J722S/TDA4VEN/TDA4AEN/AM67 Processor Silicon Revision 1.0 Texas Instruments Families of Products

Technical Reference Manual

Literature Number: SPRUJB3D
MARCH 2024 – REVISED AUGUST 2026

Table of Contents

Read This First 11
About This Manual 11
Related Documentation From Texas Instruments 11
Glossary 11
Support Resources 11
Export Control Notice 11
Release History 12
1 Introduction 13
1.1 Overview 14
1.2 Module Allocation and Instances within Device Domains 14
1.3 Functional Block Diagram 15
1.4 Device MAIN Domain 17
1.4.1 Arm® Cortex®-A53 Subsystem (A53SS) 17
1.4.2 Arm Cortex-R5F Processor (R5FSS) 17
1.4.3 Deep Learning Accelerator (C7x256v) 18
1.4.4 Vision Pre-processing Accelerator 19
1.4.5 Depth and Motion Perception Accelerator 19
1.4.6 Mailbox (MAILBOX) 20
1.4.7 Spinlock (SPINLOCK) 20
1.4.8 DDR 32-bit Subsystem (DDR32) 20
1.4.9 Data Movement Subsystem (DMSS) 20
1.4.10 General Purpose Input/Output Interface (GPIO) 21
1.4.11 Inter-Integrated Circuit Interface (I2C) 21
1.4.12 Serial Peripheral Interface (SPI) 21
1.4.13 Universal Asynchronous Receiver/Transmitter (UART) 21
1.4.14 3-port Gigabit Ethernet Switch (CPSW3G) 22
1.4.15 Universal Serial Bus (USB) Subsystem 2.0 22
1.4.16 Universal Serial Bus 3.1 Subsystem (USBSS) 22
1.4.17 Serializer/Deserializer (SERDES) 23
1.4.18 Peripheral Component Interconnect Express Subsystem (PCIE) 23
1.4.19 Enhanced Pulse-Width Modulation Module (EPWM) 23
1.4.20 Enhanced Quadrature Encoder Pulse Module (EQEP) 24
1.4.21 Enhanced Capture Module (ECAP) 24
1.4.22 Controller Area Network (MCAN) 24
1.4.23 Flash Subsystem (FSS) with Octal Serial Peripheral Interface (OSPI) 24
1.4.24 General Purpose Memory Controller (GPMC) 25
1.4.25 Error Location Module (ELM) 25
1.4.26 Multi-Media Card/Secure Digital Interface (MMCSD) 25
1.4.27 Memory Cyclic Redundancy Check 26
1.4.28 Windowed Watchdog Timer/Real Time Interrupt 26
1.4.29 TIMER 26
1.4.30 Audio Tracking Logic 26
1.4.31 Camera Subsystem 26
1.4.32 Dual Clock Comparator 27
1.4.33 Error Signaling Module (ESM) 27
1.4.34 Multi-Channel Audio Serial Port (McASP) 27
1.4.35 Camera Streaming Interface Receiver (CSI-RX) 28
1.4.36 Display Subsystem (DSS) 29
1.4.37 Open LVDS Display Interface transmitter (OLDI-TX) 29

Copyright © 2026 Texas Instruments Incorporated

1.4.38 MIPI DPHY Receiver 29
1.4.39 MIPI DPHY Transmitter 30
1.4.40 Graphics Processing Unit 30
1.4.41 Video Accelerator (CODEC) 31
1.5 Device MCU Domain 31
1.5.1 Arm Cortex-R5F Processor (R5FSS) 32
1.5.2 MCU General Purpose Input/Output Interface (MCU_GPIO) 33
1.5.3 MCU Inter-Integrated Circuit Interface (MCU_I2C) 33
1.5.4 MCU Multi-channel Serial Peripheral Interface (MCU_SPI) 33
1.5.5 MCU Universal Asynchronous Receiver/Transmitter (MCU_UART) 33
1.5.6 MCU Windowed Watchdog Timer/Real Time Interrupt 33
1.5.7 MCU TIMER 34
1.5.8 MCU Dual Clock Comparator 34
1.5.9 Memory Cyclic Redundancy Check 34
1.5.10 Controller Area Network (MCAN) 34
1.6 Device WKUP Domain 35
1.6.1 Arm Cortex-R5F Processor (R5FSS) 35
1.6.2 Inter-Integrated Circuit Interface (I2C) 35
1.6.3 Universal Asynchronous Receiver/Transmitter (UART) 36
1.6.4 MCU Error Signaling Module 36
1.6.5 Global Time Counter 36
1.6.6 Windowed Watchdog Timer/Real Time Interrupt 36
1.6.7 WKUP_TIMER 36
1.7 Device Identification 38
2 Memory Map 39
2.1 Processor Memory Map View 40
2.1.1 A53 Memory View 40
2.1.2 RAT 40
2.1.3 R5FSS Memory View 40
2.2 DDR Memory Region 43
2.3 MMU Optimization Note 43
2.4 MAIN Memory Map 43
2.5 MCU Memory Map 61
2.6 WKUP Memory Map 63
2.7 R5FSS0 Memory Map 64
2.8 MCU_R5FSS0 Memory Map 64
2.9 WKUP_R5FSS0 Memory Map 65
2.10 DMASS0 Memory Map 65
2.11 SMS0 Memory Map 65
2.12 SA3_SS0 Memory Map 65
3 System Interconnect 66
3.1 System Interconnect Overview 67
3.1.1 Terminology 67
3.2 Domain Partition 67
3.3 Initiator/Target Connectivity 68
3.4 Interrupt Condition for Interconnect 68
3.4.1 Register MMR for default_err_intr 69
3.4.2 Register MMR for default_exp 71
3.5 Coherency Features 73
3.5.1 IO Coherency Support 73
3.6 Performance Tuning 74
3.6.1 Latency Reduction 74
3.6.2 Using Quality of Service (QoS) MMR 74
3.7 Interconnect Debugging Feature 78
3.8 Quality of Service (QoS) Block 78
3.8.1 QoS Summary Table 80
3.9 Route ID 82
3.9.1 Route ID Table 82
3.10 ISC and Firewall 83
3.10.1 Initiator-Side Security Controls (ISC) 83
3.10.2 Transaction Attributes for BCDMA and PktDMA Transactions 85

Copyright © 2026 Texas Instruments Incorporated

3.10.3 Firewall Block ... 85
4 Module Integration ... 112
4.1 Memory Controllers ... 113
4.1.1 DDR32 Subsystem (DDR32SS) ... 113
4.1.2 MSRAM8KX256E ... 114
4.2 System Interconnect ... 115
4.2.1 CBASS ... 115
4.2.2 c7xv_rsws_bs_limiter Integration ... 123
4.2.3 jpgenc_rs_bw_limiter Integration ... 123
4.2.4 jpgenc_ws_bw_limiter Integration ... 124
4.2.5 gpu_rs_bw_limiter Integration ... 124
4.2.6 gpu_ws_bw_limiter Integration ... 124
4.2.7 vpac_rsws_bw_limiter Integration ... 124
4.3 Processors and Accelerators ... 125
4.3.1 Arm Cortex A53 Subsystem (A53SS) ... 125
4.3.2 Video Encoder/Decoder (CODEC) ... 128
4.3.3 Arm Cortex R5F Subsystem ... 130
4.3.4 Vision Pre-processing Accelerator (VPAC) ... 135
4.3.5 Depth and Motion Perception Accelerator (DMPAC) ... 137
4.3.6 JPGENC ... 139
4.3.7 C7X256V ... 139
4.3.8 Graphics Processing Unit (GPU) ... 144
4.4 Interprocessor Communication ... 146
4.4.1 Mailbox ... 146
4.4.2 Spinlock ... 153
4.5 Device Configuration ... 154
4.5.1 Control Module (CTRL_MMR) ... 154
4.5.2 PADCFG_CTRL ... 156
4.5.3 Voltage and Thermal Manager (VTM) ... 157
4.5.4 Power Sleep Controller (PSC) ... 159
4.5.5 Clocking ... 160
4.6 Interrupts ... 167
4.6.1 TIMESYNC_EVENT_INTROUTER ... 167
4.6.2 Generic Interrupt Controller Subsystem (GICSS) ... 168
4.6.3 main_gpiomux_introuter Integration ... 169
4.6.4 mcu_gpiomux_introuter Integration ... 173
4.7 Data Movement Architecture ... 176
4.7.1 Data Movement Subsystem (DMSS) ... 177
4.7.2 Peripheral DMA (PDMA) ... 196
4.8 Audio ... 197
4.8.1 Audio Tracking Logic (ATL) ... 197
4.8.2 Multichannel Audio Serial Port (MCASP) ... 199
4.9 General Connectivity ... 211
4.9.1 General Purpose Input/Output (GPIO) ... 211
4.9.2 Inter-Integrated Circiuit (I2C) ... 221
4.9.3 Multichannel Serial Peripheral Interface (MCSPI) ... 225
4.9.4 Universal Asynchronous Receiver/Transmitter (UART) ... 229
4.10 High-speed Serial Interfaces ... 235
4.10.1 Peripheral Component Interconnect Express (PCIe) Subsystem ... 235
4.10.2 Gigabit Ethernet Switch (CPSW) ... 238
4.10.3 Universal Serial Bus Subsystem (USB) ... 242
4.10.4 Serializer/Deserializer (SerDes) ... 248
4.11 Memory Interfaces ... 250
4.11.1 Flash Subsystem (FSS) ... 250
4.11.2 Octal Serial Peripheral Interface (OSPI) ... 251
4.11.3 General-Purpose Memory Controller (GPMC) ... 252
4.11.4 Error Location Module (ELM) ... 253
4.11.5 Multimedia Card Secure Digital (MMCSD) ... 254
4.12 Industrial and Control Interfaces ... 257
4.12.1 Modular Controller Area Network (MCAN) ... 257
4.12.2 Enhanced Capture (ECAP) ... 263

Copyright © 2026 Texas Instruments Incorporated

4.12.3 Enhanced Pulse Width Modulation (EPWM)....265
4.12.4 Enhanced Quadrature Encoder Pulse (EQEP)....268
4.13 Camera Subsystem....270
4.13.1 Camera Serial Interface Receiver (CSI_RX_IF)....270
4.13.2 MIPI D-PHY Receiver (DPHY_RX)....276
4.13.3 MIPI D-PHY Transmitter (DPHY_TX)....278
4.13.4 Camera Streaming Interface Transmitter (CSI_TX_IF)....278
4.14 Timer Modules....280
4.14.1 Global Timebase Counter (GTC)....280
4.14.2 Real Time Interrupt (RTI)....281
4.14.3 Real-Time Clock (RTC)....293
4.14.4 Timer....295
4.15 Internal Diagnostic Modules....320
4.15.1 Dual Clock Comparator (DCC)....320
4.15.2 Error Signaling Module (ESM)....335
4.15.3 Memory Cyclic Redundancy Check (MCRC64)....337
4.15.4 Programmable Built-In Self-Test (PBIST)....338
4.15.5 ECC Aggregator (ECC_AGGR)....341
4.16 Display Subsystem (DSS)....344
4.16.1 DSS_UL Unsupported Features....344
4.16.2 DSS Integration....344
4.16.3 DSS_DSI Integration....346
4.16.4 OLDI_TX_CORE....347
4.17 On-Chip Debug....349
4.17.1 DEBUGSS_WRAP Integration....349
4.17.2 debugss Integration....349
4.17.3 dbgsuspendrouter Integration....350
4.17.4 icemelter Integration....350
4.17.5 STM Integration....351
4.17.6 CTI Integration....351
4.17.7 k3_arm_atb_funnel_3_32 Integration....351
4.17.8 sam67_debug_funnel_2_32 Integration....351
4.17.9 sam67_debug_funnel_2_64 Integration....351
5 Initialization....352
5.1 Initialization Overview....353
5.1.1 ROM Code Overview....353
5.1.2 Bootloader Modes....354
5.1.3 Terminology....354
5.2 Boot Process....356
5.2.1 Public ROM Code Architecture....356
5.2.2 M4 ROM Description....357
5.2.3 Boot Process Flow....357
5.3 Boot Mode Pins....361
5.3.1 BOOTMODE Pin Mapping....362
5.4 Boot Modes....363
5.4.1 Boot with OSPI Controller....364
5.4.2 I2C Boot....376
5.4.3 SD Card Boot....377
5.4.4 eMMC Boot....379
5.4.5 Ethernet Boot....380
5.4.6 USB Boot....384
5.4.7 UART Boot....385
5.4.8 GPMC NOR Boot....387
5.4.9 GPMC NAND Boot....389
5.4.10 Serial NAND Boot....390
5.4.11 No boot/Development boot....392
5.5 PLL Configuration....393
5.6 Boot Parameter Tables....395
5.6.1 Common Header....395
5.6.2 PLL Setup....396
5.6.3 OSPI/QSPI/SPI Boot Parameter Table....397

Copyright © 2026 Texas Instruments Incorporated

5.6.4 UART Boot Parameter Table ... 398
5.6.5 I2C Boot Parameter Table ... 398
5.6.6 MMCSD/eMMC Boot Parameter Table ... 399
5.6.7 Ethernet Boot Parameter Table ... 400
5.6.8 xSPI/Fast-xSPI Boot Parameter Table ... 401
5.6.9 USB DFU Boot Parameter Table ... 402
5.6.10 USB MSC Boot Parameter Table ... 402
5.6.11 GPMC NOR Boot Parameter Table ... 403
5.6.12 GPMC NAND Boot Parameter Table ... 403
5.7 Boot Image Format ... 405
5.7.1 Overall Structure ... 405
5.7.2 X.509 Certificate ... 405
5.7.3 Organizational Identifier (OID) ... 405
5.7.4 X.509 Extensions Specific to Boot ... 406
5.7.5 Extended Boot Info Extension ... 406
5.7.6 Generating X.509 Certificates ... 409
5.8 Boot Memory Maps ... 412
5.8.1 Memory Layout/MPU ... 412
5.8.2 Global Memory Addresses Used by ROM Code ... 412
6 Device Configuration ... 413
6.1 Control Module ... 414
6.1.1 Memory Mapped Control Register Modules (CTRL_MMR) ... 414
6.1.2 General Purpose Control Register Modules Overview ... 414
6.1.3 I/O Debounce Control Registers ... 416
6.1.4 Security Control Register Modules ... 418
6.2 Power ... 418
6.2.1 Power Management Overview ... 418
6.2.2 Power Control Modules ... 418
6.2.3 Power Management Techniques ... 426
6.2.4 Power Management Subsystems ... 427
6.3 Reset ... 459
6.3.1 Overview ... 459
6.3.2 Reset Sources ... 470
6.3.3 Reset Status ... 471
6.3.4 Reset Controls ... 474
6.3.5 Reset Details ... 477
6.4 Clocking ... 499
6.4.1 Overview ... 499
6.4.2 Clock Inputs ... 500
6.4.3 Clock Outputs ... 501
6.4.4 Device Oscillators ... 504
6.4.5 PLLs ... 507
7 Processors and Accelerators ... 527
7.1 Arm® Cortex®-A53 Subsystem (A53SS) ... 528
7.1.1 A53SS Overview ... 529
7.1.2 A53SS Functional Description ... 531
7.2 Arm Cortex R5F Subsystem (R5FSS) ... 537
7.2.1 R5FSS Overview ... 538
7.2.2 R5FSS Functional Description ... 540
7.3 Cortex R5F Subsystem (R5FSS) ... 553
7.3.1 R5FSS Overview ... 553
7.3.2 R5FSS Functional Description ... 556
7.3.3 Vectored Interrupt Manager (VIM) ... 575
7.4 Device Manager Cortex R5F Subsystem (WKUP_R5FSS) ... 585
7.4.1 WKUP_R5FSS Overview ... 585
7.4.2 WKUP_R5FSS Functional Description ... 588
7.5 Vectored Interrupt Manager (VIM) ... 607
7.5.1 VIM Overview ... 607
7.5.2 VIM Functional Description ... 607
7.5.3 Interrupt Conditions ... 610
7.5.4 Memory Map ... 614

Copyright © 2026 Texas Instruments Incorporated

7.5.5 VIM Registers....615
7.5.6 Module I/O....623
7.5.7 Programmer's Guide....625
7.6 Video Encoder/Decoder (VENC/VDEC)....625
7.6.1 Introduction....625
7.6.2 Features....626
7.6.3 Block Diagram....630
7.7 Vision Pre-processing Accelerator (VPAC)....633
7.7.1 VPAC Overview....633
7.7.2 VPAC Subsystem Level....637
7.7.3 VPAC Vision Imaging Subsystem (VISS)....652
7.7.4 VPAC Lens Distortion Correction (LDC) Module....740
7.7.5 VPAC Multi-Scaler (MSC)....764
7.8 Depth and Motion Perception Accelerator (DMPAC)....786
7.8.1 DMPAC Overview....786
7.9 Graphics Accelerator (GPU)....787
7.9.1 GPU Overview....787
8 Inter-processor Communication Scheme (IPC)....789
8.1 Mailbox....790
8.1.1 Mailbox Overview....790
8.1.2 Mailbox Functional Description....791
8.1.3 Mailbox Programming Guide....795
8.2 Spinlock....797
8.2.1 Spinlock Overview....797
8.2.2 Spinlock Functional Description....797
8.2.3 Spinlock Programming Guide....799
8.3 Secure Proxy (SEC_PROXY)....801
9 Memory Controllers....802
9.1 DDR Subsystem (DDRSS)....803
9.1.1 DDRSS Overview....803
9.1.2 DDRSS Environment....805
9.1.3 DDRSS Functional Description....807
10 Interrupts....818
10.1 Interrupt Architecture....819
10.1.1 ESM Connectivity....820
10.1.2 PSIL Events....823
10.1.3 Object Missing....825
10.1.4 Utilizing Miscellaneous Signals as Interrupt....825
10.2 Interrupt Controllers....828
10.2.1 Generic Interrupt Controller (GICSS)....828
10.3 Interrupt Router (INTROUTER)....833
10.3.1 INTROUTER Integration....833
10.4 Interrupt Sources....833
10.4.1 Interrupts (Inputs)....834
10.4.2 Interrupts (Outputs)....909
11 Data Movement Architecture....984
11.1 Data Movement Architecture Overview....985
11.1.1 Overview....985
11.1.2 Definition of Terms....999
11.1.3 DMSS Hardware/Software Interface....1001
11.1.4 Operational Description....1028
11.2 Data Movement Subsystem (DMSS)....1047
11.2.1 Data Movement Subsystem (DMSS)....1048
11.2.2 Ring Accelerator (RINGACC)....1053
11.2.3 Secure Proxy (SEC_PROXY)....1064
11.2.4 Interrupt Aggregator (INTAGGR)....1070
11.2.5 Packet Streaming Interface Link (PSI-L)....1075
11.3 Peripheral DMA (PDMA)....1079
11.3.1 PDMA Controller....1079
12 Peripherals....1143
12.1 General Connectivity Peripherals....1144

Copyright © 2026 Texas Instruments Incorporated

12.1.1 General-Purpose Interface (GPIO) ... 1145
12.1.2 Inter-Integrated Circuit (I2C) Interface ... 1154
12.1.3 Multichannel Serial Peripheral Interface (MCSPI) ... 1181
12.1.4 Universal Asynchronous Receiver/Transmitter (UART) ... 1228
12.2 High-speed Serial Interfaces ... 1288
12.2.1 Gigabit Ethernet Switch (CPSW3G) ... 1289
12.2.2 Peripheral Component Interconnect Express (PCIe) Subsystem ... 1394
12.2.3 Serializer/Deserializer (SerDes) ... 1419
12.2.4 Universal Serial Bus Subsystem (USBSS) ... 1428
12.2.5 Universal Serial Bus Subsystem (USBSS) ... 1433
12.3 Memory Interfaces ... 1439
12.3.1 Flash Subsystem (FSS) ... 1440
12.3.2 Octal Serial Peripheral Interface (OSPI) ... 1445
12.3.3 General-Purpose Memory Controller (GPMC) ... 1474
12.3.4 Error Location Module (ELM) ... 1583
12.3.5 Multi-Media Card Secure Digital (MMCSD) Interface ... 1593
12.4 Industrial and Communication Interfaces ... 1666
12.4.1 Modular Controller Area Network (MCAN) ... 1667
12.4.2 Enhanced Capture (ECAP) Module ... 1704
12.4.3 Enhanced Pulse Width Modulation (EPWM) Module ... 1733
12.4.4 Enhanced Quadrature Encoder Pulse (EQEP) Module ... 1795
12.5 Audio Peripherals ... 1820
12.5.1 Audio Tracking Logic (ATL) ... 1821
12.5.2 Multichannel Audio Serial Port (MCASP) ... 1822
12.6 Camera Peripherals ... 1896
12.6.1 Camera Serial Interface Receiver (CSI_RX_IF) ... 1897
12.6.2 MIPI D-PHY Receiver (DPHY_RX) ... 1919
12.6.3 MIPI D-PHY Transmitter (DPHY_TX) ... 1926
12.6.4 Camera Streaming Interface Transmitter (CSI_TX_IF) ... 1928
12.7 Timer Modules ... 1944
12.7.1 Global Timebase Counter (GTC) ... 1944
12.7.2 RTI-Windowed Watchdog Timer (WWDT) ... 1946
12.7.3 Real-Time Clock (RTC) ... 1953
12.7.4 Timers ... 1966
12.8 Internal Diagnostics Modules ... 1989
12.8.1 Dual Clock Comparator (DCC) ... 1989
12.8.2 Error Signaling Module (ESM) ... 1997
12.8.3 Memory Cyclic Redundancy Check (MCRC) Controller ... 2012
12.8.4 ECC Aggregator ... 2026
12.8.5 ECC Aggregator Configurations ... 2052
12.8.6 Interconnect ECC Aggregators ... 2127
12.9 Display Subsystem and Peripherals ... 2312
12.9.1 Display Subsystem (DSS) ... 2313
12.9.2 MIPI Display Serial Interface (DSI) Controller ... 2399
13 On-Chip Debug ... 2462
13.1 On-Chip Debug Overview ... 2463
13.2 On-Chip Debug Features ... 2463
13.3 On-Chip Debug Functional Description ... 2464
13.3.1 On-Chip Debug Block Diagram ... 2464
13.3.2 Device Interfaces ... 2465
13.3.3 Debug and Boundary Scan Access and Control ... 2466
13.3.4 Debug Boot Modes and Boundary Scan Compliance ... 2467
13.3.5 Power, Reset, Clock Management ... 2468
13.3.6 Debug Cross Triggering ... 2468
13.3.7 Peripheral Suspend ... 2469
13.3.8 R5FSS0, MCU_R5FSS0 and WKUP_R5FSS0 Debug ... 2471
13.3.9 A53SS0 Debug ... 2472
13.3.10 C7XSS0 and C7XSS1 Debug ... 2473
13.3.11 SoC Debug and Trace ... 2474
13.3.12 Trace Traffic ... 2478
13.3.13 Application Support ... 2478

Copyright © 2026 Texas Instruments Incorporated

14 Revision History 2480

Copyright © 2026 Texas Instruments Incorporated

Read This First

About This Manual

This Technical Reference Manual (TRM) details the integration, the environment, the functional description, and the programming models for each peripheral and subsystem in the device.

Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for the device, visit the Texas Instruments website at www.ti.com.

Glossary

TI Glossary	This glossary lists and explains terms, acronyms, and definitions.

Support Resources

TI E2E™ support forums are an engineer's go-to source for fast, verified answers and design help — straight from the experts. Search existing answers or ask your own question to get the quick design help you need.

Linked content is provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

Export Control Notice

Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU, and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from disclosing party under nondisclosure obligations (if any), or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorization from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws.

Copyright © 2026 Texas Instruments Incorporated

Release History

The following table summarizes the J722S, TDA4VEN, TDA4AEN, AM67 Silicon Revision 1.0 Technical Reference Manual versions.

Version	Literature Number	Date	Notes
*	SPRUJB3	March 2024	See (1)

Trademarks

TI E2E™, C7™, Sitara™, and XDS™ are trademarks of Texas Instruments.
Neon™, CoreSight™, MPCore™, and CoreLink™, and are trademarks of Arm Ltd.
HD Radio™ is a trademark of iBiquity Digital Corporation.
Arm®, and Cortex®, and are registered trademarks of Arm Ltd.
PCI-Express® and PCIe® are registered trademarks of PCI-SIG.
Thumb® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
ARM® and are registered trademarks of ARM Ltd..
MIPI® is a registered trademark of Mobil Industry Processor Interface Alliance.
TrustZone® are registered trademarks of Arm.
All trademarks are the property of their respective owners.

Copyright © 2026 Texas Instruments Incorporated

Chapter 1
Introduction

This chapter introduces the features, subsystems, and architecture of the J722S Processor Platform high-performance System-on-Chip (SoC).

Note

This document describes the Superset architecture, processors and peripherals of the J722S Family of SoCs, which are part of the K3 Multicore SoC architecture platform. Not all features are available on each family of devices. The superset J722S devices are available for preproduction software development. Software should constrain the features used to match the intended production device. For more information on the specific features, processors and peripherals available on a particular device, refer to the Device Comparison table in the corresponding device-specific Data sheet.

The J722S Processor Platforms are hereinafter commonly referred to as J722S*platform*, *device*, or *SoC*.

TI limits support for this family of SoCs to features that are supported via Software Development Kits (SDK). The SDK “build sheet” is available for download as part of each SDK and should be referenced to understand the subset of SoC hardware functionality that is available in software.

This document describes the Superset features of the Modules integrated into this Device. See the Module Integration section of this document for a list of module features not supported by the integration on this Device.

1.1 Overview 14
1.2 Module Allocation and Instances within Device Domains 14
1.3 Functional Block Diagram 15
1.4 Device MAIN Domain 17
1.5 Device MCU Domain 31
1.6 Device WKUP Domain 35
1.7 Device Identification 38

Copyright © 2026 Texas Instruments Incorporated

1.1 Overview

1.2 Module Allocation and Instances within Device Domains

Module Full Name	Module Abbreviation	Device Domain		
		MCU	MAIN	WKUP
Quad-Core Arm Cortex A53 Subsystem	A53SS	-	1	-
Single-Core Arm Cortex-R5F Subsystem	R5FSS	1	1	1
Deep Learning Accelerator (C7™ DSP 256b + MMA)	DLA	-	2	-
Vision Pre-processing Accelerator	VPAC	-	1	-
Depth and Motion Perception Accelerator	DMPAC	-	1	-
Interprocessor Communication - Mailbox	Mailbox	-	1	-
Interprocessor Communication - Spinlock	Spinlock	-	1	-
Dual Data Rate Random Access Memory (32-Bit) Subsystem	DDRSS	-	1	-
Data Movement Subsystem	DMSS	-	2	-
Peripheral DMA	PDMA	-	3	-
General Purpose Input/Output	GPIO	1	2	-
Inter-Integrated Circuit	I2C	1	5	1
Multi-channel Serial Peripheral Interface	MCSPI	2	3	-
Universal Asynchronous Receiver/Transmitter	UART	1	7	1
3 Port Gigabit Ethernet Switch	CPSW	-	1	-
Universal Serial Bus 2.0	USB2	-	2	-
Universal Serial Bus 3.0	USB3	-	1	-
Peripheral Component Interconnect Express	PCIe	-	1	-
Serializer/Deserializer	SERDES	-	2	-
Enhanced Pulse Width Modulation Module	EPWM	-	3	-
Enhanced Quadrature Encoder Pulse Module	EQEP	-	3	-
Enhanced Capture Module	ECAP	-	3	-
Controller Area Network Interface	MCAN	2	2	-
Audio Tracking Logic	ATL	-	1	-
Flash Memory Subsystem	FSS	-	1	-
Octal Serial Peripheral Interface	OSPI	-	1	-
General Purpose Memory Controller	GPMC	-	1	-
Error Location Module	ELM	-	1	-
Multi-Media Card/Secure Digital Interface	MMCSD	-	3	-
Global Time Counter	GTC	-	-	1
Real Time Interrupt/Windowed WatchDog Timer	RTI/WWDT	1	5	1
Dual-Mode Timer	TIMER	4	8	2
Dual Clock Comparator	DCC	2	9	-
Error Signaling Module	ESM	-	1	1
Memory Cyclic Redundancy Check Controller	MCRC	1	1	-
Multi-channel Audio Serial Port	McASP	-	5	-
Camera Serial Interface Receiver	CSI-RX	-	4	-
Camera Serial Interface Transmitter	CSI-TX	-	1	-
Display Subsystem	DSS	-	2	-
Video Encoder/Decoder Engine	CODEC	-	1	-
DPHY Receiver	DPHY4RX		4	-

Copyright © 2026 Texas Instruments Incorporated

Module Full Name	Module Abbreviation	Device Domain		
		MCU	MAIN	WKUP
DPHY Transmitter	DPHYTX		1	-
Graphics Processing Unit	GPU		1	-

1.3 Functional Block Diagram

Copyright © 2026 Texas Instruments Incorporated

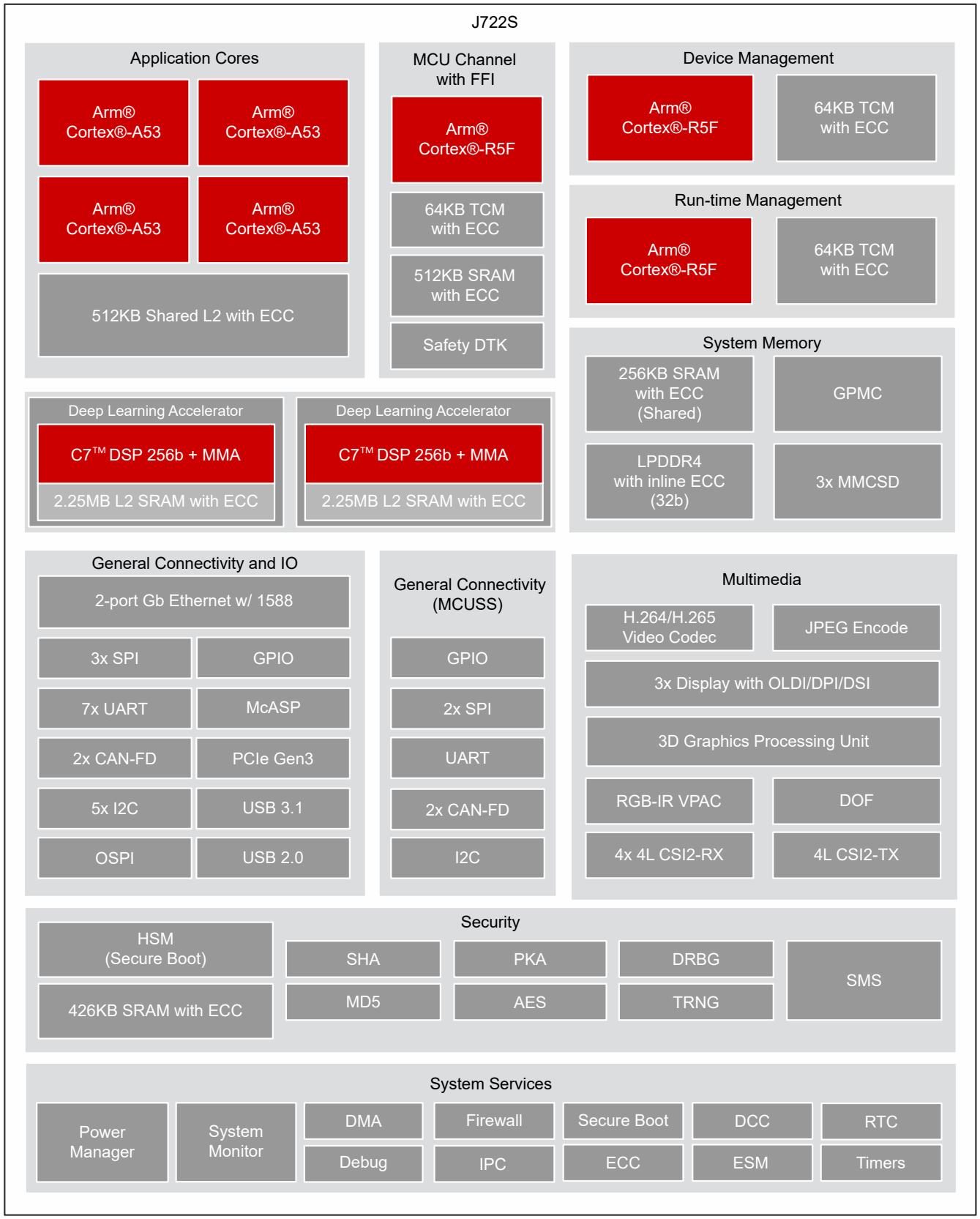


Figure 1-1. Device Block Diagram

Copyright © 2026 Texas Instruments Incorporated

1.4 Device MAIN Domain

This section describes the modules integrated in the device MAIN domain.

1.4.1 Arm® Cortex®-A53 Subsystem (A53SS)

The integrated 64-bit Arm® Cortex®-A53 subsystem (A53SS) supports the following main features:

- Full Arm®v8-A architecture compliancy
- Advanced Single Instruction Multiple Data (SIMD) and floating point extension (Arm® Neon™)
- Floating-Point Unit (FPU) VFPv4
- Armv8 Cryptography Extensions
- Arm General Interrupt Controller (GICv3) architecture
- In-order pipeline with symmetric dual-issue of most instructions
- 32KB program and 32KB data Level 1 (L1) Cache
- Support for up to four timers within each Cortex-A53 core
- Arm® CoreSight™ Debug and Trace Architecture
- ECC protection for L1 data cache and L2 Cache
- Parity protection for L1 Instruction Cache
- 128-bit wide, synchronous or asynchronous VBUSM initiator interface
- Dedicated RTI windowed watchdog timer per core
- Support for Big-Endian (BE) and Little-Endian (LE) at core level
- Interface with Arm GIC-500 Interrupt Controller (SoC level, not part of A53SS)
- Advanced power management for low power optimization

1.4.2 Arm Cortex-R5F Processor (R5FSS)

The ARM Core Cortex-R5F processor subsystem (R5FSS) supports the following main features:

- Armv7-R architecture
- R5FSS Memory System
  - 32KB Instruction Cache
    - 4x ways
    - SECDED ECC protected per 64 bits
  - Data Cache
    - 4x ways
    - SECDED ECC protected per 32 bits
  - 64KB tightly-coupled memory (TCM) per CPU
    - SECDED ECC protected per 32 bits
    - TCM hard error cache Implemented in CPU
    - Readable/writable from system
    - 32KB TCMA (ATCM)
    - 16KB TCMB0 (B0TCM)
    - 16KB TCMB1 (B1TCM)
- Full-precision Floating Point (VFPv3)
- 16-region Memory Protection Unit (MPU)
- 8 breakpoints, 8 watch points
- CoreSight Debug Access Port (DAP)
- CoreSight ETM-R5 interface (CTI, ETM)
- Performance Monitoring Unit (PMU)
- 32-bit to 36-bit Region-based Address Translation (RAT) on memory access initiators
- Integrated Vectored Interrupt Manager (VIM) per core with 256 Interrupt Inputs each
  - Programmable interrupt priority (4-bit)
  - Programmable interrupt enable mask
  - Software-generated interrupts

Copyright © 2026 Texas Instruments Incorporated

– Synchronous clock domain crossing on all core interfaces

Note

CORE0 has 64KB of TCM.

Note

These details describe a superset of the R5FSS memory configuration. For additional details on device memory availability, please refer to the device-specific Datasheet.

1.4.3 Deep Learning Accelerator (C7x256v)

- Single Core C7x/MMA subsystem
  - C7x ISA and RISC-V G ISA extensions
    - Vector DSP, 40 GFLOPS
    - 256-bit vector width
    - Deep-learning Matrix Multiply Accelerator (MMA), up to 2 TOPS
  - L1 memory architecture
    - 32KB I-cache
    - 64KB D-cache
  - L2 memory architecture
    - Repurposing of its L2/EL2 embedded memory for use by SOC resources when C7x/MMA/DRU are disabled
    - 1.25MB L2 with ECC protection on L2 SRAM
    - Unified Memory Controller (UMC) facilitates L2 SRAM accesses from CPU and SOC (DMAs) as well as EMIF accesses from CPU.
    - DRU (DMA engine) integrated that facilitates data transfer between L2, VPAC and EMIF.
      - Data compression support
      - Event bus interface integrates with SOC DMSS
      - Tightly coupled with C7x/MMA (DRU is not available for use when the C7x/MMA is disabled)
  - Interrupts
    - Local event controller (CLEC) integrated within CorePac for routing and handling of DRU interrupts, CPU generated IPC events to SOC, interrupts coming from SOC
  - Security
    - Support for C7x Authenticated Boot. Security is in-context during boot to facilitate the authentication and loading of the C7x image.
    - The C7x L2 SRAM (UMC) should not be used for storing security content as there is no protection of UMC memory from C7x itself. In a threat scenario, the C7x can be activated if in reset to execute malicious code to access data in UMC thereby defeating a protection for security data in UMC.
    - No security should be inferred from DRU as this IP is out of scope for security.
    - TrustZone debug controls (DBGEN, NIDEN, SPIDEN, SPNIDEN) are supported
  - Safety
    - Support for reporting safety errors to SOC via interrupt
  - Debug
    - Independent debug interface supporting access to embedded features
    - Debug features include: Core debug, Advanced Event Triggering, Trace, CTSETs, and an embedded CP Tracer Aggregator with bus probes on internal memory interfaces
  - Power Management
    - support for granular gated clock domains to support low power modes and to support memory interface utilization while IP disabled.
  - Advanced SOC Integration
    - Dedicated PLL for full flexibility in performance and power trade-offs
    - Dedicated windowed watchdog timer per core

Copyright © 2026 Texas Instruments Incorporated

1.4.4 Vision Pre-processing Accelerator

The Vision Pre-processing Accelerator (VPAC) provides a set of common vision primitive functions, performing various pixel data processing tasks, such as: color processing and enhancement, noise filtering, wide dynamic range (WDR) processing, lens distortion correction, pixel remap for de-warping, on-the-fly scale generation, on-the-fly pyramid generation, and offloads these common tasks from the main SoC processors (ARM, DSP, and so forth). The VPAC includes the following processing and infrastructure sub-modules:

- Vision Imaging Sub-System (VISS) which provides raw data image processing such as:
  - Wide Dynamic Range (WDR) merge
  - Defect Pixel Correction (DPC)
  - Lens Shading Correction (LSC)
  - Global/Local Brightness and Contrast Enhancement (GLBCE)
  - Advanced Spatial Noise Filter (NSF4V)
  - Edge Enhancement (EE)
  - Demosaicing
  - Color conversion
- Lens Distortion Correction (LDC) block, which provides data reading from memory (DDR or on-chip) and applies perspective transformation as well as correction of lens distortion (including fisheye lenses).
- Multi-Scalar (MSC) block reads data from memory (DDR or on-chip) to internal shared level 2 (SL2) memory and generates up to 10 scaled outputs from one or two inputs, with various scaling ratios.
- Noise Filter (NF) block reads data from memory (DDR or on-chip) to internal SL2 memory and does Bilateral filtering to remove noise.
- Hardware Thread Scheduler (HTS) provides inter-processor communication among various VPAC sub-modules (VISS, LDC, MSC and NF) as well as with the local DMA Engine (UTC).
- Internal Shared Level 2 (SL2) memory for data exchange across VPAC sub-modules (VISS, LDC, MSC and NF) and from DDR/MSMC, using the K3 Data Movement Architecture (DMA) mechanism
- Load/Store Engine (LSE) which performs data load and store tasks on a the SL2 memory for the hardware accelerator algorithm cores
- VISS to LDC direct OTF (On-the-Fly) for multi-camera
- YUV422 10 and 12 bit format support on all units except NF (for example, LDC/MSC)
- Simultaneous visual (HV) and analytics (MV) output to system memory including L3 flex connect, saving need for additional read from system memory for HV+MV processing.
- Chromatic Aberration Correction (CAC) to support lower cost lens
- RGBIR support

1.4.5 Depth and Motion Perception Accelerator

The Depth and Motion Perception Accelerator (DMPAC) computes dense stereo depth maps (*depth*) and dense optical flow vectors (*motion*) from camera inputs. The stereo and optical flow processing is partitioned into two top level sub-blocks: the Dense Optical Flow (DOF) engine and the Stereo Disparity Engine (SDE). The DOF and SDE blocks share a common local memory, DMA, external messaging and control infrastructure. The DMPAC provides the following main features, among others:

- Image resolution up to 2MPix (maximum horizontal resolution up to 2048 pixels; maximum vertical resolution up to 1024 pixels)
- Maximum throughput up to 183MPix/sec for DOF, and up to 82MPix/sec for SDE
- Simultaneous operation of Stereo (1MPix@30fps) and Optical flow (1MP@30fps)
- 12-bit fully packed luminance data input pixel data format (other formats supported through conversion)
- Packed 16-bit (disparity) or 32-bit (flow vector) output data formats
- Shared Level 2 (SL2) memory sub-system, which serves the data transfer of the DOF and SDE blocks
- Unified Transfer Controller (UTC), which serves as a DMA engine
- Hardware Thread Scheduler (HTS) block for messaging and control mechanism
- Counter, Timer and System Event Trace (CTSET) module, which provides event tracing capability for the Stereo and Optical Flow hardware threads

Copyright © 2026 Texas Instruments Incorporated

1.4.6 Mailbox (MAILBOX)

Integrated in MAIN domain: One Mailbox (MAILBOX) module to facilitate the communication between the various on-chip processor cores of the device by providing a queued mailbox-interrupt mechanism with the following main features:

- 1x clusters
- 32-bit message width
- Message reception and queue-not-full notification using interrupts
- Non-intrusive emulation

1.4.7 Spinlock (SPINLOCK)

One Spinlock module with (256 hardware semaphores) for synchronizing the processes running on multiple cores in the device.

1.4.8 DDR 32-bit Subsystem (DDR32)

Integrated in MAIN domain: One instance of DDR 32-bit Subsystem (DDR32) (also referred to as DDRSS) is used as an interface to external SDRAM devices which can be utilized for storing program or data. The DDRSS provides the following main features:

- Support of LPDDR4 memory type (up to 4000 MT/s)
- 32-bit memory bus interface with in-line ECC
- Up to 8 GB memory address range
- Support of dual rank configuration
- Support of automatic idle power saving mode when no or low activity is detected
- Class of Service (CoS) - three latency classes supported
- Dynamic change of refresh reach via SW for extended temperatures
- Prioritized refresh scheduling
- Statistical counters for performance management

1.4.9 Data Movement Subsystem (DMSS)

Integrated in the MAIN domain: One Data Movement Subsystem (DMSS) can be used for efficient transfer of data support between software, firmware, and hardware in all combinations. It consists of the following main modules:

- Packet DMA Controller
- Block Copy DMA Controller
- Ring Accelerator provides hardware acceleration to enable straightforward passing of work between a producer and a consumer and has the following main features:
  - Supports independent memory-mapped ring structures
  - Supports various modes for each ring based on usage and compatibility
  - Provides single-word deep shared incoming Transfer Response FIFO
  - Provides bit-wide source VBUSM read/write target interface for accesses from DMA controller entities
- Secure proxy module is a modified version of the proxy module and in addition has the following main features:
  - Provides proxy function to store large data bursts that a host can only access in smaller amounts
  - Supports a configurable number of threads, where each has their own independent proxy function
  - Keeps the large data burst coherent until the complete data has been accessed
  - Allows for interleaved access between multiple hosts or tasks using multiple proxy threads
  - Supports a configurable target resource. The target has a configurable number of channels, size of each channel and base address
  - Supports a programmable fixed queue for each proxy thread
  - Supports multiple producers all writing to the same queue
  - Supports programmable thresholds for when to generate events
  - Supports a max message count for outbound proxy threads limiting the number of messages a thread can produce

Copyright © 2026 Texas Instruments Incorporated

  - Optionally supports dynamic clock gating
- Interrupt Aggregator modules provide a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. Main features are as follows:
  - 64-bit VBUSP target using 64-bit registers
  - Provide a set of TI Interrupt Architecture compliant interrupt status and mask registers which are used to pass specific event status to one or more host blocks.
  - Provides an optional set of Unmapped event (UNMAP) which can take an 'unmapped' event from the ingress ETL and generate a Global event on the egress Event Transport Lane (ETL) interface.
  - Provides an optional set of Global Event Input (GEVI) counters which can count events delivered via an ingress Event Transport Lane (ETL)
  - Provides an optional set of Local Event Input (LEVI) to Global event registers which can be used to convert pulsed discrete interrupt inputs or clock synchronous rising edge events into Global events on an egress ETL
  - Provides an optional set of Global Event Input (GEVI) 'Multicast' registers which can take a Global event from an ingress ETL and generate two egress Global events on two egress ETL interfaces

1.4.10 General Purpose Input/Output Interface (GPIO)

Integrated in MAIN domain: General Purpose Input/Output (GPIO) module that provide dedicated general-purpose pins that can be configured as either inputs or outputs. The GPIO module main features include:

- Support of 9 banks x 16 GPIO pins
- Support of up to 9 banks of interrupt capable GPIOs
- Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO pin
- Set/clear functionality per individual GPIO pin

1.4.11 Inter-Integrated Circuit Interface (I2C)

Integrated in MAIN domain: Five instances of the multi-controller Inter-Integrated Circuit (I2C) interface module, each with the following main features:

- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- 8-bit-wide data access
- Support of multi-controller transmitter/peripheral receiver and receiver/peripheral transmitter modes
- Built-in FIFOs with fixed size of 32 bytes.

1.4.12 Serial Peripheral Interface (SPI)

Integrated in MAIN domain: Three instances of the Serial Peripheral Interface (SPI) module with the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of SPI word lengths, ranging from 4 to 32 bits
- Up to channels in controller mode, or single channel in receiver mode
- Support for various controller multichannel modes
- Single interrupt line for multiple interrupt source events
- Support of start-bit write command
- Support of start-bit pause and break sequence
- Built-in FIFO available for a single channel

1.4.13 Universal Asynchronous Receiver/Transmitter (UART)

Seven Instances of the configurable Universal Asynchronous Receiver/Transmitter (UART) interface module have the following main features:

- 16C750-compatible interface
- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation

Copyright © 2026 Texas Instruments Incorporated

- Programmable sleep mode
- Baud-rate from 300 bits/s up to 3.6864 Mbits/s with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

Note

Only one UART instance has support for support full modem control functions. All other UART instances will support only the TX, RX, RTS, and CTS signals.

1.4.14 3-port Gigabit Ethernet Switch (CPSW3G)

Integrated in MAIN domain: One instance of the 3-port Gigabit Ethernet Switch (CPSW3G) subsystem provides Ethernet packet communication for the device. The CPSW3G subsystem provides the following main features:

- Two external Ethernet ports with selectable RGMII and RMII interfaces and one internal Communications Port Programming Interface (CPPI) port.
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Support of eight priority level Quality Of Service (QOS) - 802.1p
- Support for Audio/Video Bridging (P802.1Qav/D6.0 and 802.1Qaz)
- Ethernet port reset isolation
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)
- Differentiated Services Code Point (DSCP) Priority Mapping (IPv4 and IPv6)
- IPV4/IPV6 UDP/TCP checksum offload
- Priority Based Flow Control (801.1QBB) and Flow Control (802.3x) Support
- Wire rate switching (802.1d)
- Store and Forward Switching
- Non-Blocking switch fabric
- Support of Time Sensitive Network - IEEE P902.3br/D2.0 Interspersing Express Traffic and IEEE 802.1Qbv/D2.2 Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE) with 512 ALE table entries
- EtherStats and 802.3 Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Maximum frame size of 2020 bytes
- Management Data Input/Output (MDIO) module for PHY Management
- Host port CPPI Streaming Packet Interface
- Emulation support

1.4.15 Universal Serial Bus (USB) Subsystem 2.0

Instantiated in MAIN domain: Two instances of the Universal Serial Bus (USB) subsystem with integrated USB2.0 PHY module has the following main features:

- Dual-Role Device (DRD) capability
- Support of Peripheral (aka Device) mode at High Speed (HS at 480 Mbps) and Full Speed (FS at 12 Mbps)
- Support of Host mode at High Speed (HS at 480 Mbps), Full Speed (FS at 12 Mbps), and Low Speed (LS at 1.5 Mbps)
- Support of Host Negotiation Protocol (HNP)
- Support of USB3 low power protocol states (U1, U2, and U3)
- USB instance contains a single xHCI compliant with xHCI 1.0 specification with internal DMA controller
- Embedded USB 2.0 PHY

1.4.16 Universal Serial Bus 3.1 Subsystem (USBSS)

Instantiated in MAIN domain: One instance of the Universal Serial Bus 3.1 subsystem (USBSS) with integrated USB2.0 PHY module has the following main features:

- Dual-Role Device (DRD) capability

Copyright © 2026 Texas Instruments Incorporated

- Compliant to the USB 3.1 Gen1 Specification
- Supports Peripheral/Device mode at High-Speed (HS at 480 Mbps), and Full-Speed (FS at 12 Mbps)
- Supports Host mode at Super-Speed (SS at 5 Gbps), High-Speed (HS at 480 Mbps), Full-Speed (FS at 12 Mbps), and Low-Speed (LS at 1.5 Mbps)
- Supports Host Negotiation Protocol (HNP)
- Supports USB3 low power protocol states (U1, U2, and U3)
- Single xHCI instance compliant to the xHCI 1.0 specification with internal DMA controller
- ECC on internal RAMs
- Embedded USB 2.0 PHY

1.4.17 Serializer/Deserializer (SERDES)

Integrated in MAIN domain: Two high-speed differential interfaces implemented with the Serializer/Deserializer (SERDES) Multi-protocol Multi-link module with the following main blocks:

- Quad lane PHY with common module for peripheral and Tx clocking handling
- PIPE Rev 4.2 Interface
- Physical coding sub-block for data translation from/to the parallel interface, as well as data encoding/decoding and symbol alignment
- MUX module for device interfaces multiplexing into a single SERDES lane (Tx and Rx)
- A wrapper for sending control and reporting status signals from the SerDes and muxes

1.4.18 Peripheral Component Interconnect Express Subsystem (PCIE)

Integrated in MAIN domain: One instance of the Peripheral Component Interconnect express (PCIe) subsystem module with shared SerDes lines, provides the following main features:

- Compliant to PCI-Express® Base Specification, Revision 4.0 (Version 0.7)
- 1-lane configuration with up to 8.0 Gbps/lane (Gen3).
- Gen3 (8 Gbps 128/130-bit encoding), Gen2 (5 Gbps 8/10-bit encoding), and Gen1 (2.5 Gbps 8/10-bit encoding) with auto-negotiation
- Dual mode: Root Port (RP) or End Point (EP) operation modes, selectable via bootstrap pins
- Dynamic PIPE width change when switching between Gen1/2/3 modes
- Constant 32-bit PIPE width for Gen1/2/3 modes
- Maximum payload size of 128 bytes
- Maximum remote read request size of 4KB
- Dual mode: Root Port (RP) or End Point (EP) operation modes
- Maximum number of non-posted outstanding transactions: 32
- Resizable Base Address Registers (BAR) capability
- Separate Reference Clock with Independent Spread (SRIS)
- Legacy, MSI and MSI-X Interrupt Support
- 32 outbound address translation regions
- Precision time measurement (PTM)

1.4.19 Enhanced Pulse-Width Modulation Module (EPWM)

Integrated in MAIN domain: Three Enhanced Pulse-Width Modulation (EPWM) modules, each with the following main features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two independent PWM outputs that can be used in different configurations (with single-edge operation, with dual-edge symmetric operation or one independent PWM output with dual-edge asymmetric operation)
- Asynchronous override control of PWM signals during fault conditions
- Programmable phase-control support for lag or lead operation relative to other EPWM modules
- Dead-band generation with independent rising and falling edge delay control
- Programmable trip zone allocation of both latched and un-latched fault conditions
- Events enabling to trigger both CPU interrupts and start of ADC conversions

Copyright © 2026 Texas Instruments Incorporated

1.4.20 Enhanced Quadrature Encoder Pulse Module (EQEP)

Integrated in MAIN domain: Three 32-bit Enhanced Quadrature Encoder Pulse (EQEP) modules for position, speed, and frequency measurements support, each with the following main features:

- Input synchronization
- Three stage/six stage digital noise filter
- Quadrature decoder unit
- Position counter and control unit for position measurement
- Quadrature edge capture unit for low speed measurement
- Unit time base for speed/frequency measurement
- Watchdog timer for detecting stalls

1.4.21 Enhanced Capture Module (ECAP)

Integrated in MAIN domain: Three Enhanced Capture (ECAP) modules provide accurate timing for different events. When not being used for event capture, its resources can be used to generate a single channel of asymmetrical PWM waveforms (configurable as either one capture input, or as one auxiliary PWM output). Each ECAP module supports the following main features:

- 32-bit time base counter
- 4 x 32 bits event time-stamp capture registers
- 4 stage sequencer (Mod4 counter), synchronized to external events
- Independent edge polarity selection for up to four sequenced time-stamp capture events
- Input capture signal pre-scaling (from 1 to 16)
- Interrupt capabilities on any of the four capture events
- Support of different capture modes (single shot capture, continuous mode capture, absolute timestamp capture or delta mode time-stamp capture)

1.4.22 Controller Area Network (MCAN)

Integrated in the domain: One Controller Area Network interfaces (MCAN) supporting both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications and having the following main features:

- Conforms with CAN Protocol version 2.0 part A, B and ISO 11898-1:2015
- Full CAN FD (up to 64 data bytes) support
- SAE J1939 support
- AUTOSAR support
- Up to 32 dedicated transmit buffers and 64 dedicated receive buffers
- Two configurable receive FIFOs, up to 64 elements each
- Configurable transmit FIFO, up to 32 elements
- Configurable transmit queue, up to 32 elements
- Configurable transmit event FIFO, up to 32 elements
- Up to 128 filter elements
- Maskable interrupts, two interrupt lines
- Timestamp Counter

1.4.23 Flash Subsystem (FSS) with Octal Serial Peripheral Interface (OSPI)

Integrated in MAIN domain: One instance of the Flash Subsystem (FSS) module provides access to external flash devices via Octal Serial Peripheral Interface (OSPI) along with encryption/decryption and in-line ECC protection. FSS supports the following main features:

- Provides one OSPI flash interfaces
- OSPI interface supports:
  - Execute in place (XIP) operation
  - 32-byte Block Copy (BC) operation
  - ECC
- OSPI supports up to 4 devices
- OSPI supports single, dual, quad, or octal SPI devices

Copyright © 2026 Texas Instruments Incorporated

- The OSPI interface has independent power management for low power operations

1.4.24 General Purpose Memory Controller (GPMC)

One instance of the General-Purpose Memory Controller (GPMC) module. The GPMC is dedicated to interfacing with external memory devices and has the following main features:

- Support of 8- or 16-bit-wide data path to external memory devices
- Supports up to 4 independent chip-select regions of programmable size and programmable base addresses on 16MB, 32MB, 64MB, or 128MB boundary in a total address space of
- Support of the following wide range of external memories/devices:
  - Asynchronous or synchronous 8-bit wide memory or device (non-burst device)
  - Asynchronous or synchronous 16-bit wide memory or device
  - 16-bit non-multiplexed NOR flash device
  - 16-bit address and data multiplexed NOR flash device
  - 8-bit and 16-bit NAND flash device
  - 16-bit pseudo-SRAM (pSRAM) device
- Supports various interface protocols when communicating with external memory or external devices:
  - Asynchronous read/write access
  - Asynchronous read page access (4, 8, and 16 Word16)
  - Synchronous read/write access
  - Synchronous read burst access without wrap capability (4, 8, and 16 Word16)
  - Synchronous read burst access with wrap capability (4, 8, and 16 Word16)
- Supports up to 16-bit on-the-fly error code detection using the Bose-Chaudhuri-Hocquenghem (BCH) or Hamming code to improve the reliability of NAND with a minimum effect on software (NAND flash with 512-byte page size or greater)

1.4.25 Error Location Module (ELM)

One instance of the Error Location Module (ELM). The ELM module works in conjunction with the GPMC and has the following main features:

- ECC calculations (up to 16-bit) for NAND support and ability to work in both page-based and continuous modes
  - 4, 8, and 16 bits per 512-byte block error-location, based on BCH algorithms
  - Eight simultaneous processing contexts
  - Page-based and continuous modes
  - Interrupt generation on error-location process completion

1.4.26 Multi-Media Card/Secure Digital Interface (MMCSD)

Integrated in MAIN domain: Three Multi-Media Card/Secure Digital (MMCSD) controller modules with the following main features:

- One controller with 8-bit wide data bus
- Two with 4-bit wide data bus
- Support of eMMC5.1 Host Specification (JESD84-B51)
- Support of SD Host Controller Standard Specification - SDIO 3.00
- Integrated DMA controller supporting SD Advanced DMA - ADMA2 and ADMA3
- eMMC Electrical Standard 5.1 (JESD84-B51)
- Multi-Media card features:
  - Backward compatible with earlier eMMC standards
  - Legacy MMC SDR: 3.3/1.8 V, 8/4/1-bit bus width, 0-25 MHz, 25/12.5/3.125 MB/s
  - High Speed SDR: 3.3/1.8 V, 8/4/1-bit bus width, 0-50 MHz, 50/25/6.25 MB/s
  - High Speed DDR: 3.3/1.8 V, 8/4-bit bus width, 0-50 MHz, 100/50 MB/s
  - HS200 SDR: 1.8 V, 0-200 MHz, 8/4-bit bus width, 200/100 MB/s
- SD card support: SDIO, SDR12, SDR25, SDR50, SDR104, DDR50

Copyright © 2026 Texas Instruments Incorporated

1.4.27 Memory Cyclic Redundancy Check

- Memory Cyclic Redundancy Check module used to perform CRC to verify the integrity of a memory system. The MCRC module has the following main feature:
  - Four Channels of CRC Compression based Signature Generation
  - 8, 16, 32, or 64-bit Data Size
  - Maximum Length PSA based on 64-bit Polynomial
  - Programmable 20 bit pattern counter per channel
  - Three Operating Modes: Auto, Semi-CPU, Full CPU
  - MCRC or CPU can perform signature verification for each Channel
  - Timeout Interrupt if CRC is not performed within the time limit

1.4.28 Windowed Watchdog Timer/Real Time Interrupt

- Windowed Watchdog Timer (WWDT), implemented by using the Digital Windowed Watchdog (DWWD) function of the Real Time Interrupt (RTI) module providing timer functionality for operation systems and benchmarking code with the following main features:
  - Two independent 64 bit counter blocks
  - Four configurable compare registers for generating operating system ticks
  - Free running counter 0 can be incremented by either the internal pre-scale counter or by an external event
  - Selectable RTI clock input (derived from any of the available clock sources)
  - Fast enabling/disabling of events

1.4.29 TIMER

- Dual mode Timer (TIMER) module with support of the following main features:
  - Free running 32-bit upward counter
  - Generates a 1-ms tick with a 32.768-kHz functional clock
  - Interrupts generated on overflow, compare and capture
  - Supported modes of operation: compare and capture, auto-reload and start-stop
  - Programmable divider clock source (2^n, where n = [0-8])
  - Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
  - On-the-fly read/write register (while counting) for systems operation and benchmarking code

1.4.30 Audio Tracking Logic

The Audio Tracking Logic (ATL) module, which is used by HD Radio™ applications to synchronize the digital audio output to the baseband clock, supports the following main features:

- Contains four ATL instances, for HD Radio support and asynchronous sample rate conversion assistance
- Each instance tracks the time error between two syncs (local Audio Word Select [AWS] and Baseband Word Select [BWS])
- Each instance selects between 8 mux choices for BWS and 8 mux choices for AWS
- Each instance generates modulated ATCLK clock signals with software-initiated pulse stealing
- Selection between interface or functional clock to run error counting timers and to derive modulated clock outputs
- Clock and reset management: receives clock and reset signals from the device PSC module
- Hardware reset
- Local software reset

1.4.31 Camera Subsystem

Camera Subsystem unites three camera streaming interfaces – receiver and transmitter, allowing the device to stream video inputs from multiple cameras to the video processing accelerator (VPAC) or to internal memory and to output CSI-2 protocol image data to any device that supports MIPI CSI-2 protocol. Main modules are as follows:

- Camera Streaming Interface Receiver (CSI_RX_IF) with the following main features:
  - Compliant to MIPI CSI-2 v1.3+ and MIPI CSI-2 v2.0

Copyright © 2026 Texas Instruments Incorporated

  - Supports up to 16 virtual channels per input
  - Supports one 4MP camera or eight 2MP camera streams
  - Supports data rate up to 2.5Gbps per lane (wire rate)
  - Supports 1, 2, 3, or 4 Data Lane connections to MIPI D-PHY Receiver (DPHY_RX)
  - Over 25 different programmable formats including YUV420, YUV422, RGB, Raw, and User Defined
  - Supports four independent (simultaneous) output streams:
    - Two VP 32-bit streams to VISS inputs of VPAC image processing accelerator
    - One (up to 4 channels) PPI 16-bit pixel retransmission interface to Camera Streaming Interface Transmitter (CSI_TX_IF)
    - One (up to 32 Channels) DMA interface through a 128-bit Packet Streaming Interface Link (PSI_L) connection to NAVSS for transfers to memory:
- Functional and data path error interrupts
- ECC support
- MIPI D-PHY Receiver (DPHY_RX) with the following main features:
  - Allows the device to input video streams from external sensor cameras and other CSI2 compliant sources
  - Compliant to MIPI D-PHY standard v1.2
  - Supports up to 4 data and 1 clock lanes
  - Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane
  - Clock lane Control / Interface logic type: CIL-SCNN for HS and low power receiving
  - Data lane Control / Interface logic type: CIL-SFAN for HS and low power receiving
  - Data lanes can be independently operated in HS or ULP mode
  - Swapping of DP/DN signals within each clock/data pair
- Camera Streaming Interface Transmitter (CSI_TX_IF) with the following main features:
  - Compliant to MIPI CSI-2 v1.3+, MIPI CSI-2 v2.0, and MIPI D-PHY v1.2
  - Data rate up to 2.5 Gbps per lane (wire rate)
  - Supports 1, 2, 3, or 4 Data Lane connections to MIPI D-PHY Transmitter (DPHY_TX)
  - Over 25 different programmable formats including YUV420, YUV422, RGB, Raw, and User Defined
  - Support of 16 virtual channels
  - Support of four configurable input streams

1.4.32 Dual Clock Comparator

- Dual Clock Comparator (DCC) modules, used to determine the accuracy of a clock signal during the time execution of an application, each having the following main features:
  - Two independent counter blocks count clock pulses from each clock source
  - Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
  - Configurable time base for error signal
  - Error signal generation when one of the clocks is out of spec
  - Clock frequency measurement

1.4.33 Error Signaling Module (ESM)

Error Signaling Module (ESM) for safety-related events and/or errors aggregation from throughout the device into one location supports the following main features:

- Selectable low and high priority interrupt error pin prioritization of each error event
- Up to 1024 level or pulse error event inputs
- Error signal routed out of device through MCU_ ESM error signal
- Configurable time base for error signal
- Error forcing capability
- Internal redundant flops on safety critical fields

1.4.34 Multi-Channel Audio Serial Port (McASP)

The McASP module supports the following features:

Copyright © 2026 Texas Instruments Incorporated

- Transmit control section consisting of:
  - Transmit state machine supporting IIS and DIT modes
  - Transmit formatter supporting IIS and DIT options
  - Transmit clock generation with auto-switch on bad clock detect
  - Transmit frame sync generation
  - Transmit TDM control
- Receive control section consisting of:
  - Receive state machine implementing supported protocols
  - Receive clock generation
  - Receive frame sync generation
  - Receive TDM control
  - Option to synchronize to transmit control section
- Multiple shift registers (2 to 16 specified at module generation):
  - Individual transmit / receive / off selection
  - Transmit in IIS or DIT mode
  - Flexible order in memory to ease DMA operation versus pinout
  - TDM mode function selection (Z, 0, 1)
- Single RX interrupt generation, with individual enables for:
  - Data Ready: all slots, last slot
  - Error: overrun, unexpected frame sync, bad clock
- Two DMA events (RX, TX)
- Slot counter exported with DMA request to enable per-slot DMA request (or left/right, at minimum)
- Automatic mute output pin option with programmable polarity on error detect

1.4.35 Camera Streaming Interface Receiver (CSI-RX)

The CSI_RX module supports the following features:

- Fully Compliant to MIPI CSI v1.3 Specification
- Supports up to 16 virtual channels per input (partial support for MIPI CSI v2.0)
- Data rate up to 2.5 Gbps per lane (wire rate)
- Supports 1, 2, 3, or 4 Data Lane connection to MIPI D-PHY
- Programmable formats including YUV420, YUV422, RGB, Raw, and User Defined (over 25 different formats supported)
- Four Independent (Simultaneous) Output Streams:
  - Two VID32 streams to VISS inputs of VPAC Image Processing Accelerator (2x 16-bit Pixels Per Clock)
  - One (Up to 4 Channel) PPI 16-bit pixel Retransmission interface to CSI-TX (2x 16-bit Pixels Per Clock)
  - One (Up to 32 Channel) DMA interface through a 128 bit PSIL Connection to NavSS/DMSS for transfers to memory.
- TI Wrapper Interrupts:
  - PSIL fifo overflow
  - VP0 frame/line mismatch
  - VP1 frame/line mismatch
- CSI2-RX core functional (informational) interrupts:
  - Stream Abort Process Complete (x4)
  - Stream Stop Process Complete (x4)
  - Reception of Generic Short Packet
  - Reception of Short Packet
  - Reception of Long Packet
  - Deskew Entry
  - ECC Spares Error (protocol non-compliance)
  - D-PHY Sleep
  - D-PHY Wakeup
- For stream 0, stream monitor interrupts:
  - Line Count Error Interrupt

Copyright © 2026 Texas Instruments Incorporated

  - Frame Mismatch Error Interrupt
  - Frame Count Error Interrupt
  - FCC Stop Interrupt
  - FCC Start Interrupt
  - Line/Byte Interrupt
  - Timer / Timer Interrupt
- CSI Core / Datapath Error Interrupts:
  - Stream FIFO Overflow (x4)
  - Invalid Short Packet Received
  - Invalid Access to Configuration Register Space
  - Data ID Error in Packet Header
  - Header ECC Error (Correctable/Uncorrectable)
  - Payload CRC Error
  - Front (D-PHY to CSI-RX) FIFO Overflow

1.4.36 Display Subsystem (DSS)

The Display Subsystem IP module supports the following features:

- Two display outputs (for a single DSS7_UL instance)2
  - Up to 24-bit per pixel parallel or embedded sync output
  - Up to 300MHz pixel clock
  - Support for RGB/YUV422 modes
  - Support for progressive/interlaced modes
- Two input display processing pipelines
  - One video pipeline supporting full RGB and 8/10-bit YUV data formats with 3/5-tap 16-phase scaler capable of 0.25x to 16x resizing.
  - One video_lite pipeline supporting full RGB and 8/10-bit YUV data formats (no resizing support)
- Two Overlay Managers with multi layer alpha blending
- One DMA controller capable of supporting up to 2K input source width.
  - 48bit addressing (256 TB reach)
- On-the-fly X/Y-axis flip of the source (Flip/Mirror mode support)
- Safety check (freeze frame detection and data correctness check)

1.4.37 Open LVDS Display Interface transmitter (OLDI-TX)

The OLDI_TX module integration (two instances) supports OLDI resolution up to:

- 1920x1440@60fps with a 200MHz pixel clock in single link mode (2x independent or duplicate displays)
- 3840x1080@60fps with a 300MHz pixel clock in dual link mode

The OLDI_TX module features include:

- OLDI Mapping
  - 3 or 4 LVDS data channels
  - VESA or JEIDA bit mapping
- Multiple OLDI Link Modes
  - Single Link (Duplicate or non-duplicate mode)
  - Dual Link (odd or even data) – Link Off
- Clock Transmission
- ANSI/TIA/EIA644-A compliant VLVDS signaling

1.4.38 MIPI DPHY Receiver

The DPHY_RX module supports the following features:

- Compliant to MIPI D-PHY standard v1.2
- Supports up to 4 data and 1 clock lanes
- Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane

Copyright © 2026 Texas Instruments Incorporated

- Clock Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-SCNN** for HS and low power receiving:
  1. **(S)** Peripheral
  2. **C**lock
  3. **N**/A forward, **N**/A reverse escape mode features
- Data Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-SFAN** for HS and low power receiving:
  1. **(S)** Peripheral
  2. **F**orward direction only for high speed mode
  3. **A**ll forward direction escape mode features are supported
  4. **N**o reverse direction escape mode features are supported
- Data lanes can be independently operated in HS or ULP mode
- Swapping of DP/DN signals within each clock/data pair (Facilitated by CSI_RX_IF controller)

1.4.39 MIPI DPHY Transmitter

The DPHY_TX0 module supports the following main features:

- Compliance to MIPI D-PHY standard version 1.2.
- Supports up to 4 data lanes and 1 clock lane.
- Supports High Speed up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane.
- Supports Escape mode:
  - Remote triggers
  - LP-DT up to 10 Mbps
  - ULPS mode
- Clock Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-MCNN**: [HS-TX, LP-TX]
  1. **(M)** Controller
  2. **C**lock
  3. **N**/A forward, **N**/A reverse escape mode features.
- Data Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-MFAA**: [HS-TX, LP-TX, LP-RX, LP-CD]
  1. **(M)** Controller
  2. **F**orward direction only for High Speed mode
  3. **A**ll forward direction escape mode features are supported
  4. **A**ll reverse direction escape mode features are supported.
- Data Lanes can be independently operated in HS or ULP mode.
- Includes a CMN block with reference generators / resistor calibration and an integrated PLL.
- Fault detection:
  - Contention detection (for example, when RX and TX sides of same link drive opposite LP levels).
  - Sequence error detection (corruption on lanes).

1.4.40 Graphics Processing Unit

The GPU module supports the following features:

- Base architecture, fully compliant with the following APIs:
  - OpenGL ES 3.2
  - OpenCL 1.2 EP
  - Vulkan 1.2
- Tile-based deferred rendering architecture for 3D graphics workloads, with concurrent processing of multiple tiles
- Support for DRM security
- Support for GPU virtualization
  - Up to 8 virtual GPUs
  - Separate IRQ for each OSID
- Multi-threaded Unified Shading Cluster (USC) engine incorporating pixel shader, vertex shader and GP-GPU (compute shader) functionality
- USC incorporates an ALU architecture with high SIMD efficiency

Copyright © 2026 Texas Instruments Incorporated

- Fully virtualized memory addressing (up to 64 GB address space), supporting unified memory architecture
- Fine-grained task switching, workload balancing and power management
- Advanced DMA driven operation for minimum host CPU interaction
- 256KB System Level Cache (SLC)
- Specialized Texture Cache Unit (TCU)
- Compressed Texture Decoding
- Lossless data compression (PVRGC) - The PowerVR's geometry compression, which is performed in the Geometry Processing phase of the 3D graphics workload.
- Dedicated RISC-V processor for firmware execution
- Separate power island for the firmware processor for low latency power domain transitions
- On-Chip Performance, Power and Statistics Registers.
- Resolution Support
  - Frame buffer max size = 8K × 8K
  - Texture max size = 8K × 8K
- Anti-aliasing
  - Maximum 4× multisampling
- Performance
  - Floating Point Operations (F32) - 64 operations per clock
  - Floating Point Operations (F16) - 128 operations per clock
  - Texture performance - 4 texels per clock (@32 BPP)
  - Pixel performance - 4 pixel(s) per clock (@32 BPP)
  - Geometry Performance - 0.25 triangles per clock
  - Max performance
    - ~50 GFLOPS, 4 GTex/s or 4 GPix/s @ 800MHz

1.4.41 Video Accelerator (CODEC)

CODEC module supports the following features:

- HEVC (H.265) Encode Main profiles @ L5.1 High-tier
- H.264 Encode BaseLine/Main/High Profiles @ L.52
- YUV420/YUV422 format
- 8b bit-depth
- I and P frame encoding
- ¼ -pel precision Mition vector
- Search Range in HEVC (+/- 128H, +/-64V)
- Search Range in H.264(+/- 64J, +/-48V)
- 4K resolution at 60 fps
- 8 channel 1080p40 Encode
- Rotation in multiples of 90 degree & vertical and horizontal Mirroring
- 4K resolution at 60fps for decode for all
- 8-channel 1080p30 decode
- 2 VBUSM initiator interfaces for data traffic (Primary and Secondary)
- 48 bit address width
- 128 bit data width
- ChannelID based on Internal Data Requestor
- Data interface at wrap level is synchronous to SoC CBA clocks
- 1 VBUSP 32-bit target interface to access config registers
- Fuse Controllevel high interrupt
- All memories are non-repair memories

1.5 Device MCU Domain

This section describes the modules integrated in the device MCU domain.

Copyright © 2026 Texas Instruments Incorporated

Note

In TI documentation, the MCU Domain may be referred to as "MCU Island", "MCU Channel", or "MCU Subsystem".

The Microcontroller Unit Domain (MCU Domain) is a “chip-in-a-chip” concept and it operates using separate clock sources and resets. This allows the MCU Domain to function continuously regardless of the state of the rest of the device, including periods in which the rest of the device is held in reset or powered down. The MCU island integrates communication peripherals such as SPI, I2C, and UART which are expected to be used for safety critical communication, so if the main SoC goes down, the MCU island can communicate this information to the rest of the system.

1.5.1 Arm Cortex-R5F Processor (R5FSS)

The ARM Core Cortex-R5F processor subsystem (R5FSS) supports the following main features:

- Armv7-R architecture
- R5FSS Memory System
  - 32KB Instruction Cache
    - 4x ways
    - SECDED ECC protected per 64 bits
  - Data Cache
    - 4x ways
    - SECDED ECC protected per 32 bits
  - 64KB tightly-coupled memory (TCM) per CPU
    - SECDED ECC protected per 32 bits
    - TCM hard error cache Implemented in CPU
    - Readable/writable from system
    - 32KB TCMA (ATCM)
    - 16KB TCMB0 (B0TCM)
    - 16KB TCMB1 (B1TCM)
- Full-precision Floating Point (VFPv3)
- 16-region Memory Protection Unit (MPU)
- 8 breakpoints, 8 watch points
- CoreSight Debug Access Port (DAP)
- CoreSight ETM-R5 interface (CTI, ETM)
- Performance Monitoring Unit (PMU)
- 32-bit to 36-bit Region-based Address Translation (RAT) on memory access initiators
- Integrated Vectored Interrupt Manager (VIM) per core with 256 Interrupt Inputs each
  - Programmable interrupt priority (4-bit)
  - Programmable interrupt enable mask
  - Software-generated interrupts
  - Synchronous clock domain crossing on all core interfaces

Note

CORE0 has 64KB of TCM.

Note

These details describe a superset of the R5FSS memory configuration. For additional details on device memory availability, please refer to the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

1.5.2 MCU General Purpose Input/Output Interface (MCU_GPIO)

Integrated in MCU domain: One General Purpose Input/Output (MCU_GPIO) module provides dedicated general-purpose pins that can be configured as either inputs or outputs. the MCU_GPIO module includes these main features:

- Support of 9 banks x 16 GPIO pins
- Support of up to 9 banks of interrupt capable GPIOs
- Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO pin
- Set/clear functionality per individual GPIO pin

1.5.3 MCU Inter-Integrated Circuit Interface (MCU_I2C)

Integrated in MCU domain:One multi-controller Inter-Integrated Circuit (MCU_I2C) interface with the following main features:

- Compliancy to the Philips I2C-bus specification version 2.1
- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- Support of multi-controller transmitter/peripheral receiver and receiver/peripheral transmitter modes
- Built-in FIFOs with programmable size of 8 to 64 bytes for buffered read or write
- 8-bit-wide data access
- Support of Auto Idle, Idle Request/Idle Acknowledge handshake, and Asynchronous Wakeup mechanisms
- Low power consumption

1.5.4 MCU Multi-channel Serial Peripheral Interface (MCU_SPI)

Integrated in MCU domain: Two Multi-channel Serial Peripheral Interface (MCU_SPI) modules where each have the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of MCSPI word lengths, ranging from 4 to 32 bits
- Up to four controller channels, or single channel in receiver mode
- Support of different controller multichannel modes
- Single interrupt line for multiple interrupt source events
- Support of start-bit write command
- Support of start-bit pause and break sequence
- Built-in FIFO available for a single channel

1.5.5 MCU Universal Asynchronous Receiver/Transmitter (MCU_UART)

Integrated in MCU domain: One configurable Universal Asynchronous Receiver/Transmitter (MCU_UART) interface with the following main features:

- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- 16C750 compatible interface
- Baud rates up to 3.6 Mbps with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

1.5.6 MCU Windowed Watchdog Timer/Real Time Interrupt

- Windowed Watchdog Timer (WWDT), implemented by using the Digital Windowed Watchdog (DWWD) function of the Real Time Interrupt (RTI) module providing timer functionality for operation systems and benchmarking code with the following main features:
  - Two independent 64 bit counter blocks
  - Four configurable compare registers for generating operating system ticks

Copyright © 2026 Texas Instruments Incorporated

- Free running counter 0 can be incremented by either the internal pre-scale counter or by an external event
- Selectable RTI clock input (derived from any of the available clock sources)
- Windowed Watchdog Timer (WWDT) feature
- Some RTI modules are pre-dedicated to specific processor cores

1.5.7 MCU TIMER

- Timer modules with support of the following main features:
  - Free running 32-bit upward counter
  - Generates a 1-ms tick with a 32.768-kHz functional clock
  - Interrupts generated on overflow, compare and capture
  - Supported modes of operation: compare and capture, auto-reload and start-stop
  - Programmable divider clock source (2n, where n = [0-8])
  - Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
  - On-the-fly read/write register (while counting) for systems operation and benchmarking code
  - Each odd numbered timer instance may be optionally cascaded with the previous even numbered timer instance to form up to a 64-bit timer

1.5.8 MCU Dual Clock Comparator

- The Dual Clock Comparator (MCU_DCC) module, used to determine the accuracy of a clock signal during the time execution of an application, each having the following main features:
  - Two independent counter blocks count clock pulses from each clock source
  - Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
  - Configurable time base for error signal
  - Error signal generation when one of the clocks is out of spec
  - Clock frequency measurement

1.5.9 Memory Cyclic Redundancy Check

- Memory Cyclic Redundancy Check module used to perform CRC to verify the integrity of a memory system. The MCRC module has the following main feature:
  - Four Channels of CRC Compression based Signature Generation
  - 8, 16, 32, or 64-bit Data Size
  - Maximum Length PSA based on 64-bit Polynomial
  - Programmable 20 bit pattern counter per channel
  - Three Operating Modes: Auto, Semi-CPU, Full CPU
  - MCRC or CPU can perform signature verification for each Channel
  - Timeout Interrupt if CRC is not performed within the time limit

1.5.10 Controller Area Network (MCAN)

Integrated in the domain: One Controller Area Network interfaces (MCAN) supporting both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications and having the following main features:

- Conforms with CAN Protocol version 2.0 part A, B and ISO 11898-1:2015
- Full CAN FD (up to 64 data bytes) support
- SAE J1939 support
- AUTOSAR support
- Up to 32 dedicated transmit buffers and 64 dedicated receive buffers
- Two configurable receive FIFOs, up to 64 elements each
- Configurable transmit FIFO, up to 32 elements
- Configurable transmit queue, up to 32 elements
- Configurable transmit event FIFO, up to 32 elements
- Up to 128 filter elements
- Maskable interrupts, two interrupt lines
- Timestamp Counter

Copyright © 2026 Texas Instruments Incorporated

1.6 Device WKUP Domain

This section describes the modules integrated in the device WKUP domain.

1.6.1 Arm Cortex-R5F Processor (R5FSS)

The ARM Core Cortex-R5F processor subsystem (R5FSS) supports the following main features:

- Armv7-R architecture
- R5FSS Memory System
  - 32KB Instruction Cache
    - 4x ways
    - SECDED ECC protected per 64 bits
  - Data Cache
    - 4x ways
    - SECDED ECC protected per 32 bits
  - 64KB tightly-coupled memory (TCM) per CPU
    - SECDED ECC protected per 32 bits
    - TCM hard error cache Implemented in CPU
    - Readable/writable from system
    - 32KB TCMA (ATCM)
    - 16KB TCMB0 (B0TCM)
    - 16KB TCMB1 (B1TCM)
- Full-precision Floating Point (VFPv3)
- 16-region Memory Protection Unit (MPU)
- 8 breakpoints, 8 watch points
- CoreSight Debug Access Port (DAP)
- CoreSight ETM-R5 interface (CTI, ETM)
- Performance Monitoring Unit (PMU)
- 32-bit to 36-bit Region-based Address Translation (RAT) on memory access initiators
- Integrated Vectored Interrupt Manager (VIM) per core with 256 Interrupt Inputs each
  - Programmable interrupt priority (4-bit)
  - Programmable interrupt enable mask
  - Software-generated interrupts
  - Synchronous clock domain crossing on all core interfaces

Note

CORE0 has 64KB of TCM.

Note

These details describe a superset of the R5FSS memory configuration. For additional details on device memory availability, please refer to the device-specific Datasheet.

1.6.2 Inter-Integrated Circuit Interface (I2C)

Multi-controller Inter-Integrated Circuit (I2C) interface module, with the following main features:

- 1x Instances with open-drain voltage buffers in compliance with the Philips I2C-bus specification version 2.1
- Support of standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Support of 7-bit and 10-bit device addressing modes
- 8-bit-wide data access
- Support of multi-controller transmitter/peripheral receiver and receiver/peripheral transmitter modes
- Built-in FIFOs with fixed size of 32 bytes.

Copyright © 2026 Texas Instruments Incorporated

1.6.3 Universal Asynchronous Receiver/Transmitter (UART)

Seven Instances of the configurable Universal Asynchronous Receiver/Transmitter (UART) interface module have the following main features:

- 16C750-compatible interface
- Support of RS-485 external transceiver auto flow control
- Dual 64-byte FIFOs – one per each received and transmitted data paths
- Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- Programmable sleep mode
- Baud-rate from 300 bits/s up to 3.6864 Mbits/s with 48 MHz functional clock
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)
- Support of IrDA 1.4 Slow Infrared (SIR), Medium Infrared (MIR), and Fast Infrared (FIR) communications
- Support of Consumer Infrared Remote control mode (CIR) with programmable data encoding

Note

Only one UART instance has support for support full modem control functions. All other UART instances will support only the TX, RX, RTS, and CTS signals.

1.6.4 MCU Error Signaling Module

- Error Signaling Module (MCU_ESM) for safety-related events and/or errors aggregation from throughout the device into one location supports the following main features:
  - Up to 1024 level or pulse error event inputs
  - Selectable low and high priority interrupt error pin prioritization of each error event
  - Single Error Pin output to signal severe device failure to outside world
  - Configurable time base for error signal
  - Triple Redundant Pulse Inputs
  - Two Priority Levels
  - Error Forcing capability
  - Internal redundant flops on safety critical fields

1.6.5 Global Time Counter

- Global Time Counter (GTC) module that can be used for time synchronization and debug trace time stamping with the following main features:
  - 64-bit up counter
  - No rollover during the lifetime of the device
  - Compatible with Armv8 system counter requirements
  - Outputs reflected binary (Gray) encoded timer value for system timer bus distribution to other modules
  - Selectable counter bit output as a push event that can be used by CPTS modules, timers or interface protocols

1.6.6 Windowed Watchdog Timer/Real Time Interrupt

- Windowed Watchdog Timer (WWDT), implemented by using the Digital Windowed Watchdog (DWWD) function of the Real Time Interrupt (RTI) module providing timer functionality for operation systems and benchmarking code with the following main features:
  - Two independent 64 bit counter blocks
  - Four configurable compare registers for generating operating system ticks
  - Free running counter 0 can be incremented by either the internal pre-scale counter or by an external event
  - Selectable RTI clock input (derived from any of the available clock sources)
  - Windowed Watchdog Timer (WWDT) feature
  - Some RTI modules are pre-dedicated to specific processor cores

1.6.7 WKUP_TIMER

- Dual mode timer module with support of the following main features:

Copyright © 2026 Texas Instruments Incorporated

- Free running 32-bit upward counter
- Generates a 1-ms tick with a 32.768-kHz functional clock
- Interrupts generated on overflow, compare and capture
- Supported modes of operation: compare and capture, auto-reload and start-stop
- Programmable divider clock source (2n, where n = [0-8])
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- On-the-fly read/write register (while counting) for systems operation and benchmarking code
- Each odd numbered timer instance may be optionally cascaded with the previous even numbered timer instance to form up to a 64-bit timer
- On-the-fly read/write register (while counting) for systems operation and benchmarking code

Copyright © 2026 Texas Instruments Incorporated

1.7 Device Identification

The manufacturer identity, the boundary scan part number, and the silicon revision of the device can be read in the WKUP_CTRL_MMR_CFG0_JTAGID register. See Table 1-1 and Table 1-2 tables for more information.

Table 1-1. Device JTAG ID

WKUP_CTRL_MMR_CFG0_JTAGID Register Field	Value	Comment
[31-28] VARIANT	See Table 1-2	Silicon Revision (SR) identifier.
[27-12] PARTNO	See Table 1-2	Part number for boundary scan.
[11-1] MFG	0x17	Manufacturer identity (TI).
[0] LSB	0x1	Always reads 1.

Table 1-2. Device JTAG ID Values

Silicon Type	VARIANT	PARTNO	WKUP_CTRL_MMR_CFG0_JTAGID Register Value
TDA4VEN SR1.0 TDA4AEN SR1.0 AM67 SR1.0	0x0	0xBBA0	0x0BBA002F

Copyright © 2026 Texas Instruments Incorporated

Chapter 2

Memory Map

The SoC level memory map is constructed using a 36b physical address and follows the guideline to put peripherals at 64KB aligned boundary. No virtual address is supported on the SoC level. However, A53 core can support virtual address internally. If software utilizes the address more than 36b, only the lower 36b is used for SoC level address decoding and the upper address bits will be ignored by the SoC. Not all the 36b memory regions are implemented. Any transactions hitting the unimplemented address range will be terminated and routed to null end point to avoid system hang. An interrupt will be asserted and the above transaction will be logged. All the SoC level peripherals and processors use the common SoC memory except the 32b only processors, such as the R5 core. For those processors and peripherals which natively only supports 32b physical address, the Region based Address Translation (RAT) module is used to remap the 32b address into the common 36b SoC address map.

2.1 Processor Memory Map View 40
2.2 DDR Memory Region 43
2.3 MMU Optimization Note 43
2.4 MAIN Memory Map 43
2.5 MCU Memory Map 61
2.6 WKUP Memory Map 63
2.7 R5FSS0 Memory Map 64
2.8 MCU_R5FSS0 Memory Map 64
2.9 WKUP_R5FSS0 Memory Map 65
2.10 DMASS0 Memory Map 65
2.11 SMS0 Memory Map 65
2.12 SA3_SS0 Memory Map 65

Copyright © 2026 Texas Instruments Incorporated

2.1 Processor Memory Map View

The device implements a 36-bit physical address space that is used by the Cortex-A53 subsystem and most other initiators in the system.

Additionally, selected initiators (R5FSS0, MCU_R5FSS0, WKUP_R5FSS0, DMASS0, SMS0, and SA3_SS0) have their own memory maps that are a mix of private endpoints and regions that map to the common 36-bit address space of the SOC.

2.1.1 A53 Memory View

All the A53 cores use the common 36b SoC memory map shown in MAIN Memory Map for its physical address map. Since A53 natively supports up to 44b physical address, if A53 issues any transaction with non-zero upper 8 physical address bits, those upper 8 address bits are ignored by SoC address decoding logic. The A53 uses an MMU to translate the virtual addresses used in operation to the physical addresses issued to the rest of the SoC, see MMU Optimization Note for further details/considerations.

2.1.2 RAT

Since the device is built on 36b physical address, there are memory regions beyond 4GB address, such as DDR data space and debug configuration space. This is not a problem for the ARMv8 core such as A53, since it supports up to 44b physical address. However, the R5 processors natively support only a 32b physical address space. In order to allow these micro controllers accessing the full 36b physical address space, a region based remapping module is dedicated for each processor.

2.1.3 R5FSS Memory View

The device has one single R5F core in the MAIN domain, one single R5F core in the MCU domain, and one single core R5F (Device Manager) in the WKUP domain. The R5F is a native 32b processor, which means the R5F core itself can only generate transactions using 32b address. A dedicated RAT module is integrated for this single core R5F to allow the R5F software to remap the 32b R5F address into the common 36b SoC address.

In addition, the R5F core has its own TCM, which the R5F core can access with its TCM with a single R5F core cycle. Those TCM memories have two sets of address, one set address is used for the R5F core for a single cycle access to those TCM. The second set of address is used by other initiators outside this R5F subsystem to access those TCM. The R5F core shall always use the first set of address to access its TCM. If the R5F core utilizes the second set of address to those TCM, the R5F core will receives an exception, since those transactions will be routed to be terminated by sending them to the null end point.

While the SoC level address for TCM is fixed, the address used by the R5F to access to its own TCM can be remapped by software. By default, the device puts the R5F's ATCM at address 0x0 and the BTCM address at 0x4101_0000.

The MCU R5F has one section of the region dedicated for peripheral access from address 0x0400_0000 to 0x07FF_FFFF. The MAIN R5F and WKUP R5F also have one section of their regions dedicated for peripheral access from address 0x2000_0000 to 0x2FFF_FFFF. These memory regions are treated as normal memory, which can't be cached. The R5F accesses this memory region through a dedicated single-issue interface, and none of the transactions accessing this memory region can be remapped by using RAT module.

The R5F core accesses its own ATCM and BTCM memory directly, and there is no address remapping for those access. All the other access than ATCM, BTCM, and address range 0x0400_0000 to 0x07FF_FFFF from the MCU R5F core, and address range 0x2000_0000 to 0x2FFF_FFFF from the MAIN R5F and WKUP R5F cores go through RAT module, which could be remapped into different address to access the rest of the SoC using the common 36b SoC memory map.

See the R5FSS0 Memory Map, MCU_R5FSS0 Memory Map, and WKUP_R5FSS0 Memory Map.

The common SoC memory map assigns the WKUP R5F's ATCM and BTCM to address 0x7800_0000 and 0x7810_0000, the MCU R5F's ATCM and BTCM to address 0x7900_0000 and 0x7902_0000, and the MAIN R5F's ATCM and BTCM to address 0x7840_0000 and 0x7850_0000. The R5F software can optionally reprogram the its internal ATCM and BTCM address to match the address assigned by the common SoC

Copyright © 2026 Texas Instruments Incorporated

memory map as well. Regardless where the R5F puts its own ATCM and BTCM, the R5F accesses its own ATCM and BTCM directly without going through RAT.

ATCM is by default disabled at address 0x0. BTCM is by default enabled and the default address for BTCM is 0x4101_0000. The base address of MCU VIM is at 0x07FF_0000 and the base address for RAT configuration is at 0x07FE_0000. The base address of MAIN and WKUP VIMs is at 0x2FFF_0000, and the base address for RAT configuration is at 0x2FFE_0000.

All of the MCU R5F transactions can go through RAT for address re-mapping function except the transactions targeted to address range 0x0400_0000 to 0x07FF_FFFF and its own ATCM and BTCM. All of the MAIN and WKUP R5F transactions can go through RAT for address re-mapping function except the transactions targeted to address range 0x2000_0000 to 0x2FFF_FFFF and their own ATCM and BTCM. It is highly recommended only remap the R5F's address range from 0x8000_0000 to 0xFFFF_FFFF to access the target region located at the common memory map between 0x8000_0000 and 0xFFFF_FFFF.

Copyright © 2026 Texas Instruments Incorporated

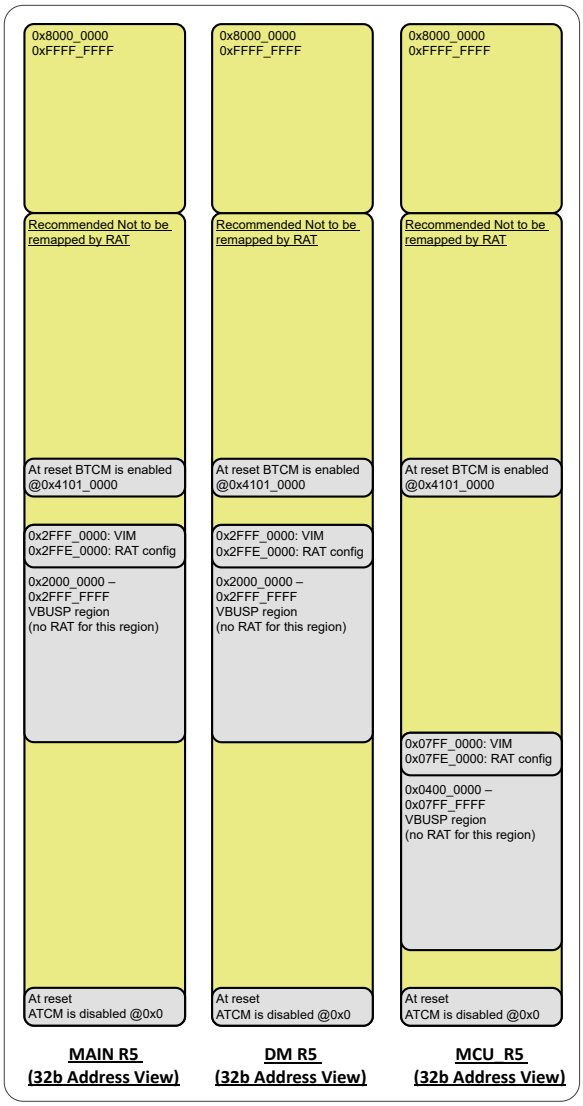


Figure 2-1. MAIN, MCU, and Device Manager R5F Memory Map View

Copyright © 2026 Texas Instruments Incorporated

2.2 DDR Memory Region

The device includes two DDR data regions: 2GB at 0x8000_0000 and 30GB at 0x8_8000_0000.

The max size of LPDDR4 is 8GB.

2.3 MMU Optimization Note

The SoC memory map is constructed to enable using MMU granule sizes of 64KB and above by placing most of the system resources on a 64KB address boundary. This also enables isolation of resources and peripherals from individual software processes through unique MMU entries.

The following is a non-exhaustive list of exceptions to this alignment rule:

- Debug SS
- MAIN GPIO0/GPIO1, MCU GPIO
- DCC
- ISC, QOS, Firewall

Resources and peripherals excluded from the 64KB addressing placement are either meant to have a single owner or are logically the same function (such as GPIO).

2.4 MAIN Memory Map

Table 2-1. MAIN Memory Map

Region Name	Start Address	End Address	Size
PSRAMECC0_RAM	0x0000000000	0x0000000400	1 KB
PADCFG_CTRL0_CFG0	0x00000F0000	0x00000F8000	32 KB
AEN_MAIN_CTRL_MMR0_CFG0	0x0000100000	0x0000120000	128 KB
AEN_MAIN_DBG_CBASS0_ERR	0x0000200000	0x0000200400	1 KB
CBASS_INFRA1_ERR	0x0000210000	0x0000210400	1 KB
CBASS_FW0_ERR	0x0000220000	0x0000220400	1 KB
CBASS_IPCSS0_ERR	0x0000230000	0x0000230400	1 KB
CBASS_MCASP0_ERR	0x0000240000	0x0000240400	1 KB
EFUSE0	0x0000300000	0x0000300100	256 B
GPU0_MEM	0x0000310000	0x0000310400	1 KB
COMPUTE_CLUSTER0_PBIST	0x0000330000	0x0000330400	1 KB
PBIST3_MEM	0x0000340000	0x0000340400	1 KB
VPAC0_MEM	0x0000350000	0x0000350400	1 KB
C7X256V0_PBIST	0x0000360000	0x0000360400	1 KB
C7X256V1_PBIST	0x0000370000	0x0000370400	1 KB
PBIST2	0x0000380000	0x0000380400	1 KB
PSCSS0	0x0000400000	0x0000401000	4 KB
PLLCTRL0	0x0000410000	0x0000410200	512 B
ESM0_CFG	0x0000420000	0x0000421000	4 KB
DFTSS0	0x0000500000	0x0000500400	1 KB
DDPA0	0x0000580000	0x0000580400	1 KB
GPIO0	0x0000600000	0x0000600100	256 B
GPIO1	0x0000601000	0x0000601100	256 B
PLL0_CFG	0x0000680000	0x00006A0000	128 KB
PSRAMECC0_ECC_AGGR	0x0000700000	0x0000700400	1 KB
PSCSS0_REGS	0x0000700400	0x0000700800	1 KB
PSRAMECC1_ECC_AGGR	0x0000701000	0x0000701400	1 KB
CPSW0_ECC	0x0000704000	0x0000704400	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
MMCSD0_ECC_AGGR_RXMEM	0x0000706000	0x0000706400	1 KB
MMCSD0_ECC_AGGR_TXMEM	0x0000707000	0x0000707400	1 KB
MMCSD1_ECC_AGGR_RXMEM	0x0000708000	0x0000708400	1 KB
MMCSD1_ECC_AGGR_TXMEM	0x0000709000	0x0000709400	1 KB
MMCSD2_ECC_AGGR_TXMEM	0x000070A000	0x000070A400	1 KB
MMCSD2_ECC_AGGR_RXMEM	0x000070B000	0x000070B400	1 KB
CSI_TX_IF0_ECC_AGGR_CFG	0x000070C000	0x000070C400	1 KB
CSI_TX_IF0_ECC_AGGR_BYTE_CFG	0x000070C400	0x000070C800	1 KB
CSI_RX_IF0_ECC_AGGR_CFG	0x000070E000	0x000070E400	1 KB
CSI_RX_IF1_ECC_AGGR_CFG	0x000070F000	0x000070F400	1 KB
MSRAM8KX256E0_ECC_AGGR_REGS	0x0000710000	0x0000710400	1 KB
SA3_SS0_ECC_AGGR	0x0000712000	0x0000712400	1 KB
CSI_RX_IF2_ECC_AGGR_CFG	0x0000714000	0x0000714400	1 KB
CSI_RX_IF3_ECC_AGGR_CFG	0x0000715000	0x0000715400	1 KB
FSS0_OSPI0_ECC_AGGR	0x0000716000	0x0000716400	1 KB
COMPUTE_CLUSTER0_SS_ECC_AGGR	0x0000718000	0x0000718400	1 KB
COMPUTE_CLUSTER0_CORE0_ECC_AGGR	0x0000718400	0x0000718800	1 KB
COMPUTE_CLUSTER0_CORE1_ECC_AGGR	0x0000718800	0x0000718C00	1 KB
COMPUTE_CLUSTER0_CORE2_ECC_AGGR	0x0000718C00	0x0000719000	1 KB
COMPUTE_CLUSTER0_CORE3_ECC_AGGR	0x0000719000	0x0000719400	1 KB
C7X256V0_ECC_AGGR	0x000071A000	0x000071A400	1 KB
C7X256V1_ECC_AGGR	0x000071B000	0x000071B400	1 KB
PCIE0_CORE_ECC_AGGR0	0x000071E000	0x000071E400	1 KB
PCIE0_CORE_ECC_AGGR1	0x000071F000	0x000071F400	1 KB
DCC0	0x0000800000	0x0000800040	64 B
DCC1	0x0000804000	0x0000804040	64 B
DCC2	0x0000808000	0x0000808040	64 B
DCC3	0x000080C000	0x000080C040	64 B
DCC4	0x0000810000	0x0000810040	64 B
DCC5	0x0000814000	0x0000814040	64 B
DCC6	0x0000818000	0x0000818040	64 B
DCC7	0x000081C000	0x000081C040	64 B
DCC8	0x0000820000	0x0000820040	64 B
PSRAMECC1_RAM	0x0000900000	0x0000900400	1 KB
MAIN_GPIOMUX_INTROUTER0_INTR_ROUTER_CFG	0x0000A00000	0x0000A00800	2 KB
TIMESYNC_EVENT_INTROUTER0_INTR_ROUTER_CFG	0x0000A40000	0x0000A40400	1 KB
PDMA0	0x0000C00000	0x0000C00400	1 KB
PDMA1	0x0000C01000	0x0000C01400	1 KB
GICSS0_GIC_TRANSLATER	0x0001000000	0x0001400000	4 MB
GICSS0_GIC	0x0001800000	0x0001900000	1 MB
TIMER0_CFG	0x0002400000	0x0002400400	1 KB
TIMER1_CFG	0x0002410000	0x0002410400	1 KB
TIMER2_CFG	0x0002420000	0x0002420400	1 KB
TIMER3_CFG	0x0002430000	0x0002430400	1 KB
TIMER4_CFG	0x0002440000	0x0002440400	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
TIMER5_CFG	0x0002450000	0x0002450400	1 KB
TIMER6_CFG	0x0002460000	0x0002460400	1 KB
TIMER7_CFG	0x0002470000	0x0002470400	1 KB
UART0	0x0002800000	0x0002800200	512 B
UART1	0x0002810000	0x0002810200	512 B
UART2	0x0002820000	0x0002820200	512 B
UART3	0x0002830000	0x0002830200	512 B
UART4	0x0002840000	0x0002840200	512 B
UART5	0x0002850000	0x0002850200	512 B
UART6	0x0002860000	0x0002860200	512 B
MCASP0_CFG	0x0002B00000	0x0002B02000	8 KB
MCASP0_DMA	0x0002B08000	0x0002B08400	1 KB
MCASP1_CFG	0x0002B10000	0x0002B12000	8 KB
MCASP1_DMA	0x0002B18000	0x0002B18400	1 KB
MCASP2_CFG	0x0002B20000	0x0002B22000	8 KB
MCASP2_DMA	0x0002B28000	0x0002B28400	1 KB
CPSW0_NUSS	0x0008000000	0x0008200000	2 MB
PCIE0_CORE_DBN_CFG_PCIE_CORE	0x000D000000	0x000D800000	8 MB
RTI0_CFG	0x000E000000	0x000E000100	256 B
RTI1_CFG	0x000E010000	0x000E010100	256 B
RTI2_CFG	0x000E020000	0x000E020100	256 B
RTI3_CFG	0x000E030000	0x000E030100	256 B
RTI4_CFG	0x000E040000	0x000E040100	256 B
RTI5_CFG	0x000E050000	0x000E050100	256 B
RTI8_CFG	0x000E0A0000	0x000E0A0100	256 B
RTI15_CFG	0x000E0F0000	0x000E0F0100	256 B
SERDES_10G0	0x000F000000	0x000F010000	64 KB
SERDES_10G1	0x000F010000	0x000F020000	64 KB
USB0_DEBUG_TRACE_MMR_TRACE_VBUSP_USB2SS_DEBUG_TRACE	0x000F080000	0x000F080200	512 B
PCIE0_CORE_USER_CFG_USER_CFG	0x000F100000	0x000F100400	1 KB
PCIE0_CORE_VMAP_OB_MMRS	0x000F101000	0x000F102000	4 KB
PCIE0_CORE_PCIE_INTD_CFG_INTD_CFG	0x000F102000	0x000F103000	4 KB
PCIE0_CORE_CPTS_CFG_CPTS_VBUSP	0x000F103000	0x000F103400	1 KB
DDR32SS0_REGS_SS_CFG_SSCFG	0x000F300000	0x000F300200	512 B
DDR32SS0_CTLPHY_WRAP_CTL_CFG_CTLCFG	0x000F308000	0x000F310000	32 KB
USB0_MMR_MMRVBP_USB2SS_CFG	0x000F900000	0x000F900800	2 KB
USB0_PHY2	0x000F908000	0x000F908400	1 KB
USB1_MMR_MMRVBP_USBSS_CMN	0x000F920000	0x000F920100	256 B
USB1_RAMS_INJ_CFG	0x000F921000	0x000F921400	1 KB
USB1_PHY2	0x000F928000	0x000F928400	1 KB
USB0_ECC_AGGR	0x000F980000	0x000F980400	1 KB
USB1_ECC_AGGR	0x000F990000	0x000F990400	1 KB
MMCSD1_CTL_CFG	0x000FA00000	0x000FA01000	4 KB
MMCSD1_SS_CFG	0x000FA08000	0x000FA08400	1 KB
MMCSD0_CTL_CFG	0x000FA10000	0x000FA11000	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
MMCSD0_SS_CFG	0x000FA18000	0x000FA18400	1 KB
MMCSD2_CTL_CFG	0x000FA20000	0x000FA21000	4 KB
MMCSD2_SS_CFG	0x000FA28000	0x000FA28400	1 KB
FSS0_CFG	0x000FC00000	0x000FC00100	256 B
FSS0_FSAS_CFG	0x000FC10000	0x000FC10100	256 B
FSS0_OTFA_CFG	0x000FC20000	0x000FC21000	4 KB
FSS0_OSPI0_CTRL	0x000FC40000	0x000FC40100	256 B
FSS0_OSPI0_SS_CFG	0x000FC44000	0x000FC44200	512 B
JPGENC0_CORE	0x000FD20000	0x000FD20100	256 B
JPGENC0_CORE_MMU	0x000FD20200	0x000FD20400	512 B
GPU0_CORE_MMRS	0x000FD80000	0x000FE00000	512 KB
MCASP3_CFG	0x000FE00000	0x000FE02000	8 KB
MCASP3_DMA	0x000FE08000	0x000FE08400	1 KB
MCASP4_CFG	0x000FE10000	0x000FE12000	8 KB
MCASP4_DMA	0x000FE18000	0x000FE18400	1 KB
I2C4_CFG	0x000FE80000	0x000FE80100	256 B
ATL0_REG	0x000FEE0000	0x000FEE0400	1 KB
CBASS_AUDIO0_ERR	0x000FEF0000	0x000FEF0400	1 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DMPAC_REGS_DMPAC_REGS_CFG_IP_MMRS	0x0010000000	0x0010000400	1 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_CP_INTD_CFG_INTD_CFG	0x0010001000	0x0010002000	4 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_HTS_S_VBUSP	0x0010008000	0x0010010000	32 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_CTSET2_WRAP_CFG_CTSET2_CFG	0x0010020000	0x0010022000	8 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DMPAC_FOCO_0_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x0010024000	0x0010024040	64 B
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DMPAC_FOCO_0_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x0010024200	0x0010024400	512 B
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DMPAC_FOCO_1_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x0010028000	0x0010028040	64 B
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DMPAC_FOCO_1_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x0010028200	0x0010028400	512 B
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_PAR_DOF_CFG_VP_MMR_VBUSP_DOFCORE	0x0010080000	0x0010081000	4 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_PAR_DOF_CFG_VP_MEM_MMRRAM_VBUSP_MMR_RAM	0x00100C0000	0x0010100000	256 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_PAR_SDE_S_VBUSP_MMR_VBUSP_MMR	0x0010100000	0x0010101000	4 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_PAR_SDE_S_VBUSP_MEM_MMRRAM_VBUSP_MMR_RAM	0x0010140000	0x0010180000	256 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU	0x0010200000	0x0010204000	16 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x0010204000	0x0010208000	16 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x0010208000	0x0010210000	32 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x0010240000	0x0010260000	128 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x0010260000	0x0010280000	128 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x0010280000	0x00102A0000	128 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHCORE	0x00102A0000	0x00102C0000	128 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x00102E0000	0x0010300000	128 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_DOF_INFRA_DMPAC_BASE_MEM_SLV_CBASS_STRIPE_MSRAM_SLV	0x0010400000	0x0010480000	512 KB
SAM67_DMPAC_WRAP0_MEM	0x0010800000	0x0010800400	1 KB
SAM67_DMPAC_WRAP0_DMPAC_TOP_CFG_SLV_KSDW_ECC_AGGR_CFG	0x0010801000	0x0010801400	1 KB
I2C0_CFG	0x0020000000	0x0020000100	256 B
I2C1_CFG	0x0020010000	0x0020010100	256 B
I2C2_CFG	0x0020020000	0x0020020100	256 B
I2C3_CFG	0x0020030000	0x0020030100	256 B
MCSPI0_CFG	0x0020100000	0x0020100400	1 KB
MCSPI1_CFG	0x0020110000	0x0020110400	1 KB
MCSPI2_CFG	0x0020120000	0x0020120400	1 KB
CBASS_MISC_PERI0_ERR	0x00201F0000	0x00201F0400	1 KB
MCAN0_SS	0x0020700000	0x0020700100	256 B
MCAN0_CFG	0x0020701000	0x0020701200	512 B
MCAN0_MSGMEM_RAM	0x0020708000	0x0020710000	32 KB
MCAN1_SS	0x0020710000	0x0020710100	256 B
MCAN1_CFG	0x0020711000	0x0020711200	512 B
MCAN1_MSGMEM_RAM	0x0020718000	0x0020720000	32 KB
EPWM0_EPWM	0x0023000000	0x0023000100	256 B
EPWM1_EPWM	0x0023010000	0x0023010100	256 B
EPWM2_EPWM	0x0023020000	0x0023020100	256 B
ECAP0_CTL_STS	0x0023100000	0x0023100100	256 B
ECAP1_CTL_STS	0x0023110000	0x0023110100	256 B
ECAP2_CTL_STS	0x0023120000	0x0023120100	256 B
EQEP0_REG	0x0023200000	0x0023200100	256 B
EQEP1_REG	0x0023210000	0x0023210100	256 B
EQEP2_REG	0x0023220000	0x0023220100	256 B
MCAN0_ECC_AGGR	0x0024018000	0x0024018400	1 KB
MCAN1_ECC_AGGR	0x0024019000	0x0024019400	1 KB
ELM0	0x0025010000	0x0025011000	4 KB
MAILBOX0_REGS0	0x0029000000	0x0029000200	512 B
MAILBOX0_REGS1	0x0029010000	0x0029010200	512 B
MAILBOX0_REGS2	0x0029020000	0x0029020200	512 B
MAILBOX0_REGS3	0x0029030000	0x0029030200	512 B
MAILBOX0_REGS4	0x0029040000	0x0029040200	512 B
MAILBOX0_REGS5	0x0029050000	0x0029050200	512 B
MAILBOX0_REGS6	0x0029060000	0x0029060200	512 B
MAILBOX0_REGS7	0x0029070000	0x0029070200	512 B

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
SPINLOCK0	0x002A000000	0x002A008000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_KSDW_ECC_AGGR_CFG	0x002B604000	0x002B604400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x002B605000	0x002B605400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x002B607000	0x002B607400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_VPAC_REGS_VPAC_REGS_CFG_IP_MMRS	0x002C000000	0x002C000400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_CTSET2_WRAP_CFG_CTSET2_CFG	0x002C002000	0x002C004000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_CP_INTD_CFG_INTD_CFG	0x002C004000	0x002C005000	4 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_HTS_S_VBUSP	0x002C010000	0x002C020000	64 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MMR_VBUSP	0x002C020000	0x002C020400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_VPAC_LDC_LSE_CFG_VP	0x002C020400	0x002C020600	512 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALY_LUTCFG_DUALY_LUT	0x002C020800	0x002C021000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALC_LUTCFG_DUALC_LUT	0x002C021000	0x002C021800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_MESH_VBUSPI_MESH_MEM	0x002C022000	0x002C024000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_Y_VBUSPI_Y_MEM	0x002C028000	0x002C030000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_CBCR_VBUSPI_CBCR_MEM	0x002C030000	0x002C038000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_MSC_CFG_VP_CFG_VP	0x002C0C0000	0x002C0C0800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_MSC_CFG_VP_LSE_CFG_VP	0x002C0C0800	0x002C0C0A00	512 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_MMR_CFG_VISS_TOP	0x002C100000	0x002C100200	512 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VPAC_VISS_LSE_CFG_VP	0x002C100400	0x002C100600	512 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_CFG_GLBCE	0x002C103800	0x002C104000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_STATMEM_CFG_GLBCE_STATMEM	0x002C104000	0x002C108000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_CFA_VBUSP_FLEXCFA	0x002C108000	0x002C110000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC	0x002C110000	0x002C110800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC1	0x002C110800	0x002C111000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC2	0x002C111000	0x002C111800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC3	0x002C111800	0x002C112000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_Y8R8	0x002C112000	0x002C112800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_C8G8	0x002C112800	0x002C113000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_S8B8	0x002C113000	0x002C113800	2 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_HIST	0x002C113800	0x002C114000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_LINE	0x002C118000	0x002C118800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_MMR_S_VBUSP_RAWFE_CFG	0x002C120000	0x002C120400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_CFG_RAWFE_H3A_CFG	0x002C120400	0x002C120500	256 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT3_RAM_RAWFE_PWL_LUT3_RAM	0x002C120800	0x002C121000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT2_RAM_RAWFE_PWL_LUT2_RAM	0x002C121000	0x002C121800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT1_RAM_RAWFE_PWL_LUT1_RAM	0x002C121800	0x002C122000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_WDR_LUT_RAM_RAWFE_WDR_LUT_RAM	0x002C122000	0x002C122800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_LUT_RAM_RAWFE_H3A_LUT_RAM	0x002C122800	0x002C123000	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_RAM_RAWFE_DPC_LUT_RAM	0x002C123000	0x002C123400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_LRAM_RAWFE_DPC_LRAM	0x002C124000	0x002C126000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LSC_RAM_RAWFE_LSC_LUT_RAM	0x002C128000	0x002C130000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_ARAM_RAWFE_H3A_ARAM	0x002C130000	0x002C132000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_LRAM_RAWFE_H3A_LRAM	0x002C132000	0x002C134000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_STATRAM_RAWFE_DPC_STATRAM	0x002C136000	0x002C138000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MMR_VBUSP_NSF4VCORE	0x002C140000	0x002C140800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_RAWHIST_HISTDATA_VBUSP_RAWHIST	0x002C140800	0x002C140A00	512 B
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_RAWHIST_HISTLUT_VBUSP_RAWHIST_LUT	0x002C141000	0x002C141800	2 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MEM_MMRRAM_VBUSP_MMR_RAM	0x002C144000	0x002C148000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_EE_VBUSP_FLEXEE	0x002C150000	0x002C158000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_CFA_VBUSP_FLEXCFA_DLUTS	0x002C158000	0x002C15C000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_MMRCFG_CAC	0x002C180000	0x002C180400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_CORE_LUT_CFG_LUT_MEM	0x002C182000	0x002C184000	8 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_LINEMEM_CFG_LINE_MEM	0x002C184000	0x002C188000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_MMRCFG_PCID	0x002C188000	0x002C188400	1 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_IR_REMAPLUT_LUT_CFG_IRREMAP_LUT	0x002C188800	0x002C189000	2 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
VPAC0_IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_CFG_LINEMEM_CFG_LINE_MEM	0x002C18C000	0x002C190000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU	0x002C200000	0x002C204000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x002C204000	0x002C208000	16 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x002C208000	0x002C210000	32 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x002C240000	0x002C260000	128 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x002C260000	0x002C280000	128 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x002C280000	0x002C2A0000	128 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHCORE	0x002C2A0000	0x002C2C0000	128 KB
VPAC0_IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x002C2E0000	0x002C300000	128 KB
CSI_RX_IF0_CP_INTD_CFG_INTD_CFG	0x0030100000	0x0030101000	4 KB
CSI_RX_IF0_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0030101000	0x0030102000	4 KB
CSI_RX_IF0_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0030102000	0x0030103000	4 KB
DPHY_RX0_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0030110000	0x0030111000	4 KB
DPHY_RX0_MMR_SLV_K3_DPHY_WRAP	0x0030111000	0x0030111100	256 B
CSI_RX_IF1_CP_INTD_CFG_INTD_CFG	0x0030120000	0x0030121000	4 KB
CSI_RX_IF1_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0030121000	0x0030122000	4 KB
CSI_RX_IF1_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0030122000	0x0030123000	4 KB
DPHY_RX1_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0030130000	0x0030131000	4 KB
DPHY_RX1_MMR_SLV_K3_DPHY_WRAP	0x0030131000	0x0030131100	256 B
CSI_RX_IF2_CP_INTD_CFG_INTD_CFG	0x0030140000	0x0030141000	4 KB
CSI_RX_IF2_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0030141000	0x0030142000	4 KB
CSI_RX_IF2_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0030142000	0x0030143000	4 KB
DPHY_RX2_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0030150000	0x0030151000	4 KB
DPHY_RX2_MMR_SLV_K3_DPHY_WRAP	0x0030151000	0x0030151100	256 B
CSI_RX_IF3_CP_INTD_CFG_INTD_CFG	0x0030160000	0x0030161000	4 KB
CSI_RX_IF3_VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x0030161000	0x0030162000	4 KB
CSI_RX_IF3_RX_SHIM_VBUSP_MMR_CSI2RXIF	0x0030162000	0x0030163000	4 KB
DPHY_RX3_VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x0030170000	0x0030171000	4 KB
DPHY_RX3_MMR_SLV_K3_DPHY_WRAP	0x0030171000	0x0030171100	256 B
CSI_TX_IF0_CP_INTD_CFG_INTD_CFG	0x0030180000	0x0030181000	4 KB
CSI_TX_IF0_VBUS2APB_WRAP_VBUSP_APB_CSI2TX_V2	0x0030181000	0x0030182000	4 KB
CSI_TX_IF0_TX_SHIM_VBUSP_MMR_CSI2TXIF_V2	0x0030182000	0x0030183000	4 KB
DPHY_TX0	0x00301C0000	0x00301C1000	4 KB
DSS0_COMMON	0x0030200000	0x0030201000	4 KB
DSS0_COMMON1	0x0030201000	0x0030202000	4 KB
DSS0_VIDL1	0x0030202000	0x0030203000	4 KB
DSS0_VID	0x0030206000	0x0030207000	4 KB
DSS0_OVR1	0x0030207000	0x0030208000	4 KB
DSS0_OVR2	0x0030208000	0x0030209000	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
DSS0_VP1	0x003020A000	0x003020B000	4 KB
DSS0_VP2	0x003020B000	0x003020C000	4 KB
CODEC0_VPU	0x0030210000	0x0030220000	64 KB
DSS1_COMMON	0x0030220000	0x0030221000	4 KB
DSS1_COMMON1	0x0030221000	0x0030222000	4 KB
DSS1_VIDL1	0x0030222000	0x0030223000	4 KB
DSS1_VID	0x0030226000	0x0030227000	4 KB
DSS1_OVR1	0x0030227000	0x0030228000	4 KB
DSS1_OVR2	0x0030228000	0x0030229000	4 KB
DSS1_VP1	0x003022A000	0x003022B000	4 KB
DSS1_VP2	0x003022B000	0x003022C000	4 KB
C7X256V0	0x0030240000	0x0030250000	64 KB
C7X256V1	0x0030250000	0x0030260000	64 KB
DSS_DSI0_DSI_WRAP_MMR_VBUSP_CFG_DSI_WRAP	0x0030270000	0x0030270100	256 B
DSS_DSI0_DSI_TOP_ECC_AGGR_SYS_CFG	0x0030271000	0x0030271400	1 KB
MCRC64_0_REGS	0x0030300000	0x0030301000	4 KB
VPAC_RSWS_BW_LIMITER7_REGS	0x0030400000	0x0030401000	4 KB
CODEC_WS_BW_LIMITER3_REGS	0x0030401000	0x0030402000	4 KB
A53_WS_BW_LIMITER1_REGS	0x0030402000	0x0030403000	4 KB
A53_RS_BW_LIMITER0_REGS	0x0030403000	0x0030404000	4 KB
JPGENC_RS_BW_LIMITER4_REGS	0x0030404000	0x0030405000	4 KB
JPGENC_WS_BW_LIMITER5_REGS	0x0030405000	0x0030406000	4 KB
C7XV_RSWS_BS_LIMITER6_REGS	0x0030406000	0x0030407000	4 KB
VPAC_RSWS_BW_LIMITER8_REGS	0x0030407000	0x0030408000	4 KB
CODEC_RS_BW_LIMITER2_REGS	0x0030408000	0x0030409000	4 KB
GPU_WS_BW_LIMITER10_REGS	0x0030409000	0x003040A000	4 KB
GPU_RS_BW_LIMITER9_REGS	0x003040A000	0x003040B000	4 KB
C7XV_RSWS_BS_LIMITER11_REGS	0x003040B000	0x003040C000	4 KB
DSS_DSI0_DSI_TOP_VBUSP_CFG_DSI_0_DSI	0x0030500000	0x0030600000	1 MB
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_CAP	0x0031000000	0x0031000020	32 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_OPER	0x0031000020	0x0031000060	64 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_PORT	0x0031000420	0x0031000440	32 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_RUNTIME	0x0031000440	0x0031000460	32 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_INTR	0x0031000460	0x00310004A0	64 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DB	0x0031000560	0x0031000760	512 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_EXTCAP	0x0031000960	0x0031000970	16 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_SUPPRTCAP2	0x0031000970	0x0031000980	16 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_SUPPRTCAP3	0x0031000980	0x00310009A0	32 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_GBL	0x003100C100	0x003100C900	2 KB
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEV	0x003100C700	0x003100CF00	2 KB
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_LINK	0x003100D000	0x003100D080	128 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEBUG	0x003100D800	0x003100DA00	512 B
USB0_VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEBUG_RAM0	0x0031040000	0x0031050000	64 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
USB1_VBP2APB_WRAP_CONTROLLER_VBP_CORE_ADDR_MAP	0x0031200000	0x0031240000	256 KB
CBASS0_ERR	0x003A000000	0x003A000400	1 KB
CBASS_RT_CFG0_ERR	0x003A010000	0x003A010400	1 KB
CBASS_RT_DATA0_ERR	0x003A020000	0x003A020400	1 KB
GPMC0_CFG	0x003B000000	0x003B000400	1 KB
R5FSS0_EVNT_BUS_VBUSP_MMRS	0x003C038000	0x003C038100	256 B
R5FSS0_CORE0_ECC_AGGR	0x003F000000	0x003F000400	1 KB
GICSS0_REGS	0x003F004000	0x003F004400	1 KB
DMASS0_ECCAGGR	0x003F005000	0x003F005400	1 KB
DMASS1_ECCAGGR	0x003F006000	0x003F006400	1 KB
ECC_AGGR0_ECC_AGGR	0x003F00F000	0x003F00F400	1 KB
CBASS_CENTRAL2_ERR	0x003F012000	0x003F012400	1 KB
PBIST0	0x003F110000	0x003F110400	1 KB
PBIST1	0x003F120000	0x003F120400	1 KB
SA3_SS0_REGS	0x0040900000	0x0040901000	4 KB
SA3_SS0_MMRA	0x0040901000	0x0040901200	512 B
SA3_SS0_EIP_76	0x0040910000	0x0040910080	128 B
SA3_SS0_EIP_29T2	0x0040920000	0x0040930000	64 KB
DEBUGSS0_SYS	0x0041000000	0x0041001000	4 KB
STM0_STIMULUS	0x0042000000	0x0043000000	16 MB
SA3_SS0_SEC_PROXY_SRC_TARGET_DATA	0x0043600000	0x0043610000	64 KB
SMS0_ECC_AGGR	0x0043700000	0x0043700400	1 KB
SMS0_HSM_ECC	0x0043701000	0x0043701400	1 KB
SA3_SS0_ECCAGGR_CFG	0x0043702000	0x0043702400	1 KB
SMS0_TIFS_DMSS_HSM_ECC	0x0043702000	0x0043702400	1 KB
SMS0_HSM_WDT_RTI	0x0043935000	0x0043935100	256 B
SMS0_HSM_CTRL_MMR	0x0043936000	0x0043937000	4 KB
SMS0_HSM_RAT_MMRS	0x0043A00000	0x0043A01000	4 KB
SMS0_HSM_SRAM0_0	0x0043C00000	0x0043C20000	128 KB
SMS0_HSM_SRAM0_1	0x0043C20000	0x0043C30000	64 KB
SMS0_HSM_SRAM1	0x0043C30000	0x0043C40000	64 KB
MSRAM8KX256E0_RAM	0x0043C40000	0x0043C80000	256 KB
SMS0_TIFS_SRAM0	0x0044040000	0x0044060000	128 KB
SMS0_TIFS_SRAM1_0	0x0044060000	0x0044068000	32 KB
SMS0_TIFS_SRAM1_1	0x0044068000	0x004406C000	16 KB
SMS0_PWR	0x0044130000	0x0044130800	2 KB
SMS0_DMTIMER0	0x0044133000	0x0044133400	1 KB
SMS0_DMTIMER1	0x0044134000	0x0044134400	1 KB
SMS0_WDT_RTI	0x0044135000	0x0044135100	256 B
SMS0_RTI	0x0044135100	0x0044135200	256 B
SMS0_RAT	0x0044200000	0x0044201000	4 KB
SMS0_SEC	0x0044230000	0x0044231000	4 KB
SMS0_SECMGR	0x0044234000	0x0044238000	16 KB
SMS0_DMTIMER2	0x0044238000	0x0044238400	1 KB
SMS0_DMTIMER3	0x0044239000	0x0044239400	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
SMS0_AES	0x004423C000	0x004423E000	8 KB
SMS0_TIFS_DMSS_HSM	0x0044800000	0x0045000000	8 MB
SA3_SS0_PSILCFG_CFG_PROXY	0x0044801000	0x0044801200	512 B
SA3_SS0_PSILSS_CFG_MMRS	0x0044802000	0x0044803000	4 KB
SA3_SS0_IPCSS_SEC_PROXY_CFG_MMRS	0x0044804000	0x0044804100	256 B
SA3_SS0_IPCSS_RINGACC_CFG_GCFG	0x0044805000	0x0044805400	1 KB
SA3_SS0_INTAGGR_CFG	0x0044808000	0x0044808020	32 B
SA3_SS0_INTAGGR_CFG_IMAP	0x0044809000	0x0044809400	1 KB
SA3_SS0_INTAGGR_CFG_MCAST	0x004480A000	0x004480A400	1 KB
SA3_SS0_INTAGGR_CFG_GCNTCFG	0x004480B000	0x004480B400	1 KB
SA3_SS0_INTAGGR_CFG_INTR	0x0044810000	0x0044818000	32 KB
SA3_SS0_INTAGGR_CFG_GCNTRTI	0x0044820000	0x0044840000	128 KB
SA3_SS0_INTAGGR_CFG_UNMAP	0x0044840000	0x0044850000	64 KB
SA3_SS0_IPCSS_SEC_PROXY_CFG_SCFG	0x0044860000	0x0044880000	128 KB
SA3_SS0_IPCSS_SEC_PROXY_CFG_RT	0x0044880000	0x00448A0000	128 KB
SA3_SS0_IPCSS_RINGACC_CFG	0x00448C0000	0x0044900000	256 KB
SA3_SS0_PKTDMA_CFG_GCFG	0x0044910000	0x0044910100	256 B
SA3_SS0_PKTDMA_CFG_RFLOW	0x0044911000	0x0044911400	1 KB
SA3_SS0_PKTDMA_CFG_RCHAN	0x0044912000	0x0044912400	1 KB
SA3_SS0_PKTDMA_CFG_TCHAN	0x0044913000	0x0044913200	512 B
SA3_SS0_PKTDMA_CFG_RCHANRT	0x0044914000	0x0044918000	16 KB
SA3_SS0_PKTDMA_CFG_TCHANRT	0x0044918000	0x004491A000	8 KB
SA3_SS0_PKTDMA_CFG_RING	0x004491A000	0x004491C000	8 KB
SA3_SS0_PKTDMA_CFG_RINGRT	0x0044940000	0x0044980000	256 KB
SA3_SS0_IPCSS_RINGACC_CFG_RT	0x0044C00000	0x0045000000	4 MB
CBASS0_FW	0x0045000000	0x0045008000	32 KB
SMS0_FW	0x0045000000	0x0046000000	16 MB
CBASS_CENTRAL2_FW	0x0045010000	0x0045011000	4 KB
PSCSS0_FW	0x0045020000	0x0045020400	1 KB
CBASS_IPCSS0_FW	0x0045028000	0x0045028800	2 KB
SMS0_CBASS_FW	0x0045080000	0x00450A0000	128 KB
SMS0_HSM_CBASS_FW	0x00450A0000	0x00450B0000	64 KB
SMS0_CBASS_ISC	0x0045808000	0x0045809000	4 KB
SMS0_HSM_CBASS_ISC	0x004580A000	0x004580B000	4 KB
SMS0_DMSS_HSM_FWMGR_CFG	0x004580B000	0x004580B400	1 KB
DMASS0_PKTDMA_CRED	0x0045810000	0x0045811000	4 KB
DMASS0_BCDMA_CRED	0x0045812000	0x0045812800	2 KB
DMASS1_BCDMA_CRED	0x0045813000	0x0045813400	1 KB
CBASS0_ISC	0x0045820000	0x0045830000	64 KB
CBASS_RT_DATA0_ISC	0x0045830000	0x0045831000	4 KB
CBASS_RT_CFG0_ISC	0x0045834000	0x0045836000	8 KB
MAIN_SEC_MMR0_CFG2	0x0045900000	0x0045920000	128 KB
MAIN_SEC_MMR0_CFG0	0x0045A00000	0x0045A20000	128 KB
SMS0_CBASS_GLB	0x0045B00000	0x0045B00400	1 KB
SMS0_HSM_CBASS_GLB	0x0045B00800	0x0045B00C00	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
CBASS_IPCSS0_GLB	0x0045B01000	0x0045B01400	1 KB
CBASS_CENTRAL2_GLB	0x0045B04000	0x0045B04400	1 KB
CBASS_RT_CFG0_GLB	0x0045B06000	0x0045B06400	1 KB
CBASS_RT_DATA0_GLB	0x0045B07000	0x0045B07400	1 KB
CBASS0_GLB	0x0045B08000	0x0045B08400	1 KB
PSCSS0_GLB	0x0045B09000	0x0045B09400	1 KB
CBASS0_QOS	0x0045D20000	0x0045D30000	64 KB
CBASS_RT_DATA0_QOS	0x0045D30000	0x0045D30800	2 KB
CBASS_RT_CFG0_QOS	0x0045D34000	0x0045D36000	8 KB
DMASS0_INTAGGR_INTR	0x0048000000	0x0048100000	1 MB
DMASS0_INTAGGR_IMAP	0x0048100000	0x0048104000	16 KB
DMASS0_INTAGGR_CFG	0x0048110000	0x0048110020	32 B
DMASS0_INTAGGR_L2G	0x0048120000	0x0048120400	1 KB
DMASS0_PSILCFG_PROXY	0x0048130000	0x0048130200	512 B
DMASS0_PSILSS_MMRS	0x0048140000	0x0048141000	4 KB
DMASS0_INTAGGR_UNMAP	0x0048180000	0x00481A0000	128 KB
DMASS0_INTAGGR_MCAST	0x0048210000	0x0048211000	4 KB
DMASS0_INTAGGR_GCNTCFG	0x0048220000	0x0048222000	8 KB
DMASS0_RINGACC_GCFG	0x0048240000	0x0048240400	1 KB
DMASS0_SEC_PROXY_MMRS	0x0048250000	0x0048250100	256 B
DMASS0_BCDMA_BCHAN	0x0048420000	0x0048422000	8 KB
DMASS0_PKTDMA_RFLOW	0x0048430000	0x0048431000	4 KB
DMASS0_PKTDMA_TCHAN	0x00484A0000	0x00484A2000	8 KB
DMASS0_BCDMA_TCHAN	0x00484A4000	0x00484A6000	8 KB
DMASS0_PKTDMA_RCHAN	0x00484C0000	0x00484C2000	8 KB
DMASS0_BCDMA_RCHAN	0x00484C2000	0x00484C4000	8 KB
DMASS0_PKTDMA_GCFG	0x00485C0000	0x00485C0100	256 B
DMASS0_BCDMA_GCFG	0x00485C0100	0x00485C0200	256 B
DMASS0_PKTDMA_RING	0x00485E0000	0x00485F0000	64 KB
DMASS0_BCDMA_RING	0x0048600000	0x0048608000	32 KB
DMASS0_RINGACC_RT	0x0049000000	0x0049400000	4 MB
DMASS0_RINGACC_CFG	0x0049800000	0x0049840000	256 KB
DMASS0_INTAGGR_GCNTRTI	0x004A000000	0x004A100000	1 MB
DMASS0_SEC_PROXY_SCFG	0x004A400000	0x004A480000	512 KB
DMASS0_SEC_PROXY_RT	0x004A600000	0x004A680000	512 KB
DMASS0_PKTDMA_RCHANRT	0x004A800000	0x004A820000	128 KB
DMASS0_BCDMA_RCHANRT	0x004A820000	0x004A840000	128 KB
DMASS0_PKTDMA_TCHANRT	0x004AA00000	0x004AA20000	128 KB
DMASS0_BCDMA_TCHANRT	0x004AA40000	0x004AA60000	128 KB
DMASS0_PKTDMA_RINGRT	0x004B800000	0x004BA00000	2 MB
DMASS0_BCDMA_RINGRT	0x004BC00000	0x004BD00000	1 MB
DMASS0_BCDMA_BCHANRT	0x004C000000	0x004C020000	128 KB
DMASS0_SEC_PROXY_SRC_TARGET_DATA	0x004D000000	0x004D080000	512 KB
DMASS1_INTAGGR_GCNTRTI	0x004E000000	0x004E020000	128 KB
DMASS1_INTAGGR_UNMAP	0x004E040000	0x004E050000	64 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
DMASS1_INTAGGR_MCAST	0x004E080000	0x004E080400	1 KB
DMASS1_INTAGGR_GCNTCFG	0x004E090000	0x004E090400	1 KB
DMASS1_INTAGGR_IMAP	0x004E0B0000	0x004E0B0800	2 KB
DMASS1_INTAGGR_CFG	0x004E0C0000	0x004E0C0020	32 B
DMASS1_BCDMA_RINGRT	0x004E100000	0x004E180000	512 KB
DMASS1_BCDMA_RCHANRT	0x004E180000	0x004E1A0000	128 KB
DMASS1_BCDMA_RCHAN	0x004E200000	0x004E202000	8 KB
DMASS1_BCDMA_RING	0x004E210000	0x004E214000	16 KB
DMASS1_PSILSS_MMRS	0x004E220000	0x004E221000	4 KB
DMASS1_BCDMA_GCFG	0x004E230000	0x004E230100	256 B
DMASS1_PSILCFG_PROXY	0x004E260000	0x004E260200	512 B
DMASS1_BCDMA_TCHAN	0x004E280000	0x004E280800	2 KB
DMASS1_BCDMA_TCHANRT	0x004E300000	0x004E308000	32 KB
DMASS1_INTAGGR_INTR	0x004E400000	0x004E440000	256 KB
GPMC0_DATA	0x0050000000	0x0058000000	128 MB
FSS0_DAT_REG1	0x0060000000	0x0068000000	128 MB
PCIE0_DAT0	0x0068000000	0x0070000000	128 MB
MSRAM8KX256E0_RAM	0x0070000000	0x0070040000	256 KB
VPAC0_MEM_SLV_DATA	0x0071000000	0x0071080000	512 KB
R5FSS0_CORE0_ICACHE	0x0076000000	0x0076800000	8 MB
R5FSS0_CORE0_DCACHE	0x0076800000	0x0077000000	8 MB
R5FSS0_CORE0_ATCM	0x0078400000	0x0078408000	32 KB
R5FSS0_CORE0_BTCM	0x0078500000	0x0078508000	32 KB
C7X256V0_UMC	0x007C000000	0x007C020000	128 KB
C7X256V0_CLEC	0x007C200000	0x007C300000	1 MB
C7X256V0_DRU	0x007C400000	0x007C404000	16 KB
C7X256V0_DRU_SET	0x007C404000	0x007C408000	16 KB
C7X256V0_DRU_QUEUE	0x007C408000	0x007C410000	32 KB
C7X256V0_DRU_CHNRT	0x007C440000	0x007C460000	128 KB
C7X256V0_DRU_CHRT	0x007C460000	0x007C480000	128 KB
C7X256V0_DRU_CHATOMIC_DEBUG	0x007C480000	0x007C4A0000	128 KB
C7X256V0_DRU_CHCORE	0x007C4A0000	0x007C4C0000	128 KB
C7X256V0_DRU_CAUSE	0x007C4E0000	0x007C500000	128 KB
C7X256V1_UMC	0x007D000000	0x007D020000	128 KB
C7X256V1_CLEC	0x007D200000	0x007D300000	1 MB
C7X256V1_DRU	0x007D400000	0x007D404000	16 KB
C7X256V1_DRU_SET	0x007D404000	0x007D408000	16 KB
C7X256V1_DRU_QUEUE	0x007D408000	0x007D410000	32 KB
C7X256V1_DRU_CHNRT	0x007D440000	0x007D460000	128 KB
C7X256V1_DRU_CHRT	0x007D460000	0x007D480000	128 KB
C7X256V1_DRU_CHATOMIC_DEBUG	0x007D480000	0x007D4A0000	128 KB
C7X256V1_DRU_CHCORE	0x007D4A0000	0x007D4C0000	128 KB
C7X256V1_DRU_CAUSE	0x007D4E0000	0x007D500000	128 KB
C7X256V0_UMC_MEM_MAIN	0x007E000000	0x007E200000	2 MB
C7X256V1_UMC_MEM_MAIN	0x007E200000	0x007E400000	2 MB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
C7X256V0_UMC_MEM_AUX	0x007F000000	0x007F080000	512 KB
C7X256V1_UMC_MEM_AUX	0x007F800000	0x007F880000	512 KB
DDR32SS0_SDRAM	0x0080000000	0x0100000000	2 GB
FSS0_DAT_REG0	0x0400000000	0x0500000000	4 GB
FSS0_DAT_REG3	0x0500000000	0x0600000000	4 GB
PCIE0_DAT1	0x0600000000	0x0700000000	4 GB
DEBUGSS_WRAP0_ROM_TABLE_0_0	0x0700000000	0x0700001000	4 KB
DEBUGSS_WRAP0_RESV0_0	0x0700001000	0x0700002000	4 KB
DEBUGSS_WRAP0_CFGAP0	0x0700002000	0x0700002100	256 B
DEBUGSS_WRAP0_APBAP0	0x0700002100	0x0700002200	256 B
DEBUGSS_WRAP0_AXIAP0	0x0700002200	0x0700002300	256 B
DEBUGSS_WRAP0_PWRAP0	0x0700002300	0x0700002400	256 B
DEBUGSS_WRAP0_PVIEW0	0x0700002400	0x0700002500	256 B
DEBUGSS_WRAP0_JTAGAP0	0x0700002500	0x0700002600	256 B
DEBUGSS_WRAP0_SECAP0	0x0700002600	0x0700002700	256 B
DEBUGSS_WRAP0_CORTEX0_CFG0	0x0700002700	0x0700002800	256 B
DEBUGSS_WRAP0_CORTEX1_CFG0	0x0700002800	0x0700002900	256 B
DEBUGSS_WRAP0_CORTEX2_CFG0	0x0700002900	0x0700002A00	256 B
DEBUGSS_WRAP0_CORTEX3_CFG0	0x0700002A00	0x0700002B00	256 B
DEBUGSS_WRAP0_CORTEX4_CFG0	0x0700002B00	0x0700002C00	256 B
DEBUGSS_WRAP0_CORTEX5_CFG0	0x0700002C00	0x0700002D00	256 B
DEBUGSS_WRAP0_CORTEX6_CFG0	0x0700002D00	0x0700002E00	256 B
DEBUGSS_WRAP0_CORTEX7_CFG0	0x0700002E00	0x0700002F00	256 B
DEBUGSS_WRAP0_CORTEX8_CFG0	0x0700002F00	0x0700003000	256 B
DEBUGSS_WRAP0_RESV1_0	0x0700003000	0x0700004000	4 KB
DEBUGSS_WRAP0_RESV2_0	0x0700004000	0x0702004000	32 MB
DEBUGSS_WRAP0_ROM_TABLE_1_0	0x0720000000	0x0720001000	4 KB
DEBUGSS_WRAP0_CSCTI0	0x0720001000	0x0720002000	4 KB
DEBUGSS_WRAP0_DRM0	0x0720002000	0x0720003000	4 KB
DEBUGSS_WRAP0_RESV3_0	0x0720003000	0x0720004000	4 KB
DEBUGSS_WRAP0_CSTPIU0	0x0720004000	0x0720005000	4 KB
DEBUGSS_WRAP0_CTF0	0x0720005000	0x0720006000	4 KB
DEBUGSS_WRAP0_RESV4_0	0x0720006000	0x0721006000	16 MB
COMPUTE_CLUSTER0_SS_ROM	0x0730000000	0x0730010000	64 KB
DEBUGSS_WRAP0_EXT_APB0	0x0730000000	0x0740000000	256 MB
COMPUTE_CLUSTER0_CORE0_DBG	0x0730010000	0x0730020000	64 KB
COMPUTE_CLUSTER0_CORE0_CTI	0x0730020000	0x0730030000	64 KB
COMPUTE_CLUSTER0_CORE0_PMU	0x0730030000	0x0730040000	64 KB
COMPUTE_CLUSTER0_CORE0_ETM	0x0730040000	0x0730050000	64 KB
COMPUTE_CLUSTER0_CORE1_DBG	0x0730110000	0x0730120000	64 KB
COMPUTE_CLUSTER0_CORE1_PMU	0x0730120000	0x0730130000	64 KB
COMPUTE_CLUSTER0_CORE1_ETM	0x0730130000	0x0730140000	64 KB
COMPUTE_CLUSTER0_CORE1_CTI	0x0730140000	0x0730150000	64 KB
COMPUTE_CLUSTER0_CORE2_DBG	0x0730210000	0x0730220000	64 KB
COMPUTE_CLUSTER0_CORE2_PMU	0x0730220000	0x0730230000	64 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
COMPUTE_CLUSTER0_CORE2_ETM	0x0730230000	0x0730240000	64 KB
COMPUTE_CLUSTER0_CORE2_CTI	0x0730240000	0x0730250000	64 KB
COMPUTE_CLUSTER0_CORE3_DBG	0x0730310000	0x0730320000	64 KB
COMPUTE_CLUSTER0_CORE3_PMU	0x0730320000	0x0730330000	64 KB
COMPUTE_CLUSTER0_CORE3_ETM	0x0730330000	0x0730340000	64 KB
COMPUTE_CLUSTER0_CORE3_CTI	0x0730340000	0x0730350000	64 KB
C7X256V0_THINMAN	0x0734000000	0x0734002000	8 KB
C7X256V0_COLOMBO	0x0734001000	0x0734003000	8 KB
C7X256V0_MATLOCK	0x0734002000	0x0734004000	8 KB
C7X256V0_CSCTI	0x0734003000	0x0734004000	4 KB
C7X256V0_CTSET2	0x0734008000	0x073400A000	8 KB
C7X256V0_CTI2	0x073400A000	0x073400B000	4 KB
C7X256V0_CTI3	0x073400B000	0x073400C000	4 KB
C7X256V0_DBG_AGR0_MMR	0x0734040000	0x0734040100	256 B
C7X256V0_DBG_AGR0_MEM_CFG	0x0734040100	0x0734040200	256 B
C7X256V0_DBG_AGR0_MEM0	0x0734060000	0x0734061000	4 KB
C7X256V0_DBG_AGR0_MEM1	0x0734061000	0x0734062000	4 KB
C7X256V0_DBG_AGR0_MEM2	0x0734062000	0x0734063000	4 KB
C7X256V0_DBG_AGR0_MEM3	0x0734063000	0x0734064000	4 KB
C7X256V0_DBG_AGR0_MEM4	0x0734064000	0x0734065000	4 KB
C7X256V0_DBG_AGR0_MEM5	0x0734065000	0x0734066000	4 KB
C7X256V0_DBG_AGR0_MEM6	0x0734066000	0x0734067000	4 KB
C7X256V0_DBG_AGR0_MEM7	0x0734067000	0x0734068000	4 KB
C7X256V0_DBG_AGR0_MEM8	0x0734068000	0x0734069000	4 KB
C7X256V0_DBG_AGR0_MEM9	0x0734069000	0x073406A000	4 KB
C7X256V0_DBG_AGR0_MEM10	0x073406A000	0x073406B000	4 KB
C7X256V0_DBG_AGR0_MEM11	0x073406B000	0x073406C000	4 KB
C7X256V0_DBG_AGR0_MEM12	0x073406C000	0x073406D000	4 KB
C7X256V0_DBG_AGR0_MEM13	0x073406D000	0x073406E000	4 KB
C7X256V0_DBG_AGR0_MEM14	0x073406E000	0x073406F000	4 KB
C7X256V0_DBG_AGR0_MEM15	0x073406F000	0x0734070000	4 KB
C7X256V0_DBG_AGR0_MEM16	0x0734070000	0x0734071000	4 KB
C7X256V0_DBG_AGR0_MEM17	0x0734071000	0x0734072000	4 KB
C7X256V0_DBG_AGR0_MEM18	0x0734072000	0x0734073000	4 KB
C7X256V0_DBG_AGR0_MEM19	0x0734073000	0x0734074000	4 KB
C7X256V0_DBG_AGR0_MEM20	0x0734074000	0x0734075000	4 KB
C7X256V0_DBG_AGR0_MEM21	0x0734075000	0x0734076000	4 KB
C7X256V0_DBG_AGR0_MEM22	0x0734076000	0x0734077000	4 KB
C7X256V0_DBG_AGR0_MEM23	0x0734077000	0x0734078000	4 KB
C7X256V0_DBG_AGR0_MEM24	0x0734078000	0x0734079000	4 KB
C7X256V0_DBG_AGR0_MEM25	0x0734079000	0x073407A000	4 KB
C7X256V0_DBG_AGR0_MEM26	0x073407A000	0x073407B000	4 KB
C7X256V0_DBG_AGR0_MEM27	0x073407B000	0x073407C000	4 KB
C7X256V0_DBG_AGR0_MEM28	0x073407C000	0x073407D000	4 KB
C7X256V0_DBG_AGR0_MEM29	0x073407D000	0x073407E000	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
C7X256V0_DBG_AGR0_MEM30	0x073407E000	0x073407F000	4 KB
C7X256V0_DBG_AGR0_MEM31	0x073407F000	0x0734080000	4 KB
C7X256V1_THINMAN	0x0738000000	0x0738002000	8 KB
C7X256V1_COLOMBO	0x0738001000	0x0738003000	8 KB
C7X256V1_MATLOCK	0x0738002000	0x0738004000	8 KB
C7X256V1_CSCTI	0x0738003000	0x0738004000	4 KB
C7X256V1_CTSET2	0x0738008000	0x073800A000	8 KB
C7X256V1_CTI2	0x073800A000	0x073800B000	4 KB
C7X256V1_CTI3	0x073800B000	0x073800C000	4 KB
C7X256V1_DBG_AGR0_MMR	0x0738040000	0x0738040100	256 B
C7X256V1_DBG_AGR0_MEM_CFG	0x0738040100	0x0738040200	256 B
C7X256V1_DBG_AGR0_MEM0	0x0738060000	0x0738061000	4 KB
C7X256V1_DBG_AGR0_MEM1	0x0738061000	0x0738062000	4 KB
C7X256V1_DBG_AGR0_MEM2	0x0738062000	0x0738063000	4 KB
C7X256V1_DBG_AGR0_MEM3	0x0738063000	0x0738064000	4 KB
C7X256V1_DBG_AGR0_MEM4	0x0738064000	0x0738065000	4 KB
C7X256V1_DBG_AGR0_MEM5	0x0738065000	0x0738066000	4 KB
C7X256V1_DBG_AGR0_MEM6	0x0738066000	0x0738067000	4 KB
C7X256V1_DBG_AGR0_MEM7	0x0738067000	0x0738068000	4 KB
C7X256V1_DBG_AGR0_MEM8	0x0738068000	0x0738069000	4 KB
C7X256V1_DBG_AGR0_MEM9	0x0738069000	0x073806A000	4 KB
C7X256V1_DBG_AGR0_MEM10	0x073806A000	0x073806B000	4 KB
C7X256V1_DBG_AGR0_MEM11	0x073806B000	0x073806C000	4 KB
C7X256V1_DBG_AGR0_MEM12	0x073806C000	0x073806D000	4 KB
C7X256V1_DBG_AGR0_MEM13	0x073806D000	0x073806E000	4 KB
C7X256V1_DBG_AGR0_MEM14	0x073806E000	0x073806F000	4 KB
C7X256V1_DBG_AGR0_MEM15	0x073806F000	0x0738070000	4 KB
C7X256V1_DBG_AGR0_MEM16	0x0738070000	0x0738071000	4 KB
C7X256V1_DBG_AGR0_MEM17	0x0738071000	0x0738072000	4 KB
C7X256V1_DBG_AGR0_MEM18	0x0738072000	0x0738073000	4 KB
C7X256V1_DBG_AGR0_MEM19	0x0738073000	0x0738074000	4 KB
C7X256V1_DBG_AGR0_MEM20	0x0738074000	0x0738075000	4 KB
C7X256V1_DBG_AGR0_MEM21	0x0738075000	0x0738076000	4 KB
C7X256V1_DBG_AGR0_MEM22	0x0738076000	0x0738077000	4 KB
C7X256V1_DBG_AGR0_MEM23	0x0738077000	0x0738078000	4 KB
C7X256V1_DBG_AGR0_MEM24	0x0738078000	0x0738079000	4 KB
C7X256V1_DBG_AGR0_MEM25	0x0738079000	0x073807A000	4 KB
C7X256V1_DBG_AGR0_MEM26	0x073807A000	0x073807B000	4 KB
C7X256V1_DBG_AGR0_MEM27	0x073807B000	0x073807C000	4 KB
C7X256V1_DBG_AGR0_MEM28	0x073807C000	0x073807D000	4 KB
C7X256V1_DBG_AGR0_MEM29	0x073807D000	0x073807E000	4 KB
C7X256V1_DBG_AGR0_MEM30	0x073807E000	0x073807F000	4 KB
C7X256V1_DBG_AGR0_MEM31	0x073807F000	0x0738080000	4 KB
DEBUGSS0_DEBUG_CELL_ROM_SLV	0x073C020000	0x073C021000	4 KB
DEBUGSS0_CTSET2_WRAP_CFG_CTSET2_CFG	0x073C022000	0x073C024000	8 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
DEBUGSS0_ATB_REPLICATOR_CFG_CXATBREPLICATOR_CFG	0x073C024000	0x073C025000	4 KB
DEBUGSS0_TBR_VBUSP_WRAP_TBR_CFG_TBR_CFG	0x073C025000	0x073C026000	4 KB
DEBUGSS0_ARM_CTI_0_CFG_CSCTI_CFG	0x073C026000	0x073C027000	4 KB
DEBUGSS0_ARM_CTI_1_CFG_CSCTI_CFG	0x073C028000	0x073C029000	4 KB
DEBUGSS0_ARM_CTI_2_CFG_CSCTI_CFG	0x073C029000	0x073C02A000	4 KB
DEBUGSS0_ARM_CTI_3_CFG_CSCTI_CFG	0x073C02A000	0x073C02B000	4 KB
DEBUGSS0_ARM_CTI_4_CFG_CSCTI_CFG	0x073C02B000	0x073C02C000	4 KB
DEBUGSS0_ARM_CTI_5_CFG_CSCTI_CFG	0x073C02C000	0x073C02D000	4 KB
DEBUGSS0_ARM_CTI_6_CFG_CSCTI_CFG	0x073C02D000	0x073C02E000	4 KB
DEBUGSS0_ARM_CTI_7_CFG_CSCTI_CFG	0x073C02E000	0x073C02F000	4 KB
DEBUGSS0_ARM_CTI_8_CFG_CSCTI_CFG	0x073C02F000	0x073C030000	4 KB
CTI0_CSCTI_CFG	0x073D000000	0x073D001000	4 KB
CTI1_CSCTI_CFG	0x073D100000	0x073D101000	4 KB
STM0_CXSTM	0x073D200000	0x073D201000	4 KB
STM0_CTI_CSCTI	0x073D201000	0x073D202000	4 KB
DBGSUSPENDROUTER0_INTR_ROUTER_CFG	0x073D300000	0x073D300800	2 KB
CPT2_AGGR0_MMR	0x073E100000	0x073E100100	256 B
CPT2_AGGR0_STP2ATB_CFG	0x073E100100	0x073E100200	256 B
CPT2_AGGR0_MEM0	0x073E120000	0x073E121000	4 KB
CPT2_AGGR0_MEM1	0x073E121000	0x073E122000	4 KB
CPT2_AGGR0_MEM2	0x073E122000	0x073E123000	4 KB
CPT2_AGGR0_MEM3	0x073E123000	0x073E124000	4 KB
CPT2_AGGR0_MEM4	0x073E124000	0x073E125000	4 KB
CPT2_AGGR0_MEM5	0x073E125000	0x073E126000	4 KB
CPT2_AGGR0_MEM6	0x073E126000	0x073E127000	4 KB
CPT2_AGGR0_MEM7	0x073E127000	0x073E128000	4 KB
CPT2_AGGR0_MEM8	0x073E128000	0x073E129000	4 KB
CPT2_AGGR0_MEM9	0x073E129000	0x073E12A000	4 KB
CPT2_AGGR0_MEM10	0x073E12A000	0x073E12B000	4 KB
CPT2_AGGR0_MEM11	0x073E12B000	0x073E12C000	4 KB
CPT2_AGGR0_MEM12	0x073E12C000	0x073E12D000	4 KB
CPT2_AGGR0_MEM13	0x073E12D000	0x073E12E000	4 KB
CPT2_AGGR0_MEM14	0x073E12E000	0x073E12F000	4 KB
CPT2_AGGR0_MEM15	0x073E12F000	0x073E130000	4 KB
CPT2_AGGR0_MEM16	0x073E130000	0x073E131000	4 KB
CPT2_AGGR0_MEM17	0x073E131000	0x073E132000	4 KB
CPT2_AGGR0_MEM18	0x073E132000	0x073E133000	4 KB
CPT2_AGGR0_MEM19	0x073E133000	0x073E134000	4 KB
CPT2_AGGR0_MEM20	0x073E134000	0x073E135000	4 KB
CPT2_AGGR0_MEM21	0x073E135000	0x073E136000	4 KB
CPT2_AGGR0_MEM22	0x073E136000	0x073E137000	4 KB
CPT2_AGGR0_MEM23	0x073E137000	0x073E138000	4 KB
CPT2_AGGR0_MEM24	0x073E138000	0x073E139000	4 KB
CPT2_AGGR0_MEM25	0x073E139000	0x073E13A000	4 KB
CPT2_AGGR0_MEM26	0x073E13A000	0x073E13B000	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
CPT2_AGGR0_MEM27	0x073E13B000	0x073E13C000	4 KB
CPT2_AGGR0_MEM28	0x073E13C000	0x073E13D000	4 KB
CPT2_AGGR0_MEM29	0x073E13D000	0x073E13E000	4 KB
CPT2_AGGR0_MEM30	0x073E13E000	0x073E13F000	4 KB
CPT2_AGGR0_MEM31	0x073E13F000	0x073E140000	4 KB
CPT2_AGGR1_MMR	0x073E140000	0x073E140100	256 B
CPT2_AGGR1_STP2ATB_CFG	0x073E140100	0x073E140200	256 B
CPT2_AGGR1_MEM0	0x073E160000	0x073E161000	4 KB
CPT2_AGGR1_MEM1	0x073E161000	0x073E162000	4 KB
CPT2_AGGR1_MEM2	0x073E162000	0x073E163000	4 KB
CPT2_AGGR1_MEM3	0x073E163000	0x073E164000	4 KB
CPT2_AGGR1_MEM4	0x073E164000	0x073E165000	4 KB
CPT2_AGGR1_MEM5	0x073E165000	0x073E166000	4 KB
CPT2_AGGR1_MEM6	0x073E166000	0x073E167000	4 KB
CPT2_AGGR1_MEM7	0x073E167000	0x073E168000	4 KB
CPT2_AGGR1_MEM8	0x073E168000	0x073E169000	4 KB
CPT2_AGGR1_MEM9	0x073E169000	0x073E16A000	4 KB
CPT2_AGGR1_MEM10	0x073E16A000	0x073E16B000	4 KB
CPT2_AGGR1_MEM11	0x073E16B000	0x073E16C000	4 KB
CPT2_AGGR1_MEM12	0x073E16C000	0x073E16D000	4 KB
CPT2_AGGR1_MEM13	0x073E16D000	0x073E16E000	4 KB
CPT2_AGGR1_MEM14	0x073E16E000	0x073E16F000	4 KB
CPT2_AGGR1_MEM15	0x073E16F000	0x073E170000	4 KB
CPT2_AGGR1_MEM16	0x073E170000	0x073E171000	4 KB
CPT2_AGGR1_MEM17	0x073E171000	0x073E172000	4 KB
CPT2_AGGR1_MEM18	0x073E172000	0x073E173000	4 KB
CPT2_AGGR1_MEM19	0x073E173000	0x073E174000	4 KB
CPT2_AGGR1_MEM20	0x073E174000	0x073E175000	4 KB
CPT2_AGGR1_MEM21	0x073E175000	0x073E176000	4 KB
CPT2_AGGR1_MEM22	0x073E176000	0x073E177000	4 KB
CPT2_AGGR1_MEM23	0x073E177000	0x073E178000	4 KB
CPT2_AGGR1_MEM24	0x073E178000	0x073E179000	4 KB
CPT2_AGGR1_MEM25	0x073E179000	0x073E17A000	4 KB
CPT2_AGGR1_MEM26	0x073E17A000	0x073E17B000	4 KB
CPT2_AGGR1_MEM27	0x073E17B000	0x073E17C000	4 KB
CPT2_AGGR1_MEM28	0x073E17C000	0x073E17D000	4 KB
CPT2_AGGR1_MEM29	0x073E17D000	0x073E17E000	4 KB
CPT2_AGGR1_MEM30	0x073E17E000	0x073E17F000	4 KB
CPT2_AGGR1_MEM31	0x073E17F000	0x073E180000	4 KB
DEBUGSS_WRAP0_ROM_TABLE_0_1	0x0740000000	0x0740001000	4 KB
DEBUGSS_WRAP0_RESV0_1	0x0740001000	0x0740002000	4 KB
DEBUGSS_WRAP0_CFGAP1	0x0740002000	0x0740002100	256 B
DEBUGSS_WRAP0_APBAP1	0x0740002100	0x0740002200	256 B
DEBUGSS_WRAP0_AXIAP1	0x0740002200	0x0740002300	256 B
DEBUGSS_WRAP0_PWRAP1	0x0740002300	0x0740002400	256 B

Copyright © 2026 Texas Instruments Incorporated

Table 2-1. MAIN Memory Map (continued)

Region Name	Start Address	End Address	Size
DEBUGSS_WRAP0_PVIEW1	0x0740002400	0x0740002500	256 B
DEBUGSS_WRAP0_JTAGAP1	0x0740002500	0x0740002600	256 B
DEBUGSS_WRAP0_SECAP1	0x0740002600	0x0740002700	256 B
DEBUGSS_WRAP0_CORTEX0_CFG1	0x0740002700	0x0740002800	256 B
DEBUGSS_WRAP0_CORTEX1_CFG1	0x0740002800	0x0740002900	256 B
DEBUGSS_WRAP0_CORTEX2_CFG1	0x0740002900	0x0740002A00	256 B
DEBUGSS_WRAP0_CORTEX3_CFG1	0x0740002A00	0x0740002B00	256 B
DEBUGSS_WRAP0_CORTEX4_CFG1	0x0740002B00	0x0740002C00	256 B
DEBUGSS_WRAP0_CORTEX5_CFG1	0x0740002C00	0x0740002D00	256 B
DEBUGSS_WRAP0_CORTEX6_CFG1	0x0740002D00	0x0740002E00	256 B
DEBUGSS_WRAP0_CORTEX7_CFG1	0x0740002E00	0x0740002F00	256 B
DEBUGSS_WRAP0_CORTEX8_CFG1	0x0740002F00	0x0740003000	256 B
DEBUGSS_WRAP0_RESV1_1	0x0740003000	0x0740004000	4 KB
DEBUGSS_WRAP0_RESV2_1	0x0740004000	0x0742004000	32 MB
DEBUGSS_WRAP0_ROM_TABLE_1_1	0x0760000000	0x0760001000	4 KB
DEBUGSS_WRAP0_CSCTI1	0x0760001000	0x0760002000	4 KB
DEBUGSS_WRAP0_DRM1	0x0760002000	0x0760003000	4 KB
DEBUGSS_WRAP0_RESV3_1	0x0760003000	0x0760004000	4 KB
DEBUGSS_WRAP0_CSTPIU1	0x0760004000	0x0760005000	4 KB
DEBUGSS_WRAP0_CTF1	0x0760005000	0x0760006000	4 KB
DEBUGSS_WRAP0_RESV4_1	0x0760006000	0x0761006000	16 MB
DEBUGSS_WRAP0_EXT_APB1	0x0770000000	0x0780000000	256 MB
DDR32SS0_SDRAM	0x0880000000	0x0900000000	2 GB
DDR32SS0_SDRAM	0x0900000000	0x0A00000000	4 GB
DDR32SS0_SDRAM	0x0A00000000	0x0C00000000	8 GB
DDR32SS0_SDRAM	0x0C00000000	0x1000000000	16 GB

2.5 MCU Memory Map

Table 2-2. MCU Memory Map

Region Name	Start Address	End Address	Size
MCU_PLLCTRL0	0x0004020000	0x0004020200	512 B
MCU_GPIO0	0x0004201000	0x0004201100	256 B
MCU_TIMEOUT0_CFG	0x0004301000	0x0004301400	1 KB
MCU_R5FSS0_EVNT_BUS_VBUSP_MMRS	0x0004400000	0x0004400100	256 B
MCU_MCAN0_ECC_AGGR	0x0004701000	0x0004701400	1 KB
MCU_MCAN1_ECC_AGGR	0x0004702000	0x0004702400	1 KB
MCU_ECC_AGGR0_ECC_AGGR	0x0004703000	0x0004703400	1 KB
MCU_ECC_AGGR1_ECC_AGGR	0x0004704000	0x0004704400	1 KB
MCU_MSRAM_256K0_ECC_AGGR_REGS	0x0004705000	0x0004705400	1 KB
MCU_MSRAM_256K1_ECC_AGGR_REGS	0x0004706000	0x0004706400	1 KB
MCU_R5FSS0_CORE0_ECC_AGGR	0x0004707000	0x0004707400	1 KB
MCU_CBASS0_ERR	0x0004720000	0x0004720400	1 KB
MCU_TIMER0_CFG	0x0004800000	0x0004800400	1 KB
MCU_TIMER1_CFG	0x0004810000	0x0004810400	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-2. MCU Memory Map (continued)

Region Name	Start Address	End Address	Size
MCU_TIMER2_CFG	0x0004820000	0x0004820400	1 KB
MCU_TIMER3_CFG	0x0004830000	0x0004830400	1 KB
MCU_RTI0_CFG	0x0004880000	0x0004880100	256 B
MCU_I2C0_CFG	0x0004900000	0x0004900100	256 B
MCU_UART0	0x0004A00000	0x0004A00200	512 B
MCU_MCSPI0_CFG	0x0004B00000	0x0004B00400	1 KB
MCU_MCSPI1_CFG	0x0004B10000	0x0004B10400	1 KB
MCU_DCC0	0x0004C00000	0x0004C00040	64 B
MCU_DCC1	0x0004C10000	0x0004C10040	64 B
MCU_MCRC64_0_REGS	0x0004D00000	0x0004D01000	4 KB
MCU_MCAN0_MSGMEM_RAM	0x0004E00000	0x0004E08000	32 KB
MCU_MCAN0_CFG	0x0004E08000	0x0004E08200	512 B
MCU_MCAN0_SS	0x0004E09000	0x0004E09100	256 B
MCU_MCAN1_MSGMEM_RAM	0x0004E10000	0x0004E18000	32 KB
MCU_MCAN1_CFG	0x0004E18000	0x0004E18200	512 B
MCU_MCAN1_SS	0x0004E19000	0x0004E19100	256 B
MCU_PBIST0	0x0004F00000	0x0004F00400	1 KB
MCU_CBASS0_ISC	0x0045818000	0x0045819000	4 KB
MCU_MCU_SEC_MMR0_CFG2	0x0045940000	0x0045940400	1 KB
MCU_MCU_SEC_MMR0_CFG0	0x0045A40000	0x0045A40400	1 KB
MCU_CBASS0_GLB	0x0045B02000	0x0045B02400	1 KB
MCU_CBASS0_QOS	0x0045D18000	0x0045D19000	4 KB
MCU_R5FSS0_CORE0_ICACHE	0x0073000000	0x0073800000	8 MB
MCU_R5FSS0_CORE0_DCACHE	0x0073800000	0x0074000000	8 MB
MCU_R5FSS0_CORE0_ATCM	0x0079000000	0x0079008000	32 KB
MCU_R5FSS0_CORE0_BTCM	0x0079020000	0x0079028000	32 KB
MCU_MSRAM_256K0_RAM	0x0079100000	0x0079140000	256 KB
MCU_MSRAM_256K1_RAM	0x0079140000	0x0079180000	256 KB
MCU_CPT2_AGGR0_MMR	0x073E180000	0x073E180100	256 B
MCU_CPT2_AGGR0_STP2ATB_CFG	0x073E180100	0x073E180200	256 B
MCU_CPT2_AGGR0_MEM0	0x073E1A0000	0x073E1A1000	4 KB
MCU_CPT2_AGGR0_MEM1	0x073E1A1000	0x073E1A2000	4 KB
MCU_CPT2_AGGR0_MEM2	0x073E1A2000	0x073E1A3000	4 KB
MCU_CPT2_AGGR0_MEM3	0x073E1A3000	0x073E1A4000	4 KB
MCU_CPT2_AGGR0_MEM4	0x073E1A4000	0x073E1A5000	4 KB
MCU_CPT2_AGGR0_MEM5	0x073E1A5000	0x073E1A6000	4 KB
MCU_CPT2_AGGR0_MEM6	0x073E1A6000	0x073E1A7000	4 KB
MCU_CPT2_AGGR0_MEM7	0x073E1A7000	0x073E1A8000	4 KB
MCU_CPT2_AGGR0_MEM8	0x073E1A8000	0x073E1A9000	4 KB
MCU_CPT2_AGGR0_MEM9	0x073E1A9000	0x073E1AA000	4 KB
MCU_CPT2_AGGR0_MEM10	0x073E1AA000	0x073E1AB000	4 KB
MCU_CPT2_AGGR0_MEM11	0x073E1AB000	0x073E1AC000	4 KB
MCU_CPT2_AGGR0_MEM12	0x073E1AC000	0x073E1AD000	4 KB
MCU_CPT2_AGGR0_MEM13	0x073E1AD000	0x073E1AE000	4 KB
MCU_CPT2_AGGR0_MEM14	0x073E1AE000	0x073E1AF000	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-2. MCU Memory Map (continued)

Region Name	Start Address	End Address	Size
MCU_CPT2_AGGR0_MEM15	0x073E1AF000	0x073E1B0000	4 KB
MCU_CPT2_AGGR0_MEM16	0x073E1B0000	0x073E1B1000	4 KB
MCU_CPT2_AGGR0_MEM17	0x073E1B1000	0x073E1B2000	4 KB
MCU_CPT2_AGGR0_MEM18	0x073E1B2000	0x073E1B3000	4 KB
MCU_CPT2_AGGR0_MEM19	0x073E1B3000	0x073E1B4000	4 KB
MCU_CPT2_AGGR0_MEM20	0x073E1B4000	0x073E1B5000	4 KB
MCU_CPT2_AGGR0_MEM21	0x073E1B5000	0x073E1B6000	4 KB
MCU_CPT2_AGGR0_MEM22	0x073E1B6000	0x073E1B7000	4 KB
MCU_CPT2_AGGR0_MEM23	0x073E1B7000	0x073E1B8000	4 KB
MCU_CPT2_AGGR0_MEM24	0x073E1B8000	0x073E1B9000	4 KB
MCU_CPT2_AGGR0_MEM25	0x073E1B9000	0x073E1BA000	4 KB
MCU_CPT2_AGGR0_MEM26	0x073E1BA000	0x073E1BB000	4 KB
MCU_CPT2_AGGR0_MEM27	0x073E1BB000	0x073E1BC000	4 KB
MCU_CPT2_AGGR0_MEM28	0x073E1BC000	0x073E1BD000	4 KB
MCU_CPT2_AGGR0_MEM29	0x073E1BD000	0x073E1BE000	4 KB
MCU_CPT2_AGGR0_MEM30	0x073E1BE000	0x073E1BF000	4 KB
MCU_CPT2_AGGR0_MEM31	0x073E1BF000	0x073E1C0000	4 KB

2.6 WKUP Memory Map

Table 2-3. WKUP Memory Map

Region Name	Start Address	End Address	Size
WKUP_GTC0_GTC_CFG0	0x0000A80000	0x0000A80400	1 KB
WKUP_GTC0_GTC_CFG1	0x0000A90000	0x0000A94000	16 KB
WKUP_GTC0_GTC_CFG2	0x0000AA0000	0x0000AA4000	16 KB
WKUP_GTC0_GTC_CFG3	0x0000AB0000	0x0000AB4000	16 KB
WKUP_VTM0_MMR_VBUSP_CFG1	0x0000B00000	0x0000B00400	1 KB
WKUP_VTM0_MMR_VBUSP_CFG2	0x0000B01000	0x0000B01400	1 KB
WKUP_VTM0_ECCAGGR_CFG	0x0000B02000	0x0000B02400	1 KB
WKUP_PSC0	0x0004000000	0x0004001000	4 KB
WKUP_ECC_AGGR2_ECC_AGGR	0x0004030000	0x0004030400	1 KB
WKUP_PLL0_CFG	0x0004040000	0x0004041000	4 KB
MCU_PADCFG_CTRL0_CFG0	0x0004080000	0x0004088000	32 KB
WKUP_ESM0_CFG	0x0004100000	0x0004101000	4 KB
WKUP_MCU_GPIOMUX_INTROUTER0_INTR_ROUTER_CFG	0x0004210000	0x0004210200	512 B
MCU_CTRL_MMR0_CFG0	0x0004500000	0x0004520000	128 KB
WKUP_CBASS_SAFE1_ERR	0x0004600000	0x0004600400	1 KB
WKUP_RTI0_CFG	0x002B000000	0x002B000100	256 B
WKUP_TIMER0_CFG	0x002B100000	0x002B100400	1 KB
WKUP_TIMER1_CFG	0x002B110000	0x002B110400	1 KB
WKUP_RTCSS0_RTC	0x002B1F0000	0x002B1F0080	128 B
WKUP_I2C0_CFG	0x002B200000	0x002B200100	256 B
WKUP_UART0	0x002B300000	0x002B300200	512 B
WKUP_CBASS0_ERR	0x002B400000	0x002B400400	1 KB
WKUP_PBIST0	0x002B500000	0x002B500400	1 KB

Copyright © 2026 Texas Instruments Incorporated

Table 2-3. WKUP Memory Map (continued)

Region Name	Start Address	End Address	Size
WKUP_PBIST1	0x002B510000	0x002B510400	1 KB
WKUP_ECC_AGGR0_ECC_AGGR	0x002B600000	0x002B600400	1 KB
WKUP_ECC_AGGR1_ECC_AGGR	0x002B601000	0x002B601400	1 KB
WKUP_PSRAMECC_8K0_REGS	0x002B608000	0x002B608400	1 KB
WKUP_R5FSS0_EVNT_BUS_VBUSP_MMRS	0x003C018000	0x003C018100	256 B
WKUP_R5FSS0_CORE0_ECC_AGGR	0x003F00D000	0x003F00D400	1 KB
WKUP_ROM0	0x0041800000	0x0041840000	256 KB
WKUP_PSRAMECC_8K0_RAM	0x0041880000	0x0041888000	32 KB
WKUPN_WKUP_CTRL_MMR0_CFG0	0x0043000000	0x0043020000	128 KB
WKUP_CBASS0_FW	0x0045008000	0x004500C000	16 KB
WKUP_CBASS0_ISC	0x0045814000	0x0045816000	8 KB
WKUP_WKUP_SEC_MMR0_CFG2	0x0045920000	0x0045940000	128 KB
WKUP_WKUP_SEC_MMR0_CFG0	0x0045A20000	0x0045A40000	128 KB
WKUP_CBASS0_GLB	0x0045B03000	0x0045B03400	1 KB
WKUP_CBASS0_QOS	0x0045D14000	0x0045D16000	8 KB
WKUP_R5FSS0_CORE0_ICACHE	0x0074000000	0x0074800000	8 MB
WKUP_R5FSS0_CORE0_DCACHE	0x0074800000	0x0075000000	8 MB
WKUP_R5FSS0_CORE0_ATCM	0x0078000000	0x0078008000	32 KB
WKUP_R5FSS0_CORE0_BTCM	0x0078100000	0x0078108000	32 KB

2.7 R5FSS0 Memory Map

Table 2-4. R5FSS0 Memory Map

Region Name	Start Address	End Address	Size
ATCM	0x0000000000	0x0000007FFF	32 KB
RAT_REGION0	0x0000010000	0x001FFFFFFF	512 MB
NON_RAT_SOC_REGION0	0x0020000000	0x002FFDFFFF	256 MB
RAT_CFG	0x002FFE0000	0x002FFE0FFF	4 KB
VIC_CFG	0x002FFF0000	0x002FFF3FFF	16 KB
RAT_REGION1	0x0030000000	0x004100FFFF	272 MB
BTCM	0x0041010000	0x0041017FFF	32 KB
RAT_REGION2	0x0041018000	0x007FFFFFFF	1008 MB
RAT_REGION3	0x0080000000	0x00FFFFFFFF	2 GB

2.8 MCU_R5FSS0 Memory Map

Table 2-5. MCU_R5FSS0 Memory Map

Region Name	Start Address	End Address	Size
ATCM	0x0000000000	0x0000007FFF	32 KB
RAT_REGION0	0x0000010000	0x000400FFFF	64 MB
NON_RAT_REGION0	0x0004000000	0x0007FDFFFF	64 MB
RAT_CFG	0x0007FE0000	0x0007FE0FFF	4 KB
VIC_CFG	0x0007FF0000	0x0007FF3FFF	16 KB
RAT_REGION1	0x0008000000	0x004100FFFF	912 MB
BTCM	0x0041010000	0x0041017FFF	32 KB
RAT_REGION2	0x0041018000	0x007FFFFFFF	1008 MB

Copyright © 2026 Texas Instruments Incorporated

Table 2-5. MCU_R5FSS0 Memory Map (continued)

Region Name	Start Address	End Address	Size
RAT_REGION3	0x0080000000	0x00FFFFFFFF	2 GB

2.9 WKUP_R5FSS0 Memory Map

Table 2-6. WKUP_R5FSS0 Memory Map

Region Name	Start Address	End Address	Size
ATCM	0x0000000000	0x0000007FFF	32 KB
RAT_REGION0	0x0000010000	0x001FFFFFFF	512 MB
NON_RAT_SOC_REGION0	0x0020000000	0x002FFDFFFF	256 MB
RAT_CFG	0x002FFE0000	0x002FFE0FFF	4 KB
VIC_CFG	0x002FFF0000	0x002FFF3FFF	16 KB
RAT_REGION1	0x0030000000	0x004100FFFF	272 MB
BTCM	0x0041010000	0x0041017FFF	32 KB
RAT_REGION2	0x0041018000	0x007FFFFFFF	1008 MB
RAT_REGION3	0x0080000000	0x00FFFFFFFF	2 GB

2.10 DMASS0 Memory Map

Table 2-7. DMASS0 Memory Map

Region Name	Start Address	End Address	Size
RINGACC__SRC__FIFOS	0x0000400000	0x00007FFFFF	4 MB

2.11 SMS0 Memory Map

Table 2-8. SMS0 Memory Map

Region Name	Start Address	End Address	Size
HSM_CTRL_MMR	0x0043936000	0x0043936FFF	4 KB

2.12 SA3_SS0 Memory Map

Table 2-9. SA3_SS0 Memory Map

Region Name	Start Address	End Address	Size
RINGACC__SRC__FIFOS	0x0000400000	0x0000405FFF	24 KB

Copyright © 2026 Texas Instruments Incorporated

Chapter 3

System Interconnect

3.1 System Interconnect Overview 67
3.2 Domain Partition 67
3.3 Initiator/Target Connectivity 68
3.4 Interrupt Condition for Interconnect 68
3.5 Coherency Features 73
3.6 Performance Tuning 74
3.7 Interconnect Debugging Feature 78
3.8 Quality of Service (QoS) Block 78
3.9 Route ID 82
3.10 ISC and Firewall 83

Copyright © 2026 Texas Instruments Incorporated

3.1 System Interconnect Overview

3.1.1 Terminology

- CBASS: this is a crossbar module to provide physical connection among the initiators and targets
- VBUSP interface: a single-issue interface
- VBUSM interface: a multi-issue interface
- Channel ID: channel ID for interface indicates a logical flow. All the transactions with the same channel ID coming from the same initiator interface are consider orthogonal and independent flow.
- OrderID: a 4 bits value associated with each transaction. All the transactions from the same initiator to the same target end point with the same orderID needs to be executed in order. OrderID is also to be used to select real-time and non-real time path for the transaction. OrderID 8-15 reserved for real-time path.
- Asel: Address Selection. Asel can be either used for select a unique memory map or be used to indicate the IO coherent transactions.
- PrivID: indicates the security access group. PrivID is assigned by the ISC block.
- ISC: stands for Initiator Security Control.
- Region based firewall: this is firewall block based on address region.
- Channelized firewall: firewall block with finer granularity down to register level with fixed region size.
- QoS: Quality of Service block is a feature which can be enabled for each CBA initiator port. Each initiator can choose to enable the QoS feature in the CBA configuration file. If an initiator port supports channel ID, the QoS block provides priority and order ID for each channel, otherwise all the transactions from that initiator port share the same priority and order ID value.

CAUTION

Since OrderID is used for ordering purpose in CBASS, changing the order ID when the system is running can cause system deadlock. Therefore, QoS configuration shall be part of the initialization steps, if the application wants to change the QoS setting from the default value.

3.2 Domain Partition

SoC is partitioned into several domains which are connected by interconnect modules.

- **MAIN Domain**: there are multiple interconnect components providing the connectivity among initiators and target interfaces for the processors and peripherals in the MAIN domain. The main application processors, such as A53SS, are located in the MAIN domain.
- **WKUP domain**: R5FSS is located in the WKUP domain. The WKUP domain is active during deepsleep mode.
- **MCU domain**: A microcontroller is located in the MCU domain. The MCU domain can be isolated from the rest of SoC during safety use case.

All modules connected to WKUP_safe interconnect can belong to both MCU domain and WKUP domain depending on the use case. The deep sleep low power mode is controlled by the WKUP Domain. However, when the MCU domain is configured for a safety use case, all of the modules connected to WKUP_safe interconnect will be part of the MCU domain.

Copyright © 2026 Texas Instruments Incorporated

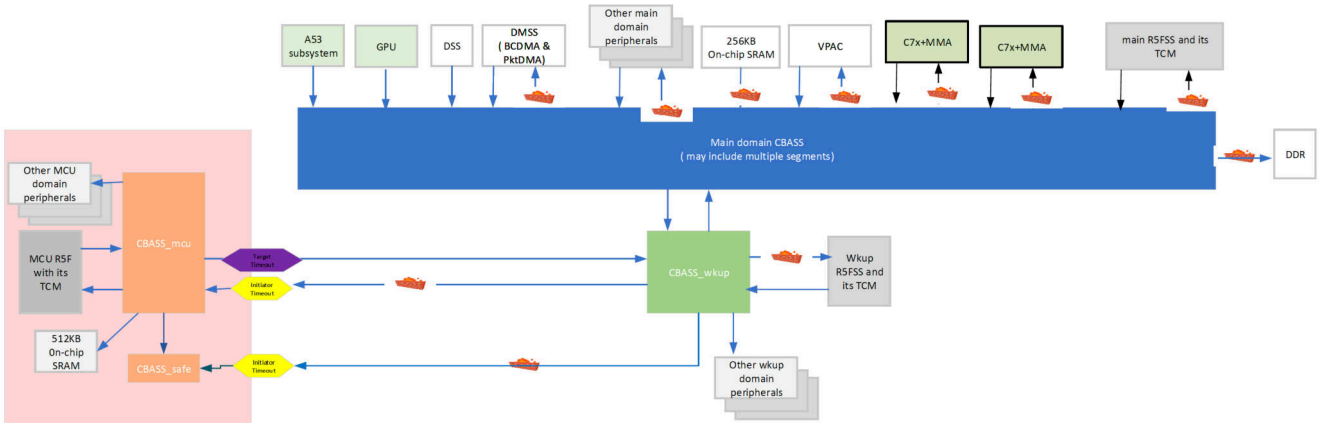


Figure 3-1. System Interconnect Overview

3.3 Initiator/Target Connectivity

All processors and the DMA can communicate with all SoC memory mapped peripherals.

3.4 Interrupt Condition for Interconnect

Each CBASS crossbar IP may generate up to two interrupts:

- default_err_intr: the intent of this interrupt is for debug purposes. Any illegal transactions will trigger this interrupt and the illegal transaction information is logged by CBASS IP.
- default_exp: this interrupt is asserted whenever there is a firewall violation. Only the CBASS IP contains firewall module that can generate this interrupt.

The illegal transactions triggering default_err_intr can be caused by multiple reasons:

- The transaction is sent to a memory region not implemented in CBASS
- The target interface or target peripheral is in disabled state

When CBASS detects an illegal transaction, it terminates the transaction gracefully to avoid a system hang. In addition, the CBASS sends return status back to the initiator with error status. If this illegal transaction is a read transaction, the read data is returned with all zeros. If this illegal transaction is a write transaction, this write transaction is terminated and returns a write error back to initiator. The interrupt and illegal transaction logging can be found in the CBASS's err_regs region.

For any transaction blocked by the firewall, the transaction is terminated gracefully and the initiator will receive error status for the blocked transaction. If this blocked transaction is read, the read data is returned with all zeros. If this blocked transaction is a write transaction, this write transaction is terminated and returns a write error back to the initiator. The default_exp interrupt and the blocked transaction information can be found in the glb_regs region.

Copyright © 2026 Texas Instruments Incorporated

Table 3-1. CBASS Interrupt

CBASS Name	default_err_intr Interrupt Supported	default_exp Interrupt Supported
CBASS0	Yes	Yes
MCU_CBASS0	Yes	No
CBASS_INFRA1	Yes	Yes
CBASS_FW0	Yes	No
CBASS_DBG0	Yes	No
CBASS_IPCSS0	Yes	Yes
CBASS_CENTRAL2	Yes	Yes
CBASS_MCASP0	Yes	No
WKUP_CBASS0	Yes	Yes

3.4.1 Register MMR for default_err_intr

Each CBASS supports default_err_intr containing an err_regs region, which includes the following registers:

Address Offset	Register Mnemonic	Register Name
0x0	PID	Revision Register
0x4	DESTINATION_ID	Destination ID Register
0x24	EXCEPTION_LOGGING_HEADER0	Exception Logging Header 0 Register
0x28	EXCEPTION_LOGGING_HEADER1	Exception Logging Header 1 Register
0x2c	EXCEPTION_LOGGING_DATA0	Exception Logging Data 0 Register
0x30	EXCEPTION_LOGGING_DATA1	Exception Logging Data 1 Register
0x34	EXCEPTION_LOGGING_DATA2	Exception Logging Data 2 Register
0x38	EXCEPTION_LOGGING_DATA3	Exception Logging Data 3 Register
0x50	ERR_INTR_RAW_STAT	Global Interrupt Raw Status Register
0x54	ERR_INTR_ENABLED_STAT	Global Interrupt Enabled Status Register
0x58	ERR_INTR_ENABLE_SET	Interrupt Enable Set Register
0x5c	ERR_INTR_ENABLE_CLR	Interrupt Enable Clear Register
0x60	ERR_EOI	EOI Register

PID (Revision Register) is located at offset 0x0. The following shows the definition of each field of this register.

The Revision Register contains the major and minor revisions for the module.

Bit	Name	Type	Reset	Description
31:30	scheme	r	0x1	PID register scheme
29:28	bu	r	0x2	Business Unit: 10 = Processors
27:16	func	r	0x600	Module ID
15:11	rtl	r	0x6	RTL revision. Will vary depending on release.
10:8	major	r	0x1	Major revision
7:6	custom	r	0x0	Custom
5:0	minor	r	0x2	Minor revision

DESTINATION_ID (Destination ID Register) has offset 0x4.

The Destination ID Register defines the destination ID value for error messages.

Bit	Name	Type	Reset	Description
31:8	reserved	r	0x0	Reserved
7:0	dest_id	rw	0x0	The destination ID.

Copyright © 2026 Texas Instruments Incorporated

EXCEPTION_LOGGING_HEADER0 (Exception Logging Header 0 Register), (0x24)

The Exception Logging Header 0 Register contains the first word of the header.

Bit	Name	Type	Reset	Description
31:24	type_f	r	0x0	Type. 7 = CBASS.
23:8	src_id	r	0x0	Source ID. Always 0.
7:0	dest_id	r	0x0	Destination ID.

EXCEPTION_LOGGING_HEADER1 (Exception Logging Header 1 Register) has offset 0x28.

The Exception Logging Header 1 Register contains the second word of the header.

Bit	Name	Type	Reset	Description
31:24	group	r	0x0	Group. Always 0.
23:16	code	r	0x0	Code. 0 = CBASS decode error.
15:0	reserved	r	0x0	Reserved

EXCEPTION_LOGGING_DATA0 (Exception Logging Data 0 Register) has offset 0x2c.

The Exception Logging Data 0 Register contains the first word of the data.

Bit	Name	Type	Reset	Description
31:0	addr_l	r	0x0	Address lower 32 bits.

EXCEPTION_LOGGING_DATA1 (Exception Logging Data 1 Register), (0x30)

The Exception Logging Data 1 Register contains the second word of the data.

Bit	Name	Type	Reset	Description
31:16	reserved	r	0x0	Reserved
15:0	addr_h	r	0x0	Address upper 16 bits.

EXCEPTION_LOGGING_DATA2 (Exception Logging Data 2 Register has offset 0x34.

The Exception Logging Data 2 Register contains the third word of the data.

Bit	Name	Type	Reset	Description
31:28	reserved	r	0x0	Reserved
27:16	routeid	r	0x0	Route ID.
15:14	reserved	r	0x0	Reserved
13	write	r	0x0	Write.
12	read	r	0x0	Read.
11	debug	r	0x0	Debug.
10	cacheable	r	0x0	Cacheable.
9	priv	r	0x0	Priv.
8	secure	r	0x0	Secure.
7:0	priv_id	r	0x0	Priv ID.

EXCEPTION_LOGGING_DATA3 (Exception Logging Data 3 Register) has offset 0x38.

The Exception Logging Data 3 Register contains the fourth word of the data.

Bit	Name	Type	Reset	Description
31:10	reserved	r	0x0	Reserved

Copyright © 2026 Texas Instruments Incorporated

Bit	Name	Type	Reset	Description
9:0	bytecnt	r	0x0	Byte count.

ERR_INTR_RAW_STAT (Global Interrupt Raw Status Register), (0x50)

The interrupt raw status register indicates if there is null interrupt regardless of interrupt enable

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	intr	rw1ts	0x0	Level Interrupt status

ERR_INTR_ENABLED_STAT (Global Interrupt Enabled Status Register), (0x54)

The interrupt status register is gated by the interrupt enable

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	enabled_intr	rw1tc	0x0	Level Enabled Interrupt status

ERR_INTR_ENABLE_SET (Interrupt Enable Set Register), (0x58)

Only when this register is set, null access will cause interrupt to be generated.

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	intr_enable_set	rw1ts	0x0	Interrupt Enable Set Register

ERR_INTR_ENABLE_CLR (Interrupt Enable Clear Register), (0x5c)

Setting this register disables the null interrupt generation

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	intr_enable_clr	rw1tc	0x0	Interrupt Enable Clear Register

ERR_EOI (EOI Register), (0x60)

Writing to EOI Register indicates that current interrupt has been serviced which then allows next interrupt to be generated

Bit	Name	Type	Reset	Description
31:16	reserved	r	0x0	Reserved
15:0	eoi_wr	rw	0x0	End Of Interrupt Register

3.4.2 Register MMR for default_exp

All of the registers for default_exp_intr are located in the glb_regs region of the CBASS which supports this feature. For each glb_reg region, the following registers are supported.

Table 3-2. glb_regs for default_exp_intr

Address Offset	Register Mnemonic	Register Name
0x0	PID	Revision Register
0x4	DESTINATION_ID	Destination ID Register
0x20	EXCEPTION_LOGGING_CONTROL	Exception Logging Control Register
0x24	EXCEPTION_LOGGING_HEADER0	Exception Logging Header 0 Register
0x28	EXCEPTION_LOGGING_HEADER1	Exception Logging Header 1 Register

Copyright © 2026 Texas Instruments Incorporated

Table 3-2. glb_regs for default_exp_intr (continued)

Address Offset	Register Mnemonic	Register Name
0x2c	EXCEPTION_LOGGING_DATA0	Exception Logging Data 0 Register
0x30	EXCEPTION_LOGGING_DATA1	Exception Logging Data 1 Register
0x34	EXCEPTION_LOGGING_DATA2	Exception Logging Data 2 Register
0x38	EXCEPTION_LOGGING_DATA3	Exception Logging Data 3 Register
0x40	EXCEPTION_PEND_SET	Exception Logging Pending Set Register
0x44	EXCEPTION_PEND_CLEAR	Exception Logging Pending Clear Register

The Revision Register contains the major and minor revisions for the module.

Bit	Name	Type	Reset	Description
31:30	scheme	r	0x1	PID register scheme
29:28	bu	r	0x2	Business Unit: 10 = Processors
27:16	func	r	0x600	Module ID
15:11	rtl	r	0x6	RTL revision. Will vary depending on release.
10:8	major	r	0x1	Major revision
7:6	custom	r	0x0	Custom
5:0	minor	r	0x2	Minor revision

The Destination ID Register defines the destination ID value for error messages.

Bit	Name	Type	Reset	Description
31:8	reserved	r	0x0	Reserved
7:0	dest_id	rw	0x0	The destination ID.

The Exception Logging Control Register controls the exception logging.

Bit	Name	Type	Reset	Description
31:2	reserved	r	0x0	Reserved
1	disable_pend	rw	0x0	Disables logging pending when set.
0	disable_f	rw	0x0	Disables logging when set.

The Exception Logging Header 0 Register contains the first word of the header.

Bit	Name	Type	Reset	Description
31:24	type_f	r	0x0	Type.
23:8	src_id	r	0x0	Source ID.
7:0	dest_id	r	0x0	Destination ID.

The Exception Logging Header 1 Register contains the second word of the header.

Bit	Name	Type	Reset	Description
31:24	group	r	0x0	Group.
23:16	code	r	0x0	Code.
15:0	reserved	r	0x0	Reserved

The Exception Logging Data 0 Register contains the first word of the data.

Copyright © 2026 Texas Instruments Incorporated

Bit	Name	Type	Reset	Description
31:0	addr_l	r	0x0	Address lower 32 bits.

The Exception Logging Data 1 Register contains the second word of the data.

Bit	Name	Type	Reset	Description
31:16	reserved	r	0x0	Reserved
15:0	addr_h	r	0x0	Address upper 16 bits.

The Exception Logging Data 2 Register contains the third word of the data.

Bit	Name	Type	Reset	Description
31:28	reserved	r	0x0	Reserved
27:16	routeid	r	0x0	Route ID.
15:14	reserved	r	0x0	Reserved
13	write	r	0x0	Write.
12	read	r	0x0	Read.
11	debug	r	0x0	Debug.
10	cacheable	r	0x0	Cacheable.
9	priv	r	0x0	Priv.
8	secure	r	0x0	Secure.
7:0	priv_id	r	0x0	Priv ID.

The Exception Logging Data 3 Register contains the fourth word of the data.

Bit	Name	Type	Reset	Description
31:10	reserved	r	0x0	Reserved
9:0	bytecnt	r	0x0	Byte count.

The Exception Logging Pending Set Register allows to set the pend signal.

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	pend_set	rw1ts	0x0	Write a 1 to set the exception pend signal.

The Exception Logging Pending Clear Register allows to clear the pend signal.

Bit	Name	Type	Reset	Description
31:1	reserved	r	0x0	Reserved
0	pend_clr	rw1tc	0x0	Write a 1 to clear the exception pend signal.

3.5 Coherency Features

3.5.1 IO Coherency Support

There are 4 bits for ASEL sideband signals. These enable a single SoC to support up to 16 different memory maps. ASEL can be used for two different purposes: it can be used as an indicator of a different memory map (ASEL value 1-12) or indicate transactions requires IO coherency (ASEL value 13-14). ASEL value can be

Copyright © 2026 Texas Instruments Incorporated

configured either through DMA configuration if the transaction is initiated by DMSS or UTC or DRU or configured through QoS MMR for the initiator who generates the transactions. Refer to QoS Programming Guide.

ASEL value of 0

- Default for the SoC Memory Map
- Will use the transaction address for memory map decoding

Non-zero ASEL values

- Assigned to a target end point, which can be used to bypass the normal address decoding
- Can be assigned to the transactions through either DMA configuration for both BCDMA and pktDMA transaction or through the QoS block inserted for each initiator interface
- ASEL value 1: Reserved for PCIe if supported by the device. Otherwise, transactions will be routed to the null end point for graceful termination.
- ASEL values 2-13: Transactions will be routed to the null end point for graceful termination.
- ASEL values 14-15: Dedicated routing to A53SS ACP Interface for transactions that require IO Coherency.
  - Write transactions with ASEL=14 cause A53SS L2 cache allocation: for cache warming feature
  - Write transactions with ASEL=15 do not cause A53SS L2 cache allocation
  - Read transactions with ASEL=14 and 15 do not cause L2 cache allocation
  - All the transactions to ACP port can be protected by firewall if firewall is inserted by the SoC level interconnect

The A53SS checks security on the transactions from the ACP port. However, it does not include supervisor or user mode checks. If different permissions are needed for the supervisor or user mode transactions, different access permissions can be granted using the region-based firewall.

The transactions initiated by the microcontroller do not support IO coherency.

3.6 Performance Tuning

3.6.1 Latency Reduction

The auto clock gating feature of WKUP domain peripherals is enabled by default.

The disabling of auto clock gating may result in improved performance.

The auto clock gating control can be modified during device initialization time through registers CLKGATE_CTRL0 and CLKGATE_CTRL1 in the WKUP_CTRL_MMR domain.

3.6.2 Using Quality of Service (QoS) MMR

The majority of the initiator has a dedicated QoS block to provide the configurability of the transaction characteristic, such as OrderID, priority/epriority, asel and etc. The user can utilize the OrderID and priority/epriority fields to fine tune the performance.

Each transaction in the system carries 3 bits priority information. The priority information is used for interconnect for arbitration decision, which implements typical priority based round robin. Priority value 0x0 is the highest priority, while 0x7 is the lowest priority. By default, QoS has priority value set to 0x7(lowest priority). The priority and epriority can be changed through the QoS block for each initiator.

Some of the modules such as DSS is able to adjust the priority of the transaction based on the system congestion condition. But majority of the transactions have static priority level set by the QoS block. But the priority setting through QoS block can be tuned to fit certain use case scenarios.

CAUTION

Priority, orderID and asel value from the QoS block shall only be modified as part of the initialization steps, when the SoC is idle. Changing the QoS block configuration during the run time is prohibited, and may cause system error or other undefined behavior.

Copyright © 2026 Texas Instruments Incorporated

3.6.2.1 QoS MMR Programming Guide

The QoS MMR is integrated as part of data plane interconnect to provide programmability of priority, orderID and QoS sideband signals for each initiator port in the data plane. Currently the QoS sideband signal is not used for any routing or schedule purpose. For most of the initiator ports, it has only one setting of the orderID and QoS field. For the initiator port supports with multiple channels, the transactions in each channel can be programmed to have different sets of priority, orderID and QoS. The orderID is a four-bit field value. All the transactions coming from the same initiator ports with the same orderID expect to receive the read data in order.

In the previous generation of the interconnect, how the traffic is split among parallel paths to the same end points was hard coded. The traffic from a certain initiator port was hard coded to choose one of the parallel paths. Due to the nature of hard coded, the system could not load balance based on the use case. The current generation of interconnect removes this limitation by splitting traffic using the OrderID value. Since the orderID value for the traffic from each initiator port is programmable, the user can load balance based on the use case scenario.

Each target port in each SCR has its own arbiter. The SoC interconnect arbitration is based on priority. For the ports with equal priority, the arbiter uses round robin to break the tie.

DDR EMIF (External Memory Interface) also uses orderID to maintain transaction ordering. All transactions with the same orderID sent to the EMIF will be executed in order. In order to maximize the EMIF performance, the user shall carefully use the full range of 16 orderID values. Therefore, the EMIF can utilize its re-ordering mechanism to improve the EMIF utilization.

Inside the EMIF controller, registers map CBA priority to AXI priority. The EMIF controller schedules transactions using a strict priority scheme based on AXI priority. In order to maximize the DDR utilization to reduce the memory page open/close and read and write turn around penalty, it is suggested to map to a single AXI priority.

3.6.2.2 QoS Summary Tables

3.6.2.2.1 CODEC0_QoS_Map

Table 3-3. CODEC0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
CODEC0_PRI_M_VBUSM_R_ASYNC	0x45D26800	5	No	No	Yes	Yes	No	Yes
CODEC0_PRI_M_VBUSM_W_ASYNC	0x45D26C00	5	No	No	Yes	Yes	No	Yes
CODEC0_SEC_M_VBUSM_R_ASYNC	0x45D27000	1	No	No	Yes	Yes	No	Yes
CODEC0_SEC_M_VBUSM_W_ASYNC	0x45D27400	1	No	No	Yes	Yes	No	Yes

3.6.2.2.2 DEBUGSS_WRAP0_QoS_Map

Table 3-4. DEBUGSS_WRAP0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
DEBUGSS_WRAP0_VBUSMW	0x45D21800	1	No	No	Yes	Yes	No	Yes
DEBUGSS_WRAP0_VBUSMR	0x45D21C00	1	No	No	Yes	Yes	No	Yes

3.6.2.2.3 COMPUTE_CLUSTER0_QoS_Map

Copyright © 2026 Texas Instruments Incorporated

Table 3-5. COMPUTE_CLUSTER0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
COMPUTE_CLUSTER0_A53_QUAD_WRAP_CBA_AXI_R	0x45D20400	1	No	No	Yes	Yes	No	Yes
COMPUTE_CLUSTER0_A53_QUAD_WRAP_CBA_AXI_W	0x45D20800	1	No	No	Yes	Yes	No	Yes

3.6.2.2.4 DSS0_QoS_Map

Table 3-6. DSS0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
DSS0_VBUSM_DMA	0x45D30000	4	No	No	Yes	No	No	Yes

3.6.2.2.5 DSS1_QoS_Map

Table 3-7. DSS1 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
DSS1_VBUSM_DMA	0x45D30400	4	No	No	Yes	No	No	Yes

3.6.2.2.6 GICSS0_QoS_Map

Table 3-8. GICSS0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
GICSS0_MEM_WR_VBUSM	0x45D22000	1	No	No	Yes	Yes	No	Yes
GICSS0_MEM_RD_VBUSM	0x45D22400	1	No	No	Yes	Yes	No	Yes

3.6.2.2.7 GPU0_QoS_Map

Table 3-9. GPU0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
GPU0_M_VBUSM_W_SYNC	0x45D29800	32	No	No	Yes	Yes	No	Yes
GPU0_M_VBUSM_R_SYNC	0x45D2A000	32	No	No	Yes	Yes	No	Yes

3.6.2.2.8 MCU_R5FSS0_QoS_Map

Table 3-10. MCU_R5FSS0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
MCU_R5FSS0_CPU0_RMST	0x45D18000	1	No	No	Yes	Yes	No	Yes
MCU_R5FSS0_CPU0_WMST	0x45D18400	1	No	No	Yes	Yes	No	Yes

Copyright © 2026 Texas Instruments Incorporated

Table 3-10. MCU_R5FSS0 QoS Map (continued)

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
MCU_R5FSS0_CPU0_PMST	0x45D18800	1	No	No	Yes	Yes	No	Yes

3.6.2.2.9 MMCSD0_QoS_Map

Table 3-11. MMCSD0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
MMCSD0_EMMCSDSS_WR	0x45D34800	1	No	No	Yes	Yes	No	Yes
MMCSD0_EMMCSDSS_RD	0x45D34C00	1	No	No	Yes	Yes	No	Yes

3.6.2.2.10 WKUP_R5FSS0_QoS_Map

Table 3-12. WKUP_R5FSS0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
WKUP_R5FSS0_CPU0_RMST	0x45D14000	1	No	No	Yes	Yes	No	Yes
WKUP_R5FSS0_CPU0_WMST	0x45D14400	1	No	No	Yes	Yes	No	Yes
WKUP_R5FSS0_CPU0_PMST	0x45D14800	1	No	No	Yes	Yes	No	Yes

3.6.2.2.11 USB0_QoS_Map

Table 3-13. USB0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
USB0_MSTR0	0x45D34000	1	No	No	Yes	Yes	No	Yes
USB0_MSTW0	0x45D34400	1	No	No	Yes	Yes	No	Yes

3.6.2.2.12 USB1_QoS_Map

Table 3-14. USB1 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
USB1_MSTR0	0x45D35000	1	No	No	Yes	Yes	No	Yes
USB1_MSTW0	0x45D35400	1	No	No	Yes	Yes	No	Yes

3.6.2.2.13 SA3_SS0_QoS_Map

Table 3-15. SA3_SS0 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
SA3_SS0_CTXCACH_EXT_DMA	0x45D25400	1	No	No	Yes	Yes	No	Yes

Copyright © 2026 Texas Instruments Incorporated

3.6.2.2.14 MMCSD1_QoS_Map

Table 3-16. MMCSD1 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
MMCSD1_EMMCSDSS_RD	0x45D23000	1	No	No	Yes	Yes	No	Yes
MMCSD1_EMMCSDSS_WR	0x45D23400	1	No	No	Yes	Yes	No	Yes

3.6.2.2.15 MMCSD2_QoS_Map

Table 3-17. MMCSD2 QoS Map

Initiator	MMR Base Address	Channel Count	QOS Capable	Virtual Id Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable
MMCSD2_EMMCSDSS_WR	0x45D23800	1	No	No	Yes	Yes	No	Yes
MMCSD2_EMMCSDSS_RD	0x45D23C00	1	No	No	Yes	Yes	No	Yes

3.7 Interconnect Debugging Feature

Interconnect routes the transaction to null end point for graceful termination under the following situation:

- There is no physical connection between initiator and target end points
- Or target interface is in disabled state
- Or the transaction is sent to an unpopulated address

The null end point returns the error status back to the initiator to prevent the interconnect hang. At the same time, the interconnect module logs this transaction information in err_reg region and issues an error interrupt (default_err_intr). This default_err_intr is sent to all of the application processors for debug purposes.

If the transaction is sent to a valid target interface, however the firewall in front of the target interface can prohibit the transaction, the transaction will be logged in the glb_regs region and the interconnect will assert a default_exp interrupt.

3.8 Quality of Service (QoS) Block

CBASS arbitrates the transaction based on priority from high to low. If the transactions going to the same end point have the same priority, then the CBASS uses round robin method for these transactions.

In the CBASS with parallel paths, orderID is used to determine how the transactions are forwarded to those parallel paths. The parallel paths are defined as two target interfaces with the exact same memory map assignment.

By default, all the target interfaces have a certain address assigned and CBASS routes the transactions based on this address. However, some of the special target interfaces can be routed through a different mechanism using the ASEL signal, such as MPU A53SS's ACP interface and PCIe target interfaces.

By default, the majority of the transactions default to the lowest priority level, ASEL value to zero and orderID to zero. The transactions from BCDMA and pktDMA have those attributes as part of the DMA configuration. ASEL value is part of the address information passed to the DMA engine. It is normally specified in descriptors such as host packet descriptor and host buffer descriptor for pktDMA, and Transfer Request (TR) packet descriptor for BCDMA. The priority and orderID information can be configured through each DMA channel's priority control register.

For the rest of the initiators, the CBASS IP provides QoS block to allow the user to program the transaction's attributes such as priority level, ASEL coding and orderID.

Copyright © 2026 Texas Instruments Incorporated

The QoS block programming is through CBASS's qos_regs region. Each initiator, except the BCDMA, pktDMA, and debugss, has an MMR to allow the user to set the priority, ASEL, and orderID value. There is another exception on priority setting. The display IP implements dynamic priority schedule based on the FIFO threshold. The QoS block inside CBASS does not have the ability to overwrite the priority value coming from the display IP.

Section *QoS Summary Table* shows the mapping between initiators and QoS block physical addresses.

Each QoS block has one 4B MAP register per channel that allows the user to program ASEL, orderID, and priority. The first MAP register is at offset 0x100 and the MAP register for channel X is located at offset 0x100 + 400x. See *QoS Block MMR*.

For any write to address range 0x4500_0000 to 0x45FF_FFFF, it is recommended to read back the value after the write to make sure the write landed. The registers in this regions are mainly for firewall configuration, ISC/DMA credential configuration, QoS MMR configuration.

On QoS MMR configuration, in order to configure certain transaction to be the highest priority, the priority field needs to be set to zero for that transaction. In order to make sure that priority field is indeed to set to zero, the following sequence should be followed:

1. Read the priority field, if it is non-zero value, follow a write to overwrite priority field to be 0x0.
2. If read back priority field is zero, write to QoS MMR register to set priority field to be 0x7. Read back the same priority field. If the read back value is 0x7, write to QoS MMR register to set priority field to be 0x0. If the read back value is 0x0, it means this register is not implemented and priority setting is not working.

Copyright © 2026 Texas Instruments Incorporated

3.8.1 QoS Summary Table

Table 3-18. QoS Summary Table

Initiator	QoS Index	Channel Count	Virtual ID Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable	QoS Block Physical Address
WKUP_R5FSS0_CPU0_RMST	80	1	N	Y	Y	N	Y	0x45D14000
WKUP_R5FSS0_CPU0_WMST	81	1	N	Y	Y	N	Y	0x45D14400
WKUP_R5FSS0_CPU0_PMST	82	1	N	Y	Y	N	Y	0x45D14800
R5FSS0_CPU0_RMST	84	1	N	Y	Y	N	Y	0x45D15000
R5FSS0_CPU0_WMST	85	1	N	Y	Y	N	Y	0x45D15400
R5FSS0_CPU0_PMST	86	1	N	Y	Y	N	Y	0x45D15800
MCU_R5FSS0_CPU0_RMST	96	1	N	Y	Y	N	Y	0x45D18000
MCU_R5FSS0_CPU0_WMST	97	1	N	Y	Y	N	Y	0x45D18400
MCU_R5FSS0_CPU0_PMST	98	1	N	Y	Y	N	Y	0x45D18800
COMPUTE_CLUSTER0_A53_QUAD_WRAP_CBA_AXI_R	129	1	N	Y	Y	N	Y	0x45D20400
COMPUTE_CLUSTER0_A53_QUAD_WRAP_CBA_AXI_W	130	1	N	Y	Y	N	Y	0x45D20800
DEBUGSS_WRAP0_VBUSMW	134	1	N	Y	Y	N	Y	0x45D21800
DEBUGSS_WRAP0_VBUSMR	135	1	N	Y	Y	N	Y	0x45D21C00
GICSS0_MEM_WR_VBUSM	136	1	N	Y	Y	N	Y	0x45D22000
GICSS0_MEM_RD_VBUSM	137	1	N	Y	Y	N	Y	0x45D22400
MMCSD1_EMMCSDSS_RD	140	1	N	Y	Y	N	Y	0x45D23000
MMCSD1_EMMCSDSS_WR	141	1	N	Y	Y	N	Y	0x45D23400
MMCSD2_EMMCSDSS_WR	142	1	N	Y	Y	N	Y	0x45D23800
MMCSD2_EMMCSDSS_RD	143	1	N	Y	Y	N	Y	0x45D23C00
USB1_MSTR0	146	8	N	Y	Y	N	Y	0x45D24800
USB1_MSTW0	147	8	N	Y	Y	N	Y	0x45D24C00
SA3_SS0_CTXCACH_EXT_DMA	149	1	N	Y	Y	N	Y	0x45D25400
JPGENC0_M_VBUSM_W	150	1	N	Y	Y	N	Y	0x45D25800
JPGENC0_M_VBUSM_R	151	2	N	Y	Y	N	Y	0x45D25C00
CODEC0_PRI_M_VBUSM_R_ASYNC	154	5	N	Y	Y	N	Y	0x45D26800
CODEC0_PRI_M_VBUSM_W_ASYNC	155	5	N	Y	Y	N	Y	0x45D26C00
CODEC0_SEC_M_VBUSM_R_ASYNC	156	1	N	Y	Y	N	Y	0x45D27000
CODEC0_SEC_M_VBUSM_W_ASYNC	157	1	N	Y	Y	N	Y	0x45D27400
C7X256V0_C7XV_SOC	158	2	N	Y	N	N	N	0x45D27800
VPAC0_LDC0_M_MST	160	3	N	Y	Y	N	Y	0x45D28000
PCIE0_PCIE_MST_RD	164	8	N	Y	Y	N	Y	0x45D29000
PCIE0_PCIE_MST_WR	165	8	N	Y	Y	N	Y	0x45D29400

Copyright © 2026 Texas Instruments Incorporated

Table 3-18. QoS Summary Table (continued)

Initiator	QoS Index	Channel Count	Virtual ID Capable	Order ID Capable	Transaction Priority Capable	Address Type Capable	Asel Capable	QoS Block Physical Address
GPU0_M_VBUSM_W_SYNC	166	32	N	Y	Y	N	Y	0x45D29800
GPU0_M_VBUSM_R_SYNC	168	32	N	Y	Y	N	Y	0x45D2A000
C7X256V1_C7XV_SOC	176	2	N	Y	N	N	N	0x45D2C000
DSS0_VBUSM_DMA	192	4	N	Y	N	N	Y	0x45D30000
DSS1_VBUSM_DMA	193	4	N	Y	N	N	Y	0x45D30400
USB0_MSTR0	208	1	N	Y	Y	N	Y	0x45D34000
USB0_MSTW0	209	1	N	Y	Y	N	Y	0x45D34400
MMCSD0_EMMCSS_WR	210	1	N	Y	Y	N	Y	0x45D34800
MMCSD0_EMMCSS_RD	211	1	N	Y	Y	N	Y	0x45D34C00

Copyright © 2026 Texas Instruments Incorporated

3.9 Route ID

The route ID is used by the interconnect to route return status and data back to the transaction initiator. Each initiator has one or multiple unique route IDs assigned to it. The route ID has no other usage. There are no software controls for the route ID. The initiator-to-route-ID assignment is shown in Table 3-19.

3.9.1 Route ID Table

Table 3-19. Route ID Table

Module Name	Initiator Interface	Which CBASS it is connected to	Route ID	Physical Address
COMPUTE_CLUSTER0	A53_QUAD_WRAP_CBA_AXI_W	CBASS0	0 - 7	0x45D20800
COMPUTE_CLUSTER0	A53_QUAD_WRAP_CBA_AXI_R	CBASS0	16 - 23	0x45D20400
C7X256V0	C7XV_SOC	CBASS0	32 - 47	0x45D27800
C7X256V1	C7XV_SOC	CBASS0	48 - 63	0x45D2C000
GPU0	M_VBUSM_W_SYNC	CBASS0	64	0x45D29800
GPU0	M_VBUSM_R_SYNC	CBASS0	65	0x45D2A000
JPGENC0	M_VBUSM_W	CBASS0	66	0x45D25800
JPGENC0	M_VBUSM_R	CBASS0	67	0x45D25C00
DSS0	VBUSM_DMA	CBASS_RT_DATA0	160	0x45D30000
DSS1	VBUSM_DMA	CBASS_RT_DATA0	161	0x45D30400
MMCSD0	EMMCSS_WR	CBASS_RT_CFG0	256	0x45D34800
MMCSD0	EMMCSS_RD	CBASS_RT_CFG0	257	0x45D34C00
DEBUGSS_WRAP0	VBUSMR	CBASS0	258	0x45D21C00
DEBUGSS_WRAP0	VBUSMW	CBASS0	259	0x45D21800
PCIE0	PCIE_MST_RD	CBASS0	267	0x45D29000
PCIE0	PCIE_MST_WR	CBASS0	268	0x45D29400
MMCSD1	EMMCSDSS_WR	CBASS0	269	0x45D23400
MMCSD1	EMMCSDSS_RD	CBASS0	270	0x45D23000
MMCSD2	EMMCSDSS_WR	CBASS0	271	0x45D23800
MMCSD2	EMMCSDSS_RD	CBASS0	272	0x45D23C00
SA3_SS0	CTXCACH_EXT_DMA	CBASS0	277	0x45D25400
GICSS0	MEM_WR_VBUSM	CBASS0	302	0x45D22000
GICSS0	MEM_RD_VBUSM	CBASS0	303	0x45D22400
USB0	MSTR0	CBASS_RT_CFG0	304	0x45D34000
USB0	MSTW0	CBASS_RT_CFG0	305	0x45D34400
USB1	MSTR0	CBASS0	306	0x45D24800
USB1	MSTW0	CBASS0	307	0x45D24C00
CODEC0	PRI_M_VBUSM_R_ASYNC	CBASS0	310	0x45D26800
CODEC0	PRI_M_VBUSM_W_ASYNC	CBASS0	311	0x45D26C00
CODEC0	SEC_M_VBUSM_R_ASYNC	CBASS0	312	0x45D27000
CODEC0	SEC_M_VBUSM_W_ASYNC	CBASS0	313	0x45D27400
VPAC0	LDC0_M_MST	CBASS0	314	0x45D28000
R5FSS0	CPU0_RMST	WKUP_CBASS0	4084	0x45D15000
R5FSS0	CPU0_WMST	WKUP_CBASS0	4085	0x45D15400
R5FSS0	CPU0_PMST	WKUP_CBASS0	4086	0x45D15800
WKUP_R5FSS0	CPU0_RMST	WKUP_CBASS0	4088	0x45D14000
WKUP_R5FSS0	CPU0_WMST	WKUP_CBASS0	4089	0x45D14400
WKUP_R5FSS0	CPU0_PMST	WKUP_CBASS0	4090	0x45D14800

Copyright © 2026 Texas Instruments Incorporated

Table 3-19. Route ID Table (continued)

Module Name	Initiator Interface	Which CBASS it is connected to	Route ID	Physical Address
MCU_R5FSS0	CPU0_RMST	MCU_CBASS0	4092	0x45D18000
MCU_R5FSS0	CPU0_WMST	MCU_CBASS0	4093	0x45D18400
MCU_R5FSS0	CPU0_PMST	MCU_CBASS0	4094	0x45D18800

3.10 ISC and Firewall

There are two basic building blocks to achieve security isolation on the SoC level: firewall block and ISC block.

3.10.1 Initiator-Side Security Controls (ISC)

The purpose of the ISC block is to provide a mechanism to either add or overwrite the attributes of the transaction, such as privID, priv, and secure. Each transaction in the system carries those attributes, either driven internally by the IP, or added/modified by the ISC block for that initiator.

All the transactions coming from pktDMA, BCDMA, and DMSS_crypto, the privID, priv, and secure attributes are driven internally by the DMA engine. There is no ISC block to override the values for these transactions. For the transactions from pktDMA, BCDMA, and XLCDMA, the secure, priv, and privID values are determined by the credential register for each DMA flow.

Transactions from all other initiators, attributes can be added or overridden by the ISC block.

The majority of the ISC blocks only contain one region, therefore all the transactions through that ISC block share the same ISC setting. There are a few ISC blocks supporting either multiple channels or multiple regions. The default setting for those ISC blocks with multiple channels, or multiple regions, are the same. However, the transaction attributes can be modified in a finer granularity by utilizing the multiple channel/region mode. If the channel mode is enabled, the starting address and the end address MMRs should be programmed to the starting and end channel number for that ISC region. If the region mode is enabled, the starting and end address MMRs are the address range the transactions are sent to.

Table 3-20 shows the default setting for privID/priv setting. It is recommended not to change those settings except some specific conditions.

3.10.1.1 ISC and Priv-ID

Table 3-20. ISC and Priv-ID

Default Priv ID	Inititator	ISC Interface	ISC ID	MMR Base Address	Region Count
4	COMPUTE_CLUSTER0	A53_QUAD_WRAP_CBA_AXI_R	129	0x45820400	1
4	COMPUTE_CLUSTER0	A53_QUAD_WRAP_CBA_AXI_W	130	0x45820800	1
32	C7X256V0	C7XV_SOC	158	0x45827800	2
33	C7X256V1	C7XV_SOC	176	0x4582C000	2
96	MCU_R5FSS0	CPU0_RMST	96	0x45818000	4
96	MCU_R5FSS0	CPU0_WMST	97	0x45818400	4
96	MCU_R5FSS0	CPU0_PMST	98	0x45818800	4
101	DMASS1	BCDMA_MEMR	194	0x45830800	1
101	DMASS1	BCDMA_MEMW	195	0x45830C00	1
102	VPAC0	DATA_MST_0	159	0x45827C00	1
103	VPAC0	DATA_MST_1	161	0x45828400	1
104	VPAC0	LDC0_M_MST	160	0x45828000	1
105	DMPAC0	DATA_MST	177	0x4582C400	32
128	MMCSD0	EMMCSS_WR	210	0x45834800	1
128	MMCSD0	EMMCSS_RD	211	0x45834C00	1
129	MMCSD1	EMMCSDSS_RD	140	0x45823000	1

Copyright © 2026 Texas Instruments Incorporated

Table 3-20. ISC and Priv-ID (continued)

Default Priv ID	Inititator	ISC Interface	ISC ID	MMR Base Address	Region Count
129	MMCSD1	EMMCSDSS_WR	141	0x45823400	1
130	MMCSD2	EMMCSDSS_WR	142	0x45823800	1
130	MMCSD2	EMMCSDSS_RD	143	0x45823C00	1
152	SA3_SS0	CTXCACH_EXT_DMA	149	0x45825400	1
154	GICSS0	MEM_WR_VBUSM	136	0x45822000	1
154	GICSS0	MEM_RD_VBUSM	137	0x45822400	1
155	USB0	MSTR0	208	0x45834000	1
155	USB0	MSTW0	209	0x45834400	1
156	usb2ss_16ffc_main_1	MSTR0	212	0x45835000	1
156	usb2ss_16ffc_main_1	MSTW0	213	0x45835400	1
157	USB1	MSTR0	146	0x45824800	8
157	USB1	MSTW0	147	0x45824C00	8
173	DSS0	VBUSM_DMA	192	0x45830000	4
174	DSS1	VBUSM_DMA	193	0x45830400	4
177	DEBUGSS_WRAP0	VBUSMW	134	0x45821800	1
177	DEBUGSS_WRAP0	VBUSMR	135	0x45821C00	1
179	PCIE0	PCIE_MST_RD	164	0x45829000	8
179	PCIE0	PCIE_MST_WR	165	0x45829400	8
187	GPU0	M_VBUSM_W_SYNC	166	0x45829800	32
187	GPU0	M_VBUSM_R_SYNC	168	0x4582A000	32
190	JPGENC0	M_VBUSM_W	150	0x45825800	1
190	JPGENC0	M_VBUSM_R	151	0x45825C00	2
191	CODEC0	PRI_M_VBUSM_R_ASYNC	154	0x45826800	5
191	CODEC0	PRI_M_VBUSM_W_ASYNC	155	0x45826C00	5
191	CODEC0	SEC_M_VBUSM_R_ASYNC	156	0x45827000	1
191	CODEC0	SEC_M_VBUSM_W_ASYNC	157	0x45827400	1
212	WKUP_R5FSS0	CPU0_RMST	80	0x45814000	4
212	WKUP_R5FSS0	CPU0_WMST	81	0x45814400	4
212	WKUP_R5FSS0	CPU0_PMST	82	0x45814800	4
214	R5FSS0	CPU0_RMST	84	0x45815000	4
214	R5FSS0	CPU0_WMST	85	0x45815400	4
214	R5FSS0	CPU0_PMST	86	0x45815800	4

3.10.1.2 ISC Registers

CBASS ISC Registers are described in Table 3-21.

Each ISC block contains a set of MMRs for each region/channel: Control MMR, start address, and end address. If an ISC block is configured to have x region/channels, this ISC block contains x+1 sets of MMRs.

Table 3-21. ISC Registers

Address Offset	Register Mnemonic	Register Name
(b*0x20), b=0..31	REGION[b]_CONTROL	ISC a Region b Control Register
(b*0x20)+0x10, b=0..31	REGION[b]_START_ADDR_L	ISC a Region b Start Address Low Register
(b*0x20)+0x14, b=0..31	REGION[b]_START_ADDR_H	ISC a Region b Start Address High Register
(b*0x20)+0x18, b=0..31	REGION[b]_END_ADDR_L	ISC a Region b End Address Low Register

Copyright © 2026 Texas Instruments Incorporated

Table 3-21. ISC Registers (continued)

Address Offset	Register Mnemonic	Register Name
(b*0x20)+0x1c, b=0..31	REGION[b]_END_ADDR_H	ISC a Region b End Address High Register

Each ISC can be configured in one of two modes: region mode and channel mode.

When the region mode is configured, the start and end address MMRs shall be programmed with an SoC address range.

When ISC is configured as channel mode, the start and end address MMRs shall be programmed using channel number.

Each region and channel can be configured to determine whether to overwrite or add values for priv, privID, and secure attributes. A default value for the control MMR, located at offset x*0x20, is used so that transactions do not map to a valid region.

The following is the SW sequence to modify the default ISC setting for region/channel b:

- If channel mode is needed, set ch_mode field to 0x1. Program the start and end address MMRs to be the channel ID
- If region mode is needed, set ch_mode field to be 0x0. Program the start and the end address MMRs to match the address range.
- Set enable field to be 0xA
- If lock field is set to 0x1, then this control MMR can't be changed until it is reset to 0x0
- PrivID: if the transaction carries privID attribute, ISC can either direct use the privID attribute from the transaction by setting the pass field to 0x0, or overwrite it using the new value define in this control register by setting the pass field to be 0x1. If the transaction does not have privID attribute, the ISC will add the privID to this transaction, regardless of the value in the pass field.
- Secure attribute: the transaction can either be forced to be non-secure mode by setting nonsec field to be 0x1, or force the transaction to be secure by setting sec field to be 0xA, or just pass through the secure attribute if the original transaction carries secure attribute. For an initiator which does not provide secure attributes, it is default set to non-secure.
- Priv attribute: by default priv attribute is not set. ISC can be used to overwrite priv attribute either to be user mode by setting nopri field to 0xF, or supervisor mode by setting priv field to be 0xF

3.10.2 Transaction Attributes for BCDMA and PktDMA Transactions

Quality of Service (QoS) Block and Initiator-Side Security Controls (ISC) provide information on how to set up the attributes on the generic transaction. But the BCDMA and pktDMA transactions do not rely on the ISC block and QoS block to set the transaction attributes. Instead, those attributes are determined by the various DMA configuration registers:

Table 3-22. BCDMA and PktDMA Transaction Attributes

	How to configure BCDMA/PktDMA Transactions	Link
ASEL	Descriptors: host packet descriptor and host buffer descriptor for pktDMA and Transfer Request (TR) descriptors for BCDMA	See DMASS Transfer Request Descriptor
priority	Channel priority control register	See Channel Priority Control Register
orderID	Channel priority control register	
secure	Flow credential registers	See Flow Credential Register
priv	Flow credential registers	
privID	Flow credential registers	
atype	Channel configuration MMR. Fixed to 2b'00.	

3.10.3 Firewall Block

There are two types of firewall: region based firewall and channelized firewall.

Copyright © 2026 Texas Instruments Incorporated

The channelized firewall is only implemented to protect the registers which control the power domain and LPSC for security enclave. User can’t modify the channelized firewall setting.

Region based firewalls are inserted at the target end point in the interconnect. Normally a group of the target interfaces share one region-based firewall. Each region-based firewall has its unique configuration of:

- FWID: this is a unique index of firewall block.
- Number of protected regions: this is to indicate how many protected regions users can define and program
- Memory regions behind this firewall: transactions sent to those memory regions will go through this firewall protection

Majority of the memory regions in the SoC are protected by region-based firewalls, except the public boot ROM, debug cell, and STM.

Region Based Firewalls shows the firewall summary on SoC.

Copyright © 2026 Texas Instruments Incorporated

3.10.3.1 Region Based Firewalls

Table 3-23. Region Based Firewalls

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	DDR32SS0	0	3	16	0x45000000	DDR32SS0	SDRAM	0x80000000	0xFFFFFFFF	No
CBASS0	DDR32SS0	0	3	16	0x45000000	DDR32SS0	SDRAM	0x880000000	0x8FFFFFFFF	No
CBASS0	DDR32SS0	0	3	16	0x45000000	DDR32SS0	SDRAM	0x900000000	0x9FFFFFFFF	No
CBASS0	DDR32SS0	0	3	16	0x45000000	DDR32SS0	SDRAM	0xA00000000	0xBFFFFFFFF	No
CBASS0	DDR32SS0	0	3	16	0x45000000	DDR32SS0	SDRAM	0xC00000000	0xFFFFFFFFF	No
CBASS0	DDR32SS0	1	3	16	0x45000400	DDR32SS0	SDRAM	0x80000000	0xFFFFFFFF	No
CBASS0	DDR32SS0	1	3	16	0x45000400	DDR32SS0	SDRAM	0x880000000	0x8FFFFFFFF	No
CBASS0	DDR32SS0	1	3	16	0x45000400	DDR32SS0	SDRAM	0x900000000	0x9FFFFFFFF	No
CBASS0	DDR32SS0	1	3	16	0x45000400	DDR32SS0	SDRAM	0xA00000000	0xBFFFFFFFF	No
CBASS0	DDR32SS0	1	3	16	0x45000400	DDR32SS0	SDRAM	0xC00000000	0xFFFFFFFFF	No
CBASS0	COMPUTE_CLUSTER0	2	3	8	0x45000800	COMPUTE_CLUSTER0				Yes
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	ROM_TABLE_0_0	0x700000000	0x700000FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV0_0	0x700001000	0x700001FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CFGAP0	0x700002000	0x7000020FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	APBAP0	0x700002100	0x7000021FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	AXIAP0	0x700002200	0x7000022FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	PWRAP0	0x700002300	0x7000023FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	PVIEW0	0x700002400	0x7000024FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	JTAGAP0	0x700002500	0x7000025FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	SECAP0	0x700002600	0x7000026FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX0_CFG0	0x700002700	0x7000027FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX1_CFG0	0x700002800	0x7000028FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX2_CFG0	0x700002900	0x7000029FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX3_CFG0	0x700002A00	0x700002AFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX4_CFG0	0x700002B00	0x700002BFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX5_CFG0	0x700002C00	0x700002CFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX6_CFG0	0x700002D00	0x700002DFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX7_CFG0	0x700002E00	0x700002EFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX8_CFG0	0x700002F00	0x700002FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV1_0	0x700003000	0x700003FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV2_0	0x700004000	0x702003FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	ROM_TABLE_1_0	0x720000000	0x720000FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CSCTI0	0x720001000	0x720001FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	DRM0	0x720002000	0x720002FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV3_0	0x720003000	0x720003FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CSTPIU0	0x720004000	0x720004FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CTF0	0x720005000	0x720005FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV4_0	0x720006000	0x721005FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	EXT_APB0	0x730000000	0x73FFFFFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	ROM_TABLE_0_1	0x740000000	0x740000FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV0_1	0x740001000	0x740001FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CFGAP1	0x740002000	0x7400020FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	APBAP1	0x740002100	0x7400021FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	AXIAP1	0x740002200	0x7400022FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	PWRAP1	0x740002300	0x7400023FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	PVIEW1	0x740002400	0x7400024FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	JTAGAP1	0x740002500	0x7400025FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	SECAP1	0x740002600	0x7400026FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX0_CFG1	0x740002700	0x7400027FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX1_CFG1	0x740002800	0x7400028FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX2_CFG1	0x740002900	0x7400029FF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX3_CFG1	0x740002A00	0x740002AFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX4_CFG1	0x740002B00	0x740002BFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX5_CFG1	0x740002C00	0x740002CFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX6_CFG1	0x740002D00	0x740002DFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX7_CFG1	0x740002E00	0x740002EFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CORTEX8_CFG1	0x740002F00	0x740002FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV1_1	0x740003000	0x740003FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV2_1	0x740004000	0x742003FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	ROM_TABLE_1_1	0x760000000	0x760000FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CSCTI1	0x760001000	0x760001FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	DRM1	0x760002000	0x760002FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV3_1	0x760003000	0x760003FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CSTPIU1	0x760004000	0x760004FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	CTF1	0x760005000	0x760005FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	RESV4_1	0x760006000	0x761005FFF	No
CBASS0	DEBUGSS_WRAP0	3	3	4	0x45000C00	DEBUGSS_WRAP0	EXT_APB1	0x770000000	0x77FFFFFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	UMC	0x7C000000	0x7C01FFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CLEC	CLEC	0x7C200000	0x7C2FFFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU	0x7C400000	0x7C403FFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_SET	0x7C404000	0x7C407FFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_QUEUE	0x7C408000	0x7C40FFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_CHNRT	0x7C440000	0x7C45FFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_CHRT	0x7C460000	0x7C47FFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_CHATOMIC_DEBUG	0x7C480000	0x7C49FFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	DRU_CHCORE	0x7C4A0000	0x7C4BFFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	dru	0x7C4E0000	0x7C4FFFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	UMC_MEM_MAIN	0x7E000000	0x7E1FFFFF	No
CBASS0	C7X256V0_CORE0	4	3	8	0x45001000	C7X256V0_CORE0	UMC_MEM_AUX	0x7F000000	0x7F07FFFF	No
CBASS0	VPAC0	5	3	8	0x45001400	VPAC0	MEM_SLV_DATA	0x71000000	0x7107FFFF	No
CBASS0	GPMC0	6	3	8	0x45001800	GPMC0	CFG	0x3B000000	0x3B0003FF	No
CBASS0	GPMC0	6	3	8	0x45001800	GPMC0	DATA	0x50000000	0x57FFFFFF	No
CBASS0	FSS0_FSAS_0	7	3	8	0x45001C00	FSS0_FSAS_0	DAT_REG0	0x400000000	0x4FFFFFFFF	No
CBASS0	FSS0_FSAS_0	7	3	8	0x45001C00	FSS0_FSAS_0	DAT_REG3	0x500000000	0x5FFFFFFFF	No
CBASS0	FSS0_FSAS_0	7	3	8	0x45001C00	FSS0_FSAS_0	DAT_REG1	0x60000000	0x67FFFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_ECC_AGGR_0	ECCAGGR	0x3F005000	0x3F0053FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_INTR	0x48000000	0x480FFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_IMAP	0x48100000	0x48103FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_CFG	0x48110000	0x4811001F	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_L2G	0x48120000	0x481203FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PSILCFG_0	PSILCFG_PROXY	0x48130000	0x481301FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PSILSS_0	PSILSS_MMRS	0x48140000	0x48140FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_UNMAP	0x48180000	0x4819FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_MCAST	0x48210000	0x48210FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_GCNTCFG	0x48220000	0x48221FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0	ETLSW_MMRS	0x48230000	0x48230FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_RINGACC_0	RINGACC_GCFG	0x48240000	0x482403FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_SEC_PROXY_0	SEC_PROXY_MMRS	0x48250000	0x482500FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_BCHAN	0x48420000	0x48421FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_RFLOW	0x48430000	0x48430FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_TCHAN	0x484A0000	0x484A1FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_TCHAN	0x484A4000	0x484A5FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_RCHAN	0x484C0000	0x484C1FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_RCHAN	0x484C2000	0x484C3FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_GCFG	0x485C0000	0x485C00FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_GCFG	0x485C0100	0x485C01FF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_RING	0x485E0000	0x485EFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_RING	0x48600000	0x48607FFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_RINGACC_0	RINGACC_RT	0x49000000	0x493FFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_RINGACC_0	RINGACC_CFG	0x49800000	0x4983FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_INTAGGR_0	INTAGGR_GCNTRTI	0x4A000000	0x4A0FFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_SEC_PROXY_0	SEC_PROXY_SCFG	0x4A400000	0x4A47FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_SEC_PROXY_0	SEC_PROXY_RT	0x4A600000	0x4A67FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_RCHANRT	0x4A800000	0x4A81FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_RCHANRT	0x4A820000	0x4A83FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_TCHANRT	0x4AA00000	0x4AA1FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_TCHANRT	0x4AA40000	0x4AA5FFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_PKTDMA_0	PKTDMA_RINGRT	0x4B800000	0x4B9FFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_RINGRT	0x4BC00000	0x4BCFFFFF	No
CBASS0	DMASS0_ECC_AGGR_0	9	3	32	0x45002400	DMASS0_BCDMA_0	BCDMA_BCHANRT	0x4C000000	0x4C01FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	J7AEN_MAIN_DBG_CBASS0	ERR	0x00200000	0x002003FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	GPU0	MEM	0x00310000	0x003103FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	COMPUTE_CLUSTER0_PBIST_0	PBIST	0x00330000	0x003303FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	PBIST3	MEM	0x00340000	0x003403FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	VPAC0	MEM	0x00350000	0x003503FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V0_PBIST	PBIST	0x00360000	0x003603FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V1_PBIST	PBIST	0x00370000	0x003703FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	PBIST2	MEM	0x00380000	0x003803FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53SS0	SS_ECC_AGGR	0x00718000	0x007183FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53SS0	CORE0_ECC_AGGR	0x00718400	0x007187FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53SS0	CORE1_ECC_AGGR	0x00718800	0x00718BFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53SS0	CORE2_ECC_AGGR	0x00718C00	0x00718FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53SS0	CORE3_ECC_AGGR	0x00719000	0x007193FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V0_ECC_AGGR	ECC_AGGR	0x0071A000	0x0071A3FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V1_ECC_AGGR	ECC_AGGR	0x0071B000	0x0071B3FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER0	CFG	0x02400000	0x024003FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER1	CFG	0x02410000	0x024103FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER2	CFG	0x02420000	0x024203FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER3	CFG	0x02430000	0x024303FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER4	CFG	0x02440000	0x024403FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER5	CFG	0x02450000	0x024503FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER6	CFG	0x02460000	0x024603FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	TIMER7	CFG	0x02470000	0x024703FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI0	CFG	0x0E000000	0x0E0000FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI1	CFG	0x0E010000	0x0E0100FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI2	CFG	0x0E020000	0x0E0200FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI3	CFG	0x0E030000	0x0E0300FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI4	CFG	0x0E040000	0x0E0400FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI5	CFG	0x0E050000	0x0E0500FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	RTI15	CFG	0x0E0F0000	0x0E0F00FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	SERDES_10G0	WIZ16B2M4CT	0x0F000000	0x0F00FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	SERDES_10G1	WIZ16B2M4CT	0x0F010000	0x0F01FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	DDR32SS0	SS_CFG	0x0F300000	0x0F3001FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	DDR32SS0	CTLPHY_WRAP_CTL_CFG_CTLCFG	0x0F308000	0x0F30FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	USB1	MMR_MMRVBP_USBSS_CMN	0x0F920000	0x0F9200FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	USB1	RAMS_INJ_CFG	0x0F921000	0x0F9213FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	USB1	PHY2	0x0F928000	0x0F9283FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	USB1	ECC_AGGR	0x0F990000	0x0F9903FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	JPGENC0	CORE	0x0FD20000	0x0FD200FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	JPGENC0	CORE_MMU	0x0FD20200	0x0FD203FF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	GPU0	CORE_MMRS	0x0FD80000	0x0FDFFFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	ELM0	MEM	0x25010000	0x25010FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	CODEC0	VPU	0x30210000	0x3021FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V0_CORE0	C7XV_CFG_MMR	0x30240000	0x3024FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7X256V1_CORE0	C7XV_CFG_MMR	0x30250000	0x3025FFFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	VPAC_RSWS_BW_LIMITER7	REGS	0x30400000	0x30400FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	CODEC_WS_BW_LIMITER3	REGS	0x30401000	0x30401FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53_WS_BW_LIMITER1	REGS	0x30402000	0x30402FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	A53_RS_BW_LIMITER0	REGS	0x30403000	0x30403FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	JPGENC_RS_BW_LIMITER4	REGS	0x30404000	0x30404FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	JPGENC_WS_BW_LIMITER5	REGS	0x30405000	0x30405FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7XV_RSWS_BS_LIMITER6	REGS	0x30406000	0x30406FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	VPAC_RSWS_BW_LIMITER8	REGS	0x30407000	0x30407FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	CODEC_RS_BW_LIMITER2	REGS	0x30408000	0x30408FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	GPU_WS_BW_LIMITER10	REGS	0x30409000	0x30409FFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	GPU_RS_BW_LIMITER9	REGS	0x3040A000	0x3040AFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	C7XV_RSWS_BS_LIMITER11	REGS	0x3040B000	0x3040BFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	USB1	VBP2APB_WRAP_CONTROLLER_VBP_CORE_ADDR_MAP	0x31200000	0x3123FFFF	No
CBASS0	J7AEN_MAIN_DBG_CBASS0	10	3	16	0x45002800	PBIST0	MEM	0x3F110000	0x3F1103FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	CBASS_MCASP0	ERR	0x00240000	0x002403FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	CPSW0	ECC	0x00704000	0x007043FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD1	ECC_AGGR_RXMEM	0x00708000	0x007083FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD1	ECC_AGGR_TXMEM	0x00709000	0x007093FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD2	ECC_AGGR_TXMEM	0x0070A000	0x0070A3FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD2	ECC_AGGR_RXMEM	0x0070B000	0x0070B3FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MSRAM8KX256E0	ECC_AGGR_REGS	0x00710000	0x007103FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0_OSPI_0	OSPI0_ECC_AGGR	0x00716000	0x007163FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_ECC_AGGR0	0x0071E000	0x0071E3FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_ECC_AGGR1	0x0071F000	0x0071F3FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	GICSS0	GIC_TRANSLATER	0x01000000	0x013FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	GICSS0	GIC	0x01800000	0x018FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	CPSW0	NUSS	0x08000000	0x081FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	PCIE_RP_CORE	0x0D000000	0x0D7FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_USER_CFG_USER_CFG	0x0F100000	0x0F1003FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_VMAP_OB_MMRS	0x0F101000	0x0F101FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_PCIE_INTD_CFG_INTD_CFG	0x0F102000	0x0F102FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	PCIE0	CORE_CPTS_CFG_CPTS_VBUSP	0x0F103000	0x0F1033FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD1	CTL_CFG	0x0FA00000	0x0FA00FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD1	SS_CFG	0x0FA08000	0x0FA083FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD2	CTL_CFG	0x0FA20000	0x0FA20FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	MMCSD2	SS_CFG	0x0FA28000	0x0FA283FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0	CFG	0x0FC00000	0x0FC000FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0_FSAS_0	FSAS_CFG	0x0FC10000	0x0FC100FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0_FSAS_0	OTFA_CFG	0x0FC20000	0x0FC20FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0_OSPI_0	OSPI0_CTRL	0x0FC40000	0x0FC400FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	FSS0_OSPI_0	OSPI0_SS_CFG	0x0FC44000	0x0FC441FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DMPAC_REGS_DMPAC_REGS_CFG_IP_MMRS	0x10000000	0x100003FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_CP_INTD_CFG_INTD_CFG	0x10001000	0x10001FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_HTS_S_VBUSP	0x10008000	0x1000FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_CTSET2_WRAP_CFG_CTSET2_CFG	0x10020000	0x10021FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DMPAC_FOCO_0_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x10024000	0x1002403F	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DMPAC_FOCO_0_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x10024200	0x100243FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DMPAC_FOCO_1_CFG_SLV_DMPAC_FOCO_CORE_FOCO_REGS_CFG_IP_MMRS	0x10028000	0x1002803F	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DMPAC_FOCO_1_CFG_SLV_VPAC_FOCO_LSE_CFG_VP	0x10028200	0x100283FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_PAR_DOF_CFG_VP_MMR_VBUSP_DOFCORE	0x10080000	0x10080FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_PAR_DOF_CFG_VP_MEM_MMRRAM_VBUSP_MMR_RAM	0x100C0000	0x100FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_PAR_SDE_S_VBUSP_MMR_VBUSP_MMR	0x10100000	0x10100FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_PAR_SDE_S_VBUSP_MEM_MMRRAM_VBUSP_MMR_RAM	0x10140000	0x1017FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU	0x10200000	0x10203FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x10204000	0x10207FFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x10208000	0x1020FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x10240000	0x1025FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x10260000	0x1027FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x10280000	0x1029FFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CHCORE	0x102A0000	0x102BFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_DRU_UTC_DMPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x102E0000	0x102FFFFF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	MEM	0x10800000	0x108003FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	DMPAC0	DMPAC_TOP_CFG_SLV_KSDW_ECC_AGGR_CFG	0x10801000	0x108013FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	CBASS0	ERR	0x3A000000	0x3A0003FF	No
CBASS0	CBASS_MCASP0	11	3	16	0x45002C00	GICSS0	REGS	0x3F004000	0x3F0043FF	No
CBASS0	MCRC64_0	13	3	1	0x45003400	MCRC64_0	REGS	0x30300000	0x30300FFF	No
CBASS0	MCASP0	14	3	8	0x45003800	MCASP0	CFG	0x02B00000	0x02B01FFF	No
CBASS0	MCASP0	14	3	8	0x45003800	MCASP0	DMA	0x02B08000	0x02B083FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	MCASP0	14	3	8	0x45003800	MCASP1	CFG	0x02B10000	0x02B11FFF	No
CBASS0	MCASP0	14	3	8	0x45003800	MCASP1	DMA	0x02B18000	0x02B183FF	No
CBASS0	MCASP0	14	3	8	0x45003800	MCASP2	CFG	0x02B20000	0x02B21FFF	No
CBASS0	MCASP0	14	3	8	0x45003800	MCASP2	DMA	0x02B28000	0x02B283FF	No
CBASS0	MCASP0	14	3	8	0x45003800	EPWM0	EPWM	0x23000000	0x230000FF	No
CBASS0	MCASP0	14	3	8	0x45003800	EPWM1	EPWM	0x23010000	0x230100FF	No
CBASS0	MCASP0	14	3	8	0x45003800	EPWM2	EPWM	0x23020000	0x230200FF	No
CBASS0	MSRAM8KX256E0	15	3	16	0x45003C00	MSRAM8KX256E0	RAM	0x43C40000	0x43C7FFFF	No
CBASS0	MSRAM8KX256E0	15	3	16	0x45003C00	MSRAM8KX256E0	RAM	0x70000000	0x7003FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	UMC	0x7D000000	0x7D01FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CLEC	CLEC	0x7D200000	0x7D2FFFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU	0x7D400000	0x7D403FFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_SET	0x7D404000	0x7D407FFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_QUEUE	0x7D408000	0x7D40FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_CHNRT	0x7D440000	0x7D45FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_CHRT	0x7D460000	0x7D47FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_CHATOMIC_DEBUG	0x7D480000	0x7D49FFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	DRU_CHCORE	0x7D4A0000	0x7D4BFFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	dru	0x7D4E0000	0x7D4FFFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	UMC_MEM_MAIN	0x7E200000	0x7E3FFFFF	No
CBASS0	C7X256V1_CORE0	16	3	8	0x45004000	C7X256V1_CORE0	UMC_MEM_AUX	0x7F800000	0x7F87FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	MMCSD0	ECC_AGGR_RXMEM	0x00706000	0x007063FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	MMCSD0	ECC_AGGR_TXMEM	0x00707000	0x007073FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_TX_IF0	ECC_AGGR_CFG	0x0070C000	0x0070C3FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_TX_IF0	ECC_AGGR_BYTE_CFG	0x0070C400	0x0070C7FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF0	ECC_AGGR_CFG	0x0070E000	0x0070E3FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF1	ECC_AGGR_CFG	0x0070F000	0x0070F3FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF2	ECC_AGGR_CFG	0x00714000	0x007143FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF3	ECC_AGGR_CFG	0x00715000	0x007153FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	DEBUG_TRACE_MMR_TRACE_VBUSP_USB2SS_DEBUG_TRACE	0x0F080000	0x0F0801FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	MMR_MMRVBP_USB2SS_CFG	0x0F900000	0x0F9007FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	PHY2	0x0F908000	0x0F9083FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	ECC_AGGR_CFG	0x0F980000	0x0F9803FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	MMCSD0	17	3	16	0x45004400	MMCSD0	CTL_CFG	0x0FA10000	0x0FA10FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	MMCSD0	SS_CFG	0x0FA18000	0x0FA183FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF0	CP_INTD_CFG_INTD_CFG	0x30100000	0x30100FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF0	VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x30101000	0x30101FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF0	RX_SHIM_VBUSP_MMR_CSI2RXIF	0x30102000	0x30102FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX0	VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x30110000	0x30110FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX0	MMR_SLV_K3_DPHY_WRAP	0x30111000	0x301110FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF1	CP_INTD_CFG_INTD_CFG	0x30120000	0x30120FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF1	VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x30121000	0x30121FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF1	RX_SHIM_VBUSP_MMR_CSI2RXIF	0x30122000	0x30122FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX1	VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x30130000	0x30130FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX1	MMR_SLV_K3_DPHY_WRAP	0x30131000	0x301310FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF2	CP_INTD_CFG_INTD_CFG	0x30140000	0x30140FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF2	VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x30141000	0x30141FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF2	RX_SHIM_VBUSP_MMR_CSI2RXIF	0x30142000	0x30142FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX2	VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x30150000	0x30150FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX2	MMR_SLV_K3_DPHY_WRAP	0x30151000	0x301510FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF3	CP_INTD_CFG_INTD_CFG	0x30160000	0x30160FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF3	VBUS2APB_WRAP_VBUSP_APB_CSI2RX	0x30161000	0x30161FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_RX_IF3	RX_SHIM_VBUSP_MMR_CSI2RXIF	0x30162000	0x30162FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX3	VBUS2APB_WRAP_VBUSP_K3_DPHY_RX	0x30170000	0x30170FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_RX3	MMR_SLV_K3_DPHY_WRAP	0x30171000	0x301710FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_TX_IF0	CP_INTD_CFG_INTD_CFG	0x30180000	0x30180FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_TX_IF0	VBUS2APB_WRAP_VBUSP_APB_CSI2TX_V2	0x30181000	0x30181FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CSI_TX_IF0	TX_SHIM_VBUSP_MMR_CSI2TXIF_V2	0x30182000	0x30182FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DPHY_TX0	WIZ16B8M4CDT3	0x301C0000	0x301C0FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	COMMON	0x30200000	0x30200FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	COMMON1	0x30201000	0x30201FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	VIDL1	0x30202000	0x30202FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	VID	0x30206000	0x30206FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	OVR1	0x30207000	0x30207FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	OVR2	0x30208000	0x30208FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	VP1	0x3020A000	0x3020AFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS0	VP2	0x3020B000	0x3020BFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	COMMON	0x30220000	0x30220FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	COMMON1	0x30221000	0x30221FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	VIDL1	0x30222000	0x30222FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	VID	0x30226000	0x30226FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	OVR1	0x30227000	0x30227FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	OVR2	0x30228000	0x30228FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	VP1	0x3022A000	0x3022AFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS1	VP2	0x3022B000	0x3022BFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS_DSI0	dsi	0x30270000	0x302700FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS_DSI0	DSI_TOP_ECC_AGGR_SYS_CFG	0x30271000	0x302713FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DSS_DSI0	DSI_TOP_VBUSP_CFG_DSI_0_DSI	0x30500000	0x305FFFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_CAP	0x31000000	0x3100001F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_OPER	0x31000020	0x3100005F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_PORT	0x31000420	0x3100043F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_RUNTIME	0x31000440	0x3100045F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_INTR	0x31000460	0x3100049F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DB	0x31000560	0x3100075F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_EXTCAP	0x31000960	0x3100096F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_SUPPRTCAP2	0x31000970	0x3100097F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_SUPPRTCAP3	0x31000980	0x3100099F	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_GBL	0x3100C100	0x3100C8FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEV	0x3100C700	0x3100CEFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_LINK	0x3100D000	0x3100D07F	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEBUG	0x3100D800	0x3100D9FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	USB0	VBP2AHB_WRAP_CONTROLLER_VBP_USB3_CORE_DEBUG_RAM0	0x31040000	0x3104FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CBASS_RT_CFG0	ERR	0x3A010000	0x3A0103FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	CBASS_RT_DATA0	ERR	0x3A020000	0x3A0203FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_ECC_AGGR_0	ECCAGGR	0x3F006000	0x3F0063FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	PBIST1	MEM	0x3F120000	0x3F1203FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_GCNTRTI	0x4E000000	0x4E01FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_UNMAP	0x4E040000	0x4E04FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_MCAST	0x4E080000	0x4E0803FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_GCNTCFG	0x4E090000	0x4E0903FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_IMAP	0x4E0B0000	0x4E0B07FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_CFG	0x4E0C0000	0x4E0C001F	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_RINGRT	0x4E100000	0x4E17FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_RCHANRT	0x4E180000	0x4E19FFFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_RCHAN	0x4E200000	0x4E201FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_RING	0x4E210000	0x4E213FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_PSILSS_0	PSILSS_MMRS	0x4E220000	0x4E220FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_GCFG	0x4E230000	0x4E2300FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_PSILCFG_0	PSILCFG_PROXY	0x4E260000	0x4E2601FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_TCHAN	0x4E280000	0x4E2807FF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_BCDMA_0	BCDMA_TCHANRT	0x4E300000	0x4E307FFF	No
CBASS0	MMCSD0	17	3	16	0x45004400	DMASS1_INTAGGR_0	INTAGGR_INTR	0x4E400000	0x4E43FFFF	No
CBASS0	PCIE0	18	3	8	0x45004800	PCIE0	PCIE_RP_DAT1	0x600000000	0x6FFFFFFFF	Yes
CBASS0	PCIE0	18	3	8	0x45004800	PCIE0	PCIE_RP_DAT0	0x68000000	0x6FFFFFFF	Yes
CBASS0	MCASP3	21	3	8	0x45005400	MCASP3	CFG	0x0FE00000	0x0FE01FFF	No
CBASS0	MCASP3	21	3	8	0x45005400	MCASP3	DMA	0x0FE08000	0x0FE083FF	No
CBASS0	MCASP3	21	3	8	0x45005400	MCASP4	CFG	0x0FE10000	0x0FE11FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS0	MCASP3	21	3	8	0x45005400	MCASP4	DMA	0x0FE18000	0x0FE183FF	No
CBASS0	MCASP3	21	3	8	0x45005400	I2C4	CFG	0x0FE80000	0x0FE800FF	No
CBASS0	MCASP3	21	3	8	0x45005400	ATL0	REG	0x0FEE0000	0x0FEE03FF	No
CBASS0	MCASP3	21	3	8	0x45005400	CBASS_AUDIO0	ERR	0x0FEF0000	0x0FEF03FF	No
CBASS0	DMPAC0	22	3	8	0x45005800	DMPAC0	DMPAC_TOP_DOF_INFRA_DMPAC_BASE_MEM_SLV_CBASS_STRIPE_MSRAM_SLV	0x10400000	0x1047FFFF	No
WKUP_CBASS0	WKUP_R5FSS0_CORE0	32	3	4	0x45008000	WKUP_R5FSS0_CORE0	CORE0_ICACHE	0x74000000	0x747FFFFF	No
WKUP_CBASS0	WKUP_R5FSS0_CORE0	32	3	4	0x45008000	WKUP_R5FSS0_CORE0	CORE0_DCACHE	0x74800000	0x74FFFFFF	No
WKUP_CBASS0	WKUP_R5FSS0_CORE0	32	3	4	0x45008000	WKUP_R5FSS0_CORE0	CORE0_ATCM	0x78000000	0x78007FFF	No
WKUP_CBASS0	WKUP_R5FSS0_CORE0	32	3	4	0x45008000	WKUP_R5FSS0_CORE0	CORE0_BTCM	0x78100000	0x78107FFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSRAMECC0	RAM	0x00000000	0x000003FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PADCFG_CTRL0	CFG0	0x000F0000	0x000F7FFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	CTRL_MMR0	CFG0	0x00100000	0x0011FFFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	CBASS_INFRA1	ERR	0x00210000	0x002103FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	EFUSE0	MEM	0x00300000	0x003000FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSC0	psc	0x00400000	0x00400FFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PLLCTRL0	MEM	0x00410000	0x004101FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	ESM0	CFG	0x00420000	0x00420FFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DFTSS0	MEM	0x00500000	0x005003FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DDPA0	DDPA	0x00580000	0x005803FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	GPIO0	MEM	0x00600000	0x006000FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	GPIO1	MEM	0x00601000	0x006010FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PLL0	CFG	0x00680000	0x0069FFFF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSRAMECC0	REGS	0x00700000	0x007003FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSC0_ECC_AGGR_0	REGS	0x00700400	0x007007FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSRAMECC1	REGS	0x00701000	0x007013FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC0	CFG	0x00800000	0x0080003F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC1	CFG	0x00804000	0x0080403F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC2	CFG	0x00808000	0x0080803F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC3	CFG	0x0080C000	0x0080C03F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC4	CFG	0x00810000	0x0081003F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC5	CFG	0x00814000	0x0081403F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC6	CFG	0x00818000	0x0081803F	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC7	CFG	0x0081C000	0x0081C03F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	DCC8	CFG	0x00820000	0x0082003F	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	PSRAMECC1	RAM	0x00900000	0x009003FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	MAIN_GPIOMUX_INTROUTER0	INTR_ROUTER_CFG	0x00A00000	0x00A007FF	No
WKUP_CBASS0	PSRAMECC0	33	3	16	0x45008400	TIMESYNC_EVENT_INTROUTER0	INTR_ROUTER_CFG	0x00A40000	0x00A403FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_GTC0	CFG0	0x00A80000	0x00A803FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_GTC0	CFG1	0x00A90000	0x00A93FFF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_GTC0	CFG2	0x00AA0000	0x00AA3FFF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_GTC0	CFG3	0x00AB0000	0x00AB3FFF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_VTM0	VTM_CFG1	0x00B00000	0x00B003FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_VTM0	VTM_CFG2	0x00B01000	0x00B013FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_VTM0	ECCAGGR_CFG	0x00B02000	0x00B023FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_RTI0	CFG	0x2B000000	0x2B0000FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_RTCSS0	RTC	0x2B1F0000	0x2B1F007F	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_I2C0	CFG	0x2B200000	0x2B2000FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_UART0	MEM	0x2B300000	0x2B3001FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_CBASS0	ERR	0x2B400000	0x2B4003FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_PBIST0	MEM	0x2B500000	0x2B5003FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_PBIST1	MEM	0x2B510000	0x2B5103FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_ECC_AGGR0	REGS	0x2B600000	0x2B6003FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_ECC_AGGR1	REGS	0x2B601000	0x2B6013FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_PSRAMECC_8K0	REGS	0x2B608000	0x2B6083FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_R5FSS0_COMMON0	EVNT_BUS_VBUSP_MMRS	0x3C018000	0x3C0180FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	R5FSS0_COMMON0	EVNT_BUS_VBUSP_MMRS	0x3C038000	0x3C0380FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	R5FSS0_COMMON0	CORE0_ECC_AGGR	0x3F000000	0x3F0003FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_R5FSS0_COMMON0	CORE0_ECC_AGGR	0x3F00D000	0x3F00D3FF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_PSRAMECC_8K0	RAM	0x41880000	0x41887FFF	No
WKUP_CBASS0	WKUP_GTC0	34	3	16	0x45008800	WKUP_CTRL_MMR0	CFG0	0x43000000	0x4301FFFF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_PSC0	psc	0x04000000	0x04000FFF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	MCU_PLLCTRL0	MEM	0x04020000	0x040201FF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_ECC_AGGR2	REGS	0x04030000	0x040303FF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_PLL0	CFG	0x04040000	0x04040FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	MCU_PADCFG_CTRL0	CFG0	0x04080000	0x04087FFF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_ESM0	CFG	0x04100000	0x04100FFF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	MCU_GPIO0	MEM	0x04201000	0x042010FF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_MCU_GPIOMUX_INTROUTER0	INTR_ROUTER_CFG	0x04210000	0x042101FF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	MCU_CTRL_MMR0	CFG0	0x04500000	0x0451FFFF	No
WKUP_CBASS0	WKUP_PSC0	35	3	16	0x45008C00	WKUP_CBASS_SAFE1	ERR	0x04600000	0x046003FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_TIMEOUT0	CFG	0x04301000	0x043013FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	EVNT_BUS_VBUSP_MMRS	0x04400000	0x044000FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN0	ECC_AGGR	0x04701000	0x047013FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN1	ECC_AGGR	0x04702000	0x047023FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_ECC_AGGR0	REGS	0x04703000	0x047033FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_ECC_AGGR1	REGS	0x04704000	0x047043FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MSRAM_256K0	ECC_AGGR_REGS	0x04705000	0x047053FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MSRAM_256K1	ECC_AGGR_REGS	0x04706000	0x047063FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	CORE0_ECC_AGGR	0x04707000	0x047073FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_CBASS0	ERR	0x04720000	0x047203FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_TIMER0	CFG	0x04800000	0x048003FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_TIMER1	CFG	0x04810000	0x048103FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_TIMER2	CFG	0x04820000	0x048203FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_TIMER3	CFG	0x04830000	0x048303FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_RTI0	CFG	0x04880000	0x048800FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_I2C0	CFG	0x04900000	0x049000FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_UART0	MEM	0x04A00000	0x04A001FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCSPI0	CFG	0x04B00000	0x04B003FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCSPI1	CFG	0x04B10000	0x04B103FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_DCC0	CFG	0x04C00000	0x04C0003F	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_DCC1	CFG	0x04C10000	0x04C1003F	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCRC64_0	REGS	0x04D00000	0x04D00FFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN0	MSGMEM_RAM	0x04E00000	0x04E07FFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN0	CFG	0x04E08000	0x04E081FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN0	SS	0x04E09000	0x04E090FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN1	MSGMEM_RAM	0x04E10000	0x04E17FFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN1	CFG	0x04E18000	0x04E181FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MCAN1	SS	0x04E19000	0x04E190FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_PBIST0	MEM	0x04F00000	0x04F003FF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	CORE0_ICACHE	0x73000000	0x737FFFFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	CORE0_DCACHE	0x73800000	0x73FFFFFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	CORE0_ATCM	0x79000000	0x79007FFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_R5FSS0_CORE0	CORE0_BTCM	0x79020000	0x79027FFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MSRAM_256K0	RAM	0x79100000	0x7913FFFF	No
WKUP_CBASS0	MCU_TIMEOUT0	36	3	8	0x45009000	MCU_MSRAM_256K1	RAM	0x79140000	0x7917FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_KSDW_ECC_AGGR_CFG	0x2B604000	0x2B6043FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x2B605000	0x2B6053FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_KSDW_ECC_AGGR_CFG	0x2B607000	0x2B6073FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_VPAC_REGS_VPAC_REGS_CFG_IP_MMRS	0x2C000000	0x2C0003FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_CTSET2_WRAP_CFG_CTSET2_CFG	0x2C002000	0x2C003FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_CP_INTD_CFG_INTD_CFG	0x2C004000	0x2C004FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_HTS_S_VBUSP	0x2C010000	0x2C01FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MMR_VBUSP	0x2C020000	0x2C0203FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_VPAC_LDC_LSE_CFG_VP	0x2C020400	0x2C0205FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALY_LUTCFG_DUALY_LUT	0x2C020800	0x2C020FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_PIXWRINTF_DUALC_LUTCFG_DUALC_LUT	0x2C021000	0x2C0217FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_MESH_VBUSPI_MESH_MEM	0x2C022000	0x2C023FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_Y_VBUSPI_Y_MEM	0x2C028000	0x2C02FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_LDC0_S_VBUSP_MEMCFG_LOOP_CBCR_VBUSPI_CBCR_MEM	0x2C030000	0x2C037FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_MSC_CFG_VP_CFG_VP	0x2C0C0000	0x2C0C07FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_MSC_CFG_VP_LSE_CFG_VP	0x2C0C0800	0x2C0C09FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_MMR_CFG_VISS_TOP	0x2C100000	0x2C1001FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VPAC_VISS_LSE_CFG_VP	0x2C100400	0x2C1005FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_CFG_GLBCE	0x2C103800	0x2C103FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_K3_GLBCE_TOP_STATMEM_CFG_GLBCE_STATMEM	0x2C104000	0x2C107FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_CFA_VBUSP_FLEXCFA	0x2C108000	0x2C10FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC	0x2C110000	0x2C1107FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC1	0x2C110800	0x2C110FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC2	0x2C111000	0x2C1117FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_CONTRASTC3	0x2C111800	0x2C111FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_Y8R8	0x2C112000	0x2C1127FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_C8G8	0x2C112800	0x2C112FFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_S8B8	0x2C113000	0x2C1137FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_HIST	0x2C113800	0x2C113FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_FCC_VBUSP_FLEXCC_LINE	0x2C118000	0x2C1187FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_MMR_S_VBUSP_RAWFE_CFG	0x2C120000	0x2C1203FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_CFG_RAWFE_H3A_CFG	0x2C120400	0x2C1204FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT3_RAM_RAWFE_PWL_LUT3_RAM	0x2C120800	0x2C120FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT2_RAM_RAWFE_PWL_LUT2_RAM	0x2C121000	0x2C1217FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LUT1_RAM_RAWFE_PWL_LUT1_RAM	0x2C121800	0x2C121FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_WDR_LUT_RAM_RAWFE_WDR_LUT_RAM	0x2C122000	0x2C1227FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_LUT_RAM_RAWFE_H3A_LUT_RAM	0x2C122800	0x2C122FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_RAM_RAWFE_DPC_LUT_RAM	0x2C123000	0x2C1233FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_LRAM_RAWFE_DPC_LRAM	0x2C124000	0x2C125FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_LSC_RAM_RAWFE_LSC_LUT_RAM	0x2C128000	0x2C12FFFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_ARAM_RAWFE_H3A_ARAM	0x2C130000	0x2C131FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_H3A_WRAP_LRAM_RAWFE_H3A_LRAM	0x2C132000	0x2C133FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_RAWFE_CFG_DPC_STATRAM_RAWFE_DPC_STATRAM	0x2C136000	0x2C137FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MMR_VBUSP_NSF4VCORE	0x2C140000	0x2C1407FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_RAWHIST_HISTDATA_VBUSP_RAWHIST	0x2C140800	0x2C1409FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_RAWHIST_HISTLUT_VBUSP_RAWHIST_LUT	0x2C141000	0x2C1417FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_NSF4V_CFG_MEM_MMRRAM_VBUSP_MMR_RAM	0x2C144000	0x2C147FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_EE_VBUSP_FLEXEE	0x2C150000	0x2C157FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_FCP_CFA_VBUSP_FLEXCFA_DLUTS	0x2C158000	0x2C15BFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_MMRCFG_CAC	0x2C180000	0x2C1803FF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_CORE_LUT_CFG_LUT_MEM	0x2C182000	0x2C183FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_CAC_S_VBUSP_LINEMEM_CFG_LINE_MEM	0x2C184000	0x2C187FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_MMRCFG_PCID	0x2C188000	0x2C1883FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_IR_REMAPLUT_LUT_CFG_IRREMAP_LUT	0x2C188800	0x2C188FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_PAR_VPAC_VISS0_S_VBUSP_VISS_PCID_S_VBUSP_CFG_LINEMEM_CFG_LINE_MEM	0x2C18C000	0x2C18FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU	0x2C200000	0x2C203FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_SET	0x2C204000	0x2C207FFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_QUEUE	0x2C208000	0x2C20FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHNRT	0x2C240000	0x2C25FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHRT	0x2C260000	0x2C27FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHATOMIC_DEBUG	0x2C280000	0x2C29FFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CHCORE	0x2C2A0000	0x2C2BFFFF	No
WKUP_CBASS0	VPAC0	37	3	4	0x45009400	VPAC0	IVPAC_TOP_0_CFG_SLV_DRU_UTC_VPAC0_DRU_MMR_CFG_DRU_DRU_CAUSE	0x2C2E0000	0x2C2FFFFF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	PDMA0	REGS	0x00C00000	0x00C003FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	PDMA1	REGS	0x00C01000	0x00C013FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART0	MEM	0x02800000	0x028001FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART1	MEM	0x02810000	0x028101FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART2	MEM	0x02820000	0x028201FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART3	MEM	0x02830000	0x028301FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART4	MEM	0x02840000	0x028401FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART5	MEM	0x02850000	0x028501FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	UART6	MEM	0x02860000	0x028601FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	RTI8	CFG	0x0E0A0000	0x0E0A00FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	I2C0	CFG	0x20000000	0x200000FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	I2C1	CFG	0x20010000	0x200100FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	I2C2	CFG	0x20020000	0x200200FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	I2C3	CFG	0x20030000	0x200300FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCSPI0	CFG	0x20100000	0x201003FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCSPI1	CFG	0x20110000	0x201103FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCSPI2	CFG	0x20120000	0x201203FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	CBASS_MISC_PERI0	ERR	0x201F0000	0x201F03FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN0	SS	0x20700000	0x207000FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN0	CFG	0x20701000	0x207011FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN0	MSGMEM_RAM	0x20708000	0x2070FFFF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN1	SS	0x20710000	0x207100FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN1	CFG	0x20711000	0x207111FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN1	MSGMEM_RAM	0x20718000	0x2071FFFF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	ECAP0	CTL_STS	0x23100000	0x231000FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	ECAP1	CTL_STS	0x23110000	0x231100FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	ECAP2	CTL_STS	0x23120000	0x231200FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	EQEP0	REG	0x23200000	0x232000FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	EQEP1	REG	0x23210000	0x232100FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	EQEP2	REG	0x23220000	0x232200FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN0	ECC_AGGR	0x24018000	0x240183FF	No
WKUP_CBASS0	PDMA0	38	3	16	0x45009800	MCAN1	ECC_AGGR	0x24019000	0x240193FF	No
WKUP_CBASS0	R5FSS0_CORE0	42	3	1	0x4500A800	R5FSS0_CORE0	CORE0_ICACHE	0x76000000	0x767FFFFF	No
WKUP_CBASS0	R5FSS0_CORE0	42	3	1	0x4500A800	R5FSS0_CORE0	CORE0_DCACHE	0x76800000	0x76FFFFFF	No
WKUP_CBASS0	R5FSS0_CORE0	42	3	1	0x4500A800	R5FSS0_CORE0	CORE0_ATCM	0x78400000	0x78407FFF	No
WKUP_CBASS0	R5FSS0_CORE0	42	3	1	0x4500A800	R5FSS0_CORE0	CORE0_BTCM	0x78500000	0x78507FFF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	CBASS_FW0	ERR	0x00220000	0x002203FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	CBASS_IPCSS0	ERR	0x00230000	0x002303FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SA3_SS0_SA_UL_0	ECC_AGGR	0x00712000	0x007123FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_0	REGS0	0x29000000	0x290001FF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_1	REGS1	0x29010000	0x290101FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_2	REGS2	0x29020000	0x290201FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_3	REGS3	0x29030000	0x290301FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_4	REGS4	0x29040000	0x290401FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_5	REGS5	0x29050000	0x290501FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_6	REGS6	0x29060000	0x290601FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	MAILBOX0_MAILBOX_CLUSTER_7	REGS7	0x29070000	0x290701FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SPINLOCK0	REGS	0x2A000000	0x2A007FFF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	ECC_AGGR0	REGS	0x3F00F000	0x3F00F3FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	CBASS_CENTRAL2	ERR	0x3F012000	0x3F0123FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SA3_SS0_SA_UL_0	REGS	0x40900000	0x40900FFF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SA3_SS0_SA_UL_0	MMRA	0x40901000	0x409011FF	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SA3_SS0_SA_UL_0	EIP_76	0x40910000	0x4091007F	No
CBASS_CENTRAL2	CBASS_FW0	66	3	4	0x45010800	SA3_SS0_SA_UL_0	EIP_29T2	0x40920000	0x4092FFFF	No
AM62A_PSC_CHFW_CBASS	PSC0	128	3	5	0x45020000	PSC0	psc	0x00400A6C	0x00400A6F	No
AM62A_PSC_CHFW_CBASS	PSC0	128	3	5	0x45020000	PSC0	psc	0x00400A74	0x00400A77	No
AM62A_PSC_CHFW_CBASS	PSC0	128	3	5	0x45020000	PSC0	psc	0x00400A78	0x00400A7B	No
AM62A_PSC_CHFW_CBASS	PSC0	128	3	5	0x45020000	PSC0	psc	0x00400A7C	0x00400A7F	No
AM62A_PSC_CHFW_CBASS	PSC0	128	3	5	0x45020000	PSC0	psc	0x00400A80	0x00400A83	No
CBASS_IPCSS0	DMASS0_SEC_PROXY_0	160	3	8	0x45028000	DMASS0_SEC_PROXY_0	SEC_PROXY_SRC_TARGET_DATA	0x4D000000	0x4D07FFFF	No

Copyright © 2026 Texas Instruments Incorporated

Table 3-23. Region Based Firewalls (continued)

IP	Target	Firewall ID	Number of Priv IDs per Region	Firewall Regions	Firewall Physical Address	Instance	Interface	Start Address	End Address	ASEL Capable
CBASS_IPCSS0	SA3_SS0_SEC_PROXY_0	161	3	8	0x45028400	SA3_SS0_SEC_PROXY_0	SEC_PROXY_SRC_TARGET_DATA	0x43600000	0x4360FFFF	No

Copyright © 2026 Texas Instruments Incorporated

3.10.3.2 Region Based Firewall Programming

Each firewall block can be configured to support multiple regions and each region can be configured individually.

See Region Based Firewalls.

Copyright © 2026 Texas Instruments Incorporated

Chapter 4

Module Integration

4.1 Memory Controllers 113
4.2 System Interconnect 115
4.3 Processors and Accelerators 125
4.4 Interprocessor Communication 146
4.5 Device Configuration 154
4.6 Interrupts 167
4.7 Data Movement Architecture 176
4.8 Audio 197
4.9 General Connectivity 211
4.10 High-speed Serial Interfaces 235
4.11 Memory Interfaces 250
4.12 Industrial and Control Interfaces 257
4.13 Camera Subsystem 270
4.14 Timer Modules 280
4.15 Internal Diagnostic Modules 320
4.16 Display Subsystem (DSS) 344
4.17 On-Chip Debug 349

Copyright © 2026 Texas Instruments Incorporated

4.1 Memory Controllers

4.1.1 DDR32 Subsystem (DDR32SS)

This section contains the integration details for the 32-bit DDR module on this device. For further information, see the *DDR32 Subsystem (DDR32SS)* section of the *Memory Controllers* chapter.

4.1.1.1 DDR32SS Unsupported Features

The following features are not supported on this family of devices:

- DDR3, DDR3L, DDR3U, and LPDDR4x Devices
- Two independent 16-bit channels for LPDDR4
- DIMMS
- Data bus obfuscation / data encryption
- Fail-safe reset I/O to maintain reset state during SoC power OFF
- Automatic periodic scrubbing of SDRAM for ECC
- Coherence of transactions arriving across the RT and NRT bus interfaces
- Maximum of 17 row bits are supported for LPDDR4. LPDDR4 with 18 row bits are not supported
- LPDDR4 devices with byte mode die configurations
- The ECC engine of the DDR controller
- 1T command timing (only 2T timing is supported)

4.1.1.2 ddr32ss Integration

Table 4-1. DDR32SS Module Allocations

Instance	MAIN	MCU	WKUP
DDR32SS0	✓		

Table 4-2. Clock Integration for ddr32ss

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DDR32SS0	DDRSS_DDR_PLL_CLK	MAIN_PLL12_HSDIV0_CLKOUT		None
DDR32SS0	PLL_CTRL_CLK	MAIN_SYSCLK0		None

Table 4-3. Hardware Requests for ddr32ss

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DDR32SS0	DDR32SS0_ddrss_controller_0	GICSS0_spi_151	GICSS0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_controller_0	R5FSS0_CORE0_intr_151	R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_controller_0	WKUP_R5FSS0_CORE0_intr_151	WKUP_R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_controller_0	MCU_R5FSS0_CORE0_cpu0_intr_151	MCU_R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_controller_0	C7X256V0_CLEC_gic_spi_151	C7X256V0_CLEC	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_controller_0	C7X256V1_CLEC_gic_spi_151	C7X256V1_CLEC	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_dram_ecc_corr_err_lvl_0	ESM0_esm_lvl_event_174	ESM0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_dram_ecc_uncorr_err_lvl_0	ESM0_esm_lvl_event_175	ESM0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	GICSS0_spi_172	GICSS0	DDR32SS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-3. Hardware Requests for ddr32ss (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	R5FSS0_CORE0_intr_181	R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	WKUP_R5FSS0_CORE0_intr_181	WKUP_R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	MCU_R5FSS0_CORE0_cpu0_intr_181	MCU_R5FSS0_CORE0	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	C7X256V0_CLEC_gic_spi_172	C7X256V0_CLEC	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_pll_freq_change_req_0	C7X256V1_CLEC_gic_spi_172	C7X256V1_CLEC	DDR32SS0 interrupt request	level
DDR32SS0	DDR32SS0_ddrss_v2a_other_err_lvl_0	ESM0_esm_lvl_event_176	ESM0	DDR32SS0 interrupt request	level

4.1.2 MSRAM8KX256E

This section contains the integration details for the MSRAM8KX256E module on this device.

4.1.2.1 MSRAM8KX256E Unsupported Features

The following features are not supported on this family of devices:

- Coherency
- Exclusive access
- Cache line wrap addressing mode

4.1.2.2 msram8kx256e Integration

Table 4-4. MSRAM8KX256E Module Allocations

Instance	MAIN	MCU	WKUP
msram8kx256e0	✓		

Table 4-5. Integration Attributes for msram8kx256e

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
msram8kx256e0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso

Table 4-6. Clock Integration for msram8kx256e

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
msram8kx256e0	CCLK_CLK	MAIN_SYSCLK0/2		None
msram8kx256e0	VCLK_CLK	MAIN_SYSCLK0		None

Table 4-7. Reset Integration for msram8kx256e

Module Instance	Source	Description
msram8kx256e0	PSC0	msram8kx256e0 reset

Table 4-8. Hardware Requests for msram8kx256e

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
msram8kx256e0	msram8kx256e0_ecc_corr_level_0	ESM0_esm_lvl_event_5	ESM0	msram8kx256e0 interrupt request	level
msram8kx256e0	msram8kx256e0_ecc_uncorr_level_0	ESM0_esm_lvl_event_6	ESM0	msram8kx256e0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.2 System Interconnect

4.2.1 CBASS

4.2.1.1 cbass Integration

Table 4-9. CBASS Module Allocations

Instance	MAIN	MCU	WKUP
CBASS0	✓		

Table 4-10. Clock Integration for cbass

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS0	CLK	MAIN_SYSCLK0		None
CBASS0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
CBASS0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None
CBASS0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-11. Hardware Requests for cbass

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS0	CBASS0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS0 interrupt request	level
CBASS0	CBASS0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS0 interrupt request	level
CBASS0	CBASS0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS0 interrupt request	level
CBASS0	CBASS0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS0 interrupt request	level
CBASS0	CBASS0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS0 interrupt request	level
CBASS0	CBASS0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS0 interrupt request	level
CBASS0	CBASS0_default_exp_0	ESM0_esm_lvl_event_86	ESM0	CBASS0 interrupt request	level

4.2.1.2 cbass_audio Integration

Table 4-12. CBASS_AUDIO Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_AUDIO0	✓		

Table 4-13. Clock Integration for cbass_audio

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_AUDIO0	CLK	MAIN_SYSCLK0/2		None
CBASS_AUDIO0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None
CBASS_AUDIO0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-14. Hardware Requests for cbass_audio

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_AUDIO0 interrupt request	level
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_AUDIO0 interrupt request	level
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_AUDIO0 interrupt request	level
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_AUDIO0 interrupt request	level
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_AUDIO0 interrupt request	level
CBASS_AUDIO0	CBASS_AUDIO0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_AUDIO0 interrupt request	level

4.2.1.3 cbass_fw Integration

Table 4-15. CBASS_FW Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_FW0	✓		
MCU_CBASS_FW0		✓	

Table 4-16. Clock Integration for cbass_fw

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_FW0	DM_CLK_1_CLK	DM_CLK		None
CBASS_FW0	DM_CLK_4_CLK	DM_CLK/4		None
CBASS_FW0	HSM_CLK_1_CLK	MAIN_PLL15_HSDIV0_CLKOUT		None
CBASS_FW0	HSM_CLK_2_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None
CBASS_FW0	HSM_CLK_4_CLK	MAIN_PLL15_HSDIV0_CLKOUT/4		None
CBASS_FW0	MCU_SYSCLK0_4_CLK	MCU_SYSCLK0/4		None
CBASS_FW0	CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None
CBASS_FW0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
CBASS_FW0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None
CBASS_FW0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None
MCU_CBASS_FW0	MCU_SYSCLK0_1_CLK	MCU_SYSCLK0		None
MCU_CBASS_FW0	MCU_SYSCLK0_4_CLK	MCU_SYSCLK0/4		None
MCU_CBASS_FW0	CLK	MCU_SYSCLK0/4		None

4.2.1.4 cbass_infra Integration

Table 4-17. CBASS_INFRA Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_INFRA1	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-18. Clock Integration for cbass_infra

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_INFRA1	CLK	MAIN_SYSCLK0/4		None
CBASS_INFRA1	MAIN_SYSCLK0_16_CLK	MAIN_SYSCLK0/16		None
CBASS_INFRA1	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-19. Hardware Requests for cbass_infra

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_INFRA1 interrupt request	level
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_INFRA1 interrupt request	level
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_INFRA1 interrupt request	level
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_INFRA1 interrupt request	level
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_INFRA1 interrupt request	level
CBASS_INFRA1	CBASS_INFRA1_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_INFRA1 interrupt request	level

4.2.1.5 cbass_ipcss Integration

Table 4-20. CBASS_IPCSS Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_IPCSS0	✓		

Table 4-21. Clock Integration for cbass_ipcss

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_IPCSS0	HSM_CLK_2_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None
CBASS_IPCSS0	CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None
CBASS_IPCSS0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
CBASS_IPCSS0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None

Table 4-22. Hardware Requests for cbass_ipcss

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_IPCSS0 interrupt request	level
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_IPCSS0 interrupt request	level
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_IPCSS0 interrupt request	level
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_IPCSS0 interrupt request	level
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_IPCSS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-22. Hardware Requests for cbass_ipcss (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_IPCSS0	CBASS_IPCSS0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_IPCSS0 interrupt request	level
CBASS_IPCSS0	CBASS_IPCSS0_default_exp_0	ESM0_esm_lvl_event_110	ESM0	CBASS_IPCSS0 interrupt request	level

4.2.1.6 cbass_mcasp Integration

Table 4-23. CBASS_MCASP Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_MCASP0	✓		

Table 4-24. Clock Integration for cbass_mcasp

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_MCASP0	CLK	MAIN_SYSCLK0/2		None
CBASS_MCASP0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None

Table 4-25. Hardware Requests for cbass_mcasp

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_MCASP0 interrupt request	level
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_MCASP0 interrupt request	level
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_MCASP0 interrupt request	level
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_MCASP0 interrupt request	level
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_MCASP0 interrupt request	level
CBASS_MCASP0	CBASS_MCASP0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_MCASP0 interrupt request	level

4.2.1.7 cbass_misc_peri Integration

Table 4-26. CBASS_MISC_PERI Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_MISC_PERI0	✓		

Table 4-27. Clock Integration for cbass_misc_peri

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_MISC_PERI0	CLK	MAIN_SYSCLK0/4		None
CBASS_MISC_PERI0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-28. Hardware Requests for cbass_misc_peri

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_MISC_PERI0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-28. Hardware Requests for cbass_misc_peri (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_MISC_PERI0 interrupt request	level
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_MISC_PERI0 interrupt request	level
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_MISC_PERI0 interrupt request	level
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_MISC_PERI0 interrupt request	level
CBASS_MISC_PERI0	CBASS_MISC_PERI0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_MISC_PERI0 interrupt request	level

4.2.1.8 cbass_rt_cfg Integration

Table 4-29. CBASS_RT_CFG Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_RT_CFG0	✓		

Table 4-30. Clock Integration for cbass_rt_cfg

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_RT_CFG0	DSS_FCLK_1_CLK	MAIN_PLL1_HSDIV4_CLKOUT		None
CBASS_RT_CFG0	CLK	MAIN_SYSCLK0/2		None
CBASS_RT_CFG0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
CBASS_RT_CFG0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None
CBASS_RT_CFG0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-31. Hardware Requests for cbass_rt_cfg

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_RT_CFG0 interrupt request	level
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_RT_CFG0 interrupt request	level
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_RT_CFG0 interrupt request	level
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_RT_CFG0 interrupt request	level
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_RT_CFG0 interrupt request	level
CBASS_RT_CFG0	CBASS_RT_CFG0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_RT_CFG0 interrupt request	level

4.2.1.9 cbass_rt_data Integration

Table 4-32. CBASS_RT_DATA Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_RT_DATA0	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-33. Clock Integration for cbass_rt_data

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_RT_DATA0	DSS_FCLK_1_CLK	MAIN_PLL1_HSDIV4_CLKOUT		None
CBASS_RT_DATA0	CLK	MAIN_SYSCLK0		None
CBASS_RT_DATA0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None

Table 4-34. Hardware Requests for cbass_rt_data

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	GICSS0_spi_133	GICSS0	CBASS_RT_DATA0 interrupt request	level
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	CBASS_RT_DATA0 interrupt request	level
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	CBASS_RT_DATA0 interrupt request	level
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	CBASS_RT_DATA0 interrupt request	level
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	CBASS_RT_DATA0 interrupt request	level
CBASS_RT_DATA0	CBASS_RT_DATA0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	CBASS_RT_DATA0 interrupt request	level

4.2.1.10 cbass_rt_fw Integration

Table 4-35. CBASS_RT_FW Module Allocations

Instance	MAIN	MCU	WKUP
CBASS_RT_FW0	✓		

Table 4-36. Clock Integration for cbass_rt_fw

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CBASS_RT_FW0	CLK	MAIN_SYSCLK0/2		None
CBASS_RT_FW0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
CBASS_RT_FW0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None

4.2.1.11 cbass_safe Integration

Table 4-37. CBASS_SAFE Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_CBASS_SAFE1			✓

Table 4-38. Clock Integration for cbass_safe

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_CBASS_SAFE1	MCU_SYSCLK0_4_CLK	MCU_SYSCLK0/4		None
WKUP_CBASS_SAFE1	CLK	MCU_SYSCLK0/4		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-39. Hardware Requests for cbass_safe

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	GICSS0_spi_133	GICSS0	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	R5FSS0_CORE0_intr_31	R5FSS0_CORE0	WKUP_CBASS_SAFE1 interrupt request	level
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_31	MCU_R5FSS0_CORE0	WKUP_CBASS_SAFE1 interrupt request	level

4.2.1.12 mcu_cbass Integration

Table 4-40. MCU_CBASS Module Allocations

Instance	MAIN	MCU	WKUP
MCU_CBASS0		✓	

Table 4-41. Clock Integration for mcu_cbass

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_CBASS0	MCU_SYSCLK0_1_CLK	MCU_SYSCLK0		None
MCU_CBASS0	MCU_SYSCLK0_2_CLK	MCU_SYSCLK0/2		None
MCU_CBASS0	MCU_SYSCLK0_4_CLK	MCU_SYSCLK0/4		None
MCU_CBASS0	CLK	MCU_SYSCLK0		None

Table 4-42. Hardware Requests for mcu_cbass

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	GICSS0_spi_133	GICSS0	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	MCU_CBASS0 interrupt request	level
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	R5FSS0_CORE0_intr_31	R5FSS0_CORE0	MCU_CBASS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-42. Hardware Requests for mcu_cbass (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CBASS0	MCU_CBASS0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_31	MCU_R5FSS0_CORE0	MCU_CBASS0 interrupt request	level

4.2.1.13 wkup_cbass Integration

Table 4-43. WKUP_CBASS Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_CBASS0			✓

Table 4-44. Clock Integration for wkup_cbass

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_CBASS0	DM_CLK_1_CLK	DM_CLK		None
WKUP_CBASS0	DM_CLK_2_CLK	DM_CLK/2		None
WKUP_CBASS0	DM_CLK_4_CLK	DM_CLK/4		None
WKUP_CBASS0	DM_CLK_8_CLK	DM_CLK/8		None
WKUP_CBASS0	MCU_SYSCLK0_1_CLK	MCU_SYSCLK0		None
WKUP_CBASS0	MCU_SYSCLK0_4_CLK	MCU_SYSCLK0/4		None
WKUP_CBASS0	CLK	DM_CLK		None
WKUP_CBASS0	MAIN_SYSCLK0_1_CLK	MAIN_SYSCLK0		None
WKUP_CBASS0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-45. Hardware Requests for wkup_cbass

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	GICSS0_spi_133	GICSS0	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_exp_0	WKUP_ESM0_esm_lvl_event_3	WKUP_ESM0	WKUP_CBASS0 interrupt request	level
WKUP_CBASS0	WKUP_CBASS0_default_exp_0	ESM0_esm_lvl_event_183	ESM0	WKUP_CBASS0 interrupt request	level

4.2.1.14 J7AEN_main_dbg_cbass Integration

Table 4-46. J7AEN_MAIN_DBG_CBASS Module Allocations

Instance	MAIN	MCU	WKUP
J7AEN_main_dbg_cbass0	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-47. Clock Integration for J7AEN_main_dbg_cbass

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
J7AEN_main_dbg_cbass0	DM_CLK_2_CLK	DM_CLK/2		None
J7AEN_main_dbg_cbass0	CLK	MAIN_SYSCLK0/4		None
J7AEN_main_dbg_cbass0	MAIN_SYSCLK0_2_CLK	MAIN_SYSCLK0/2		None
J7AEN_main_dbg_cbass0	MAIN_SYSCLK0_4_CLK	MAIN_SYSCLK0/4		None

Table 4-48. Hardware Requests for J7AEN_main_dbg_cbass

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	GICSS0_spi_133	GICSS0	J7AEN_main_dbg_cbass0 interrupt request	level
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	R5FSS0_CORE0_intr_147	R5FSS0_CORE0	J7AEN_main_dbg_cbass0 interrupt request	level
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	WKUP_R5FSS0_CORE0_intr_147	WKUP_R5FSS0_CORE0	J7AEN_main_dbg_cbass0 interrupt request	level
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_147	MCU_R5FSS0_CORE0	J7AEN_main_dbg_cbass0 interrupt request	level
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	C7X256V0_CLEC_gic_spi_133	C7X256V0_CLEC	J7AEN_main_dbg_cbass0 interrupt request	level
J7AEN_main_dbg_cbass0	J7AEN_main_dbg_cbass0_default_err_intr_0	C7X256V1_CLEC_gic_spi_133	C7X256V1_CLEC	J7AEN_main_dbg_cbass0 interrupt request	level

4.2.2 c7xv_rsws_bs_limiter Integration

Table 4-49. C7XV_RSWS_BS_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
C7XV_RSWS_BS_LIMITER6	✓		
C7XV_RSWS_BS_LIMITER11	✓		

Table 4-50. Clock Integration for c7xv_rsws_bs_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
C7XV_RSWS_BS_LIMITER6	CLK_CLK	MAIN_SYSCLK0		None
C7XV_RSWS_BS_LIMITER11	CLK_CLK	MAIN_SYSCLK0		None

4.2.3 jpgenc_rs_bw_limiter Integration

Table 4-51. JPGENC_RS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
JPGENC_RS_BW_LIMITER4	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-52. Clock Integration for jpgenc_rs_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
JPGENC_RS_BW_LIMITER4	CLK_CLK	MAIN_SYSCLK0/2		None

4.2.4 jpgenc_ws_bw_limiter Integration

Table 4-53. JPGENC_WS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
JPGENC_WS_BW_LIMITER5	✓		

Table 4-54. Clock Integration for jpgenc_ws_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
JPGENC_WS_BW_LIMITER5	CLK_CLK	MAIN_SYSCLK0/2		None

4.2.5 gpu_rs_bw_limiter Integration

Table 4-55. GPU_RS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
GPU_RS_BW_LIMITER9	✓		

Table 4-56. Clock Integration for gpu_rs_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GPU_RS_BW_LIMITER9	CLK	MAIN_SYSCLK0		None

4.2.6 gpu_ws_bw_limiter Integration

Table 4-57. GPU_WS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
GPU_WS_BW_LIMITER10	✓		

Table 4-58. Clock Integration for gpu_ws_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GPU_WS_BW_LIMITER10	CLK	MAIN_SYSCLK0		None

4.2.7 vpac_rsws_bw_limiter Integration

Table 4-59. VPAC_RSWS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
VPAC_RSWS_BW_LIMITER8	✓		
VPAC_RSWS_BW_LIMITER7	✓		

Table 4-60. Clock Integration for vpac_rsws_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
VPAC_RSWS_BW_LIMITER8	CLK_CLK	MAIN_SYSCLK0		None
VPAC_RSWS_BW_LIMITER7	CLK_CLK	MAIN_SYSCLK0		None

Copyright © 2026 Texas Instruments Incorporated

4.3 Processors and Accelerators

4.3.1 Arm Cortex A53 Subsystem (A53SS)

This section contains the integration details for the A53SS module on this device. For further information, see the *Arm Cortex A53 Subsystem (A53SS)* section of the *Processors and Accelerators* chapter.

4.3.1.1 A53SS Integration

Table 4-61. A53SS Module Allocations

Instance	MAIN	MCU	WKUP
A53SS0_CORE_0	✓		
A53SS0_CORE_1	✓		
A53SS0_CORE_2	✓		
A53SS0_CORE_3	✓		
A53SS0	✓		

Table 4-62. Clock Integration for A53SS

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
A53SS0_CORE_0	CLK	MAIN_PLL8_HSDIV0_CLKOUT		None
A53SS0_CORE_1	CLK	MAIN_PLL8_HSDIV0_CLKOUT		None
A53SS0_CORE_2	CLK	MAIN_PLL8_HSDIV0_CLKOUT		None
A53SS0_CORE_3	A53_CORE3_ARM_CLK_CLK	MAIN_PLL8_HSDIV0_CLKOUT		None
A53SS0	COREPAC_ARM_CLK	MAIN_PLL8_HSDIV0_CLKOUT		a53ss clock
A53SS0	PLL_CTRL_CLK	MAIN_SYSCLK0		None

Table 4-63. Hardware Requests for A53SS

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
A53SS0	A53SS0_cnthpirq0_0	GICSS0_ppi0_0_26	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cnthpirq1_0	GICSS0_ppi0_1_26	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cnthpirq2_0	GICSS0_ppi0_2_26	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cnthpirq3_0	GICSS0_ppi0_3_26	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpnsirq0_0	GICSS0_ppi0_0_30	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpnsirq1_0	GICSS0_ppi0_1_30	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpnsirq2_0	GICSS0_ppi0_2_30	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpnsirq3_0	GICSS0_ppi0_3_30	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpsirq0_0	GICSS0_ppi0_0_29	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpsirq1_0	GICSS0_ppi0_1_29	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpsirq2_0	GICSS0_ppi0_2_29	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntpsirq3_0	GICSS0_ppi0_3_29	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntvirq0_0	GICSS0_ppi0_0_27	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntvirq1_0	GICSS0_ppi0_1_27	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntvirq2_0	GICSS0_ppi0_2_27	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_cntvirq3_0	GICSS0_ppi0_3_27	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_commirq0_0	GICSS0_ppi0_0_22	GICSS0	A53SS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-63. Hardware Requests for A53SS (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
A53SS0	A53SS0_commirq1_0	GICSS0_ppi0_1_22	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_commirq2_0	GICSS0_ppi0_2_22	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_commirq3_0	GICSS0_ppi0_3_22	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq0_0	GICSS0_ppi0_0_24	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq0_0	GICSS0_ppi0_1_17	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq0_0	GICSS0_ppi0_2_17	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq0_0	GICSS0_ppi0_3_17	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq1_0	GICSS0_ppi0_0_18	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq1_0	GICSS0_ppi0_1_24	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq1_0	GICSS0_ppi0_2_18	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq1_0	GICSS0_ppi0_3_18	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq2_0	GICSS0_ppi0_0_19	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq2_0	GICSS0_ppi0_1_19	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq2_0	GICSS0_ppi0_2_24	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq2_0	GICSS0_ppi0_3_19	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq3_0	GICSS0_ppi0_0_20	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq3_0	GICSS0_ppi0_1_20	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq3_0	GICSS0_ppi0_2_20	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ctiirq3_0	GICSS0_ppi0_3_24	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr0_corrected_err_level_0	ESM0_esm_lvl_event_24	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr0_uncorrected_err_level_0	ESM0_esm_lvl_event_94	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr1_corrected_err_level_0	ESM0_esm_lvl_event_25	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr1_uncorrected_err_level_0	ESM0_esm_lvl_event_93	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr2_corrected_err_level_0	ESM0_esm_lvl_event_45	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr2_uncorrected_err_level_0	ESM0_esm_lvl_event_46	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr3_corrected_err_level_0	ESM0_esm_lvl_event_47	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr3_uncorrected_err_level_0	ESM0_esm_lvl_event_48	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr_corepac_corrected_err_level_0	ESM0_esm_lvl_event_26	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_ecc_eccaggr_corepac_uncorrected_err_level_0	ESM0_esm_lvl_event_95	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_exterrirq_0	ESM0_esm_lvl_event_144	ESM0	A53SS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-63. Hardware Requests for A53SS (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
A53SS0	A53SS0_interrirq_0	ESM0_esm_lvl_event_145	ESM0	A53SS0 interrupt request	level
A53SS0	A53SS0_pmuirq0_0	GICSS0_ppi0_0_23	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_pmuirq1_0	GICSS0_ppi0_1_23	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_pmuirq2_0	GICSS0_ppi0_2_23	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_pmuirq3_0	GICSS0_ppi0_3_23	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_vcpumntirq0_0	GICSS0_ppi0_0_25	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_vcpumntirq1_0	GICSS0_ppi0_1_25	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_vcpumntirq2_0	GICSS0_ppi0_2_25	GICSS0	A53SS0 interrupt request	level
A53SS0	A53SS0_vcpumntirq3_0	GICSS0_ppi0_3_25	GICSS0	A53SS0 interrupt request	level

4.3.1.2 a53_rs_bw_limiter Integration

Table 4-64. A53_RS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
A53_RS_BW_LIMITER0	✓		

Table 4-65. Clock Integration for a53_rs_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
A53_RS_BW_LIMITER0	CLK	MAIN_SYSCLK0		None

4.3.1.3 a53_ws_bw_limiter Integration

Table 4-66. A53_WS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
A53_WS_BW_LIMITER1	✓		

Table 4-67. Clock Integration for a53_ws_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
A53_WS_BW_LIMITER1	CLK	MAIN_SYSCLK0		None

Copyright © 2026 Texas Instruments Incorporated

4.3.2 Video Encoder/Decoder (CODEC)

This section contains the integration details for the CODEC module on this device. For further information, see the *Video Encoder/Decoder (CODEC)* section of the *Processors and Accelerators* chapter.

4.3.2.1 CODEC Unsupported Features

The following features are not supported on this family of devices:

- YUV422 format output in Decode
- YUV422 10-bit support
- Slice-level processing/synchronization
- MMU/RAT for > 32-bit address support. Address mapping > 32-bit can be accomplished by IP assigning the higher addresses for frame buffers to support 36-bit.

4.3.2.2 CODEC Integration

Table 4-68. CODEC Module Allocations

Instance	MAIN	MCU	WKUP
CODEC0	✓		

Table 4-69. Integration Attributes for CODEC

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
CODEC0	PSC0	PD_CODEC	LPSC_main_codec	65	OFF	YES	LPSC_main_ip

Table 4-70. Clock Integration for CODEC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CODEC0	ACLK	MAIN_PLL2_HSDIV4_CLKOUT		None
CODEC0	BCLK	MAIN_PLL2_HSDIV4_CLKOUT		None
CODEC0	CCLK	MAIN_PLL2_HSDIV4_CLKOUT		None
CODEC0	PCLK	MAIN_PLL2_HSDIV4_CLKOUT		None

Table 4-71. Reset Integration for CODEC

Module Instance	Source	Description
CODEC0	PSC0	CODEC0 reset

Table 4-72. Hardware Requests for CODEC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CODEC0	CODEC0_vpu_wave521cl_intr_0	GICSS0_spi_257	GICSS0	CODEC0 interrupt request	level
CODEC0	CODEC0_vpu_wave521cl_intr_0	R5FSS0_CORE0_intr_133	R5FSS0_CORE0	CODEC0 interrupt request	level
CODEC0	CODEC0_vpu_wave521cl_intr_0	WKUP_R5FSS0_CORE0_intr_133	WKUP_R5FSS0_CORE0	CODEC0 interrupt request	level
CODEC0	CODEC0_vpu_wave521cl_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_133	MCU_R5FSS0_CORE0	CODEC0 interrupt request	level
CODEC0	CODEC0_vpu_wave521cl_intr_0	C7X256V0_CLEC_gic_spi_257	C7X256V0_CLEC	CODEC0 interrupt request	level
CODEC0	CODEC0_vpu_wave521cl_intr_0	C7X256V1_CLEC_gic_spi_257	C7X256V1_CLEC	CODEC0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.3.2.3 codec_rs_bw_limiter Integration

Table 4-73. CODEC_RS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
CODEC_RS_BW_LIMITER2	✓		

Table 4-74. Clock Integration for codec_rs_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CODEC_RS_BW_LIMITER2	CLK	MAIN_SYSCLK0		None

4.3.2.4 codec_ws_bw_limiter Integration

Table 4-75. CODEC_WS_BW_LIMITER Module Allocations

Instance	MAIN	MCU	WKUP
CODEC_WS_BW_LIMITER3	✓		

Table 4-76. Clock Integration for codec_ws_bw_limiter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CODEC_WS_BW_LIMITER3	CLK	MAIN_SYSCLK0		None

4.3.2.5 VPU_PBIST Integration

Table 4-77. VPU_PBIST Module Allocations

Instance	MAIN	MCU	WKUP
PBIST3	✓		

Table 4-78. Clock Integration for VPU_PBIST

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PBIST3	CLK8_CLK	MAIN_SYSCLK0/4		None

Table 4-79. Hardware Requests for VPU_PBIST

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	PBIST3 interrupt request	pulse
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	PBIST3 interrupt request	pulse
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	PBIST3 interrupt request	pulse
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event0_232	ESM0	PBIST3 interrupt request	pulse
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event1_232	ESM0	PBIST3 interrupt request	pulse
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event2_232	ESM0	PBIST3 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-79. Hardware Requests for VPU_PBIST (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PBIST3	PBIST3_k3_pbist_8c28p_4bit_wrap__dft_pbist_safety_error_0	ESM0_esm_lvl_event_148	ESM0	PBIST3 interrupt request	level

4.3.3 Arm Cortex R5F Subsystem

4.3.3.1 R5FSS_SS0 Integration

Table 4-80. R5FSS_SS0 Module Allocations

Instance	MAIN	MCU	WKUP
R5FSS0_SS0	✓		
WKUP_R5FSS0_SS0			✓
MCU_R5FSS0_SS0		✓	

4.3.3.2 Arm Cortex R5F Subsystem (R5FSS)

This section contains the integration details for the R5FSS module on this device. For further information, see the Arm Cortex R5F Subsystem (R5FSS) section of the Processors and Accelerators chapter.

4.3.3.2.1 R5FSS Unsupported Features

The following features are not supported on this family of devices:

- Lockstep
- 32-bit VBUSM Initiator Peripheral Port normal or virtual
- BTCM0 / BTCM1 interleaving on MSB option

4.3.3.2.2 R5FSS Integration

Table 4-81. R5FSS Module Allocations

Instance	MAIN	MCU	WKUP
R5FSS0	✓		

4.3.3.2.3 R5FSS_CORE0 Integration

Table 4-82. R5FSS_CORE0 Module Allocations

Instance	MAIN	MCU	WKUP
R5FSS0_CORE0	✓		

Table 4-83. Integration Attributes for R5FSS_CORE0

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
R5FSS0_CORE0	PSC0	PD_MAIN_MCUSS0	LPSC_main_mcuss0_core0	79	OFF	YES	LPSC_main_ip

Table 4-84. Clock Integration for R5FSS_CORE0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
R5FSS0_CORE0	CLK	MAIN_PLL15_HSDIV3_CLKOUT		None
R5FSS0_CORE0	ICLK	MAIN_PLL15_HSDIV3_CLKOUT		interface clock

Table 4-85. Reset Integration for R5FSS_CORE0

Module Instance	Source	Description
R5FSS0_CORE0	PSC0	R5FSS0_CORE0 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-86. Hardware Requests for R5FSS_CORE0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
R5FSS0_CORE0	R5FSS0_CORE0_cti_0	R5FSS0_CORE0_intr_175	R5FSS0_CORE0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_ecc_corrected_level_0	ESM0_esm_lvl_event_208	ESM0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_ecc_uncorrected_level_0	ESM0_esm_lvl_event_209	ESM0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_exp_intr_0	ESM0_esm_lvl_event_210	ESM0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_exp_intr_0	R5FSS0_CORE0_intr_4	R5FSS0_CORE0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_pmu_0	R5FSS0_CORE0_intr_94	R5FSS0_CORE0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_valfiq_0	R5FSS0_CORE0_intr_95	R5FSS0_CORE0	R5FSS0_CORE0 interrupt request	level
R5FSS0_CORE0	R5FSS0_CORE0_valirq_0	R5FSS0_CORE0_intr_96	R5FSS0_CORE0	R5FSS0_CORE0 interrupt request	level

4.3.3.2.4 R5FSS_COMMON0 Integration

Table 4-87. R5FSS_COMMON0 Module Allocations

Instance	MAIN	MCU	WKUP
R5FSS0_COMMON0	✓		

Table 4-88. Hardware Requests for R5FSS_COMMON0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
R5FSS0_COMMON0	R5FSS0_COMMON0_commrx_level_0_0	R5FSS0_CORE0_intr_90	R5FSS0_CORE0	R5FSS0_COMMON0 interrupt request	level
R5FSS0_COMMON0	R5FSS0_COMMON0_commtx_level_0_0	R5FSS0_CORE0_intr_91	R5FSS0_CORE0	R5FSS0_COMMON0 interrupt request	level
R5FSS0_COMMON0	R5FSS0_COMMON0_ecc_de_to_esm_0_0	ESM0_esm_lvl_event_211	ESM0	R5FSS0_COMMON0 interrupt request	level
R5FSS0_COMMON0	R5FSS0_COMMON0_ecc_se_to_esm_0_0	ESM0_esm_lvl_event_212	ESM0	R5FSS0_COMMON0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.3.3.3 Arm Cortex R5F Subsystem (MCU_R5FSS)

This section contains the integration details for the MCU_R5FSS module on this device. For Further information, see the Arm Cortex R5F Subsystem (MCU_R5FSS) section of the Processors and Accelerators chapter

4.3.3.3.1 MCU_R5FSS Unsupported Features

The following features are not supported on this family of devices:

- Lockstep
- 32-bit VBUSM Initiator Peripheral Port normal or virtual
- Non-Maskable Interrupts (NMI)

4.3.3.3.2 MCU_R5FSS Integration

Table 4-89. MCU_R5FSS Module Allocations

Instance	MAIN	MCU	WKUP
MCU_R5FSS0		✓	
MCU_R5FSS0_CORE0		✓	
MCU_R5FSS0_COMMON0		✓	

Table 4-90. Integration Attributes for MCU_R5FSS

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCU_R5FSS0_CORE0	WKUP_PSC0	PD_MCUSS	LPSC_mcu_r5	6	OFF	YES	LPSC_MCU_common

Table 4-91. Clock Integration for MCU_R5FSS

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_R5FSS0_CORE0	CPU0_CLK	MCU_PLL0_HSDIV3_CLKOUT		None
MCU_R5FSS0_CORE0	ICLK	MCU_PLL0_HSDIV3_CLKOUT		interface clock

Table 4-92. Reset Integration for MCU_R5FSS

Module Instance	Source	Description
MCU_R5FSS0_CORE0	WKUP_PSC0	MCU_R5FSS0_CORE0 reset

Table 4-93. Hardware Requests for MCU_R5FSS

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_commrx_level_0_0	MCU_R5FSS0_CORE0_cpu0_intr_90	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_commtx_level_0_0	MCU_R5FSS0_CORE0_cpu0_intr_91	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_cti_0	MCU_R5FSS0_CORE0_cpu0_intr_175	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_exp_intr_0	WKUP_ESM0_esm_lvl_event_30	WKUP_ESM0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_exp_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_4	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_pmu_0	MCU_R5FSS0_CORE0_cpu0_intr_94	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_valfiq_0	MCU_R5FSS0_CORE0_cpu0_intr_95	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_cpu0_valirq_0	MCU_R5FSS0_CORE0_cpu0_intr_96	MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-93. Hardware Requests for MCU_R5FSS (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_R5FSS0_COMMON0	MCU_R5FSS0_COMMON0_ecc_de_to_esm_0_0	WKUP_ESM0_esm_lvl_event_28	WKUP_ESM0	MCU_R5FSS0_COMMON0 interrupt request	level
MCU_R5FSS0_COMMON0	MCU_R5FSS0_COMMON0_ecc_se_to_esm_0_0	WKUP_ESM0_esm_lvl_event_29	WKUP_ESM0	MCU_R5FSS0_COMMON0 interrupt request	level

4.3.3.3.3 R5FSS_CPU0_ECC_AGGR_0 Integration

Table 4-94. R5FSS_CPU0_ECC_AGGR_0 Module Allocations

Instance	MAIN	MCU	WKUP
MCU_R5FSS0_CPU0_ECC_AGGR_0		✓	

Table 4-95. Clock Integration for R5FSS_CPU0_ECC_AGGR_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_R5FSS0_CPU0_ECC_AGGR_0	ECC_CLK	MCU_PLL0_HSDIV3_CLKOUT		ecc aggregator clock

Table 4-96. Hardware Requests for R5FSS_CPU0_ECC_AGGR_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_R5FSS0_CPU0_ECC_AGGR_0	MCU_R5FSS0_CPU0_ECC_AGGR_0_cpu0_ecc_corrected_level_0	WKUP_ESM0_esm_lvl_event_4	WKUP_ESM0	MCU_R5FSS0_CPU0_ECC_AGGR_0 interrupt request	level
MCU_R5FSS0_CPU0_ECC_AGGR_0	MCU_R5FSS0_CPU0_ECC_AGGR_0_cpu0_ecc_uncorrected_level_0	WKUP_ESM0_esm_lvl_event_5	WKUP_ESM0	MCU_R5FSS0_CPU0_ECC_AGGR_0 interrupt request	level

4.3.3.4 Device Manager Cortex R5F Subsystem (WKUP_R5FSS)

This section contains the integration details for the WKUP_R5FSS module on this device. For further information, see the Device Manager Cortex R5F Subsystem (WKUP_R5FSS) section of the Processors and Accelerators chapter.

4.3.3.4.1 WKUP_R5FSS Unsupported Features

The following features are not supported on this family of devices:

- Lockstep
- 32-bit VBUSM Initiator Peripheral Port normal or virtual

4.3.3.4.2 WKUP_R5FSS Integration

Table 4-97. WKUP_R5FSS Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_R5FSS0			✓
WKUP_R5FSS0_CORE0			✓
WKUP_R5FSS0_COMMON0			✓

Copyright © 2026 Texas Instruments Incorporated

Table 4-98. Integration Attributes for WKUP_R5FSS

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_R5FSS0_CORE0	PSC0	GP_CORE	LPSC_main_dm	1	OFF	YES	LPSC_main_alwayson

Table 4-99. Clock Integration for WKUP_R5FSS

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_R5FSS0_CORE0	CLK	MAIN_PLL15_HSDIV2_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_CLKSEL[0:0]=0	None
WKUP_R5FSS0_CORE0	CLK	MCU_PLL0_HSDIV0_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_CLKSEL[0:0]=1	None
WKUP_R5FSS0_CORE0	ICLK	MAIN_PLL15_HSDIV2_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_CLKSEL[0:0]=0	interface clock
WKUP_R5FSS0_CORE0	ICLK	MCU_PLL0_HSDIV0_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_CLKSEL[0:0]=1	interface clock

Table 4-100. Reset Integration for WKUP_R5FSS

Module Instance	Source	Description
WKUP_R5FSS0_CORE0	PSC0	WKUP_R5FSS0_CORE0 reset

Table 4-101. Hardware Requests for WKUP_R5FSS

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_cti_0	WKUP_R5FSS0_CORE0_intr_175	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_ecc_corrected_level_0	ESM0_esm_lvl_event_30	ESM0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_ecc_uncorrected_level_0	ESM0_esm_lvl_event_91	ESM0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_exp_intr_0	ESM0_esm_lvl_event_124	ESM0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_exp_intr_0	WKUP_R5FSS0_CORE0_intr_4	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_pmu_0	WKUP_R5FSS0_CORE0_intr_58	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_valfiq_0	WKUP_R5FSS0_CORE0_intr_59	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_valirq_0	WKUP_R5FSS0_CORE0_intr_60	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0 interrupt request	level
WKUP_R5FSS0_COMMON0	WKUP_R5FSS0_COMMON0_commrx_level_0_0	WKUP_R5FSS0_CORE0_intr_5	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_COMMON0 interrupt request	level
WKUP_R5FSS0_COMMON0	WKUP_R5FSS0_COMMON0_commtx_level_0_0	WKUP_R5FSS0_CORE0_intr_6	WKUP_R5FSS0_CORE0	WKUP_R5FSS0_COMMON0 interrupt request	level
WKUP_R5FSS0_COMMON0	WKUP_R5FSS0_COMMON0_ecc_de_to_esm_0_0	ESM0_esm_lvl_event_40	ESM0	WKUP_R5FSS0_COMMON0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-101. Hardware Requests for WKUP_R5FSS (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_R5FSS0_COMMON0	WKUP_R5FSS0_COMMON0_ecc_se_to_esm_0_0	ESM0_esm_lvl_event_42	ESM0	WKUP_R5FSS0_COMMON0 interrupt request	level

4.3.4 Vision Pre-processing Accelerator (VPAC)

This section contains the integration details for the VPAC module on this device. For further information, see the *Vision Pre-processing Accelerator (VPAC)* section of the *Processors and Accelerators* chapter.

4.3.4.1 VPAC3L Unsupported Features

The following features are not supported on this family of devices:

- MMU
- Direct Image Access

4.3.4.2 VPAC Integration

Table 4-102. VPAC Module Allocations

Instance	MAIN	MCU	WKUP
VPAC0	✓		

Table 4-103. Clock Integration for VPAC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
VPAC0	PLL_CTRL_CLK	MAIN_SYSCLK0		None
VPAC0	VPAC_PLL_CFG_CLK	MAIN_PLL15_HSDIV0_CLKOUT		None
VPAC0	VPAC_PLL_CLK	MAIN_PLL5_HSDIV0_CLKOUT		None

Table 4-104. Hardware Requests for VPAC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
VPAC0	VPAC0_ecc_intr0_corr_level_0	ESM0_esm_lvl_event_168	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_ecc_intr0_uncorr_level_0	ESM0_esm_lvl_event_169	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_ecc_intr1_corr_level_0	ESM0_esm_lvl_event_170	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_ecc_intr1_uncorr_level_0	ESM0_esm_lvl_event_171	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_ecc_intr3_corr_level_0	ESM0_esm_lvl_event_172	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_ecc_intr3_uncorr_level_0	ESM0_esm_lvl_event_173	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	VPAC0 interrupt request	pulse
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	pulse
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	VPAC0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-104. Hardware Requests for VPAC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event0_237	ESM0	VPAC0 interrupt request	pulse
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event1_237	ESM0	VPAC0 interrupt request	pulse
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event2_237	ESM0	VPAC0 interrupt request	pulse
VPAC0	VPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_safety_error_0	ESM0_esm_lvl_event_177	ESM0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	GICSS0_spi_168	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	R5FSS0_CORE0_intr_178	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	WKUP_R5FSS0_CORE0_intr_178	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	MCU_R5FSS0_CORE0_cpu0_intr_178	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	C7X256V0_CLEC_gic_spi_168	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_0	C7X256V1_CLEC_gic_spi_168	C7X256V1_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	GICSS0_spi_169	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	R5FSS0_CORE0_intr_179	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	WKUP_R5FSS0_CORE0_intr_179	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	MCU_R5FSS0_CORE0_cpu0_intr_179	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	C7X256V0_CLEC_gic_spi_169	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_1	C7X256V1_CLEC_gic_spi_169	C7X256V1_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	GICSS0_spi_170	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	R5FSS0_CORE0_intr_180	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	WKUP_R5FSS0_CORE0_intr_180	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	MCU_R5FSS0_CORE0_cpu0_intr_180	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	C7X256V0_CLEC_gic_spi_170	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_2	C7X256V1_CLEC_gic_spi_170	C7X256V1_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_3	GICSS0_spi_189	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_3	R5FSS0_CORE0_intr_182	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_3	WKUP_R5FSS0_CORE0_intr_182	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_3	MCU_R5FSS0_CORE0_cpu0_intr_182	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-104. Hardware Requests for VPAC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
VPAC0	VPAC0_vpac_level_3	C7X256V0_CLEC_gic_spi_189	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_3	C7X256V1_CLEC_gic_spi_189	C7X256V1_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	GICSS0_spi_190	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	R5FSS0_CORE0_intr_189	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	WKUP_R5FSS0_CORE0_intr_189	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	MCU_R5FSS0_CORE0_cpu0_intr_189	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	C7X256V0_CLEC_gic_spi_190	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_4	C7X256V1_CLEC_gic_spi_190	C7X256V1_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	GICSS0_spi_191	GICSS0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	R5FSS0_CORE0_intr_191	R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	WKUP_R5FSS0_CORE0_intr_191	WKUP_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	MCU_R5FSS0_CORE0_cpu0_intr_191	MCU_R5FSS0_CORE0	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	C7X256V0_CLEC_gic_spi_191	C7X256V0_CLEC	VPAC0 interrupt request	level
VPAC0	VPAC0_vpac_level_5	C7X256V1_CLEC_gic_spi_191	C7X256V1_CLEC	VPAC0 interrupt request	level

4.3.5 Depth and Motion Perception Accelerator (DMPAC)

This section contains the integration details for the DMPAC module on this device. For further information, see the *Depth and Motion Perception Accelerator (DMPAC)* section of the *Processors and Accelerators* chapter.

4.3.5.1 DMPAC Integration

Table 4-105. DMPAC Module Allocations

Instance	MAIN	MCU	WKUP
DMPAC0	✓		

Table 4-106. Clock Integration for DMPAC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DMPAC0	DMPAC_PLL_CLK	MAIN_PLL5_HSDIV2_CLKOUT		None
DMPAC0	PLL_CTRL_CLK	MAIN_SYSCLK0		None

Table 4-107. Hardware Requests for DMPAC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMPAC0	DMPAC0_dft_lbist_bist_done_0	WKUP_R5FSS0_CORE0_intr_3	WKUP_R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dft_lbist_bist_done_0	MCU_R5FSS0_CORE0_cpu0_intr_3	MCU_R5FSS0_CORE0	DMPAC0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-107. Hardware Requests for DMPAC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMPAC0	DMPAC0_dmpac_level_0	MAIN_GPIOMUX_INTROUTER0_in_196	MAIN_GPIOMUX_INTROUTER0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_0	R5FSS0_CORE0_intr_5	R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_0	WKUP_R5FSS0_CORE0_intr_62	WKUP_R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_0	MCU_R5FSS0_CORE0_cpu0_intr_5	MCU_R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_0	C7X256V0_CLEC_gic_spi_50	C7X256V0_CLEC	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_0	C7X256V1_CLEC_gic_spi_50	C7X256V1_CLEC	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	MAIN_GPIOMUX_INTROUTER0_in_197	MAIN_GPIOMUX_INTROUTER0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	R5FSS0_CORE0_intr_6	R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	WKUP_R5FSS0_CORE0_intr_96	WKUP_R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	MCU_R5FSS0_CORE0_cpu0_intr_6	MCU_R5FSS0_CORE0	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	C7X256V0_CLEC_gic_spi_51	C7X256V0_CLEC	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_dmpac_level_1	C7X256V1_CLEC_gic_spi_51	C7X256V1_CLEC	DMPAC0 interrupt request	level
DMPAC0	DMPAC0_ecc_corrected_err_pulse_0	ESM0_esm_pls_event0_250	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_ecc_corrected_err_pulse_0	ESM0_esm_pls_event1_250	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_ecc_corrected_err_pulse_0	ESM0_esm_pls_event2_250	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_ecc_uncorrected_err_pulse_0	ESM0_esm_pls_event0_251	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_ecc_uncorrected_err_pulse_0	ESM0_esm_pls_event1_251	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_ecc_uncorrected_err_pulse_0	ESM0_esm_pls_event2_251	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event0_244	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event1_244	ESM0	DMPAC0 interrupt request	pulse
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_cpu_0	ESM0_esm_pls_event2_244	ESM0	DMPAC0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-107. Hardware Requests for DMPAC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMPAC0	DMPAC0_k3_pbist_8c28p_4bit_wrap__dft_pbist_safety_error_0	ESM0_esm_lvl_event_106	ESM0	DMPAC0 interrupt request	level

4.3.6 JPGENC

This section contains the integration details for the JPGENC module on this device.

4.3.6.1 JPGENC Integration

Table 4-108. JPGENC Module Allocations

Instance	MAIN	MCU	WKUP
JPGENC0	✓		

Table 4-109. Integration Attributes for JPGENC

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
JPGENC0	PSC0	GP_CORE	LPSC_main_jpeg	39	OFF	YES	LPSC_main_ip

Table 4-110. Clock Integration for JPGENC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
JPGENC0	CORE_CLK	MAIN_SYSCLK0/2		None

Table 4-111. Reset Integration for JPGENC

Module Instance	Source	Description
JPGENC0	PSC0	JPGENC0 reset

Table 4-112. Hardware Requests for JPGENC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
JPGENC0	JPGENC0_irq_0	GICSS0_spi_130	GICSS0	JPGENC0 interrupt request	level
JPGENC0	JPGENC0_irq_0	R5FSS0_CORE0_intr_100	R5FSS0_CORE0	JPGENC0 interrupt request	level
JPGENC0	JPGENC0_irq_0	WKUP_R5FSS0_CORE0_intr_100	WKUP_R5FSS0_CORE0	JPGENC0 interrupt request	level
JPGENC0	JPGENC0_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_100	MCU_R5FSS0_CORE0	JPGENC0 interrupt request	level
JPGENC0	JPGENC0_irq_0	C7X256V0_CLEC_gic_spi_130	C7X256V0_CLEC	JPGENC0 interrupt request	level
JPGENC0	JPGENC0_irq_0	C7X256V1_CLEC_gic_spi_130	C7X256V1_CLEC	JPGENC0 interrupt request	level

4.3.7 C7X256V

This section contains the integration details for the C7X256V module on this device.

4.3.7.1 C7x256v Integration

Table 4-113. C7X256V Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0	✓		
C7X256V1	✓		

Copyright © 2026 Texas Instruments Incorporated

4.3.7.2 C7X256V_C7XV_CORE_0 Integration

Table 4-114. C7X256V_C7XV_CORE_0 Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_C7XV_CORE_0	✓		
C7X256V1_C7XV_CORE_0	✓		

Table 4-115. Clock Integration for C7X256V_C7XV_CORE_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
C7X256V0_C7XV_CORE_0	C7XV_CLK	MAIN_PLL7_HSDIV0_CLKOUT		None
C7X256V1_C7XV_CORE_0	C7XV_CLK	MAIN_PLL7_HSDIV1_CLKOUT		None

4.3.7.3 C7X256V_CLEC Integration

Table 4-116. C7X256V_CLEC Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_CLEC	✓		
C7X256V1_CLEC	✓		

Table 4-117. Hardware Requests for C7X256V_CLEC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C7X256V0_CLEC	C7X256V0_CLEC_arm_0_eventi_out_0	COMPUTE_CLUSTER0_cpu_evnt_eventi_0	COMPUTE_CLUSTER0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event0_226	ESM0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event1_226	ESM0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event2_226	ESM0	C7X256V0_CLEC interrupt request	pulse
C7X256V0_CLEC	C7X256V0_CLEC_dft_pbist_safety_error_0	ESM0_esm_lvl_event_149	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_0	ESM0_esm_lvl_event_160	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_1	ESM0_esm_lvl_event_161	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_2	ESM0_esm_lvl_event_162	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_3	ESM0_esm_lvl_event_163	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_4	ESM0_esm_lvl_event_164	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_5	ESM0_esm_lvl_event_165	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_6	ESM0_esm_lvl_event_166	ESM0	C7X256V0_CLEC interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-117. Hardware Requests for C7X256V_CLEC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C7X256V0_CLEC	C7X256V0_CLEC_esm_events_out_level_7	ESM0_esm_lvl_event_167	ESM0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_0	GICSS0_spi_280	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_1	GICSS0_spi_281	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_2	GICSS0_spi_282	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_3	GICSS0_spi_283	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_4	GICSS0_spi_284	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_5	GICSS0_spi_285	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_6	GICSS0_spi_286	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_7	GICSS0_spi_287	GICSS0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_8	MCU_R5FSS0_CORE0_cpu0_intr_249	MCU_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_9	MCU_R5FSS0_CORE0_cpu0_intr_250	MCU_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_10	MCU_R5FSS0_CORE0_cpu0_intr_254	MCU_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_11	MCU_R5FSS0_CORE0_cpu0_intr_255	MCU_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_12	WKUP_R5FSS0_CORE0_intr_249	WKUP_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_13	WKUP_R5FSS0_CORE0_intr_250	WKUP_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_14	WKUP_R5FSS0_CORE0_intr_254	WKUP_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_15	WKUP_R5FSS0_CORE0_intr_255	WKUP_R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_20	C7X256V1_CLEC_soc_events_in_12	C7X256V1_CLEC	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_21	C7X256V1_CLEC_soc_events_in_13	C7X256V1_CLEC	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_22	C7X256V1_CLEC_soc_events_in_14	C7X256V1_CLEC	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_23	C7X256V1_CLEC_soc_events_in_15	C7X256V1_CLEC	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_24	R5FSS0_CORE0_intr_249	R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_25	R5FSS0_CORE0_intr_250	R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_26	R5FSS0_CORE0_intr_254	R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V0_CLEC	C7X256V0_CLEC_soc_events_out_level_27	R5FSS0_CORE0_intr_255	R5FSS0_CORE0	C7X256V0_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_arm_0_eventi_out_0	COMPUTE_CLUSTER0_cpu_evnt_eventi_0	COMPUTE_CLUSTER0	C7X256V1_CLEC interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-117. Hardware Requests for C7X256V_CLEC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event0_245	ESM0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event1_245	ESM0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_cpu_0	ESM0_esm_pls_event2_245	ESM0	C7X256V1_CLEC interrupt request	pulse
C7X256V1_CLEC	C7X256V1_CLEC_dft_pbist_safety_error_0	ESM0_esm_lvl_event_64	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_0	ESM0_esm_lvl_event_192	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_1	ESM0_esm_lvl_event_193	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_2	ESM0_esm_lvl_event_194	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_3	ESM0_esm_lvl_event_195	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_4	ESM0_esm_lvl_event_196	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_5	ESM0_esm_lvl_event_197	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_6	ESM0_esm_lvl_event_198	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_esm_events_out_level_7	ESM0_esm_lvl_event_199	ESM0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_0	GICSS0_spi_50	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_1	GICSS0_spi_51	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_2	GICSS0_spi_52	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_3	GICSS0_spi_53	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_4	GICSS0_spi_54	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_5	GICSS0_spi_55	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_6	GICSS0_spi_56	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_7	GICSS0_spi_57	GICSS0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_8	MCU_R5FSS0_CORE0_cpu0_intr_244	MCU_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_9	MCU_R5FSS0_CORE0_cpu0_intr_245	MCU_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_10	MCU_R5FSS0_CORE0_cpu0_intr_246	MCU_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-117. Hardware Requests for C7X256V_CLEC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_11	MCU_R5FSS0_CORE0_cpu0_intr_247	MCU_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_12	WKUP_R5FSS0_CORE0_intr_120	WKUP_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_13	WKUP_R5FSS0_CORE0_intr_121	WKUP_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_14	WKUP_R5FSS0_CORE0_intr_122	WKUP_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_15	WKUP_R5FSS0_CORE0_intr_165	WKUP_R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_20	C7X256V0_CLEC_soc_events_in_12	C7X256V0_CLEC	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_21	C7X256V0_CLEC_soc_events_in_13	C7X256V0_CLEC	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_22	C7X256V0_CLEC_soc_events_in_14	C7X256V0_CLEC	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_23	C7X256V0_CLEC_soc_events_in_15	C7X256V0_CLEC	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_24	R5FSS0_CORE0_intr_244	R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_25	R5FSS0_CORE0_intr_245	R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_26	R5FSS0_CORE0_intr_246	R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level
C7X256V1_CLEC	C7X256V1_CLEC_soc_events_out_level_27	R5FSS0_CORE0_intr_247	R5FSS0_CORE0	C7X256V1_CLEC interrupt request	level

4.3.7.4 C7X256V_CLK Integration

Table 4-118. C7X256V_CLK Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_CLK	✓		
C7X256V1_CLK	✓		

Table 4-119. Clock Integration for C7X256V_CLK

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
C7X256V0_CLK	C7XV_CLK	MAIN_PLL7_HSDIV0_CLKOUT		None
C7X256V0_CLK	PLL_CTRL_CLK	MAIN_SYSCLK0		None
C7X256V1_CLK	C7XV_CLK	MAIN_PLL7_HSDIV1_CLKOUT		None
C7X256V1_CLK	PLL_CTRL_CLK	MAIN_SYSCLK0		None

4.3.7.5 C7X256V_CORE0 Integration

Table 4-120. C7X256V_CORE0 Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_CORE0	✓		
C7X256V1_CORE0	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-121. Clock Integration for C7X256V_CORE0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
C7X256V0_CORE0	DIVH_CLK2_SOC_GCLK	C7X256V0_CLK (INSTANCE)		None
C7X256V0_CORE0	DIVH_CLK4_GCLK	C7X256V0_CLK (INSTANCE)		None
C7X256V0_CORE0	DIVH_CLK4_SOC_GCLK	C7X256V0_CLK (INSTANCE)		None
C7X256V0_CORE0	DIVP_CLK1_GCLK	C7X256V0_CLK (INSTANCE)		None
C7X256V0_CORE0	DIVP_CLK1_SOC_GCLK	C7X256V0_CLK (INSTANCE)		None
C7X256V1_CORE0	DIVH_CLK2_SOC_GCLK	C7X256V1_CLK (INSTANCE)		None
C7X256V1_CORE0	DIVH_CLK4_GCLK	C7X256V1_CLK (INSTANCE)		None
C7X256V1_CORE0	DIVH_CLK4_SOC_GCLK	C7X256V1_CLK (INSTANCE)		None
C7X256V1_CORE0	DIVP_CLK1_GCLK	C7X256V1_CLK (INSTANCE)		None
C7X256V1_CORE0	DIVP_CLK1_SOC_GCLK	C7X256V1_CLK (INSTANCE)		None

4.3.7.6 C7X256V_DEBUG Integration

Table 4-122. C7X256V_DEBUG Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_DEBUG	✓		
C7X256V1_DEBUG	✓		

4.3.7.7 C7X256V_ECC_AGGR Integration

Table 4-123. C7X256V_ECC_AGGR Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_ECC_AGGR	✓		
C7X256V1_ECC_AGGR	✓		

4.3.7.8 C7X256V_PBIST Integration

Table 4-124. C7X256V_PBIST Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_PBIST	✓		
C7X256V1_PBIST	✓		

4.3.8 Graphics Processing Unit (GPU)

This section contains the integration details for the GPU module on this device. For further information, see the *Graphics Processing Unit (GPU)* section of the *Processors and Accelerators* chapter.

4.3.8.1 GPU Unsupported Features

The following features are not supported on this family of devices:

- Multi-GPU support

Copyright © 2026 Texas Instruments Incorporated

4.3.8.2 GPU Integration

Table 4-125. GPU Module Allocations

Instance	MAIN	MCU	WKUP
GPU0	✓		

Table 4-126. Clock Integration for GPU

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GPU0	PLL_CLK	MAIN_PLL6_HSDIV0_CLKOUT		None
GPU0	CTRL_CLK	MAIN_SYSCLK0		None

Table 4-127. Hardware Requests for GPU

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPU0	GPU0_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_cpu_0	ESM0_esm_pls_event0_238	ESM0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_cpu_0	ESM0_esm_pls_event1_238	ESM0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_cpu_0	ESM0_esm_pls_event2_238	ESM0	GPU0 interrupt request	pulse
GPU0	GPU0_dft_pbist_safety_error_0	ESM0_esm_lvl_event_156	ESM0	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	GICSS0_spi_120	GICSS0	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	R5FSS0_CORE0_intr_248	R5FSS0_CORE0	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	WKUP_R5FSS0_CORE0_intr_31	WKUP_R5FSS0_CORE0	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	MCU_R5FSS0_CORE0_cpu0_intr_248	MCU_R5FSS0_CORE0	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	C7X256V0_CLEC_gic_spi_120	C7X256V0_CLEC	GPU0 interrupt request	level
GPU0	GPU0_gpu_pwrctrl_req_0	C7X256V1_CLEC_gic_spi_120	C7X256V1_CLEC	GPU0 interrupt request	level
GPU0	GPU0_gpu_safety_irq_0	ESM0_esm_lvl_event_184	ESM0	GPU0 interrupt request	level
GPU0	GPU0_os_irq_0	GICSS0_spi_273	GICSS0	GPU0 interrupt request	level
GPU0	GPU0_os_irq_0	C7X256V0_CLEC_gic_spi_273	C7X256V0_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_0	C7X256V1_CLEC_gic_spi_273	C7X256V1_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_1	GICSS0_spi_274	GICSS0	GPU0 interrupt request	level
GPU0	GPU0_os_irq_1	C7X256V0_CLEC_gic_spi_274	C7X256V0_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_1	C7X256V1_CLEC_gic_spi_274	C7X256V1_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_2	GICSS0_spi_275	GICSS0	GPU0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-127. Hardware Requests for GPU (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPU0	GPU0_os_irq_2	C7X256V0_CLEC_gic_spi_275	C7X256V0_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_2	C7X256V1_CLEC_gic_spi_275	C7X256V1_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_3	GICSS0_spi_276	GICSS0	GPU0 interrupt request	level
GPU0	GPU0_os_irq_3	C7X256V0_CLEC_gic_spi_276	C7X256V0_CLEC	GPU0 interrupt request	level
GPU0	GPU0_os_irq_3	C7X256V1_CLEC_gic_spi_276	C7X256V1_CLEC	GPU0 interrupt request	level

4.4 Interprocessor Communication

4.4.1 Mailbox

This section contains the integration details for the Mailbox module on this device. For further information, see the *Mailbox* section of the I*nterprocessor Communication* chapter.

4.4.1.1 Mailbox Unsupported Features

The following features are not supported on this family of devices:

- 'User' protection of mailbox in hardware. If required, must be implemented by software.

4.4.1.2 mailbox Integration

Table 4-128. MAILBOX Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0	✓		

4.4.1.3 MAILBOX_MAILBOX_CLUSTER_0 Integration

Table 4-129. MAILBOX_MAILBOX_CLUSTER_0 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_0	✓		

Table 4-130. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_0

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_0	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-131. Clock Integration for MAILBOX_MAILBOX_CLUSTER_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_0	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-132. Reset Integration for MAILBOX_MAILBOX_CLUSTER_0

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_0	PSC0	MAILBOX0_MAILBOX_CLUSTER_0 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-133. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_mailbox_cluster_pend_0	GICSS0_spi_108	GICSS0	MAILBOX0_MAILBOX_CLUSTER_0 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_mailbox_cluster_pend_0	C7X256V0_CLEC_gic_spi_108	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_0 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_mailbox_cluster_pend_0	C7X256V1_CLEC_gic_spi_108	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_0 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_mailbox_cluster_pend_2	MCU_R5FSS0_CORE0_cpu0_intr_240	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_0 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_mailbox_cluster_pend_3	WKUP_R5FSS0_CORE0_intr_240	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_0 interrupt request	level

4.4.1.4 MAILBOX_MAILBOX_CLUSTER_1 Integration

Table 4-134. MAILBOX_MAILBOX_CLUSTER_1 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_1	✓		

Table 4-135. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_1

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_1	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-136. Clock Integration for MAILBOX_MAILBOX_CLUSTER_1

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_1	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-137. Reset Integration for MAILBOX_MAILBOX_CLUSTER_1

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_1	PSC0	MAILBOX0_MAILBOX_CLUSTER_1 reset

Table 4-138. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_1

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_mailbox_cluster_pend_0	GICSS0_spi_109	GICSS0	MAILBOX0_MAILBOX_CLUSTER_1 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_mailbox_cluster_pend_0	C7X256V0_CLEC_gic_spi_109	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_1 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_mailbox_cluster_pend_0	C7X256V1_CLEC_gic_spi_109	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_1 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_mailbox_cluster_pend_2	MCU_R5FSS0_CORE0_cpu0_intr_241	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-138. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_1 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_mailbox_cluster_pend_3	WKUP_R5FSS0_CORE0_intr_241	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_1 interrupt request	level

4.4.1.5 MAILBOX_MAILBOX_CLUSTER_2 Integration

Table 4-139. MAILBOX_MAILBOX_CLUSTER_2 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_2	✓		

Table 4-140. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_2

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_2	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-141. Clock Integration for MAILBOX_MAILBOX_CLUSTER_2

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_2	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-142. Reset Integration for MAILBOX_MAILBOX_CLUSTER_2

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_2	PSC0	MAILBOX0_MAILBOX_CLUSTER_2 reset

Table 4-143. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_2

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_0	GICSS0_spi_140	GICSS0	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_0	C7X256V0_CLEC_gic_spi_140	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_0	C7X256V1_CLEC_gic_spi_140	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_6	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_2	MCU_R5FSS0_CORE0_cpu0_intr_242	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_6	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_115	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_mailbox_cluster_pend_3	WKUP_R5FSS0_CORE0_intr_242	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_2 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.4.1.6 MAILBOX_MAILBOX_CLUSTER_3 Integration

Table 4-144. MAILBOX_MAILBOX_CLUSTER_3 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_3	✓		

Table 4-145. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_3

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_3	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-146. Clock Integration for MAILBOX_MAILBOX_CLUSTER_3

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_3	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-147. Reset Integration for MAILBOX_MAILBOX_CLUSTER_3

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_3	PSC0	MAILBOX0_MAILBOX_CLUSTER_3 reset

Table 4-148. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_3

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_0	GICSS0_spi_141	GICSS0	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_0	C7X256V0_CLEC_gic_spi_141	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_0	C7X256V1_CLEC_gic_spi_141	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_7	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_2	MCU_R5FSS0_CORE0_cpu0_intr_243	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_7	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_116	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_mailbox_cluster_pend_3	WKUP_R5FSS0_CORE0_intr_243	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_3 interrupt request	level

4.4.1.7 MAILBOX_MAILBOX_CLUSTER_4 Integration

Table 4-149. MAILBOX_MAILBOX_CLUSTER_4 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_4	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-150. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_4

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_4	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-151. Clock Integration for MAILBOX_MAILBOX_CLUSTER_4

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_4	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-152. Reset Integration for MAILBOX_MAILBOX_CLUSTER_4

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_4	PSC0	MAILBOX0_MAILBOX_CLUSTER_4 reset

Table 4-153. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_4

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_4	MAILBOX0_MAILBOX_CLUSTER_4_mailbox_cluster_pend_0	WKUP_R5FSS0_CORE0_intr_115	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_4 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_4	MAILBOX0_MAILBOX_CLUSTER_4_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_8	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_4 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_4	MAILBOX0_MAILBOX_CLUSTER_4_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_8	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_4 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_4	MAILBOX0_MAILBOX_CLUSTER_4_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_240	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_4 interrupt request	level

4.4.1.8 MAILBOX_MAILBOX_CLUSTER_5 Integration

Table 4-154. MAILBOX_MAILBOX_CLUSTER_5 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_5	✓		

Table 4-155. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_5

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_5	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-156. Clock Integration for MAILBOX_MAILBOX_CLUSTER_5

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_5	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-157. Reset Integration for MAILBOX_MAILBOX_CLUSTER_5

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_5	PSC0	MAILBOX0_MAILBOX_CLUSTER_5 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-158. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_5

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_5	MAILBOX0_MAILBOX_CLUSTER_5_mailbox_cluster_pend_0	WKUP_R5FSS0_CORE0_intr_116	WKUP_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_5 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_5	MAILBOX0_MAILBOX_CLUSTER_5_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_9	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_5 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_5	MAILBOX0_MAILBOX_CLUSTER_5_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_9	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_5 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_5	MAILBOX0_MAILBOX_CLUSTER_5_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_241	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_5 interrupt request	level

4.4.1.9 MAILBOX_MAILBOX_CLUSTER_6 Integration

Table 4-159. MAILBOX_MAILBOX_CLUSTER_6 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_6	✓		

Table 4-160. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_6

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_6	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-161. Clock Integration for MAILBOX_MAILBOX_CLUSTER_6

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_6	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-162. Reset Integration for MAILBOX_MAILBOX_CLUSTER_6

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_6	PSC0	MAILBOX0_MAILBOX_CLUSTER_6 reset

Table 4-163. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_6

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_6	MAILBOX0_MAILBOX_CLUSTER_6_mailbox_cluster_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_115	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_6 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_6	MAILBOX0_MAILBOX_CLUSTER_6_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_10	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_6 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_6	MAILBOX0_MAILBOX_CLUSTER_6_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_10	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_6 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_6	MAILBOX0_MAILBOX_CLUSTER_6_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_242	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_6 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.4.1.10 MAILBOX_MAILBOX_CLUSTER_7 Integration

Table 4-164. MAILBOX_MAILBOX_CLUSTER_7 Module Allocations

Instance	MAIN	MCU	WKUP
MAILBOX0_MAILBOX_CLUSTER_7	✓		

Table 4-165. Integration Attributes for MAILBOX_MAILBOX_CLUSTER_7

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAILBOX0_MAILBOX_CLUSTER_7	PSC0	GP_CORE	LPSC_main_sms_common	27	ON	YES	LPSC_main_alwayson

Table 4-166. Clock Integration for MAILBOX_MAILBOX_CLUSTER_7

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MAILBOX0_MAILBOX_CLUSTER_7	VCLK_CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-167. Reset Integration for MAILBOX_MAILBOX_CLUSTER_7

Module Instance	Source	Description
MAILBOX0_MAILBOX_CLUSTER_7	PSC0	MAILBOX0_MAILBOX_CLUSTER_7 reset

Table 4-168. Hardware Requests for MAILBOX_MAILBOX_CLUSTER_7

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_mailbox_cluster_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_116	MCU_R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_7 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_mailbox_cluster_pend_1	C7X256V0_CLEC_soc_events_in_11	C7X256V0_CLEC	MAILBOX0_MAILBOX_CLUSTER_7 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_mailbox_cluster_pend_2	C7X256V1_CLEC_soc_events_in_11	C7X256V1_CLEC	MAILBOX0_MAILBOX_CLUSTER_7 interrupt request	level
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_mailbox_cluster_pend_3	R5FSS0_CORE0_intr_243	R5FSS0_CORE0	MAILBOX0_MAILBOX_CLUSTER_7 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.4.2 Spinlock

This section contains the integration details for the Spinlock module on this device. For further information, see the *Spinlock* section of the *Interprocessor Communication* chapter.

4.4.2.1 SPINLOCK Unsupported Features

The following features are not supported on this family of devices:

- ARM architectural Spinlock instructions (LDREX,STREX)
- Any use model other than binary mutex (for example, counting semaphore, lockless programming): Spinlock MMR is a single binary 0,1 implemented by a state machine.
- 64-bit accesses: Spinlock MMRs are aligned on 32-bit boundaries meaning a 64-bit access affects the state of 2 Spinlocks.

4.4.2.2 SPINLOCK Integration

Table 4-169. SPINLOCK Module Allocations

Instance	MAIN	MCU	WKUP
SPINLOCK0	✓		

Table 4-170. Clock Integration for SPINLOCK

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
SPINLOCK0	FICLK	MAIN_PLL15_HSDIV0_CLKOUT/2		functional and interface clock

Copyright © 2026 Texas Instruments Incorporated

4.5 Device Configuration

4.5.1 Control Module (CTRL_MMR)

This section contains the integration details for the CTRL_MMR module on this device. For further information, see the *Memory Mapped Control Register Modules (CTRL_MMR)* section of the *Device Configuration* chapter.

4.5.1.1 MAIN_CTRL_MMR Integration

Table 4-171. MAIN_CTRL_MMR Module Allocations

Instance	MAIN	MCU	WKUP
CTRL_MMR0	✓		

Table 4-172. Clock Integration for MAIN_CTRL_MMR

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CTRL_MMR0	VBUSP_CLK	MAIN_SYSCLK0/4		None

Table 4-173. Hardware Requests for MAIN_CTRL_MMR

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CTRL_MMR0	CTRL_MMR0_IPC_SET0_ipc_set_ipcfg_0	C7X256V0_CLEC_soc_events_in_5	C7X256V0_CLEC	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_IPC_SET1_ipc_set_ipcfg_0	C7X256V1_CLEC_soc_events_in_5	C7X256V1_CLEC	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_39	R5FSS0_CORE0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_39	MCU_R5FSS0_CORE0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	GICSS0_spi_129	GICSS0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_128	R5FSS0_CORE0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	WKUP_R5FSS0_CORE0_intr_128	WKUP_R5FSS0_CORE0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_128	MCU_R5FSS0_CORE0	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	C7X256V0_CLEC_gic_spi_129	C7X256V0_CLEC	CTRL_MMR0 interrupt request	level
CTRL_MMR0	CTRL_MMR0_access_err_0	C7X256V1_CLEC_gic_spi_129	C7X256V1_CLEC	CTRL_MMR0 interrupt request	level

4.5.1.2 WKUP_CTRL_MMR Integration

Table 4-174. WKUP_CTRL_MMR Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_CTRL_MMR0			✓

Table 4-175. Clock Integration for WKUP_CTRL_MMR

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_CTRL_MMR0	VBUSP_CLK	DM_CLK/4		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-176. Hardware Requests for WKUP_CTRL_MMR

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_39	R5FSS0_CORE0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_39	MCU_R5FSS0_CORE0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	GICSS0_spi_129	GICSS0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_128	R5FSS0_CORE0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	WKUP_R5FSS0_CORE0_intr_128	WKUP_R5FSS0_CORE0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_128	MCU_R5FSS0_CORE0	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	C7X256V0_CLEC_gic_spi_129	C7X256V0_CLEC	WKUP_CTRL_MMR0 interrupt request	level
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_access_err_0	C7X256V1_CLEC_gic_spi_129	C7X256V1_CLEC	WKUP_CTRL_MMR0 interrupt request	level

4.5.1.3 MCU_CTRL_MMR Integration

Table 4-177. MCU_CTRL_MMR Module Allocations

Instance	MAIN	MCU	WKUP
MCU_CTRL_MMR0		✓	

Table 4-178. Clock Integration for MCU_CTRL_MMR

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_CTRL_MMR0	VBUSP_CLK	MCU_SYSCLK0/4		None

Table 4-179. Hardware Requests for MCU_CTRL_MMR

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_CTRL_MMR0	MCU_CTRL_MMR0_IPC_SET0_ipc_set_ipcfg_0	WKUP_R5FSS0_CORE0_intr_0	WKUP_R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_39	R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_39	MCU_R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	GICSS0_spi_129	GICSS0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	R5FSS0_CORE0_intr_128	R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	WKUP_R5FSS0_CORE0_intr_128	WKUP_R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_128	MCU_R5FSS0_CORE0	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	C7X256V0_CLEC_gic_spi_129	C7X256V0_CLEC	MCU_CTRL_MMR0 interrupt request	level
MCU_CTRL_MMR0	MCU_CTRL_MMR0_access_err_0	C7X256V1_CLEC_gic_spi_129	C7X256V1_CLEC	MCU_CTRL_MMR0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.5.2 PADCFG_CTRL

This section contains the integration details for the PADCFG_CTRL module on this device. For further information, see *Pad Configuration Registers.*

4.5.2.1 MAIN_PADCFG_CTRL Integration

Table 4-180. MAIN_PADCFG_CTRL Module Allocations

Instance	MAIN	MCU	WKUP
PADCFG_CTRL0	✓		

Table 4-181. Clock Integration for MAIN_PADCFG_CTRL

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PADCFG_CTRL0	VBUSP_CLK	MAIN_SYSCLK0/4		None

Table 4-182. Hardware Requests for MAIN_PADCFG_CTRL

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	R5FSS0_CORE0_intr_39	R5FSS0_CORE0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_39	MCU_R5FSS0_CORE0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	GICSS0_spi_129	GICSS0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	R5FSS0_CORE0_intr_128	R5FSS0_CORE0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	WKUP_R5FSS0_CORE0_intr_128	WKUP_R5FSS0_CORE0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_128	MCU_R5FSS0_CORE0	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	C7X256V0_CLEC_gic_spi_129	C7X256V0_CLEC	PADCFG_CTRL0 interrupt request	level
PADCFG_CTRL0	PADCFG_CTRL0_access_err_0	C7X256V1_CLEC_gic_spi_129	C7X256V1_CLEC	PADCFG_CTRL0 interrupt request	level

4.5.2.2 MCU_PADCFG_CTRL Integration

Table 4-183. MCU_PADCFG_CTRL Module Allocations

Instance	MAIN	MCU	WKUP
MCU_PADCFG_CTRL0		✓	

Table 4-184. Clock Integration for MCU_PADCFG_CTRL

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_PADCFG_CTRL0	VBUSP_CLK	MCU_SYSCLK0/4		None

Table 4-185. Hardware Requests for MCU_PADCFG_CTRL

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	R5FSS0_CORE0_intr_39	R5FSS0_CORE0	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_39	MCU_R5FSS0_CORE0	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	GICSS0_spi_129	GICSS0	MCU_PADCFG_CTRL0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-185. Hardware Requests for MCU_PADCFG_CTRL (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	R5FSS0_CORE0_intr_128	R5FSS0_CORE0	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	WKUP_R5FSS0_CORE0_intr_128	WKUP_R5FSS0_CORE0	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	MCU_R5FSS0_CORE0_cpu0_intr_128	MCU_R5FSS0_CORE0	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	C7X256V0_CLEC_gic_spi_129	C7X256V0_CLEC	MCU_PADCFG_CTRL0 interrupt request	level
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_access_err_0	C7X256V1_CLEC_gic_spi_129	C7X256V1_CLEC	MCU_PADCFG_CTRL0 interrupt request	level

4.5.3 Voltage and Thermal Manager (VTM)

This section contains the integration details for the VTM module on this device. For further information, see the *Voltage and Thermal Manager (VTM)* in *Power* section of the *Device Configuration* chapter.

4.5.3.1 VTM Unsupported Features

The following features are not supported on this family of devices:

- AVS Class 0 support
- Maximum temperature can assert even if the vbusclock is halted or the device is in reset.

4.5.3.2 VTM Integration

Table 4-186. VTM Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_VTM0			✓

Table 4-187. Integration Attributes for VTM

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_VTM0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-188. Clock Integration for VTM

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_VTM0	FIX_REF2_CLK	CLK_12M_RC		reference clock
WKUP_VTM0	FIX_REF_CLK	CLK_12M_RC		reference clock
WKUP_VTM0	FIX_REF_CLK	HFOSC0 (INSTANCE)		reference clock
WKUP_VTM0	ICLK	DM_CLK/4		interface clock

Table 4-189. Reset Integration for VTM

Module Instance	Source	Description
WKUP_VTM0	PSC0	WKUP_VTM0 reset

Table 4-190. Hardware Requests for VTM

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_VTM0	WKUP_VTM0_corr_level_0	WKUP_ESM0_esm_lvl_event_11	WKUP_ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_corr_level_0	ESM0_esm_lvl_event_139	ESM0	WKUP_VTM0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-190. Hardware Requests for VTM (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	WKUP_ESM0_esm_lvl_event_8	WKUP_ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	ESM0_esm_lvl_event_137	ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	GICSS0_spi_183	GICSS0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	R5FSS0_CORE0_intr_183	R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	WKUP_R5FSS0_CORE0_intr_183	WKUP_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_183	MCU_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	C7X256V0_CLEC_gic_spi_183	C7X256V0_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th1_intr_0	C7X256V1_CLEC_gic_spi_183	C7X256V1_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	WKUP_ESM0_esm_lvl_event_10	WKUP_ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	ESM0_esm_lvl_event_138	ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	GICSS0_spi_184	GICSS0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	R5FSS0_CORE0_intr_184	R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	WKUP_R5FSS0_CORE0_intr_184	WKUP_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_184	MCU_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	C7X256V0_CLEC_gic_spi_184	C7X256V0_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_gt_th2_intr_0	C7X256V1_CLEC_gic_spi_184	C7X256V1_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	WKUP_ESM0_esm_lvl_event_9	WKUP_ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	ESM0_esm_lvl_event_136	ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	GICSS0_spi_185	GICSS0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	R5FSS0_CORE0_intr_185	R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	WKUP_R5FSS0_CORE0_intr_185	WKUP_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_185	MCU_R5FSS0_CORE0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	C7X256V0_CLEC_gic_spi_185	C7X256V0_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_therm_lvl_lt_th0_intr_0	C7X256V1_CLEC_gic_spi_185	C7X256V1_CLEC	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_uncorr_level_0	WKUP_ESM0_esm_lvl_event_12	WKUP_ESM0	WKUP_VTM0 interrupt request	level
WKUP_VTM0	WKUP_VTM0_uncorr_level_0	ESM0_esm_lvl_event_140	ESM0	WKUP_VTM0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.5.4 Power Sleep Controller (PSC)

This section contains the integration details for the PSC module on this device. For further information, see the *Power Sleep Controller (PSC)* in *Power* section of the *Device Configuration* chapter.

4.5.4.1 PSC Integration

Table 4-191. PSC Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_PSC0			✓
PSC0	✓		

Table 4-192. Clock Integration for PSC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_PSC0	FICLK	MCU_SYSCLK0/4		functional and interface clock
WKUP_PSC0	SLOW_SYSCLK	MCU_SYSCLK0/24		slow system clock
PSC0	CLK	MAIN_SYSCLK0/24		None
PSC0	SLOW_CLK	MAIN_SYSCLK0/24		None

Table 4-193. Hardware Requests for PSC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_PSC0	WKUP_PSC0_psc_allint_0	R5FSS0_CORE0_intr_145	R5FSS0_CORE0	WKUP_PSC0 interrupt request	pulse
WKUP_PSC0	WKUP_PSC0_psc_allint_0	WKUP_R5FSS0_CORE0_intr_145	WKUP_R5FSS0_CORE0	WKUP_PSC0 interrupt request	pulse
WKUP_PSC0	WKUP_PSC0_psc_allint_0	MCU_R5FSS0_CORE0_cpu0_intr_145	MCU_R5FSS0_CORE0	WKUP_PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	GICSS0_spi_203	GICSS0	PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	R5FSS0_CORE0_intr_146	R5FSS0_CORE0	PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	WKUP_R5FSS0_CORE0_intr_146	WKUP_R5FSS0_CORE0	PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	MCU_R5FSS0_CORE0_cpu0_intr_146	MCU_R5FSS0_CORE0	PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	C7X256V0_CLEC_gic_spi_203	C7X256V0_CLEC	PSC0 interrupt request	pulse
PSC0	PSC0_psc_allint_0	C7X256V1_CLEC_gic_spi_203	C7X256V1_CLEC	PSC0 interrupt request	pulse

4.5.4.2 PSC_ECC_AGGR_0 Integration

Table 4-194. PSC_ECC_AGGR_0 Module Allocations

Instance	MAIN	MCU	WKUP
PSC0_ECC_AGGR_0	✓		

Table 4-195. Clock Integration for PSC_ECC_AGGR_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PSC0_ECC_AGGR_0	FICLK	MAIN_SYSCLK0/24		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-196. Hardware Requests for PSC_ECC_AGGR_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PSC0_ECC_AGGR_0	PSC0_ECC_AGGR_0_fw_ch_br_ecc_aggr_corr_level_0	ESM0_esm_lvl_event_22	ESM0	PSC0_ECC_AGGR_0 interrupt request	level
PSC0_ECC_AGGR_0	PSC0_ECC_AGGR_0_fw_ch_br_ecc_aggr_uncorr_level_0	ESM0_esm_lvl_event_23	ESM0	PSC0_ECC_AGGR_0 interrupt request	level

4.5.4.3 PSC_FW_0 Integration

Table 4-197. PSC_FW_0 Module Allocations

Instance	MAIN	MCU	WKUP
PSC0_FW_0	✓		

Table 4-198. Clock Integration for PSC_FW_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PSC0_FW_0	FICLK	MAIN_SYSCLK0/24		functional and interface clock

Table 4-199. Hardware Requests for PSC_FW_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PSC0_FW_0	PSC0_FW_0_fw_ch_br_default_exp_0	ESM0_esm_lvl_event_76	ESM0	PSC0_FW_0 interrupt request	level

4.5.5 Clocking

4.5.5.1 PLLCTRL Integration

Table 4-200. PLLCTRL Module Allocations

Instance	MAIN	MCU	WKUP
PLLCTRL0	✓		
MCU_PLLCTRL0		✓	

Table 4-201. Clock Integration for PLLCTRL

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PLLCTRL0	PLL_CLKOUT_CLK	MAIN_PLL0_HSDIV0_CLKOUT		None
PLLCTRL0	PLL_REFCLK_CLK	CLK_12M_RC		None
PLLCTRL0	PLL_REFCLK_CLK	HFOSC0 (INSTANCE)		None
PLLCTRL0	VBUS_SLV_REFCLK_CLK	MAIN_SYSCLK0/4		None
MCU_PLLCTRL0	PLL_CLKOUT_CLK	MCU_PLL0_HSDIV0_CLKOUT		None
MCU_PLLCTRL0	PLL_REFCLK_CLK	CLK_12M_RC		None
MCU_PLLCTRL0	PLL_REFCLK_CLK	HFOSC0 (INSTANCE)		None
MCU_PLLCTRL0	VBUS_SLV_REFCLK_CLK	MCU_SYSCLK0/4		None

Copyright © 2026 Texas Instruments Incorporated

4.5.5.2 main_pll Integration

Table 4-202. MAIN_PLL Module Allocations

Instance	MAIN	MCU	WKUP
PLL0	✓		

Table 4-203. Integration Attributes for main_pll

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
PLL0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-204. Clock Integration for main_pll

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PLL0	VBUSP_CLK	MAIN_SYSCLK0/4		None

Table 4-205. Reset Integration for main_pll

Module Instance	Source	Description
PLL0	PSC0	PLL0 reset

4.5.5.3 MCU_PLL Integration

Table 4-206. MCU_PLL Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_PLL0			✓

Table 4-207. Integration Attributes for MCU_PLL

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_PLL0	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	

Table 4-208. Clock Integration for MCU_PLL

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_PLL0	VBUSP_CLK	MCU_SYSCLK0/4		None

Table 4-209. Reset Integration for MCU_PLL

Module Instance	Source	Description
WKUP_PLL0	WKUP_PSC0	WKUP_PLL0 reset

4.5.5.4 pllfracf2_ssmod_16fft Integration

Table 4-210. PLLFRACF2_SSMOD_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
pllfracf2_ssmod_16fft0	✓		
pllfracf2_ssmod_16fft1	✓		
pllfracf2_ssmod_16fft12	✓		
pllfracf2_ssmod_16fft15	✓		
pllfracf2_ssmod_16fft16	✓		
pllfracf2_ssmod_16fft17	✓		
pllfracf2_ssmod_16fft18	✓		
pllfracf2_ssmod_16fft2	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-210. PLLFRACF2_SSMOD_16FFT Module Allocations (continued)

Instance	MAIN	MCU	WKUP
pllfracf2_ssmod_16fft5	✓		
pllfracf2_ssmod_16fft6	✓		
pllfracf2_ssmod_16fft7	✓		
pllfracf2_ssmod_16fft8	✓		
MCU_pllfracf2_ssmod_16fft0		✓	

Table 4-211. Clock Integration for pllfracf2_ssmod_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
pllfracf2_ssmod_16fft0	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft0	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft1	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft1	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft12	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft12	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft15	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft15	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft16	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft16	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft17	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft17	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft18	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft18	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft2	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft2	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft5	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft5	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft6	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft6	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft7	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft7	FREF_CLK	HFOSC0 (INSTANCE)		None
pllfracf2_ssmod_16fft8	FREF_CLK	CLK_12M_RC		None
pllfracf2_ssmod_16fft8	FREF_CLK	HFOSC0 (INSTANCE)		None
MCU_pllfracf2_ssmod_16fft0	FREF_CLK	CLK_12M_RC		None
MCU_pllfracf2_ssmod_16fft0	FREF_CLK	HFOSC0 (INSTANCE)		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-212. Hardware Requests for pllfracf2_ssmod_16fft

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
pllfracf2_ssmod_16fft0	pllfracf2_ssmod_16fft0_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_54	WKUP_ESM0	pllfracf2_ssmod_16fft0 interrupt request	level
pllfracf2_ssmod_16fft0	pllfracf2_ssmod_16fft0_lockloss_ipcfg_0	ESM0_esm_lvl_event_128	ESM0	pllfracf2_ssmod_16fft0 interrupt request	level
pllfracf2_ssmod_16fft1	pllfracf2_ssmod_16fft1_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_55	WKUP_ESM0	pllfracf2_ssmod_16fft1 interrupt request	level
pllfracf2_ssmod_16fft1	pllfracf2_ssmod_16fft1_lockloss_ipcfg_0	ESM0_esm_lvl_event_129	ESM0	pllfracf2_ssmod_16fft1 interrupt request	level
pllfracf2_ssmod_16fft12	pllfracf2_ssmod_16fft12_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_58	WKUP_ESM0	pllfracf2_ssmod_16fft12 interrupt request	level
pllfracf2_ssmod_16fft12	pllfracf2_ssmod_16fft12_lockloss_ipcfg_0	ESM0_esm_lvl_event_132	ESM0	pllfracf2_ssmod_16fft12 interrupt request	level
pllfracf2_ssmod_16fft15	pllfracf2_ssmod_16fft15_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_59	WKUP_ESM0	pllfracf2_ssmod_16fft15 interrupt request	level
pllfracf2_ssmod_16fft15	pllfracf2_ssmod_16fft15_lockloss_ipcfg_0	ESM0_esm_lvl_event_133	ESM0	pllfracf2_ssmod_16fft15 interrupt request	level
pllfracf2_ssmod_16fft16	pllfracf2_ssmod_16fft16_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_52	WKUP_ESM0	pllfracf2_ssmod_16fft16 interrupt request	level
pllfracf2_ssmod_16fft16	pllfracf2_ssmod_16fft16_lockloss_ipcfg_0	ESM0_esm_lvl_event_27	ESM0	pllfracf2_ssmod_16fft16 interrupt request	level
pllfracf2_ssmod_16fft17	pllfracf2_ssmod_16fft17_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_62	WKUP_ESM0	pllfracf2_ssmod_16fft17 interrupt request	level
pllfracf2_ssmod_16fft17	pllfracf2_ssmod_16fft17_lockloss_ipcfg_0	ESM0_esm_lvl_event_7	ESM0	pllfracf2_ssmod_16fft17 interrupt request	level
pllfracf2_ssmod_16fft18	pllfracf2_ssmod_16fft18_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_51	WKUP_ESM0	pllfracf2_ssmod_16fft18 interrupt request	level
pllfracf2_ssmod_16fft18	pllfracf2_ssmod_16fft18_lockloss_ipcfg_0	ESM0_esm_lvl_event_43	ESM0	pllfracf2_ssmod_16fft18 interrupt request	level
pllfracf2_ssmod_16fft2	pllfracf2_ssmod_16fft2_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_56	WKUP_ESM0	pllfracf2_ssmod_16fft2 interrupt request	level
pllfracf2_ssmod_16fft2	pllfracf2_ssmod_16fft2_lockloss_ipcfg_0	ESM0_esm_lvl_event_130	ESM0	pllfracf2_ssmod_16fft2 interrupt request	level
pllfracf2_ssmod_16fft5	pllfracf2_ssmod_16fft5_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_60	WKUP_ESM0	pllfracf2_ssmod_16fft5 interrupt request	level
pllfracf2_ssmod_16fft5	pllfracf2_ssmod_16fft5_lockloss_ipcfg_0	ESM0_esm_lvl_event_17	ESM0	pllfracf2_ssmod_16fft5 interrupt request	level
pllfracf2_ssmod_16fft6	pllfracf2_ssmod_16fft6_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_53	WKUP_ESM0	pllfracf2_ssmod_16fft6 interrupt request	level
pllfracf2_ssmod_16fft6	pllfracf2_ssmod_16fft6_lockloss_ipcfg_0	ESM0_esm_lvl_event_19	ESM0	pllfracf2_ssmod_16fft6 interrupt request	level
pllfracf2_ssmod_16fft7	pllfracf2_ssmod_16fft7_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_61	WKUP_ESM0	pllfracf2_ssmod_16fft7 interrupt request	level
pllfracf2_ssmod_16fft7	pllfracf2_ssmod_16fft7_lockloss_ipcfg_0	ESM0_esm_lvl_event_18	ESM0	pllfracf2_ssmod_16fft7 interrupt request	level
pllfracf2_ssmod_16fft8	pllfracf2_ssmod_16fft8_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_57	WKUP_ESM0	pllfracf2_ssmod_16fft8 interrupt request	level
pllfracf2_ssmod_16fft8	pllfracf2_ssmod_16fft8_lockloss_ipcfg_0	ESM0_esm_lvl_event_131	ESM0	pllfracf2_ssmod_16fft8 interrupt request	level
MCU_pllfracf2_ssmod_16fft0	MCU_pllfracf2_ssmod_16fft0_lockloss_ipcfg_0	WKUP_ESM0_esm_lvl_event_63	WKUP_ESM0	MCU_pllfracf2_ssmod_16fft0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-212. Hardware Requests for pllfracf2_ssmod_16fft (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_pllfracf2_ssmod_16fft0	MCU_pllfracf2_ssmod_16fft0_lockloss_ipcfg_0	ESM0_esm_lvl_event_134	ESM0	MCU_pllfracf2_ssmod_16fft0 interrupt request	level

4.5.5.5 hsdiv0_16fft Integration

Table 4-213. HSDIV0_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
hsdiv0_16fft12	✓		
hsdiv0_16fft16	✓		
hsdiv0_16fft17	✓		
hsdiv0_16fft18	✓		
hsdiv0_16fft6	✓		
hsdiv0_16fft8	✓		
MCU_hsdiv0_16fft32		✓	

Table 4-214. Clock Integration for hsdiv0_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
hsdiv0_16fft12	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft12	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv0_16fft16	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft16	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv0_16fft17	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft17	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv0_16fft18	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft18	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv0_16fft6	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft6	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv0_16fft8	FREF_CLK	CLK_12M_RC		None
hsdiv0_16fft8	FREF_CLK	HFOSC0 (INSTANCE)		None
MCU_hsdiv0_16fft32	FREF_CLK	HFOSC0 (INSTANCE)		None
MCU_hsdiv0_16fft32	VCO_CLKIN_CLK	CLK_12M_RC		None
MCU_hsdiv0_16fft32	VCO_CLKIN_CLK	HFOSC0 (INSTANCE)		None

4.5.5.6 hsdiv1_16fft Integration

Table 4-215. HSDIV1_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
hsdiv1_16fft7	✓		

Table 4-216. Clock Integration for hsdiv1_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
hsdiv1_16fft7	FREF_CLK	CLK_12M_RC		None
hsdiv1_16fft7	FREF_CLK	HFOSC0 (INSTANCE)		None

Copyright © 2026 Texas Instruments Incorporated

4.5.5.7 hsdiv2_16fft Integration

Table 4-217. HSDIV2_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
hsdiv2_16fft5	✓		

Table 4-218. Clock Integration for hsdiv2_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
hsdiv2_16fft5	FREF_CLK	CLK_12M_RC		None
hsdiv2_16fft5	FREF_CLK	HFOSC0 (INSTANCE)		None

4.5.5.8 hsdiv3_16fft Integration

Table 4-219. HSDIV3_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
hsdiv3_16fft15	✓		

Table 4-220. Clock Integration for hsdiv3_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
hsdiv3_16fft15	FREF_CLK	CLK_12M_RC		None
hsdiv3_16fft15	FREF_CLK	HFOSC0 (INSTANCE)		None

4.5.5.9 hsdiv4_16fft Integration

Table 4-221. HSDIV4_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
hsdiv4_16fft0	✓		
hsdiv4_16fft1	✓		
hsdiv4_16fft2	✓		
MCU_hsdiv4_16fft0		✓	

Table 4-222. Clock Integration for hsdiv4_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
hsdiv4_16fft0	FREF_CLK	CLK_12M_RC		None
hsdiv4_16fft0	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv4_16fft1	FREF_CLK	CLK_12M_RC		None
hsdiv4_16fft1	FREF_CLK	HFOSC0 (INSTANCE)		None
hsdiv4_16fft2	FREF_CLK	CLK_12M_RC		None
hsdiv4_16fft2	FREF_CLK	HFOSC0 (INSTANCE)		None
MCU_hsdiv4_16fft0	FREF_CLK	CLK_12M_RC		None
MCU_hsdiv4_16fft0	FREF_CLK	HFOSC0 (INSTANCE)		None

4.5.5.10 postdiv1_16fft Integration

Table 4-223. POSTDIV1_16FFT Module Allocations

Instance	MAIN	MCU	WKUP
postdiv1_16fft1	✓		
MCU_postdiv1_16fft0		✓	

Copyright © 2026 Texas Instruments Incorporated

Table 4-224. Clock Integration for postdiv1_16fft

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
postdiv1_16fft1	FREF_CLK	CLK_12M_RC		None
postdiv1_16fft1	FREF_CLK	HFOSC0 (INSTANCE)		None
MCU_postdiv1_16fft0	FREF_CLK	CLK_12M_RC		None
MCU_postdiv1_16fft0	FREF_CLK	HFOSC0 (INSTANCE)		None

4.5.5.11 postdiv4_16ff Integration

Table 4-225. POSTDIV4_16FF Module Allocations

Instance	MAIN	MCU	WKUP
postdiv4_16ff0	✓		
postdiv4_16ff2	✓		

Table 4-226. Clock Integration for postdiv4_16ff

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
postdiv4_16ff0	FREF_CLK	CLK_12M_RC		None
postdiv4_16ff0	FREF_CLK	HFOSC0 (INSTANCE)		None
postdiv4_16ff2	FREF_CLK	CLK_12M_RC		None
postdiv4_16ff2	FREF_CLK	HFOSC0 (INSTANCE)		None

Copyright © 2026 Texas Instruments Incorporated

4.6 Interrupts

4.6.1 TIMESYNC_EVENT_INTROUTER

This section contains the integration details for the TIMESYNC_EVENT_INTROUTER modules on this device.

4.6.1.1 timesync_event_introuter Integration

Table 4-227. TIMESYNC_EVENT_INTROUTER Module Allocations

Instance	MAIN	MCU	WKUP
TIMESYNC_EVENT_INTROUTER0	✓		

Table 4-228. Integration Attributes for timesync_event_introuter

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
TIMESYNC_EVENT_INTROUTER0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-229. Clock Integration for timesync_event_introuter

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMESYNC_EVENT_INTROUTER0	FICLK	MAIN_SYSCLK0/4		functional and interface clock

Table 4-230. Reset Integration for timesync_event_introuter

Module Instance	Source	Description
TIMESYNC_EVENT_INTROUTER0	PSC0	TIMESYNC_EVENT_INTROUTER0 reset

Table 4-231. Hardware Requests for timesync_event_introuter

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_0	DMASS0_INTAGGR_0_intaggr_levi_pend_8	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_1	DMASS0_INTAGGR_0_intaggr_levi_pend_9	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_2	DMASS0_INTAGGR_0_intaggr_levi_pend_10	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_3	DMASS0_INTAGGR_0_intaggr_levi_pend_11	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_4	DMASS0_INTAGGR_0_intaggr_levi_pend_12	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_5	DMASS0_INTAGGR_0_intaggr_levi_pend_13	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_6	DMASS0_INTAGGR_0_intaggr_levi_pend_14	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_7	DMASS0_INTAGGR_0_intaggr_levi_pend_15	DMASS0_INTAGGR_0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_8	PCIE0_pcie_cpts_hw2_push_0	PCIE0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_10	CPSW0_cpts_hw1_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-231. Hardware Requests for timesync_event_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_11	CPSW0_cpts_hw2_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_12	CPSW0_cpts_hw3_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_13	CPSW0_cpts_hw4_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_14	CPSW0_cpts_hw5_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_15	CPSW0_cpts_hw6_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_16	CPSW0_cpts_hw7_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_17	CPSW0_cpts_hw8_push_0	CPSW0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_19	EPWM0_epwm_syncin_0	EPWM0	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_20	PINFUNCTION_SYNC0_OUTout_SYNC0_OUT_0	PINFUNCTION_SYNC0_OUTout	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_21	PINFUNCTION_SYNC1_OUTout_SYNC1_OUT_0	PINFUNCTION_SYNC1_OUTout	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_22	PINFUNCTION_SYNC2_OUTout_SYNC2_OUT_0	PINFUNCTION_SYNC2_OUTout	TIMESYNC_EVENT_INTROUTER0 interrupt request	level
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_outl_23	PINFUNCTION_SYNC3_OUTout_SYNC3_OUT_0	PINFUNCTION_SYNC3_OUTout	TIMESYNC_EVENT_INTROUTER0 interrupt request	level

4.6.2 Generic Interrupt Controller Subsystem (GICSS)

This section contains the integration details for the GICSS modules on this device. For Further information, see the *Generic Interrupt Controller (GICSS)* section of the *Interrupts* chapter.

4.6.2.1 GICSS Unsupported Features

The following features are not supported on this family of devices:

- GICv2 backwards compatibility
- AWID-based LPI mapping

4.6.2.2 gicss Integration

Table 4-232. GICSS Module Allocations

Instance	MAIN	MCU	WKUP
GICSS0	✓		

Table 4-233. Integration Attributes for gicss

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
GICSS0	PSC0	GP_CORE	LPSC_main_gic	36	ON	YES	LPSC_main_ip

Table 4-234. Clock Integration for gicss

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GICSS0	FICLK	MAIN_SYSCLK0/2		functional and interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-235. Reset Integration for gicss

Module Instance	Source	Description
GICSS0	PSC0	GICSS0 reset

Table 4-236. Hardware Requests for gicss

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GICSS0	GICSS0_axim_err_0	ESM0_esm_pls_event 0_230	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_axim_err_0	ESM0_esm_pls_event 1_230	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_axim_err_0	ESM0_esm_pls_event 2_230	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_ecc_aggr_co rr_level_0	ESM0_esm_lvl_event _12	ESM0	GICSS0 interrupt request	level
GICSS0	GICSS0_ecc_aggr_un corr_level_0	ESM0_esm_lvl_event _75	ESM0	GICSS0 interrupt request	level
GICSS0	GICSS0_ecc_fatal_0	ESM0_esm_pls_event 0_231	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_ecc_fatal_0	ESM0_esm_pls_event 1_231	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_ecc_fatal_0	ESM0_esm_pls_event 2_231	ESM0	GICSS0 interrupt request	pulse
GICSS0	GICSS0_gic_pwr0_wa ke_request_0	WKUP_R5FSS0_COR E0_intr_154	WKUP_R5FSS0_COR E0	GICSS0 interrupt request	level
GICSS0	GICSS0_gic_pwr0_wa ke_request_1	WKUP_R5FSS0_COR E0_intr_155	WKUP_R5FSS0_COR E0	GICSS0 interrupt request	level
GICSS0	GICSS0_gic_pwr0_wa ke_request_2	WKUP_R5FSS0_COR E0_intr_156	WKUP_R5FSS0_COR E0	GICSS0 interrupt request	level
GICSS0	GICSS0_gic_pwr0_wa ke_request_3	WKUP_R5FSS0_COR E0_intr_157	WKUP_R5FSS0_COR E0	GICSS0 interrupt request	level

4.6.2.3 GIC Integration

Table 4-237. GIC Module Allocations

Instance	MAIN	MCU	WKUP
C7X256V0_GICSS	✓		
C7X256V1_GICSS	✓		

4.6.3 main_gpiomux_introuter Integration

Table 4-238. MAIN_GPIOMUX_INTROUTER Module Allocations

Instance	MAIN	MCU	WKUP
MAIN_GPIOMUX_INTROUTER0	✓		

Table 4-239. Integration Attributes for main_gpiomux_introuter

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MAIN_GPIOMUX_INTROUTER0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-240. Reset Integration for main_gpiomux_introuter

Module Instance	Source	Description
MAIN_GPIOMUX_INTROUTER0	PSC0	MAIN_GPIOMUX_INTROUTER0 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-241. Hardware Requests for main_gpiomux_introuter

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_0	GICSS0_spi_32	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_0	WKUP_R5FSS0_CORE0_intr_32	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_0	C7X256V0_CLEC_gic_spi_32	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_0	C7X256V1_CLEC_gic_spi_32	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_1	GICSS0_spi_33	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_1	WKUP_R5FSS0_CORE0_intr_33	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_1	C7X256V0_CLEC_gic_spi_33	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_1	C7X256V1_CLEC_gic_spi_33	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_2	GICSS0_spi_34	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_2	WKUP_R5FSS0_CORE0_intr_34	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_2	C7X256V0_CLEC_gic_spi_34	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_2	C7X256V1_CLEC_gic_spi_34	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_3	GICSS0_spi_35	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_3	WKUP_R5FSS0_CORE0_intr_35	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_3	C7X256V0_CLEC_gic_spi_35	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_3	C7X256V1_CLEC_gic_spi_35	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_4	GICSS0_spi_36	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_4	WKUP_R5FSS0_CORE0_intr_36	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_4	C7X256V0_CLEC_gic_spi_36	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_4	C7X256V1_CLEC_gic_spi_36	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_5	GICSS0_spi_37	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_5	WKUP_R5FSS0_CORE0_intr_37	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_5	C7X256V0_CLEC_gic_spi_37	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_5	C7X256V1_CLEC_gic_spi_37	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_6	GICSS0_spi_38	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_6	WKUP_R5FSS0_CORE0_intr_38	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-241. Hardware Requests for main_gpiomux_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_6	C7X256V0_CLEC_gic_spi_38	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_6	C7X256V1_CLEC_gic_spi_38	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_7	GICSS0_spi_39	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_7	WKUP_R5FSS0_CORE0_intr_39	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_7	C7X256V0_CLEC_gic_spi_39	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_7	C7X256V1_CLEC_gic_spi_39	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_8	GICSS0_spi_40	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_8	WKUP_R5FSS0_CORE0_intr_40	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_8	C7X256V0_CLEC_gic_spi_40	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_8	C7X256V1_CLEC_gic_spi_40	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_9	GICSS0_spi_41	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_9	WKUP_R5FSS0_CORE0_intr_41	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_9	C7X256V0_CLEC_gic_spi_41	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_9	C7X256V1_CLEC_gic_spi_41	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_10	GICSS0_spi_42	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_10	WKUP_R5FSS0_CORE0_intr_42	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_10	C7X256V0_CLEC_gic_spi_42	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_10	C7X256V1_CLEC_gic_spi_42	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_11	GICSS0_spi_43	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_11	WKUP_R5FSS0_CORE0_intr_43	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_11	C7X256V0_CLEC_gic_spi_43	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_11	C7X256V1_CLEC_gic_spi_43	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_12	GICSS0_spi_44	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_12	WKUP_R5FSS0_CORE0_intr_44	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_12	C7X256V0_CLEC_gic_spi_44	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_12	C7X256V1_CLEC_gic_spi_44	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-241. Hardware Requests for main_gpiomux_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_13	GICSS0_spi_45	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_13	WKUP_R5FSS0_CORE0_intr_45	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_13	C7X256V0_CLEC_gic_spi_45	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_13	C7X256V1_CLEC_gic_spi_45	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_14	GICSS0_spi_46	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_14	WKUP_R5FSS0_CORE0_intr_46	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_14	C7X256V0_CLEC_gic_spi_46	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_14	C7X256V1_CLEC_gic_spi_46	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_15	GICSS0_spi_47	GICSS0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_15	WKUP_R5FSS0_CORE0_intr_47	WKUP_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_15	C7X256V0_CLEC_gic_spi_47	C7X256V0_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_15	C7X256V1_CLEC_gic_spi_47	C7X256V1_CLEC	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_16	R5FSS0_CORE0_intr_104	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_17	R5FSS0_CORE0_intr_105	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_18	R5FSS0_CORE0_intr_106	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_19	R5FSS0_CORE0_intr_107	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_20	R5FSS0_CORE0_intr_56	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_21	R5FSS0_CORE0_intr_57	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_22	DMASS0_INTAGGR_0_intaggr_levi_pend_24	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_23	DMASS0_INTAGGR_0_intaggr_levi_pend_25	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_24	DMASS0_INTAGGR_0_intaggr_levi_pend_16	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_25	DMASS0_INTAGGR_0_intaggr_levi_pend_17	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_26	DMASS0_INTAGGR_0_intaggr_levi_pend_18	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_27	DMASS0_INTAGGR_0_intaggr_levi_pend_19	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-241. Hardware Requests for main_gpiomux_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_28	DMASS0_INTAGGR_0_intaggr_levi_pend_20	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_29	DMASS0_INTAGGR_0_intaggr_levi_pend_21	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_30	DMASS0_INTAGGR_0_intaggr_levi_pend_22	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_31	DMASS0_INTAGGR_0_intaggr_levi_pend_23	DMASS0_INTAGGR_0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_32	R5FSS0_CORE0_intr_58	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_33	R5FSS0_CORE0_intr_59	R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_34	MCU_R5FSS0_CORE0_cpu0_intr_32	MCU_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_outp_35	MCU_R5FSS0_CORE0_cpu0_intr_33	MCU_R5FSS0_CORE0	MAIN_GPIOMUX_INTROUTER0 interrupt request	pulse

4.6.4 mcu_gpiomux_introuter Integration

Table 4-242. MCU_GPIOMUX_INTROUTER Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_MCU_GPIOMUX_INTROUTER0		✓	

Table 4-243. Integration Attributes for mcu_gpiomux_introuter

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	

Table 4-244. Reset Integration for mcu_gpiomux_introuter

Module Instance	Source	Description
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_PSC0	WKUP_MCU_GPIOMUX_INTROUTER0 reset

Table 4-245. Hardware Requests for mcu_gpiomux_introuter

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_0	GICSS0_spi_104	GICSS0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_0	WKUP_R5FSS0_CORE0_intr_104	WKUP_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_0	C7X256V0_CLEC_gic_spi_104	C7X256V0_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_0	C7X256V1_CLEC_gic_spi_104	C7X256V1_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-245. Hardware Requests for mcu_gpiomux_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_1	GICSS0_spi_105	GICSS0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_1	WKUP_R5FSS0_CORE0_intr_105	WKUP_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_1	C7X256V0_CLEC_gic_spi_105	C7X256V0_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_1	C7X256V1_CLEC_gic_spi_105	C7X256V1_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_2	GICSS0_spi_106	GICSS0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_2	WKUP_R5FSS0_CORE0_intr_106	WKUP_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_2	C7X256V0_CLEC_gic_spi_106	C7X256V0_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_2	C7X256V1_CLEC_gic_spi_106	C7X256V1_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_3	GICSS0_spi_107	GICSS0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_3	WKUP_R5FSS0_CORE0_intr_107	WKUP_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_3	C7X256V0_CLEC_gic_spi_107	C7X256V0_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_3	C7X256V1_CLEC_gic_spi_107	C7X256V1_CLEC	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_4	MCU_R5FSS0_CORE0_cpu0_intr_104	MCU_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_5	MCU_R5FSS0_CORE0_cpu0_intr_105	MCU_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_6	MCU_R5FSS0_CORE0_cpu0_intr_106	MCU_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_7	MCU_R5FSS0_CORE0_cpu0_intr_107	MCU_R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_8	WKUP_ESM0_esm_pls_event0_88	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_8	WKUP_ESM0_esm_pls_event1_88	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-245. Hardware Requests for mcu_gpiomux_introuter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_8	WKUP_ESM0_esm_pls_event2_88	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_9	WKUP_ESM0_esm_pls_event0_89	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_9	WKUP_ESM0_esm_pls_event1_89	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_9	WKUP_ESM0_esm_pls_event2_89	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_10	WKUP_ESM0_esm_pls_event0_90	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_10	WKUP_ESM0_esm_pls_event1_90	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_10	WKUP_ESM0_esm_pls_event2_90	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_11	WKUP_ESM0_esm_pls_event0_91	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_11	WKUP_ESM0_esm_pls_event1_91	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_11	WKUP_ESM0_esm_pls_event2_91	WKUP_ESM0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_12	R5FSS0_CORE0_intr_32	R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_13	R5FSS0_CORE0_intr_33	R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_14	R5FSS0_CORE0_intr_60	R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_outp_15	R5FSS0_CORE0_intr_61	R5FSS0_CORE0	WKUP_MCU_GPIOMUX_INTROUTER0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.7 Data Movement Architecture

Copyright © 2026 Texas Instruments Incorporated

4.7.1 Data Movement Subsystem (DMSS)

This section contains the integration details for the DMSS module on this device. For Further information, see the Data Movement Subsystem (DMSS) section of the Data Movement Architecture chapter

4.7.1.1 DMSS Unsupported Features

The following features are not supported on this family of devices:

- Ring Accelerator QM and CREDENTIAL Modes

4.7.1.2 DMASS Integration

Table 4-246. DMASS Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0	✓		
DMASS1	✓		

4.7.1.3 DMASS_CBASS_0 Integration

Table 4-247. DMASS_CBASS_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_CBASS_0	✓		

4.7.1.4 DMASS_COMMON_0 Integration

Table 4-248. DMASS_COMMON_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_COMMON_0	✓		
DMASS1_COMMON_0	✓		

4.7.1.5 DMASS_ECC_AGGR_0 Integration

Table 4-249. DMASS_ECC_AGGR_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_ECC_AGGR_0	✓		
DMASS1_ECC_AGGR_0	✓		

Table 4-250. Clock Integration for DMASS_ECC_AGGR_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DMASS0_ECC_AGGR_0	FICLK	MAIN_SYSCLK0/2		None
DMASS1_ECC_AGGR_0	FICLK	MAIN_SYSCLK0		None

Table 4-251. Hardware Requests for DMASS_ECC_AGGR_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_ECC_AGGR_0	DMASS0_ECC_AGGR_0_ecc_corrected_err_level_0	ESM0_esm_lvl_event_9	ESM0	DMASS0_ECC_AGGR_0 interrupt request	level
DMASS0_ECC_AGGR_0	DMASS0_ECC_AGGR_0_ecc_uncorrected_err_level_0	ESM0_esm_lvl_event_10	ESM0	DMASS0_ECC_AGGR_0 interrupt request	level
DMASS1_ECC_AGGR_0	DMASS1_ECC_AGGR_0_ecc_corrected_err_level_0	ESM0_esm_lvl_event_63	ESM0	DMASS1_ECC_AGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-251. Hardware Requests for DMASS_ECC_AGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS1_ECC_AGGR_0	DMASS1_ECC_AGGR_0_ecc_uncorrected_err_level_0	ESM0_esm_lvl_event_62	ESM0	DMASS1_ECC_AGGR_0 interrupt request	level

4.7.1.6 DMASS_INTAGGR_0 Integration

Table 4-252. DMASS_INTAGGR_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_INTAGGR_0	✓		
DMASS1_INTAGGR_0	✓		

Table 4-253. Hardware Requests for DMASS_INTAGGR_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_0	GICSS0_spi_64	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_0	C7X256V0_CLEC_gic_spi_64	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_0	C7X256V1_CLEC_gic_spi_64	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_1	GICSS0_spi_65	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_1	C7X256V0_CLEC_gic_spi_65	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_1	C7X256V1_CLEC_gic_spi_65	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_2	GICSS0_spi_66	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_2	C7X256V0_CLEC_gic_spi_66	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_2	C7X256V1_CLEC_gic_spi_66	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_3	GICSS0_spi_67	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_3	C7X256V0_CLEC_gic_spi_67	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_3	C7X256V1_CLEC_gic_spi_67	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_4	GICSS0_spi_68	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_4	C7X256V0_CLEC_gic_spi_68	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_4	C7X256V1_CLEC_gic_spi_68	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_5	GICSS0_spi_69	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_5	C7X256V0_CLEC_gic_spi_69	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_5	C7X256V1_CLEC_gic_spi_69	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_6	GICSS0_spi_70	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_6	C7X256V0_CLEC_gic_spi_70	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_6	C7X256V1_CLEC_gic_spi_70	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_7	GICSS0_spi_71	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_7	C7X256V0_CLEC_gic_spi_71	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_7	C7X256V1_CLEC_gic_spi_71	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_8	GICSS0_spi_72	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_8	C7X256V0_CLEC_gic_spi_72	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_8	C7X256V1_CLEC_gic_spi_72	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_9	GICSS0_spi_73	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_9	C7X256V0_CLEC_gic_spi_73	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_9	C7X256V1_CLEC_gic_spi_73	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_10	GICSS0_spi_74	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_10	C7X256V0_CLEC_gic_spi_74	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_10	C7X256V1_CLEC_gic_spi_74	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_11	GICSS0_spi_75	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_11	C7X256V0_CLEC_gic_spi_75	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_11	C7X256V1_CLEC_gic_spi_75	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_12	GICSS0_spi_76	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_12	C7X256V0_CLEC_gic_spi_76	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_12	C7X256V1_CLEC_gic_spi_76	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_13	GICSS0_spi_77	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_13	C7X256V0_CLEC_gic_spi_77	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_13	C7X256V1_CLEC_gic_spi_77	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_14	GICSS0_spi_78	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_14	C7X256V0_CLEC_gic_spi_78	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_14	C7X256V1_CLEC_gic_spi_78	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_15	GICSS0_spi_79	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_15	C7X256V0_CLEC_gic_spi_79	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_15	C7X256V1_CLEC_gic_spi_79	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_16	GICSS0_spi_80	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_16	C7X256V0_CLEC_gic_spi_80	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_16	C7X256V1_CLEC_gic_spi_80	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_17	GICSS0_spi_81	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_17	C7X256V0_CLEC_gic_spi_81	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_17	C7X256V1_CLEC_gic_spi_81	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_18	GICSS0_spi_82	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_18	C7X256V0_CLEC_gic_spi_82	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_18	C7X256V1_CLEC_gic_spi_82	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_19	GICSS0_spi_83	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_19	C7X256V0_CLEC_gic_spi_83	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_19	C7X256V1_CLEC_gic_spi_83	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_20	GICSS0_spi_84	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_20	C7X256V0_CLEC_gic_spi_84	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_20	C7X256V1_CLEC_gic_spi_84	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_21	GICSS0_spi_85	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_21	C7X256V0_CLEC_gic_spi_85	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_21	C7X256V1_CLEC_gic_spi_85	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_22	GICSS0_spi_86	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_22	C7X256V0_CLEC_gic_spi_86	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_22	C7X256V1_CLEC_gic_spi_86	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_23	GICSS0_spi_87	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_23	C7X256V0_CLEC_gic_spi_87	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_23	C7X256V1_CLEC_gic_spi_87	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_24	GICSS0_spi_88	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_24	C7X256V0_CLEC_gic_spi_88	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_24	C7X256V1_CLEC_gic_spi_88	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_25	GICSS0_spi_89	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_25	C7X256V0_CLEC_gic_spi_89	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_25	C7X256V1_CLEC_gic_spi_89	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_26	GICSS0_spi_90	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_26	C7X256V0_CLEC_gic_spi_90	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_26	C7X256V1_CLEC_gic_spi_90	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_27	GICSS0_spi_91	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_27	C7X256V0_CLEC_gic_spi_91	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_27	C7X256V1_CLEC_gic_spi_91	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_28	GICSS0_spi_92	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_28	C7X256V0_CLEC_gic_spi_92	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_28	C7X256V1_CLEC_gic_spi_92	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_29	GICSS0_spi_93	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_29	C7X256V0_CLEC_gic_spi_93	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_29	C7X256V1_CLEC_gic_spi_93	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_30	GICSS0_spi_94	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_30	C7X256V0_CLEC_gic_spi_94	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_30	C7X256V1_CLEC_gic_spi_94	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_31	GICSS0_spi_95	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_31	C7X256V0_CLEC_gic_spi_95	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_31	C7X256V1_CLEC_gic_spi_95	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_32	GICSS0_spi_96	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_32	C7X256V0_CLEC_gic_spi_96	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_32	C7X256V1_CLEC_gic_spi_96	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_33	GICSS0_spi_97	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_33	C7X256V0_CLEC_gic_spi_97	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_33	C7X256V1_CLEC_gic_spi_97	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_34	GICSS0_spi_98	GICSS0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_34	C7X256V0_CLEC_gic_spi_98	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_34	C7X256V1_CLEC_gic_spi_98	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_35	GICSS0_spi_99	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_35	C7X256V0_CLEC_gic_spi_99	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_35	C7X256V1_CLEC_gic_spi_99	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_36	GICSS0_spi_100	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_36	C7X256V0_CLEC_gic_spi_100	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_36	C7X256V1_CLEC_gic_spi_100	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_37	GICSS0_spi_101	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_37	C7X256V0_CLEC_gic_spi_101	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_37	C7X256V1_CLEC_gic_spi_101	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_38	GICSS0_spi_102	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_38	C7X256V0_CLEC_gic_spi_102	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_38	C7X256V1_CLEC_gic_spi_102	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_39	GICSS0_spi_103	GICSS0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_39	C7X256V0_CLEC_gic_spi_103	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_39	C7X256V1_CLEC_gic_spi_103	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_40	WKUP_R5FSS0_CORE0_intr_64	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_41	WKUP_R5FSS0_CORE0_intr_65	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_42	WKUP_R5FSS0_CORE0_intr_66	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_43	WKUP_R5FSS0_CORE0_intr_67	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_44	WKUP_R5FSS0_CORE0_intr_68	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_45	WKUP_R5FSS0_CORE0_intr_69	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_46	WKUP_R5FSS0_CORE0_intr_70	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_47	WKUP_R5FSS0_CORE0_intr_71	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_48	WKUP_R5FSS0_CORE0_intr_72	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_49	WKUP_R5FSS0_CORE0_intr_73	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_50	WKUP_R5FSS0_CORE0_intr_74	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_51	WKUP_R5FSS0_CORE0_intr_75	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_52	WKUP_R5FSS0_CORE0_intr_76	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_53	WKUP_R5FSS0_CORE0_intr_77	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_54	WKUP_R5FSS0_CORE0_intr_78	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_55	WKUP_R5FSS0_CORE0_intr_79	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_56	WKUP_R5FSS0_CORE0_intr_80	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_57	WKUP_R5FSS0_CORE0_intr_81	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_58	WKUP_R5FSS0_CORE0_intr_82	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_59	WKUP_R5FSS0_CORE0_intr_83	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_60	WKUP_R5FSS0_CORE0_intr_84	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_61	WKUP_R5FSS0_CORE0_intr_85	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_62	WKUP_R5FSS0_CORE0_intr_86	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_63	WKUP_R5FSS0_CORE0_intr_87	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_64	WKUP_R5FSS0_CORE0_intr_88	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_65	WKUP_R5FSS0_CORE0_intr_89	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_66	WKUP_R5FSS0_CORE0_intr_90	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_67	WKUP_R5FSS0_CORE0_intr_91	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_68	WKUP_R5FSS0_CORE0_intr_92	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_69	WKUP_R5FSS0_CORE0_intr_93	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_70	WKUP_R5FSS0_CORE0_intr_94	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_71	WKUP_R5FSS0_CORE0_intr_95	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_72	WKUP_R5FSS0_CORE0_intr_8	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_73	WKUP_R5FSS0_CORE0_intr_9	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_74	WKUP_R5FSS0_CORE0_intr_10	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_75	WKUP_R5FSS0_CORE0_intr_11	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_76	WKUP_R5FSS0_CORE0_intr_12	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_77	WKUP_R5FSS0_CORE0_intr_13	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_78	WKUP_R5FSS0_CORE0_intr_14	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_79	WKUP_R5FSS0_CORE0_intr_15	WKUP_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_84	C7X256V0_CLEC_soc_events_in_16	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_85	C7X256V0_CLEC_soc_events_in_17	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_86	C7X256V0_CLEC_soc_events_in_18	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_87	C7X256V0_CLEC_soc_events_in_19	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_88	C7X256V0_CLEC_soc_events_in_20	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_89	C7X256V0_CLEC_soc_events_in_21	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_90	C7X256V0_CLEC_soc_events_in_22	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_91	C7X256V0_CLEC_soc_events_in_23	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_92	C7X256V0_CLEC_soc_events_in_24	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_93	C7X256V0_CLEC_soc_events_in_25	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_94	C7X256V0_CLEC_soc_events_in_26	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_95	C7X256V0_CLEC_soc_events_in_27	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_96	C7X256V0_CLEC_soc_events_in_28	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_97	C7X256V0_CLEC_soc_events_in_29	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_98	C7X256V0_CLEC_soc_events_in_30	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_99	C7X256V0_CLEC_soc_events_in_31	C7X256V0_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_100	C7X256V1_CLEC_soc_events_in_16	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_101	C7X256V1_CLEC_soc_events_in_17	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_102	C7X256V1_CLEC_soc_events_in_18	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_103	C7X256V1_CLEC_soc_events_in_19	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_104	C7X256V1_CLEC_soc_events_in_20	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_105	C7X256V1_CLEC_soc_events_in_21	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_106	C7X256V1_CLEC_soc_events_in_22	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_107	C7X256V1_CLEC_soc_events_in_23	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_108	C7X256V1_CLEC_soc_events_in_24	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_109	C7X256V1_CLEC_soc_events_in_25	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_110	C7X256V1_CLEC_soc_events_in_26	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_111	C7X256V1_CLEC_soc_events_in_27	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_112	C7X256V1_CLEC_soc_events_in_28	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_113	C7X256V1_CLEC_soc_events_in_29	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_114	C7X256V1_CLEC_soc_events_in_30	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_115	C7X256V1_CLEC_soc_events_in_31	C7X256V1_CLEC	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_152	R5FSS0_CORE0_intr_64	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_153	R5FSS0_CORE0_intr_65	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_154	R5FSS0_CORE0_intr_66	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_155	R5FSS0_CORE0_intr_67	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_156	R5FSS0_CORE0_intr_68	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_157	R5FSS0_CORE0_intr_69	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_158	R5FSS0_CORE0_intr_70	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_159	R5FSS0_CORE0_intr_71	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_160	R5FSS0_CORE0_intr_72	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_161	R5FSS0_CORE0_intr_73	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_162	R5FSS0_CORE0_intr_74	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_163	R5FSS0_CORE0_intr_75	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_164	R5FSS0_CORE0_intr_76	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_165	R5FSS0_CORE0_intr_77	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_166	R5FSS0_CORE0_intr_78	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_167	R5FSS0_CORE0_intr_79	R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_168	MCU_R5FSS0_CORE0_cpu0_intr_64	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_169	MCU_R5FSS0_CORE0_cpu0_intr_65	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_170	MCU_R5FSS0_CORE0_cpu0_intr_66	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_171	MCU_R5FSS0_CORE0_cpu0_intr_67	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_172	MCU_R5FSS0_CORE0_cpu0_intr_68	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_173	MCU_R5FSS0_CORE0_cpu0_intr_69	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_174	MCU_R5FSS0_CORE0_cpu0_intr_70	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_175	MCU_R5FSS0_CORE0_cpu0_intr_71	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_176	MCU_R5FSS0_CORE0_cpu0_intr_72	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_177	MCU_R5FSS0_CORE0_cpu0_intr_73	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_178	MCU_R5FSS0_CORE0_cpu0_intr_74	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_179	MCU_R5FSS0_CORE0_cpu0_intr_75	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_180	MCU_R5FSS0_CORE0_cpu0_intr_76	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_181	MCU_R5FSS0_CORE0_cpu0_intr_77	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_182	MCU_R5FSS0_CORE0_cpu0_intr_78	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_intaggr_vintr_pend_183	MCU_R5FSS0_CORE0_cpu0_intr_79	MCU_R5FSS0_CORE0	DMASS0_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_0	GICSS0_spi_237	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_0	C7X256V0_CLEC_gic_spi_237	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_0	C7X256V1_CLEC_gic_spi_237	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_1	GICSS0_spi_238	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_1	C7X256V0_CLEC_gic_spi_238	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_1	C7X256V1_CLEC_gic_spi_238	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_2	GICSS0_spi_239	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_2	C7X256V0_CLEC_gic_spi_239	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_2	C7X256V1_CLEC_gic_spi_239	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_3	GICSS0_spi_240	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_3	C7X256V0_CLEC_gic_spi_240	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_3	C7X256V1_CLEC_gic_spi_240	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_4	GICSS0_spi_241	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_4	C7X256V0_CLEC_gic_spi_241	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_4	C7X256V1_CLEC_gic_spi_241	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_5	GICSS0_spi_242	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_5	C7X256V0_CLEC_gic_spi_242	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_5	C7X256V1_CLEC_gic_spi_242	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_6	GICSS0_spi_243	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_6	C7X256V0_CLEC_gic_spi_243	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_6	C7X256V1_CLEC_gic_spi_243	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_7	GICSS0_spi_244	GICSS0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_7	C7X256V0_CLEC_gic_spi_244	C7X256V0_CLEC	DMASS1_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_7	C7X256V1_CLEC_gic_spi_244	C7X256V1_CLEC	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_8	WKUP_R5FSS0_CORE0_intr_129	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_9	WKUP_R5FSS0_CORE0_intr_130	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_10	WKUP_R5FSS0_CORE0_intr_131	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_11	WKUP_R5FSS0_CORE0_intr_132	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_12	WKUP_R5FSS0_CORE0_intr_150	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_13	WKUP_R5FSS0_CORE0_intr_158	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_14	R5FSS0_CORE0_intr_158	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_14	WKUP_R5FSS0_CORE0_intr_159	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_15	R5FSS0_CORE0_intr_159	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_15	WKUP_R5FSS0_CORE0_intr_160	WKUP_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_16	MCU_R5FSS0_CORE0_cpu0_intr_129	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_17	MCU_R5FSS0_CORE0_cpu0_intr_130	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_18	MCU_R5FSS0_CORE0_cpu0_intr_131	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_19	MCU_R5FSS0_CORE0_cpu0_intr_132	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_20	MCU_R5FSS0_CORE0_cpu0_intr_150	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_21	MCU_R5FSS0_CORE0_cpu0_intr_158	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_22	R5FSS0_CORE0_intr_160	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_22	MCU_R5FSS0_CORE0_cpu0_intr_159	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_23	R5FSS0_CORE0_intr_150	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_23	MCU_R5FSS0_CORE0_cpu0_intr_160	MCU_R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_24	R5FSS0_CORE0_intr_129	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_25	R5FSS0_CORE0_intr_130	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_26	R5FSS0_CORE0_intr_131	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_27	R5FSS0_CORE0_intr_132	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_28	R5FSS0_CORE0_intr_138	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_29	R5FSS0_CORE0_intr_139	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_30	R5FSS0_CORE0_intr_143	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_31	R5FSS0_CORE0_intr_144	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_32	R5FSS0_CORE0_intr_8	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_33	R5FSS0_CORE0_intr_9	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_34	R5FSS0_CORE0_intr_10	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_35	R5FSS0_CORE0_intr_11	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_36	R5FSS0_CORE0_intr_12	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_37	R5FSS0_CORE0_intr_13	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_38	R5FSS0_CORE0_intr_14	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-253. Hardware Requests for DMASS_INTAGGR_0 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_intaggr_vintr_pend_39	R5FSS0_CORE0_intr_15	R5FSS0_CORE0	DMASS1_INTAGGR_0 interrupt request	level

4.7.1.7 DMASS_IPCSS_0 Integration

Table 4-254. DMASS_IPCSS_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_IPCSS_0	✓		

4.7.1.8 DMASS_PSILSS_0 Integration

Table 4-255. DMASS_PSILSS_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_PSILSS_0	✓		
DMASS1_PSILSS_0	✓		

4.7.1.9 DMASS_PSILCFG_0 Integration

Table 4-256. DMASS_PSILCFG_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_PSILCFG_0	✓		
DMASS1_PSILCFG_0	✓		

4.7.1.10 DMASS_RINGACC_0 Integration

Table 4-257. DMASS_RINGACC_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_RINGACC_0	✓		

4.7.1.11 DMASS_PKTDMA_0 Integration

Table 4-258. DMASS_PKTDMA_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_PKTDMA_0	✓		

4.7.1.12 DMASS_BCDMA_0 Integration

Table 4-259. DMASS_BCDMA_0 Module Allocations

Instance	MAIN	MCU	WKUP
DMASS0_BCDMA_0	✓		
DMASS1_BCDMA_0	✓		

4.7.1.13 Global Event Map

The global event map for all DMSS events is shown in Table 4-260.

Table 4-260. Global Event Map

Destination		Offset	PSIL Routed Slots	Actual Slots
DMSS Instance or External	Port			
DMSS	DMSS INTAGGR SEVI	0 (0k)	8192 (8k)	1536
DMSS	DMSS INTAGGR MEVI	8192 (8k)	2048 (2k)	128

Copyright © 2026 Texas Instruments Incorporated

Table 4-260. Global Event Map (continued)

Destination		Offset	PSIL Routed Slots	Actual Slots
DMSS Instance or External	Port			
DMSS	DMSS INTAGGR GEVI	10240 (10k)	2048 (2k)	256
External	DMSC IA SEVI	18432 (18k)	2048 (2k)	-
DMSS	DMSS INTAGGR LEVI	32768 (32k)	32 (0.03k)	32
External	PDMA_MAIN0 LEVI	41984 (41.000k)	128	-
External	PDMA_MAIN1 LEVI	42112 (41.125k)	128	-
DMSS	DMSS BCDMA Triggers	50176 (49k)	1024 (1k)	156
External	UTC0	62464	128	-
External	UTC1	62720	128	-
External	DRU0	62976	64	-
External	DRU1	63232	64	-

4.7.1.14 PSI-L System Thread Map

Table 4-261. DMSS PSILSS Thread Map

Thread Number	NAV Instance	Endpoint
0x0000	DMSS	PSILCFG CFGSTRM
0x0020	DMSS	PKTDMA CFGSTRM
0x0021	DMSS	BCDMA CFGSTRM
0x1000-0x1FFF	DMSS	PKTDMA Threads
0x2000-0x2FFF	DMSS	BCDMA Threads
0x3000-0x42FF	DMSS	Reserved
0x4300-0x43FF	DMSS	PDMA_MAIN0
0x4400-0x44FF	DMSS	PDMA_MAIN1
0x4500-0x457F	DMSS	PDMA_MCASP
0x4580-0x45FF	DMSS	PDMA_MCASP1
0x4600-0x46FF	DMSS	CPSW2
0x4700-0x74FF	DMSS	Reserved
0x7500-0x7506	DMSS	SAUL0

Copyright © 2026 Texas Instruments Incorporated

4.7.2 Peripheral DMA (PDMA)

This section contains the integration details for the PDMA module on this device. For further information, see the *Peripheral DMA (PDMA)* section of the *Data Movement Architecture* chapter.

4.7.2.1 PDMA Unsupported Features

The following features are not supported on this family of devices:

- Dynamic transfer requests
- Cross-channel triggering

4.7.2.2 PDMA Integration

Table 4-262. PDMA Module Allocations

Instance	MAIN	MCU	WKUP
PDMA2	✓		
PDMA1	✓		
PDMA3	✓		
PDMA0	✓		

Table 4-263. Integration Attributes for PDMA

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
PDMA2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
PDMA1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
PDMA0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso

Table 4-264. Clock Integration for PDMA

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PDMA2	FICLK	MAIN_SYSCLK0/2		None
PDMA1	VCLK	MAIN_SYSCLK0/4		None
PDMA3	VCLK	MAIN_SYSCLK0/2		None
PDMA0	VCLK	MAIN_SYSCLK0/4		None

Table 4-265. Reset Integration for PDMA

Module Instance	Source	Description
PDMA2	PSC0	PDMA2 reset
PDMA1	PSC0	PDMA1 reset
PDMA0	PSC0	PDMA0 reset

Table 4-266. Hardware Requests for PDMA

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PDMA1	PDMA1_ecc_ded_pend_0	ESM0_esm_lvl_event_89	ESM0	PDMA1 interrupt request	level
PDMA1	PDMA1_ecc_sec_pend_0	ESM0_esm_lvl_event_28	ESM0	PDMA1 interrupt request	level
PDMA0	PDMA0_ecc_ded_pend_0	ESM0_esm_lvl_event_88	ESM0	PDMA0 interrupt request	level
PDMA0	PDMA0_ecc_sec_pend_0	ESM0_esm_lvl_event_15	ESM0	PDMA0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.8 Audio

4.8.1 Audio Tracking Logic (ATL)

This section contains the integration details for the ATL module on this device. For further information, see the *Audio Tracking Logic (ATL)* section of the *Peripherals* chapter.

4.8.1.1 atl Integration

Table 4-267. ATL Module Allocations

Instance	MAIN	MCU	WKUP
atl0	✓		

Table 4-268. Clock Integration for atl

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
atl0	ATL_CLK	MAIN_PLL2_HSDIV8_CLKOUT/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=0	None
atl0	ATL_CLK	MAIN_PLL1_HSDIV6_CLKOUT/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=1	None
atl0	ATL_CLK	MAIN_PLL5_HSDIV1_CLKOUT/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=2	None
atl0	ATL_CLK	MAIN_PLL0_HSDIV7_CLKOUT/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=4	None
atl0	ATL_CLK	MCU_EXT_REFCLK0/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=5	None
atl0	ATL_CLK	EXT_REFCLK1/2	MAIN_CTRL_MMR_CFG0_ATL_CLKSEL[2:0]=6	None
atl0	ATL_IO_PORT_AWS	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS0_SEL[3:0]=10	None
atl0	ATL_IO_PORT_AWS	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS0_SEL[3:0]=11	None
atl0	ATL_IO_PORT_AWS	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS0_SEL[3:0]=12	None
atl0	ATL_IO_PORT_AWS	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS0_SEL[3:0]=2	None
atl0	ATL_IO_PORT_AWS	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS0_SEL[3:0]=7	None
atl0	ATL_IO_PORT_AWS_1	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS1_SEL[3:0]=10	None
atl0	ATL_IO_PORT_AWS_1	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS1_SEL[3:0]=11	None
atl0	ATL_IO_PORT_AWS_1	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS1_SEL[3:0]=12	None

Copyright © 2026 Texas Instruments Incorporated

Table 4-268. Clock Integration for atl (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
atl0	ATL_IO_PORT_AWS_1	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS1_SEL[3:0]=2	None
atl0	ATL_IO_PORT_AWS_1	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS1_SEL[3:0]=7	None
atl0	ATL_IO_PORT_AWS_2	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS2_SEL[3:0]=10	None
atl0	ATL_IO_PORT_AWS_2	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS2_SEL[3:0]=11	None
atl0	ATL_IO_PORT_AWS_2	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS2_SEL[3:0]=12	None
atl0	ATL_IO_PORT_AWS_2	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS2_SEL[3:0]=2	None
atl0	ATL_IO_PORT_AWS_2	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS2_SEL[3:0]=7	None
atl0	ATL_IO_PORT_AWS_3	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS3_SEL[3:0]=10	None
atl0	ATL_IO_PORT_AWS_3	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS3_SEL[3:0]=11	None
atl0	ATL_IO_PORT_AWS_3	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS3_SEL[3:0]=12	None
atl0	ATL_IO_PORT_AWS_3	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS3_SEL[3:0]=2	None
atl0	ATL_IO_PORT_AWS_3	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_AWS3_SEL[3:0]=7	None
atl0	ATL_IO_PORT_BWS	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS0_SEL[3:0]=10	None
atl0	ATL_IO_PORT_BWS	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS0_SEL[3:0]=11	None
atl0	ATL_IO_PORT_BWS	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS0_SEL[3:0]=12	None
atl0	ATL_IO_PORT_BWS	MCASP2_AFSR (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS0_SEL[3:0]=2	None
atl0	ATL_IO_PORT_BWS	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS0_SEL[3:0]=7	None
atl0	ATL_IO_PORT_BWS_1	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS1_SEL[3:0]=10	None

Copyright © 2026 Texas Instruments Incorporated

Table 4-268. Clock Integration for atl (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
atl0	ATL_IO_PORT_BWS_1	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS1_SEL[3:0]=11	None
atl0	ATL_IO_PORT_BWS_1	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS1_SEL[3:0]=12	None
atl0	ATL_IO_PORT_BWS_1	MCASP2_AFSR (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS1_SEL[3:0]=2	None
atl0	ATL_IO_PORT_BWS_1	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS1_SEL[3:0]=7	None
atl0	ATL_IO_PORT_BWS_2	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS2_SEL[3:0]=10	None
atl0	ATL_IO_PORT_BWS_2	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS2_SEL[3:0]=11	None
atl0	ATL_IO_PORT_BWS_2	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS2_SEL[3:0]=12	None
atl0	ATL_IO_PORT_BWS_2	MCASP2_AFSR (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS2_SEL[3:0]=2	None
atl0	ATL_IO_PORT_BWS_2	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS2_SEL[3:0]=7	None
atl0	ATL_IO_PORT_BWS_3	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS3_SEL[3:0]=10	None
atl0	ATL_IO_PORT_BWS_3	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS3_SEL[3:0]=11	None
atl0	ATL_IO_PORT_BWS_3	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS3_SEL[3:0]=12	None
atl0	ATL_IO_PORT_BWS_3	MCASP2_AFSR (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS3_SEL[3:0]=2	None
atl0	ATL_IO_PORT_BWS_3	MCASP2_AFSX (PIN)	MAIN_CTRL_MMR_CFG0_ATL_BWS3_SEL[3:0]=7	None
atl0	VBUS_CLK	MAIN_SYSCLK0/2		None

4.8.2 Multichannel Audio Serial Port (MCASP)

This section contains the integration details for the MCASP module on this device. For further information, see the *Multichannel Audio Serial Port (MCASP)* section of the *Peripherals* chapter.

4.8.2.1 MCASP Unsupported Features

The following features are not supported on this family of devices:

- Muting output (AMUTE)
- Muting input (AMUTEIN)
- Instances may not support all pin options. See device specific Datasheet for details on pin support of each MCASP instance.

Copyright © 2026 Texas Instruments Incorporated

4.8.2.2 MCASP Integration

Table 4-269. MCASP Module Allocations

Instance	MAIN	MCU	WKUP
MCASP0	✓		
MCASP1	✓		
MCASP2	✓		
MCASP3	✓		
MCASP4	✓		

Table 4-270. Integration Attributes for MCASP

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCASP0	PSC0	GP_CORE	LPSC_main_mcasp0	17	OFF	YES	LPSC_main_ip
MCASP1	PSC0	GP_CORE	LPSC_main_mcasp1	18	OFF	YES	LPSC_main_ip
MCASP2	PSC0	GP_CORE	LPSC_main_mcasp2	19	OFF	YES	LPSC_main_ip

Table 4-271. Clock Integration for MCASP

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP0	AUX_CLK	MAIN_PLL2_HSDIV8_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=0	auxillary clock input. audio clocks are derived from this clock.
MCASP0	AUX_CLK	MAIN_PLL1_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=1	auxillary clock input. audio clocks are derived from this clock.
MCASP0	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=4	auxillary clock input. audio clocks are derived from this clock.
MCASP0	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=5	auxillary clock input. audio clocks are derived from this clock.
MCASP0	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=6	auxillary clock input. audio clocks are derived from this clock.
MCASP0	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_CLKSEL[2:0]=7	auxillary clock input. audio clocks are derived from this clock.
MCASP0	ACLKR_I	MCASP0_ACLKR (PIN)		receive audio data clock
MCASP0	ACLKX_I	MCASP0_ACLKX (PIN)		transmit audio data clock
MCASP0	AHCLKR_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=0	receive audio high frequency clock
MCASP0	AHCLKR_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP0	AHCLKR_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP0	AHCLKR_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=2	receive audio high frequency clock
MCASP0	AHCLKR_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=3	receive audio high frequency clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP0	AHCLKR_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=4	receive audio high frequency clock
MCASP0	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=5	receive audio high frequency clock
MCASP0	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=6	receive audio high frequency clock
MCASP0	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=7	receive audio high frequency clock
MCASP0	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[3:0]=8	receive audio high frequency clock
MCASP0	AHCLKX_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=0	transmit audio high frequency clock
MCASP0	AHCLKX_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP0	AHCLKX_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP0	AHCLKX_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=2	transmit audio high frequency clock
MCASP0	AHCLKX_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=3	transmit audio high frequency clock
MCASP0	AHCLKX_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=4	transmit audio high frequency clock
MCASP0	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=5	transmit audio high frequency clock
MCASP0	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=6	transmit audio high frequency clock
MCASP0	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=7	transmit audio high frequency clock
MCASP0	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP0_AHCLKSEL[11:8]=8	transmit audio high frequency clock
MCASP0	FICLK	MAIN_SYSCLK0/2		functional and interface clock
MCASP1	AUX_CLK	MAIN_PLL2_HSDIV8_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=0	auxillary clock input. audio clocks are derived from this clock.
MCASP1	AUX_CLK	MAIN_PLL1_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=1	auxillary clock input. audio clocks are derived from this clock.
MCASP1	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=4	auxillary clock input. audio clocks are derived from this clock.

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP1	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=5	auxillary clock input. audio clocks are derived from this clock.
MCASP1	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=6	auxillary clock input. audio clocks are derived from this clock.
MCASP1	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_CLKSEL[2:0]=7	auxillary clock input. audio clocks are derived from this clock.
MCASP1	ACLKR_I	MCASP1_ACLKR (PIN)		receive audio data clock
MCASP1	ACLKX_I	MCASP1_ACLKX (PIN)		transmit audio data clock
MCASP1	AHCLKR_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=0	receive audio high frequency clock
MCASP1	AHCLKR_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP1	AHCLKR_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP1	AHCLKR_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=2	receive audio high frequency clock
MCASP1	AHCLKR_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=3	receive audio high frequency clock
MCASP1	AHCLKR_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=4	receive audio high frequency clock
MCASP1	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=5	receive audio high frequency clock
MCASP1	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=6	receive audio high frequency clock
MCASP1	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=7	receive audio high frequency clock
MCASP1	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[3:0]=8	receive audio high frequency clock
MCASP1	AHCLKX_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=0	transmit audio high frequency clock
MCASP1	AHCLKX_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP1	AHCLKX_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP1	AHCLKX_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=2	transmit audio high frequency clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP1	AHCLKX_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=3	transmit audio high frequency clock
MCASP1	AHCLKX_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=4	transmit audio high frequency clock
MCASP1	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=5	transmit audio high frequency clock
MCASP1	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=6	transmit audio high frequency clock
MCASP1	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=7	transmit audio high frequency clock
MCASP1	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP1_AHCLKSEL[11:8]=8	transmit audio high frequency clock
MCASP1	FICLK	MAIN_SYSCLK0/2		functional and interface clock
MCASP2	AUX_CLK	MAIN_PLL2_HSDIV8_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=0	auxillary clock input. audio clocks are derived from this clock.
MCASP2	AUX_CLK	MAIN_PLL1_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=1	auxillary clock input. audio clocks are derived from this clock.
MCASP2	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=4	auxillary clock input. audio clocks are derived from this clock.
MCASP2	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=5	auxillary clock input. audio clocks are derived from this clock.
MCASP2	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=6	auxillary clock input. audio clocks are derived from this clock.
MCASP2	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP2_CLKSEL[2:0]=7	auxillary clock input. audio clocks are derived from this clock.
MCASP2	ACLKR_I	MCASP2_ACLKR (PIN)		receive audio data clock
MCASP2	ACLKX_I	MCASP2_ACLKX (PIN)		transmit audio data clock
MCASP2	AHCLKR_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP2_AHCLKSEL[3:0]=0	receive audio high frequency clock
MCASP2	AHCLKR_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP2_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP2	AHCLKR_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP2_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP2	AHCLKR_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP2_AHCLKSEL[3:0]=2	receive audio high frequency clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP2	AHCLKR_I	AUDIO_EXT_REFCLK 1 (PIN)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=3	receive audio high frequency clock
MCASP2	AHCLKR_I	AUDIO_EXT_REFCLK 2 (PIN)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=4	receive audio high frequency clock
MCASP2	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=5	receive audio high frequency clock
MCASP2	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=6	receive audio high frequency clock
MCASP2	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=7	receive audio high frequency clock
MCASP2	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[3:0]=8	receive audio high frequency clock
MCASP2	AHCLKX_I	EXT_REFCLK1	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=0	transmit audio high frequency clock
MCASP2	AHCLKX_I	CLK_12M_RC	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=1	transmit audio high frequency clock
MCASP2	AHCLKX_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=1	transmit audio high frequency clock
MCASP2	AHCLKX_I	AUDIO_EXT_REFCLK 0 (PIN)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=2	transmit audio high frequency clock
MCASP2	AHCLKX_I	AUDIO_EXT_REFCLK 1 (PIN)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=3	transmit audio high frequency clock
MCASP2	AHCLKX_I	AUDIO_EXT_REFCLK 2 (PIN)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=4	transmit audio high frequency clock
MCASP2	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=5	transmit audio high frequency clock
MCASP2	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=6	transmit audio high frequency clock
MCASP2	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=7	transmit audio high frequency clock
MCASP2	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_C FG0_MCASP2_AHCL KSEL[11:8]=8	transmit audio high frequency clock
MCASP2	FICLK	MAIN_SYSCLK0/2		functional and interface clock
MCASP3	AUX_CLK	MAIN_PLL2_HSDIV8_ CLKOUT	MAIN_CTRL_MMR_C FG0_MCASP3_CLKS EL[2:0]=0	auxillary clock input. audio clocks are derived from this clock.
MCASP3	AUX_CLK	MAIN_PLL1_HSDIV6_ CLKOUT	MAIN_CTRL_MMR_C FG0_MCASP3_CLKS EL[2:0]=1	auxillary clock input. audio clocks are derived from this clock.

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP3	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_CLKSEL[2:0]=4	auxillary clock input. audio clocks are derived from this clock.
MCASP3	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_CLKSEL[2:0]=5	auxillary clock input. audio clocks are derived from this clock.
MCASP3	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_CLKSEL[2:0]=6	auxillary clock input. audio clocks are derived from this clock.
MCASP3	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_CLKSEL[2:0]=7	auxillary clock input. audio clocks are derived from this clock.
MCASP3	ACLKR_I	MCASP3_ACLKR (PIN)		receive audio data clock
MCASP3	ACLKX_I	MCASP3_ACLKX (PIN)		transmit audio data clock
MCASP3	AHCLKR_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=0	receive audio high frequency clock
MCASP3	AHCLKR_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP3	AHCLKR_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP3	AHCLKR_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=2	receive audio high frequency clock
MCASP3	AHCLKR_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=3	receive audio high frequency clock
MCASP3	AHCLKR_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=4	receive audio high frequency clock
MCASP3	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=5	receive audio high frequency clock
MCASP3	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=6	receive audio high frequency clock
MCASP3	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=7	receive audio high frequency clock
MCASP3	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[3:0]=8	receive audio high frequency clock
MCASP3	AHCLKX_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=0	transmit audio high frequency clock
MCASP3	AHCLKX_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP3	AHCLKX_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=1	transmit audio high frequency clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP3	AHCLKX_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=2	transmit audio high frequency clock
MCASP3	AHCLKX_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=3	transmit audio high frequency clock
MCASP3	AHCLKX_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=4	transmit audio high frequency clock
MCASP3	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=5	transmit audio high frequency clock
MCASP3	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=6	transmit audio high frequency clock
MCASP3	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=7	transmit audio high frequency clock
MCASP3	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP3_AHCLKSEL[11:8]=8	transmit audio high frequency clock
MCASP3	FICLK	MAIN_SYSCLK0/2		functional and interface clock
MCASP4	AUX_CLK	MAIN_PLL2_HSDIV8_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=0	auxillary clock input. audio clocks are derived from this clock.
MCASP4	AUX_CLK	MAIN_PLL1_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=1	auxillary clock input. audio clocks are derived from this clock.
MCASP4	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=4	auxillary clock input. audio clocks are derived from this clock.
MCASP4	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=5	auxillary clock input. audio clocks are derived from this clock.
MCASP4	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=6	auxillary clock input. audio clocks are derived from this clock.
MCASP4	AUX_CLK	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_CLKSEL[2:0]=7	auxillary clock input. audio clocks are derived from this clock.
MCASP4	ACLKR_I	MCASP4_ACLKR (PIN)		receive audio data clock
MCASP4	ACLKX_I	MCASP4_ACLKX (PIN)		transmit audio data clock
MCASP4	AHCLKR_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=0	receive audio high frequency clock
MCASP4	AHCLKR_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=1	receive audio high frequency clock
MCASP4	AHCLKR_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=1	receive audio high frequency clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-271. Clock Integration for MCASP (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCASP4	AHCLKR_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=2	receive audio high frequency clock
MCASP4	AHCLKR_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=3	receive audio high frequency clock
MCASP4	AHCLKR_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=4	receive audio high frequency clock
MCASP4	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=5	receive audio high frequency clock
MCASP4	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=6	receive audio high frequency clock
MCASP4	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=7	receive audio high frequency clock
MCASP4	AHCLKR_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[3:0]=8	receive audio high frequency clock
MCASP4	AHCLKX_I	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=0	transmit audio high frequency clock
MCASP4	AHCLKX_I	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP4	AHCLKX_I	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=1	transmit audio high frequency clock
MCASP4	AHCLKX_I	AUDIO_EXT_REFCLK0 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=2	transmit audio high frequency clock
MCASP4	AHCLKX_I	AUDIO_EXT_REFCLK1 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=3	transmit audio high frequency clock
MCASP4	AHCLKX_I	AUDIO_EXT_REFCLK2 (PIN)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=4	transmit audio high frequency clock
MCASP4	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=5	transmit audio high frequency clock
MCASP4	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=6	transmit audio high frequency clock
MCASP4	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=7	transmit audio high frequency clock
MCASP4	AHCLKX_I	ATL0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCASP4_AHCLKSEL[11:8]=8	transmit audio high frequency clock
MCASP4	FICLK	MAIN_SYSCLK0/2		functional and interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-272. Reset Integration for MCASP

Module Instance	Source	Description
MCASP0	PSC0	MCASP0 reset
MCASP1	PSC0	MCASP1 reset
MCASP2	PSC0	MCASP2 reset

Table 4-273. Hardware Requests for MCASP

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCASP0	MCASP0_rec_dma_event_req_0	PDMA2_mcasp_main_0_rx_0	PDMA2	MCASP0 interrupt request	pulse
MCASP0	MCASP0_rec_intr_pend_0	GICSS0_spi_267	GICSS0	MCASP0 interrupt request	level
MCASP0	MCASP0_rec_intr_pend_0	R5FSS0_CORE0_intr_120	R5FSS0_CORE0	MCASP0 interrupt request	level
MCASP0	MCASP0_rec_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_120	MCU_R5FSS0_CORE0	MCASP0 interrupt request	level
MCASP0	MCASP0_rec_intr_pend_0	C7X256V0_CLEC_gic_spi_267	C7X256V0_CLEC	MCASP0 interrupt request	level
MCASP0	MCASP0_rec_intr_pend_0	C7X256V1_CLEC_gic_spi_267	C7X256V1_CLEC	MCASP0 interrupt request	level
MCASP0	MCASP0_xmit_dma_event_req_0	PDMA2_mcasp_main_0_tx_0	PDMA2	MCASP0 interrupt request	pulse
MCASP0	MCASP0_xmit_intr_pend_0	GICSS0_spi_268	GICSS0	MCASP0 interrupt request	level
MCASP0	MCASP0_xmit_intr_pend_0	R5FSS0_CORE0_intr_121	R5FSS0_CORE0	MCASP0 interrupt request	level
MCASP0	MCASP0_xmit_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_121	MCU_R5FSS0_CORE0	MCASP0 interrupt request	level
MCASP0	MCASP0_xmit_intr_pend_0	C7X256V0_CLEC_gic_spi_268	C7X256V0_CLEC	MCASP0 interrupt request	level
MCASP0	MCASP0_xmit_intr_pend_0	C7X256V1_CLEC_gic_spi_268	C7X256V1_CLEC	MCASP0 interrupt request	level
MCASP1	MCASP1_rec_dma_event_req_0	PDMA2_mcasp_main_1_rx_0	PDMA2	MCASP1 interrupt request	pulse
MCASP1	MCASP1_rec_intr_pend_0	GICSS0_spi_269	GICSS0	MCASP1 interrupt request	level
MCASP1	MCASP1_rec_intr_pend_0	R5FSS0_CORE0_intr_122	R5FSS0_CORE0	MCASP1 interrupt request	level
MCASP1	MCASP1_rec_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_122	MCU_R5FSS0_CORE0	MCASP1 interrupt request	level
MCASP1	MCASP1_rec_intr_pend_0	C7X256V0_CLEC_gic_spi_269	C7X256V0_CLEC	MCASP1 interrupt request	level
MCASP1	MCASP1_rec_intr_pend_0	C7X256V1_CLEC_gic_spi_269	C7X256V1_CLEC	MCASP1 interrupt request	level
MCASP1	MCASP1_xmit_dma_event_req_0	PDMA2_mcasp_main_1_tx_0	PDMA2	MCASP1 interrupt request	pulse
MCASP1	MCASP1_xmit_intr_pend_0	GICSS0_spi_270	GICSS0	MCASP1 interrupt request	level
MCASP1	MCASP1_xmit_intr_pend_0	R5FSS0_CORE0_intr_123	R5FSS0_CORE0	MCASP1 interrupt request	level
MCASP1	MCASP1_xmit_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_123	MCU_R5FSS0_CORE0	MCASP1 interrupt request	level
MCASP1	MCASP1_xmit_intr_pend_0	C7X256V0_CLEC_gic_spi_270	C7X256V0_CLEC	MCASP1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-273. Hardware Requests for MCASP (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCASP1	MCASP1_xmit_intr_pend_0	C7X256V1_CLEC_gic_spi_270	C7X256V1_CLEC	MCASP1 interrupt request	level
MCASP2	MCASP2_rec_dma_event_req_0	PDMA2_mcasp_main_2_rx_0	PDMA2	MCASP2 interrupt request	pulse
MCASP2	MCASP2_rec_intr_pend_0	GICSS0_spi_271	GICSS0	MCASP2 interrupt request	level
MCASP2	MCASP2_rec_intr_pend_0	R5FSS0_CORE0_intr_124	R5FSS0_CORE0	MCASP2 interrupt request	level
MCASP2	MCASP2_rec_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_124	MCU_R5FSS0_CORE0	MCASP2 interrupt request	level
MCASP2	MCASP2_rec_intr_pend_0	C7X256V0_CLEC_gic_spi_271	C7X256V0_CLEC	MCASP2 interrupt request	level
MCASP2	MCASP2_rec_intr_pend_0	C7X256V1_CLEC_gic_spi_271	C7X256V1_CLEC	MCASP2 interrupt request	level
MCASP2	MCASP2_xmit_dma_event_req_0	PDMA2_mcasp_main_2_tx_0	PDMA2	MCASP2 interrupt request	pulse
MCASP2	MCASP2_xmit_intr_pend_0	GICSS0_spi_272	GICSS0	MCASP2 interrupt request	level
MCASP2	MCASP2_xmit_intr_pend_0	R5FSS0_CORE0_intr_125	R5FSS0_CORE0	MCASP2 interrupt request	level
MCASP2	MCASP2_xmit_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_125	MCU_R5FSS0_CORE0	MCASP2 interrupt request	level
MCASP2	MCASP2_xmit_intr_pend_0	C7X256V0_CLEC_gic_spi_272	C7X256V0_CLEC	MCASP2 interrupt request	level
MCASP2	MCASP2_xmit_intr_pend_0	C7X256V1_CLEC_gic_spi_272	C7X256V1_CLEC	MCASP2 interrupt request	level
MCASP3	MCASP3_rec_dma_event_req_0	PDMA3_mcasp_main_3_rx_0	PDMA3	MCASP3 interrupt request	pulse
MCASP3	MCASP3_rec_intr_pend_0	MAIN_GPIOMUX_INTROUTER0_in_186	MAIN_GPIOMUX_INTROUTER0	MCASP3 interrupt request	level
MCASP3	MCASP3_rec_intr_pend_0	R5FSS0_CORE0_intr_118	R5FSS0_CORE0	MCASP3 interrupt request	level
MCASP3	MCASP3_rec_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_118	MCU_R5FSS0_CORE0	MCASP3 interrupt request	level
MCASP3	MCASP3_xmit_dma_event_req_0	PDMA3_mcasp_main_3_tx_0	PDMA3	MCASP3 interrupt request	pulse
MCASP3	MCASP3_xmit_intr_pend_0	MAIN_GPIOMUX_INTROUTER0_in_187	MAIN_GPIOMUX_INTROUTER0	MCASP3 interrupt request	level
MCASP3	MCASP3_xmit_intr_pend_0	R5FSS0_CORE0_intr_117	R5FSS0_CORE0	MCASP3 interrupt request	level
MCASP3	MCASP3_xmit_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_117	MCU_R5FSS0_CORE0	MCASP3 interrupt request	level
MCASP4	MCASP4_rec_dma_event_req_0	PDMA3_mcasp_main_4_rx_0	PDMA3	MCASP4 interrupt request	pulse
MCASP4	MCASP4_rec_intr_pend_0	MAIN_GPIOMUX_INTROUTER0_in_188	MAIN_GPIOMUX_INTROUTER0	MCASP4 interrupt request	level
MCASP4	MCASP4_rec_intr_pend_0	R5FSS0_CORE0_intr_252	R5FSS0_CORE0	MCASP4 interrupt request	level
MCASP4	MCASP4_rec_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_252	MCU_R5FSS0_CORE0	MCASP4 interrupt request	level
MCASP4	MCASP4_xmit_dma_event_req_0	PDMA3_mcasp_main_4_tx_0	PDMA3	MCASP4 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-273. Hardware Requests for MCASP (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCASP4	MCASP4_xmit_intr_pend_0	MAIN_GPIOMUX_INTROUTER0_in_189	MAIN_GPIOMUX_INTROUTER0	MCASP4 interrupt request	level
MCASP4	MCASP4_xmit_intr_pend_0	R5FSS0_CORE0_intr_251	R5FSS0_CORE0	MCASP4 interrupt request	level
MCASP4	MCASP4_xmit_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_251	MCU_R5FSS0_CORE0	MCASP4 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.9 General Connectivity

4.9.1 General Purpose Input/Output (GPIO)

This section contains the integration details for the GPIO modules on this device. For further information, see the *General-Purpose Interface* section of the *Peripherals* chapter.

4.9.1.1 GPIO Unsupported Features

The following features are not supported on this family of devices:

- The following apply to MCU_GPIO0:
  - MCU_GPIO0_[143:24] are not pinned out.
  - Interrupts [143:24] are not pinned out.
  - Bank Interrupts [8:2] are not pinned out.
- The following apply to GPIO0:
  - GPIO0_[143:87] are not pinned out.
  - Interrupts [143:92] are not pinned out.
  - Bank Interrupts [8:6] are not pinned out.
- The following apply to GPIO1:
  - GPIO1_[143:73], GPIO1_[41:32] are not pinned out.
  - Interrupts [143:72] are not pinned out
  - Bank Interrupts [8:4] are not pinned out.

4.9.1.2 GPIO Integration

Table 4-274. GPIO Module Allocations

Instance	MAIN	MCU	WKUP
GPIO0	✓		
GPIO1	✓		
MCU_GPIO0		✓	

Table 4-275. Integration Attributes for GPIO

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
GPIO0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
GPIO1	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
MCU_GPIO0	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	

Table 4-276. Clock Integration for GPIO

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GPIO0	FICLK	MAIN_SYSCLK0/4		functional and interface clock
GPIO1	FICLK	MAIN_SYSCLK0/4		functional and interface clock
MCU_GPIO0	FICLK	MCU_SYSCLK0/4	MCU_CTRL_MMR_CFG0_MCU_GPIO_CLKSEL[1:0]=0	functional and interface clock
MCU_GPIO0	FICLK	LFOSC0 (INSTANCE)/4	MCU_CTRL_MMR_CFG0_MCU_GPIO_CLKSEL[1:0]=1	functional and interface clock
MCU_GPIO0	FICLK	CLK_32K_RC/4	MCU_CTRL_MMR_CFG0_MCU_GPIO_CLKSEL[1:0]=2	functional and interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-276. Clock Integration for GPIO (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_GPIO0	FICLK	CLK_12M_RC/4	MCU_CTRL_MMR_CFG0_MCU_GPIO_CLKSEL[1:0]=3	functional and interface clock

Table 4-277. Reset Integration for GPIO

Module Instance	Source	Description
GPIO0	PSC0	GPIO0 reset
GPIO1	PSC0	GPIO1 reset
MCU_GPIO0	WKUP_PSC0	MCU_GPIO0 reset

Table 4-278. Hardware Requests for GPIO

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO0_gpio_0	MAIN_GPIOMUX_INTROUTER0_in_0	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_1	MAIN_GPIOMUX_INTROUTER0_in_1	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_2	MAIN_GPIOMUX_INTROUTER0_in_2	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_3	MAIN_GPIOMUX_INTROUTER0_in_3	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_4	MAIN_GPIOMUX_INTROUTER0_in_4	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_5	MAIN_GPIOMUX_INTROUTER0_in_5	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_6	MAIN_GPIOMUX_INTROUTER0_in_6	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_7	MAIN_GPIOMUX_INTROUTER0_in_7	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_8	MAIN_GPIOMUX_INTROUTER0_in_8	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_9	MAIN_GPIOMUX_INTROUTER0_in_9	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_10	MAIN_GPIOMUX_INTROUTER0_in_10	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_11	MAIN_GPIOMUX_INTROUTER0_in_11	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_12	MAIN_GPIOMUX_INTROUTER0_in_12	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_13	MAIN_GPIOMUX_INTROUTER0_in_13	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_14	MAIN_GPIOMUX_INTROUTER0_in_14	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_15	MAIN_GPIOMUX_INTROUTER0_in_15	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_16	MAIN_GPIOMUX_INTROUTER0_in_16	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_17	MAIN_GPIOMUX_INTROUTER0_in_17	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_18	MAIN_GPIOMUX_INTROUTER0_in_18	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO0_gpio_19	MAIN_GPIOMUX_INTROUTER0_in_19	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_20	MAIN_GPIOMUX_INTROUTER0_in_20	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_21	MAIN_GPIOMUX_INTROUTER0_in_21	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_22	MAIN_GPIOMUX_INTROUTER0_in_22	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_23	MAIN_GPIOMUX_INTROUTER0_in_23	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_24	MAIN_GPIOMUX_INTROUTER0_in_24	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_25	MAIN_GPIOMUX_INTROUTER0_in_25	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_26	MAIN_GPIOMUX_INTROUTER0_in_26	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_27	MAIN_GPIOMUX_INTROUTER0_in_27	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_28	MAIN_GPIOMUX_INTROUTER0_in_28	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_29	MAIN_GPIOMUX_INTROUTER0_in_29	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_30	MAIN_GPIOMUX_INTROUTER0_in_30	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_31	MAIN_GPIOMUX_INTROUTER0_in_31	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_32	MAIN_GPIOMUX_INTROUTER0_in_32	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_33	MAIN_GPIOMUX_INTROUTER0_in_33	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_34	MAIN_GPIOMUX_INTROUTER0_in_34	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_35	MAIN_GPIOMUX_INTROUTER0_in_35	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_36	MAIN_GPIOMUX_INTROUTER0_in_36	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_37	MAIN_GPIOMUX_INTROUTER0_in_37	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_38	MAIN_GPIOMUX_INTROUTER0_in_38	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_39	MAIN_GPIOMUX_INTROUTER0_in_39	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_40	MAIN_GPIOMUX_INTROUTER0_in_40	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_41	MAIN_GPIOMUX_INTROUTER0_in_41	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_42	MAIN_GPIOMUX_INTROUTER0_in_42	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_43	MAIN_GPIOMUX_INTROUTER0_in_43	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_44	MAIN_GPIOMUX_INTROUTER0_in_44	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO0_gpio_45	MAIN_GPIOMUX_INTROUTER0_in_45	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_46	MAIN_GPIOMUX_INTROUTER0_in_46	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_47	MAIN_GPIOMUX_INTROUTER0_in_47	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_48	MAIN_GPIOMUX_INTROUTER0_in_48	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_49	MAIN_GPIOMUX_INTROUTER0_in_49	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_50	MAIN_GPIOMUX_INTROUTER0_in_50	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_51	MAIN_GPIOMUX_INTROUTER0_in_51	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_52	MAIN_GPIOMUX_INTROUTER0_in_52	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_53	MAIN_GPIOMUX_INTROUTER0_in_53	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_54	MAIN_GPIOMUX_INTROUTER0_in_54	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_55	MAIN_GPIOMUX_INTROUTER0_in_55	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_56	MAIN_GPIOMUX_INTROUTER0_in_56	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_57	MAIN_GPIOMUX_INTROUTER0_in_57	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_58	MAIN_GPIOMUX_INTROUTER0_in_58	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_59	MAIN_GPIOMUX_INTROUTER0_in_59	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_60	MAIN_GPIOMUX_INTROUTER0_in_60	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_61	MAIN_GPIOMUX_INTROUTER0_in_61	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_62	MAIN_GPIOMUX_INTROUTER0_in_62	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_63	MAIN_GPIOMUX_INTROUTER0_in_63	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_64	MAIN_GPIOMUX_INTROUTER0_in_64	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_65	MAIN_GPIOMUX_INTROUTER0_in_65	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_66	MAIN_GPIOMUX_INTROUTER0_in_66	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_67	MAIN_GPIOMUX_INTROUTER0_in_67	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_68	MAIN_GPIOMUX_INTROUTER0_in_68	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_69	MAIN_GPIOMUX_INTROUTER0_in_69	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_70	MAIN_GPIOMUX_INTROUTER0_in_70	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO0_gpio_71	MAIN_GPIOMUX_INTROUTER0_in_71	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_72	MAIN_GPIOMUX_INTROUTER0_in_72	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_73	MAIN_GPIOMUX_INTROUTER0_in_73	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_74	MAIN_GPIOMUX_INTROUTER0_in_74	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_75	MAIN_GPIOMUX_INTROUTER0_in_75	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_76	MAIN_GPIOMUX_INTROUTER0_in_76	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_77	MAIN_GPIOMUX_INTROUTER0_in_77	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_78	MAIN_GPIOMUX_INTROUTER0_in_78	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_79	MAIN_GPIOMUX_INTROUTER0_in_79	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_80	MAIN_GPIOMUX_INTROUTER0_in_80	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_81	MAIN_GPIOMUX_INTROUTER0_in_81	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_82	MAIN_GPIOMUX_INTROUTER0_in_82	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_83	MAIN_GPIOMUX_INTROUTER0_in_83	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_84	MAIN_GPIOMUX_INTROUTER0_in_84	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_85	MAIN_GPIOMUX_INTROUTER0_in_85	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_86	MAIN_GPIOMUX_INTROUTER0_in_86	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_87	MAIN_GPIOMUX_INTROUTER0_in_87	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_88	MAIN_GPIOMUX_INTROUTER0_in_88	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_89	MAIN_GPIOMUX_INTROUTER0_in_89	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_90	MAIN_GPIOMUX_INTROUTER0_in_176	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_91	MAIN_GPIOMUX_INTROUTER0_in_177	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_0	MAIN_GPIOMUX_INTROUTER0_in_190	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_0	WKUP_R5FSS0_CORE0_intr_56	WKUP_R5FSS0_CORE0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_1	MAIN_GPIOMUX_INTROUTER0_in_191	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_1	WKUP_R5FSS0_CORE0_intr_57	WKUP_R5FSS0_CORE0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_2	MAIN_GPIOMUX_INTROUTER0_in_192	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO0	GPIO0_gpio_bank_3	MAIN_GPIOMUX_INTROUTER0_in_193	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_4	MAIN_GPIOMUX_INTROUTER0_in_194	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO0	GPIO0_gpio_bank_5	MAIN_GPIOMUX_INTROUTER0_in_195	MAIN_GPIOMUX_INTROUTER0	GPIO0 interrupt request	pulse
GPIO1	GPIO1_gpio_0	MAIN_GPIOMUX_INTROUTER0_in_90	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_1	MAIN_GPIOMUX_INTROUTER0_in_91	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_2	MAIN_GPIOMUX_INTROUTER0_in_92	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_3	MAIN_GPIOMUX_INTROUTER0_in_93	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_4	MAIN_GPIOMUX_INTROUTER0_in_94	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_5	MAIN_GPIOMUX_INTROUTER0_in_95	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_6	MAIN_GPIOMUX_INTROUTER0_in_96	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_7	MAIN_GPIOMUX_INTROUTER0_in_97	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_8	MAIN_GPIOMUX_INTROUTER0_in_98	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_9	MAIN_GPIOMUX_INTROUTER0_in_99	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_10	MAIN_GPIOMUX_INTROUTER0_in_100	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_11	MAIN_GPIOMUX_INTROUTER0_in_101	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_12	MAIN_GPIOMUX_INTROUTER0_in_102	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_13	MAIN_GPIOMUX_INTROUTER0_in_103	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_14	MAIN_GPIOMUX_INTROUTER0_in_104	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_15	MAIN_GPIOMUX_INTROUTER0_in_105	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_16	MAIN_GPIOMUX_INTROUTER0_in_106	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_17	MAIN_GPIOMUX_INTROUTER0_in_107	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_18	MAIN_GPIOMUX_INTROUTER0_in_108	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_19	MAIN_GPIOMUX_INTROUTER0_in_109	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_20	MAIN_GPIOMUX_INTROUTER0_in_110	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_21	MAIN_GPIOMUX_INTROUTER0_in_111	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_22	MAIN_GPIOMUX_INTROUTER0_in_112	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO1	GPIO1_gpio_23	MAIN_GPIOMUX_INTROUTER0_in_113	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_24	MAIN_GPIOMUX_INTROUTER0_in_114	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_25	MAIN_GPIOMUX_INTROUTER0_in_115	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_26	MAIN_GPIOMUX_INTROUTER0_in_116	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_27	MAIN_GPIOMUX_INTROUTER0_in_117	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_28	MAIN_GPIOMUX_INTROUTER0_in_118	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_29	MAIN_GPIOMUX_INTROUTER0_in_119	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_30	MAIN_GPIOMUX_INTROUTER0_in_120	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_31	MAIN_GPIOMUX_INTROUTER0_in_121	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_32	MAIN_GPIOMUX_INTROUTER0_in_122	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_33	MAIN_GPIOMUX_INTROUTER0_in_123	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_34	MAIN_GPIOMUX_INTROUTER0_in_124	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_35	MAIN_GPIOMUX_INTROUTER0_in_125	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_36	MAIN_GPIOMUX_INTROUTER0_in_126	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_37	MAIN_GPIOMUX_INTROUTER0_in_127	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_38	MAIN_GPIOMUX_INTROUTER0_in_128	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_39	MAIN_GPIOMUX_INTROUTER0_in_129	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_40	MAIN_GPIOMUX_INTROUTER0_in_130	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_41	MAIN_GPIOMUX_INTROUTER0_in_131	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_42	MAIN_GPIOMUX_INTROUTER0_in_132	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_43	MAIN_GPIOMUX_INTROUTER0_in_133	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_44	MAIN_GPIOMUX_INTROUTER0_in_134	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_45	MAIN_GPIOMUX_INTROUTER0_in_135	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_46	MAIN_GPIOMUX_INTROUTER0_in_136	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_47	MAIN_GPIOMUX_INTROUTER0_in_137	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_48	MAIN_GPIOMUX_INTROUTER0_in_138	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO1	GPIO1_gpio_49	MAIN_GPIOMUX_INTROUTER0_in_139	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_50	MAIN_GPIOMUX_INTROUTER0_in_140	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_51	MAIN_GPIOMUX_INTROUTER0_in_141	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_52	MAIN_GPIOMUX_INTROUTER0_in_142	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_53	MAIN_GPIOMUX_INTROUTER0_in_143	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_54	MAIN_GPIOMUX_INTROUTER0_in_144	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_55	MAIN_GPIOMUX_INTROUTER0_in_145	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_56	MAIN_GPIOMUX_INTROUTER0_in_146	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_57	MAIN_GPIOMUX_INTROUTER0_in_147	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_58	MAIN_GPIOMUX_INTROUTER0_in_148	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_59	MAIN_GPIOMUX_INTROUTER0_in_149	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_60	MAIN_GPIOMUX_INTROUTER0_in_150	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_61	MAIN_GPIOMUX_INTROUTER0_in_151	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_62	MAIN_GPIOMUX_INTROUTER0_in_152	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_63	MAIN_GPIOMUX_INTROUTER0_in_153	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_64	MAIN_GPIOMUX_INTROUTER0_in_154	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_65	MAIN_GPIOMUX_INTROUTER0_in_155	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_66	MAIN_GPIOMUX_INTROUTER0_in_156	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_67	MAIN_GPIOMUX_INTROUTER0_in_157	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_68	MAIN_GPIOMUX_INTROUTER0_in_158	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_69	MAIN_GPIOMUX_INTROUTER0_in_159	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_70	MAIN_GPIOMUX_INTROUTER0_in_160	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_71	MAIN_GPIOMUX_INTROUTER0_in_161	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_bank_0	MAIN_GPIOMUX_INTROUTER0_in_180	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_bank_1	MAIN_GPIOMUX_INTROUTER0_in_181	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_bank_2	MAIN_GPIOMUX_INTROUTER0_in_182	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPIO1	GPIO1_gpio_bank_3	MAIN_GPIOMUX_INTROUTER0_in_183	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_bank_4	MAIN_GPIOMUX_INTROUTER0_in_184	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
GPIO1	GPIO1_gpio_bank_5	MAIN_GPIOMUX_INTROUTER0_in_185	MAIN_GPIOMUX_INTROUTER0	GPIO1 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_0	WKUP_MCU_GPIOMUX_INTROUTER0_in_0	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_1	WKUP_MCU_GPIOMUX_INTROUTER0_in_1	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_2	WKUP_MCU_GPIOMUX_INTROUTER0_in_2	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_3	WKUP_MCU_GPIOMUX_INTROUTER0_in_3	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_4	WKUP_MCU_GPIOMUX_INTROUTER0_in_4	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_5	WKUP_MCU_GPIOMUX_INTROUTER0_in_5	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_6	WKUP_MCU_GPIOMUX_INTROUTER0_in_6	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_7	WKUP_MCU_GPIOMUX_INTROUTER0_in_7	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_8	WKUP_MCU_GPIOMUX_INTROUTER0_in_8	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_9	WKUP_MCU_GPIOMUX_INTROUTER0_in_9	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_10	WKUP_MCU_GPIOMUX_INTROUTER0_in_10	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_11	WKUP_MCU_GPIOMUX_INTROUTER0_in_11	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_12	WKUP_MCU_GPIOMUX_INTROUTER0_in_12	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_13	WKUP_MCU_GPIOMUX_INTROUTER0_in_13	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_14	WKUP_MCU_GPIOMUX_INTROUTER0_in_14	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_15	WKUP_MCU_GPIOMUX_INTROUTER0_in_15	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-278. Hardware Requests for GPIO (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_GPIO0	MCU_GPIO0_gpio_16	WKUP_MCU_GPIOMUX_INTROUTER0_in_16	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_17	WKUP_MCU_GPIOMUX_INTROUTER0_in_17	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_18	WKUP_MCU_GPIOMUX_INTROUTER0_in_18	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_19	WKUP_MCU_GPIOMUX_INTROUTER0_in_19	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_20	WKUP_MCU_GPIOMUX_INTROUTER0_in_20	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_21	WKUP_MCU_GPIOMUX_INTROUTER0_in_21	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_22	WKUP_MCU_GPIOMUX_INTROUTER0_in_22	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_23	WKUP_MCU_GPIOMUX_INTROUTER0_in_23	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_bank_0	WKUP_MCU_GPIOMUX_INTROUTER0_in_30	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_bank_1	WKUP_MCU_GPIOMUX_INTROUTER0_in_31	WKUP_MCU_GPIOMUX_INTROUTER0	MCU_GPIO0 interrupt request	pulse
MCU_GPIO0	MCU_GPIO0_gpio_lvl_0	WKUP_R5FSS0_CORE0_intr_18	WKUP_R5FSS0_CORE0	MCU_GPIO0 interrupt request	level
MCU_GPIO0	MCU_GPIO0_gpio_lvl_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_2	WKUP_DEEPSLEEP_SOURCES0	MCU_GPIO0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.9.2 Inter-Integrated Circiuit (I2C)

This section contains the integration details for the I2C module on this device. For further information, see the *Inter-Integrated Circiuit (I2C)* section of the *Peripherals* chapter.

4.9.2.1 I2C Unsupported Features

The following features are not supported on this family of devices:

- Serial Camera Control Bus (SCCB) protocol
- DMA Mode
- Debug suspend mode

The following apply to Main domain instances:

- Full I^2C electrical compliance for I2C modules using device pins with LVCMOS voltage buffers (Refer to chapter 4 of the device-specific Datasheet for details related to device *Terminal Configuration and Functions*)
- High-speed (3.4-Mbps) operation for I2C modules using device pins with LVCMOS voltage buffers (Refer to chapter 4 of the device-specific Datasheet for details related to device *Terminal Configuration and Functions*)
- Asynchronous wakeup (via IO Daisy Chain)

The following apply to MCU domain instances:

- Direct asynchronous wakeup for MCU domain instances

4.9.2.2 I2C Integration

Table 4-279. I2C Module Allocations

Instance	MAIN	MCU	WKUP
I2C0	✓		
I2C1	✓		
I2C2	✓		
I2C3	✓		
I2C4	✓		
MCU_I2C0		✓	
WKUP_I2C0			✓

Table 4-280. Integration Attributes for I2C

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
I2C0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
I2C1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
I2C2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
I2C3	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCU_I2C0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
WKUP_I2C0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-281. Clock Integration for I2C

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
I2C0	ICLK	MAIN_SYSCLK0/4		interface clock
I2C0	FCLK	MAIN_PLL1_HSDIV0_CLKOUT/2		functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-281. Clock Integration for I2C (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
I2C1	ICLK	MAIN_SYSCLK0/4		interface clock
I2C1	FCLK	MAIN_PLL1_HSDIV0_CLKOUT/2		functional clock
I2C2	ICLK	MAIN_SYSCLK0/4		interface clock
I2C2	FCLK	MAIN_PLL1_HSDIV0_CLKOUT/2		functional clock
I2C3	ICLK	MAIN_SYSCLK0/4		interface clock
I2C3	FCLK	MAIN_PLL1_HSDIV0_CLKOUT/2		functional clock
I2C4	ICLK	MAIN_SYSCLK0/4		interface clock
I2C4	FCLK	MAIN_PLL1_HSDIV0_CLKOUT/2		functional clock
MCU_I2C0	ICLK	MCU_SYSCLK0/4		interface clock
MCU_I2C0	FCLK	MCU_PLL0_HSDIV1_CLKOUT		functional clock
WKUP_I2C0	ICLK	DM_CLK/4		interface clock
WKUP_I2C0	FCLK	MCU_PLL0_HSDIV1_CLKOUT		functional clock

Table 4-282. Reset Integration for I2C

Module Instance	Source	Description
I2C0	PSC0	I2C0 reset
I2C1	PSC0	I2C1 reset
I2C2	PSC0	I2C2 reset
I2C3	PSC0	I2C3 reset
MCU_I2C0	WKUP_PSC0	MCU_I2C0 reset
WKUP_I2C0	PSC0	WKUP_I2C0 reset

Table 4-283. Hardware Requests for I2C

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
I2C0	I2C0_pointrpend_0	GICSS0_spi_193	GICSS0	I2C0 interrupt request	level
I2C0	I2C0_pointrpend_0	R5FSS0_CORE0_intr_193	R5FSS0_CORE0	I2C0 interrupt request	level
I2C0	I2C0_pointrpend_0	WKUP_R5FSS0_CORE0_intr_193	WKUP_R5FSS0_CORE0	I2C0 interrupt request	level
I2C0	I2C0_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_193	MCU_R5FSS0_CORE0	I2C0 interrupt request	level
I2C0	I2C0_pointrpend_0	C7X256V0_CLEC_gic_spi_193	C7X256V0_CLEC	I2C0 interrupt request	level
I2C0	I2C0_pointrpend_0	C7X256V1_CLEC_gic_spi_193	C7X256V1_CLEC	I2C0 interrupt request	level
I2C1	I2C1_pointrpend_0	GICSS0_spi_194	GICSS0	I2C1 interrupt request	level
I2C1	I2C1_pointrpend_0	R5FSS0_CORE0_intr_194	R5FSS0_CORE0	I2C1 interrupt request	level
I2C1	I2C1_pointrpend_0	WKUP_R5FSS0_CORE0_intr_194	WKUP_R5FSS0_CORE0	I2C1 interrupt request	level
I2C1	I2C1_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_194	MCU_R5FSS0_CORE0	I2C1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-283. Hardware Requests for I2C (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
I2C1	I2C1_pointrpend_0	C7X256V0_CLEC_gic_spi_194	C7X256V0_CLEC	I2C1 interrupt request	level
I2C1	I2C1_pointrpend_0	C7X256V1_CLEC_gic_spi_194	C7X256V1_CLEC	I2C1 interrupt request	level
I2C2	I2C2_pointrpend_0	GICSS0_spi_195	GICSS0	I2C2 interrupt request	level
I2C2	I2C2_pointrpend_0	R5FSS0_CORE0_intr_195	R5FSS0_CORE0	I2C2 interrupt request	level
I2C2	I2C2_pointrpend_0	WKUP_R5FSS0_CORE0_intr_195	WKUP_R5FSS0_CORE0	I2C2 interrupt request	level
I2C2	I2C2_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_195	MCU_R5FSS0_CORE0	I2C2 interrupt request	level
I2C2	I2C2_pointrpend_0	C7X256V0_CLEC_gic_spi_195	C7X256V0_CLEC	I2C2 interrupt request	level
I2C2	I2C2_pointrpend_0	C7X256V1_CLEC_gic_spi_195	C7X256V1_CLEC	I2C2 interrupt request	level
I2C3	I2C3_pointrpend_0	GICSS0_spi_196	GICSS0	I2C3 interrupt request	level
I2C3	I2C3_pointrpend_0	R5FSS0_CORE0_intr_196	R5FSS0_CORE0	I2C3 interrupt request	level
I2C3	I2C3_pointrpend_0	WKUP_R5FSS0_CORE0_intr_196	WKUP_R5FSS0_CORE0	I2C3 interrupt request	level
I2C3	I2C3_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_196	MCU_R5FSS0_CORE0	I2C3 interrupt request	level
I2C3	I2C3_pointrpend_0	C7X256V0_CLEC_gic_spi_196	C7X256V0_CLEC	I2C3 interrupt request	level
I2C3	I2C3_pointrpend_0	C7X256V1_CLEC_gic_spi_196	C7X256V1_CLEC	I2C3 interrupt request	level
I2C4	I2C4_pointrpend_0	MAIN_GPIOMUX_INTROUTER0_in_178	MAIN_GPIOMUX_INTROUTER0	I2C4 interrupt request	level
I2C4	I2C4_pointrpend_0	R5FSS0_CORE0_intr_239	R5FSS0_CORE0	I2C4 interrupt request	level
I2C4	I2C4_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_239	MCU_R5FSS0_CORE0	I2C4 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	GICSS0_spi_139	GICSS0	MCU_I2C0 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	R5FSS0_CORE0_intr_197	R5FSS0_CORE0	MCU_I2C0 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	WKUP_R5FSS0_CORE0_intr_197	WKUP_R5FSS0_CORE0	MCU_I2C0 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_197	MCU_R5FSS0_CORE0	MCU_I2C0 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	C7X256V0_CLEC_gic_spi_139	C7X256V0_CLEC	MCU_I2C0 interrupt request	level
MCU_I2C0	MCU_I2C0_pointrpend_0	C7X256V1_CLEC_gic_spi_139	C7X256V1_CLEC	MCU_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_clkstop_wakeup_0	WKUP_R5FSS0_CORE0_intr_143	WKUP_R5FSS0_CORE0	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_clkstop_wakeup_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_0	WKUP_DEEPSLEEP_SOURCES0	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_pointrpend_0	GICSS0_spi_197	GICSS0	WKUP_I2C0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-283. Hardware Requests for I2C (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_I2C0	WKUP_I2C0_pointrpend_0	R5FSS0_CORE0_intr_190	R5FSS0_CORE0	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_pointrpend_0	WKUP_R5FSS0_CORE0_intr_190	WKUP_R5FSS0_CORE0	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_pointrpend_0	MCU_R5FSS0_CORE0_cpu0_intr_190	MCU_R5FSS0_CORE0	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_pointrpend_0	C7X256V0_CLEC_gic_spi_197	C7X256V0_CLEC	WKUP_I2C0 interrupt request	level
WKUP_I2C0	WKUP_I2C0_pointrpend_0	C7X256V1_CLEC_gic_spi_197	C7X256V1_CLEC	WKUP_I2C0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.9.3 Multichannel Serial Peripheral Interface (MCSPI)

This section contains the integration details for the MCSPI module on this device. For further information, see the *Multichannel Serial Peripheral Interface (MCSPI)* section of the *Peripherals* chapter.

4.9.3.1 MCSPI Unsupported Features

The following features are not supported on this family of devices:

- Peripheral mode wakeup.
- Retention during power down

4.9.3.2 MCSPI Integration

Table 4-284. MCSPI Module Allocations

Instance	MAIN	MCU	WKUP
MCSPI0	✓		
MCSPI1	✓		
MCSPI2	✓		
MCU_MCSPI0		✓	
MCU_MCSPI1		✓	

Table 4-285. Integration Attributes for MCSPI

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCSPI0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCSPI1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCSPI2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCU_MCSPI0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
MCU_MCSPI1	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO

Table 4-286. Clock Integration for MCSPI

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCSPI0	FCLK	MAIN_PLL2_HSDIV5_CLKOUT/10		functional clock
MCSPI0	ICLK	MAIN_SYSCLK0/4		interface clock
MCSPI1	FCLK	MAIN_PLL2_HSDIV5_CLKOUT/10		functional clock
MCSPI1	ICLK	MAIN_SYSCLK0/4		interface clock
MCSPI2	FCLK	MAIN_PLL2_HSDIV5_CLKOUT/10		functional clock
MCSPI2	ICLK	MAIN_SYSCLK0/4		interface clock
MCU_MCSPI0	FCLK	MCU_PLL0_HSDIV6_CLKOUT/8		functional clock
MCU_MCSPI0	ICLK	MCU_SYSCLK0/2		interface clock
MCU_MCSPI1	FCLK	MCU_PLL0_HSDIV6_CLKOUT/8		functional clock
MCU_MCSPI1	ICLK	MCU_SYSCLK0/2		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-287. Reset Integration for MCSPI

Module Instance	Source	Description
MCSPI0	PSC0	MCSPI0 reset
MCSPI1	PSC0	MCSPI1 reset
MCSPI2	PSC0	MCSPI2 reset
MCU_MCSPI0	WKUP_PSC0	MCU_MCSPI0 reset
MCU_MCSPI1	WKUP_PSC0	MCU_MCSPI1 reset

Table 4-288. Hardware Requests for MCSPI

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCSPI0	MCSPI0_dma_read_event_0	PDMA0_spi_main_0_rx_0	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_read_event_1	PDMA0_spi_main_0_rx_1	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_read_event_2	PDMA0_spi_main_0_rx_2	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_read_event_3	PDMA0_spi_main_0_rx_3	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_write_event_0	PDMA0_spi_main_0_tx_0	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_write_event_1	PDMA0_spi_main_0_tx_1	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_write_event_2	PDMA0_spi_main_0_tx_2	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_dma_write_event_3	PDMA0_spi_main_0_tx_3	PDMA0	MCSPI0 interrupt request	pulse
MCSPI0	MCSPI0_intr_spi_0	GICSS0_spi_204	GICSS0	MCSPI0 interrupt request	level
MCSPI0	MCSPI0_intr_spi_0	R5FSS0_CORE0_intr_204	R5FSS0_CORE0	MCSPI0 interrupt request	level
MCSPI0	MCSPI0_intr_spi_0	WKUP_R5FSS0_CORE0_intr_204	WKUP_R5FSS0_CORE0	MCSPI0 interrupt request	level
MCSPI0	MCSPI0_intr_spi_0	MCU_R5FSS0_CORE0_cpu0_intr_204	MCU_R5FSS0_CORE0	MCSPI0 interrupt request	level
MCSPI0	MCSPI0_intr_spi_0	C7X256V0_CLEC_gic_spi_204	C7X256V0_CLEC	MCSPI0 interrupt request	level
MCSPI0	MCSPI0_intr_spi_0	C7X256V1_CLEC_gic_spi_204	C7X256V1_CLEC	MCSPI0 interrupt request	level
MCSPI1	MCSPI1_dma_read_event_0	PDMA0_spi_main_1_rx_0	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_read_event_1	PDMA0_spi_main_1_rx_1	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_read_event_2	PDMA0_spi_main_1_rx_2	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_read_event_3	PDMA0_spi_main_1_rx_3	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_write_event_0	PDMA0_spi_main_1_tx_0	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_write_event_1	PDMA0_spi_main_1_tx_1	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_write_event_2	PDMA0_spi_main_1_tx_2	PDMA0	MCSPI1 interrupt request	pulse
MCSPI1	MCSPI1_dma_write_event_3	PDMA0_spi_main_1_tx_3	PDMA0	MCSPI1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-288. Hardware Requests for MCSPI (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCSPI1	MCSPI1_intr_spi_0	GICSS0_spi_205	GICSS0	MCSPI1 interrupt request	level
MCSPI1	MCSPI1_intr_spi_0	R5FSS0_CORE0_intr_205	R5FSS0_CORE0	MCSPI1 interrupt request	level
MCSPI1	MCSPI1_intr_spi_0	WKUP_R5FSS0_CORE0_intr_205	WKUP_R5FSS0_CORE0	MCSPI1 interrupt request	level
MCSPI1	MCSPI1_intr_spi_0	MCU_R5FSS0_CORE0_cpu0_intr_205	MCU_R5FSS0_CORE0	MCSPI1 interrupt request	level
MCSPI1	MCSPI1_intr_spi_0	C7X256V0_CLEC_gic_spi_205	C7X256V0_CLEC	MCSPI1 interrupt request	level
MCSPI1	MCSPI1_intr_spi_0	C7X256V1_CLEC_gic_spi_205	C7X256V1_CLEC	MCSPI1 interrupt request	level
MCSPI2	MCSPI2_dma_read_event_0	PDMA0_spi_main_2_rx_0	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_read_event_1	PDMA0_spi_main_2_rx_1	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_read_event_2	PDMA0_spi_main_2_rx_2	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_read_event_3	PDMA0_spi_main_2_rx_3	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_write_event_0	PDMA0_spi_main_2_tx_0	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_write_event_1	PDMA0_spi_main_2_tx_1	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_write_event_2	PDMA0_spi_main_2_tx_2	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_dma_write_event_3	PDMA0_spi_main_2_tx_3	PDMA0	MCSPI2 interrupt request	pulse
MCSPI2	MCSPI2_intr_spi_0	GICSS0_spi_206	GICSS0	MCSPI2 interrupt request	level
MCSPI2	MCSPI2_intr_spi_0	R5FSS0_CORE0_intr_206	R5FSS0_CORE0	MCSPI2 interrupt request	level
MCSPI2	MCSPI2_intr_spi_0	WKUP_R5FSS0_CORE0_intr_206	WKUP_R5FSS0_CORE0	MCSPI2 interrupt request	level
MCSPI2	MCSPI2_intr_spi_0	MCU_R5FSS0_CORE0_cpu0_intr_206	MCU_R5FSS0_CORE0	MCSPI2 interrupt request	level
MCSPI2	MCSPI2_intr_spi_0	C7X256V0_CLEC_gic_spi_206	C7X256V0_CLEC	MCSPI2 interrupt request	level
MCSPI2	MCSPI2_intr_spi_0	C7X256V1_CLEC_gic_spi_206	C7X256V1_CLEC	MCSPI2 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	GICSS0_spi_208	GICSS0	MCU_MCSPI0 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	R5FSS0_CORE0_intr_207	R5FSS0_CORE0	MCU_MCSPI0 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	WKUP_R5FSS0_CORE0_intr_207	WKUP_R5FSS0_CORE0	MCU_MCSPI0 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	MCU_R5FSS0_CORE0_cpu0_intr_207	MCU_R5FSS0_CORE0	MCU_MCSPI0 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	C7X256V0_CLEC_gic_spi_208	C7X256V0_CLEC	MCU_MCSPI0 interrupt request	level
MCU_MCSPI0	MCU_MCSPI0_intr_spi_0	C7X256V1_CLEC_gic_spi_208	C7X256V1_CLEC	MCU_MCSPI0 interrupt request	level
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	GICSS0_spi_209	GICSS0	MCU_MCSPI1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-288. Hardware Requests for MCSPI (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	R5FSS0_CORE0_intr_208	R5FSS0_CORE0	MCU_MCSPI1 interrupt request	level
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	WKUP_R5FSS0_CORE0_intr_208	WKUP_R5FSS0_CORE0	MCU_MCSPI1 interrupt request	level
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	MCU_R5FSS0_CORE0_cpu0_intr_208	MCU_R5FSS0_CORE0	MCU_MCSPI1 interrupt request	level
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	C7X256V0_CLEC_gic_spi_209	C7X256V0_CLEC	MCU_MCSPI1 interrupt request	level
MCU_MCSPI1	MCU_MCSPI1_intr_spi_0	C7X256V1_CLEC_gic_spi_209	C7X256V1_CLEC	MCU_MCSPI1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.9.4 Universal Asynchronous Receiver/Transmitter (UART)

This section contains the integration details for the UART module on this device. For further information, see the *Universal Asynchronous Receiver/Transmitter (UART)* section of the *Peripherals* chapter.

4.9.4.1 UART Unsupported Features

The following features are not supported on this family of devices:

- 12Mbps for MCU and WKUP domains
- Full modem handshaking is not available on all instances. See device Datasheet for instances supporting full modem handshaking.
- Synchronous mode: SCLK is not pinned out
- ISO7816 mode
- SCR DMA mode 2
- Multi-drop transmission
- 9-bit mode

4.9.4.2 UART Integration

Table 4-289. UART Module Allocations

Instance	MAIN	MCU	WKUP
UART0	✓		
UART1	✓		
UART2	✓		
UART3	✓		
UART4	✓		
UART5	✓		
UART6	✓		
MCU_UART0		✓	
WKUP_UART0			✓

Table 4-290. Integration Attributes for UART

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
UART0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART3	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART4	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART5	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
UART6	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCU_UART0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
WKUP_UART0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Copyright © 2026 Texas Instruments Incorporated

Table 4-291. Clock Integration for UART

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
UART0	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART0_CLKSEL[0:0]=0 USART0_CLK_CTRL[1:0]	functional clock
UART0	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART0_CLKSEL[0:0]=1	functional clock
UART0	ICLK	MAIN_SYSCLK0/4		interface clock
UART1	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART1_CLKSEL[0:0]=0 USART1_CLK_CTRL[1:0]	functional clock
UART1	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART1_CLKSEL[0:0]=1	functional clock
UART1	ICLK	MAIN_SYSCLK0/4		interface clock
UART2	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART2_CLKSEL[0:0]=0 USART2_CLK_CTRL[1:0]	functional clock
UART2	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART2_CLKSEL[0:0]=1	functional clock
UART2	ICLK	MAIN_SYSCLK0/4		interface clock
UART3	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART3_CLKSEL[0:0]=0 USART3_CLK_CTRL[1:0]	functional clock
UART3	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART3_CLKSEL[0:0]=1	functional clock
UART3	ICLK	MAIN_SYSCLK0/4		interface clock
UART4	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART4_CLKSEL[0:0]=0 USART4_CLK_CTRL[1:0]	functional clock
UART4	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART4_CLKSEL[0:0]=1	functional clock
UART4	ICLK	MAIN_SYSCLK0/4		interface clock
UART5	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART5_CLKSEL[0:0]=0 USART5_CLK_CTRL[1:0]	functional clock
UART5	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART5_CLKSEL[0:0]=1	functional clock
UART5	ICLK	MAIN_SYSCLK0/4		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-291. Clock Integration for UART (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
UART6	FCLK	MAIN_PLL1_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_USART6_CLKSEL[0:0]=0 USART6_CLK_CTRL[1:0]	functional clock
UART6	FCLK	MAIN_PLL1_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_USART6_CLKSEL[0:0]=1	functional clock
UART6	ICLK	MAIN_SYSCLK0/4		interface clock
MCU_UART0	FCLK	MCU_PLL0_HSDIV2_CLKOUT		functional clock
MCU_UART0	ICLK	MCU_SYSCLK0/2		interface clock
WKUP_UART0	FCLK	MCU_PLL0_HSDIV2_CLKOUT		functional clock
WKUP_UART0	ICLK	DM_CLK/4		interface clock

Table 4-292. Reset Integration for UART

Module Instance	Source	Description
UART0	PSC0	UART0 reset
UART1	PSC0	UART1 reset
UART2	PSC0	UART2 reset
UART3	PSC0	UART3 reset
UART4	PSC0	UART4 reset
UART5	PSC0	UART5 reset
UART6	PSC0	UART6 reset
MCU_UART0	WKUP_PSC0	MCU_UART0 reset
WKUP_UART0	PSC0	WKUP_UART0 reset

Table 4-293. Hardware Requests for UART

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
UART0	UART0_usart_dma_0	PDMA1_usart_main_0_tx_0	PDMA1	UART0 interrupt request	level
UART0	UART0_usart_dma_1	PDMA1_usart_main_0_rx_0	PDMA1	UART0 interrupt request	level
UART0	UART0_usart_irq_0	GICSS0_spi_210	GICSS0	UART0 interrupt request	level
UART0	UART0_usart_irq_0	R5FSS0_CORE0_intr_210	R5FSS0_CORE0	UART0 interrupt request	level
UART0	UART0_usart_irq_0	WKUP_R5FSS0_CORE0_intr_210	WKUP_R5FSS0_CORE0	UART0 interrupt request	level
UART0	UART0_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_210	MCU_R5FSS0_CORE0	UART0 interrupt request	level
UART0	UART0_usart_irq_0	C7X256V0_CLEC_gic_spi_210	C7X256V0_CLEC	UART0 interrupt request	level
UART0	UART0_usart_irq_0	C7X256V1_CLEC_gic_spi_210	C7X256V1_CLEC	UART0 interrupt request	level
UART1	UART1_usart_dma_0	PDMA1_usart_main_1_tx_0	PDMA1	UART1 interrupt request	level
UART1	UART1_usart_dma_1	PDMA1_usart_main_1_rx_0	PDMA1	UART1 interrupt request	level
UART1	UART1_usart_irq_0	GICSS0_spi_211	GICSS0	UART1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-293. Hardware Requests for UART (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
UART1	UART1_usart_irq_0	R5FSS0_CORE0_intr_211	R5FSS0_CORE0	UART1 interrupt request	level
UART1	UART1_usart_irq_0	WKUP_R5FSS0_CORE0_intr_211	WKUP_R5FSS0_CORE0	UART1 interrupt request	level
UART1	UART1_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_211	MCU_R5FSS0_CORE0	UART1 interrupt request	level
UART1	UART1_usart_irq_0	C7X256V0_CLEC_gic_spi_211	C7X256V0_CLEC	UART1 interrupt request	level
UART1	UART1_usart_irq_0	C7X256V1_CLEC_gic_spi_211	C7X256V1_CLEC	UART1 interrupt request	level
UART2	UART2_usart_dma_0	PDMA1_usart_main_2_tx_0	PDMA1	UART2 interrupt request	level
UART2	UART2_usart_dma_1	PDMA1_usart_main_2_rx_0	PDMA1	UART2 interrupt request	level
UART2	UART2_usart_irq_0	GICSS0_spi_212	GICSS0	UART2 interrupt request	level
UART2	UART2_usart_irq_0	R5FSS0_CORE0_intr_212	R5FSS0_CORE0	UART2 interrupt request	level
UART2	UART2_usart_irq_0	WKUP_R5FSS0_CORE0_intr_212	WKUP_R5FSS0_CORE0	UART2 interrupt request	level
UART2	UART2_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_212	MCU_R5FSS0_CORE0	UART2 interrupt request	level
UART2	UART2_usart_irq_0	C7X256V0_CLEC_gic_spi_212	C7X256V0_CLEC	UART2 interrupt request	level
UART2	UART2_usart_irq_0	C7X256V1_CLEC_gic_spi_212	C7X256V1_CLEC	UART2 interrupt request	level
UART3	UART3_usart_dma_0	PDMA1_usart_main_3_tx_0	PDMA1	UART3 interrupt request	level
UART3	UART3_usart_dma_1	PDMA1_usart_main_3_rx_0	PDMA1	UART3 interrupt request	level
UART3	UART3_usart_irq_0	GICSS0_spi_213	GICSS0	UART3 interrupt request	level
UART3	UART3_usart_irq_0	R5FSS0_CORE0_intr_213	R5FSS0_CORE0	UART3 interrupt request	level
UART3	UART3_usart_irq_0	WKUP_R5FSS0_CORE0_intr_213	WKUP_R5FSS0_CORE0	UART3 interrupt request	level
UART3	UART3_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_213	MCU_R5FSS0_CORE0	UART3 interrupt request	level
UART3	UART3_usart_irq_0	C7X256V0_CLEC_gic_spi_213	C7X256V0_CLEC	UART3 interrupt request	level
UART3	UART3_usart_irq_0	C7X256V1_CLEC_gic_spi_213	C7X256V1_CLEC	UART3 interrupt request	level
UART4	UART4_usart_dma_0	PDMA1_usart_main_4_tx_0	PDMA1	UART4 interrupt request	level
UART4	UART4_usart_dma_1	PDMA1_usart_main_4_rx_0	PDMA1	UART4 interrupt request	level
UART4	UART4_usart_irq_0	GICSS0_spi_214	GICSS0	UART4 interrupt request	level
UART4	UART4_usart_irq_0	R5FSS0_CORE0_intr_214	R5FSS0_CORE0	UART4 interrupt request	level
UART4	UART4_usart_irq_0	WKUP_R5FSS0_CORE0_intr_214	WKUP_R5FSS0_CORE0	UART4 interrupt request	level
UART4	UART4_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_214	MCU_R5FSS0_CORE0	UART4 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-293. Hardware Requests for UART (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
UART4	UART4_usart_irq_0	C7X256V0_CLEC_gic_spi_214	C7X256V0_CLEC	UART4 interrupt request	level
UART4	UART4_usart_irq_0	C7X256V1_CLEC_gic_spi_214	C7X256V1_CLEC	UART4 interrupt request	level
UART5	UART5_usart_dma_0	PDMA1_usart_main_5_tx_0	PDMA1	UART5 interrupt request	level
UART5	UART5_usart_dma_1	PDMA1_usart_main_5_rx_0	PDMA1	UART5 interrupt request	level
UART5	UART5_usart_irq_0	GICSS0_spi_215	GICSS0	UART5 interrupt request	level
UART5	UART5_usart_irq_0	R5FSS0_CORE0_intr_215	R5FSS0_CORE0	UART5 interrupt request	level
UART5	UART5_usart_irq_0	WKUP_R5FSS0_CORE0_intr_215	WKUP_R5FSS0_CORE0	UART5 interrupt request	level
UART5	UART5_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_215	MCU_R5FSS0_CORE0	UART5 interrupt request	level
UART5	UART5_usart_irq_0	C7X256V0_CLEC_gic_spi_215	C7X256V0_CLEC	UART5 interrupt request	level
UART5	UART5_usart_irq_0	C7X256V1_CLEC_gic_spi_215	C7X256V1_CLEC	UART5 interrupt request	level
UART6	UART6_usart_dma_0	PDMA1_usart_main_6_tx_0	PDMA1	UART6 interrupt request	level
UART6	UART6_usart_dma_1	PDMA1_usart_main_6_rx_0	PDMA1	UART6 interrupt request	level
UART6	UART6_usart_irq_0	GICSS0_spi_216	GICSS0	UART6 interrupt request	level
UART6	UART6_usart_irq_0	R5FSS0_CORE0_intr_216	R5FSS0_CORE0	UART6 interrupt request	level
UART6	UART6_usart_irq_0	WKUP_R5FSS0_CORE0_intr_216	WKUP_R5FSS0_CORE0	UART6 interrupt request	level
UART6	UART6_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_216	MCU_R5FSS0_CORE0	UART6 interrupt request	level
UART6	UART6_usart_irq_0	C7X256V0_CLEC_gic_spi_216	C7X256V0_CLEC	UART6 interrupt request	level
UART6	UART6_usart_irq_0	C7X256V1_CLEC_gic_spi_216	C7X256V1_CLEC	UART6 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	GICSS0_spi_217	GICSS0	MCU_UART0 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	R5FSS0_CORE0_intr_217	R5FSS0_CORE0	MCU_UART0 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	WKUP_R5FSS0_CORE0_intr_217	WKUP_R5FSS0_CORE0	MCU_UART0 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_217	MCU_R5FSS0_CORE0	MCU_UART0 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	C7X256V0_CLEC_gic_spi_217	C7X256V0_CLEC	MCU_UART0 interrupt request	level
MCU_UART0	MCU_UART0_usart_irq_0	C7X256V1_CLEC_gic_spi_217	C7X256V1_CLEC	MCU_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_clkstop_wakeup_0	WKUP_R5FSS0_CORE0_intr_144	WKUP_R5FSS0_CORE0	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_clkstop_wakeup_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_1	WKUP_DEEPSLEEP_SOURCES0	WKUP_UART0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-293. Hardware Requests for UART (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_UART0	WKUP_UART0_usart_irq_0	GICSS0_spi_218	GICSS0	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_usart_irq_0	R5FSS0_CORE0_intr_219	R5FSS0_CORE0	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_usart_irq_0	WKUP_R5FSS0_CORE0_intr_219	WKUP_R5FSS0_CORE0	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_usart_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_219	MCU_R5FSS0_CORE0	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_usart_irq_0	C7X256V0_CLEC_gic_spi_218	C7X256V0_CLEC	WKUP_UART0 interrupt request	level
WKUP_UART0	WKUP_UART0_usart_irq_0	C7X256V1_CLEC_gic_spi_218	C7X256V1_CLEC	WKUP_UART0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.10 High-speed Serial Interfaces

4.10.1 Peripheral Component Interconnect Express (PCIe) Subsystem

This section contains the integration details for the PCIe module on this device. For further information, see the *Peripheral Component Interconnect Express (PCIe)* section of the *Peripherals* chapter.

4.10.1.1 PCIE Unsupported Features

The following features are not supported on this family of devices:

- Single Root I/O Virtualization (SR-IOV)
- ATS capability to support virtualization in root port mode
- PCIe beacon for in-band wake
- Vendor Defined Messages (VDM)
- I/O access in inbound direction in RP or EP modes
- Addressing modes other than incremental for burst transactions. Thus, the PCIe addresses cannot be in cacheable memory space.
- L2 power states
- Hot plug

4.10.1.2 PCIe Integration

Table 4-294. PCIE Module Allocations

Instance	MAIN	MCU	WKUP
PCIE0	✓		

Table 4-295. Integration Attributes for PCIe

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
PCIE0	PSC0	PD_PCIE	LPSC_main_pcie0	75	OFF	YES	LPSC_main_serdes1

Table 4-296. Clock Integration for PCIe

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PCIE0	PCIE_CBA_CLK	MAIN_SYSCLK0/2		None
PCIE0	PCIE_CPTS_RCLK_CLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=0	None
PCIE0	PCIE_CPTS_RCLK_CLK	MAIN_PLL0_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=1	None
PCIE0	PCIE_CPTS_RCLK_CLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=2	None
PCIE0	PCIE_CPTS_RCLK_CLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=4	None
PCIE0	PCIE_CPTS_RCLK_CLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=5	None
PCIE0	PCIE_CPTS_RCLK_CLK	SERDES_10G1 (INSTANCE)	MAIN_CTRL_MMR_CFG0_PCIE0_CLKSEL[2:0]=6	None
PCIE0	PCIE_LANE0_REFCLK	SERDES_10G1 (INSTANCE)		None
PCIE0	PCIE_LANE0_RXCLK	SERDES_10G1 (INSTANCE)		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-296. Clock Integration for PCIe (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
PCIE0	PCIE_LANE0_RXFCLK	SERDES_10G1 (INSTANCE)		None
PCIE0	PCIE_LANE0_TXFCLK	SERDES_10G1 (INSTANCE)		None
PCIE0	PCIE_LANE0_TXMCLK	SERDES_10G1 (INSTANCE)		None
PCIE0	PCIE_PM_CLK	CLK_12M_RC		None

Table 4-297. Reset Integration for PCIe

Module Instance	Source	Description
PCIE0	PSC0	PCIE0 reset

Table 4-298. Hardware Requests for PCIe

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PCIE0	PCIE0_pcie_cpts_comp_0	EPWM0_epwm_syncin_0	EPWM0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	DMASS0_INTAGGR_0_intaggr_levi_pend_1	DMASS0_INTAGGR_0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	GICSS0_spi_49	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	R5FSS0_CORE0_intr_49	R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	WKUP_R5FSS0_CORE0_intr_49	WKUP_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	MCU_R5FSS0_CORE0_cpu0_intr_49	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	C7X256V0_CLEC_gic_spi_49	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_comp_0	C7X256V1_CLEC_gic_spi_49	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_genf0_0	TIMESYNC_EVENT_INTROUTER0_in_4	TIMESYNC_EVENT_INTROUTER0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_hw1_push_0	TIMESYNC_EVENT_INTROUTER0_in_5	TIMESYNC_EVENT_INTROUTER0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_cpts_pend_0	GICSS0_spi_121	GICSS0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_cpts_pend_0	R5FSS0_CORE0_intr_53	R5FSS0_CORE0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_cpts_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_53	MCU_R5FSS0_CORE0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_cpts_pend_0	C7X256V0_CLEC_gic_spi_121	C7X256V0_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_cpts_pend_0	C7X256V1_CLEC_gic_spi_121	C7X256V1_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_cpts_sync_0	TIMESYNC_EVENT_INTROUTER0_in_6	TIMESYNC_EVENT_INTROUTER0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_downstream_pulse_0	GICSS0_spi_162	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_downstream_pulse_0	R5FSS0_CORE0_intr_99	R5FSS0_CORE0	PCIE0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-298. Hardware Requests for PCIe (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PCIE0	PCIE0_pcie_downstream_pulse_0	WKUP_R5FSS0_CORE0_intr_98	WKUP_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_downstream_pulse_0	MCU_R5FSS0_CORE0_cpu0_intr_99	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_downstream_pulse_0	C7X256V0_CLEC_gic_spi_162	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_downstream_pulse_0	C7X256V1_CLEC_gic_spi_162	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	GICSS0_spi_163	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	R5FSS0_CORE0_intr_126	R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	WKUP_R5FSS0_CORE0_intr_99	WKUP_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	MCU_R5FSS0_CORE0_cpu0_intr_126	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	C7X256V0_CLEC_gic_spi_163	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_dpa_pulse_0	C7X256V1_CLEC_gic_spi_163	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ecc0_corr_level_0	ESM0_esm_lvl_event_180	ESM0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_ecc0_uncorr_level_0	ESM0_esm_lvl_event_181	ESM0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_ecc1_uncorr_level_0	ESM0_esm_lvl_event_182	ESM0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_error_pulse_0	GICSS0_spi_125	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_error_pulse_0	C7X256V0_CLEC_gic_spi_125	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_error_pulse_0	C7X256V1_CLEC_gic_spi_125	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_flr_pulse_0	GICSS0_spi_126	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_flr_pulse_0	C7X256V0_CLEC_gic_spi_126	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_flr_pulse_0	C7X256V1_CLEC_gic_spi_126	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	GICSS0_spi_123	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	R5FSS0_CORE0_intr_55	R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	WKUP_R5FSS0_CORE0_intr_55	WKUP_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	MCU_R5FSS0_CORE0_cpu0_intr_55	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	C7X256V0_CLEC_gic_spi_123	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_hot_reset_pulse_0	C7X256V1_CLEC_gic_spi_123	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_legacy_pulse_0	GICSS0_spi_127	GICSS0	PCIE0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-298. Hardware Requests for PCIe (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PCIE0	PCIE0_pcie_legacy_pulse_0	C7X256V0_CLEC_gic_spi_127	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_legacy_pulse_0	C7X256V1_CLEC_gic_spi_127	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_link_state_pulse_0	GICSS0_spi_131	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_link_state_pulse_0	C7X256V0_CLEC_gic_spi_131	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_link_state_pulse_0	C7X256V1_CLEC_gic_spi_131	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_local_level_0	GICSS0_spi_137	GICSS0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_local_level_0	C7X256V0_CLEC_gic_spi_137	C7X256V0_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_local_level_0	C7X256V1_CLEC_gic_spi_137	C7X256V1_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_phy_level_0	GICSS0_spi_142	GICSS0	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_phy_level_0	C7X256V0_CLEC_gic_spi_142	C7X256V0_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_phy_level_0	C7X256V1_CLEC_gic_spi_142	C7X256V1_CLEC	PCIE0 interrupt request	level
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	TIMESYNC_EVENT_INTROUTER0_in_7	TIMESYNC_EVENT_INTROUTER0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	GICSS0_spi_124	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	R5FSS0_CORE0_intr_165	R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	MCU_R5FSS0_CORE0_cpu0_intr_165	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	C7X256V0_CLEC_gic_spi_124	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_ptm_valid_pulse_0	C7X256V1_CLEC_gic_spi_124	C7X256V1_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	GICSS0_spi_122	GICSS0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	R5FSS0_CORE0_intr_54	R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	WKUP_R5FSS0_CORE0_intr_54	WKUP_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	MCU_R5FSS0_CORE0_cpu0_intr_54	MCU_R5FSS0_CORE0	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	C7X256V0_CLEC_gic_spi_122	C7X256V0_CLEC	PCIE0 interrupt request	pulse
PCIE0	PCIE0_pcie_pwr_state_pulse_0	C7X256V1_CLEC_gic_spi_122	C7X256V1_CLEC	PCIE0 interrupt request	pulse

4.10.2 Gigabit Ethernet Switch (CPSW)

This section contains the integration details for the CPSW module on this device. For further information, see the *Gigabit Ethernet Switch (CPSW)* section of the *Peripherals* chapter.

Copyright © 2026 Texas Instruments Incorporated

4.10.2.1 CPSW Unsupported Features

The following features are not supported on this family of devices:

- Maximum frame size of 9600 bytes
- GMII Mode
- MACSEC
- Synchronous Ethernet

4.10.2.2 cpsw Integration

Table 4-299. CPSW Module Allocations

Instance	MAIN	MCU	WKUP
CPSW0	✓		

Table 4-300. Integration Attributes for cpsw

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
CPSW0	PSC0	GP_CORE	LPSC_main_cpsw	42	OFF	YES	LPSC_main_serdes1 LPSC_main_serdes0 LPSC_main_ip

Table 4-301. Clock Integration for cpsw

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CPSW0	CPPI_ICLK	MAIN_SYSCLK0/2		cppi streaming packet interface clock
CPSW0	CPTS_RFT_CLK	MAIN_PLL2_HSDIV5_CLKOUT	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=0	cpts reference clock
CPSW0	CPTS_RFT_CLK	MAIN_PLL0_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=1	cpts reference clock
CPSW0	CPTS_RFT_CLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=2	cpts reference clock
CPSW0	CPTS_RFT_CLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=4	cpts reference clock
CPSW0	CPTS_RFT_CLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=5	cpts reference clock
CPSW0	CPTS_RFT_CLK	SERDES_10G1 (INSTANCE)	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=6	cpts reference clock
CPSW0	CPTS_RFT_CLK	MAIN_SYSCLK0	MAIN_CTRL_MMR_CFG0_CPSW_CLKSEL[2:0]=7	cpts reference clock
CPSW0	GMII1_MR_CLK	MAIN_PLL2_HSDIV1_CLKOUT/10		gmii input receive reference clock
CPSW0	GMII1_MT_CLK	MAIN_PLL2_HSDIV1_CLKOUT/10		gmii input transmit reference clock
CPSW0	GMII2_MR_CLK	MAIN_PLL2_HSDIV1_CLKOUT/10		gmii input receive reference clock
CPSW0	GMII2_MT_CLK	MAIN_PLL2_HSDIV1_CLKOUT/10		gmii input transmit reference clock
CPSW0	GMII_RFT_CLK	MAIN_PLL2_HSDIV1_CLKOUT/2		gmii input 125-mhz reference clock
CPSW0	RGMII_MHZ_250_CLK	MAIN_PLL2_HSDIV1_CLKOUT		rgmii 250-mhz reference clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-301. Clock Integration for cpsw (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CPSW0	RGMII_MHZ_50_CLK	MAIN_PLL2_HSDIV1_CLKOUT/5		rgmii 50-mhz reference clock
CPSW0	RGMII_MHZ_5_CLK	MAIN_PLL2_HSDIV1_CLKOUT/50		rgmii 5-mhz reference clock
CPSW0	RMII1_MHZ_50_CLK	RMII1_REF_CLK (PIN)		rmii 50-mhz reference clock
CPSW0	RMII2_MHZ_50_CLK	RMII2_REF_CLK (PIN)		rmii 50-mhz reference clock

Table 4-302. Reset Integration for cpsw

Module Instance	Source	Description
CPSW0	PSC0	CPSW0 reset

Table 4-303. Hardware Requests for cpsw

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CPSW0	CPSW0_cpts_comp_0	EPWM0_epwm_synci n_0	EPWM0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_COMPout_CP_GEMAC_CPTS0_TS_COMP_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_COMPout	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	DMASS0_INTAGGR_0_intaggr_levi_pend_0	DMASS0_INTAGGR_0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	GICSS0_spi_48	GICSS0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	R5FSS0_CORE0_intr_48	R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	WKUP_R5FSS0_CORE0_intr_48	WKUP_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	MCU_R5FSS0_CORE0_cpu0_intr_48	MCU_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	C7X256V0_CLEC_gic_spi_48	C7X256V0_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_comp_0	C7X256V1_CLEC_gic_spi_48	C7X256V1_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_genf0_0	TIMESYNC_EVENT_INTROUTER0_in_16	TIMESYNC_EVENT_INTROUTER0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_genf1_0	TIMESYNC_EVENT_INTROUTER0_in_17	TIMESYNC_EVENT_INTROUTER0	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_sync_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_SYNCout_CP_GEMAC_CPTS0_TS_SYNC_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_SYNCout	CPSW0 interrupt request	level
CPSW0	CPSW0_cpts_sync_0	TIMESYNC_EVENT_INTROUTER0_in_18	TIMESYNC_EVENT_INTROUTER0	CPSW0 interrupt request	level
CPSW0	CPSW0_ecc_ded_pend_0	ESM0_esm_lvl_event_67	ESM0	CPSW0 interrupt request	level
CPSW0	CPSW0_ecc_sec_pend_0	ESM0_esm_lvl_event_3	ESM0	CPSW0 interrupt request	level
CPSW0	CPSW0_evnt_pend_0	GICSS0_spi_134	GICSS0	CPSW0 interrupt request	level
CPSW0	CPSW0_evnt_pend_0	R5FSS0_CORE0_intr_134	R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_evnt_pend_0	WKUP_R5FSS0_CORE0_intr_134	WKUP_R5FSS0_CORE0	CPSW0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-303. Hardware Requests for cpsw (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CPSW0	CPSW0_evnt_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_134	MCU_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_evnt_pend_0	C7X256V0_CLEC_gic_spi_134	C7X256V0_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_evnt_pend_0	C7X256V1_CLEC_gic_spi_134	C7X256V1_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	GICSS0_spi_135	GICSS0	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	R5FSS0_CORE0_intr_135	R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	WKUP_R5FSS0_CORE0_intr_135	WKUP_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_135	MCU_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	C7X256V0_CLEC_gic_spi_135	C7X256V0_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_mdio_pend_0	C7X256V1_CLEC_gic_spi_135	C7X256V1_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	GICSS0_spi_136	GICSS0	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	R5FSS0_CORE0_intr_136	R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	WKUP_R5FSS0_CORE0_intr_136	WKUP_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_136	MCU_R5FSS0_CORE0	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	C7X256V0_CLEC_gic_spi_136	C7X256V0_CLEC	CPSW0 interrupt request	level
CPSW0	CPSW0_stat_pend_0	C7X256V1_CLEC_gic_spi_136	C7X256V1_CLEC	CPSW0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.10.3 Universal Serial Bus Subsystem (USB)

This section contains the integration details for the USB module on this device. For further information, see the *Universal Serial Bus 2.0 Subsystem (USB2SS)* and *Universal Serial Bus 3.0 Subsystem (USB3SS)* sections of the *Peripherals* chapter.

4.10.3.1 USB2SS Unsupported Features

The following features are not supported on this family of devices:

- OTG: no Session Request Protocol (SRP) support
- ID pin signaling from USB2 PHY

4.10.3.2 USB3SS Unsupported Features

The following features are not supported on this family of devices:

- Virtualization (SR-IOV, xHCI, or Arm SMMU)
- Direct Debug USB (DDUSB) interface for trace
- Attach Detection Protocol (ADP)
- 5-Volt compatible VBUS pin
- Force event command TRB operating in device mode
- Negotiate Bandwidth command TRB when operating in device mode
- Get Port Bandwidth command TRB when operating in device mode
- Bandwidth request event TRB when operating in device mode
- Doorbell event TRB when operating in device mode
- Device Notification event TRB when operating in device mode
- NRDY event TRB when operating in host mode
- Halt Endpoint command TRB when operating in host mode
- OTG protocol for USB Type-C applications
- Light HC Reset Capability (LHRC)
- xHCI debug capability (Debug Port)
- xHCI save and restore operations
- HIMD when operating in host mode
- Latency tolerance messaging
- Negotiate bandwidth command
- Number of slots limited to 64
- Number of active periodic endpoints limited to 96 when operating in host mode
- Smart ISO
- Secondary streams
- PTM when operating in device mode

4.10.3.3 USB Integration

Table 4-304. USB Module Allocations

Instance	MAIN	MCU	WKUP
USB0	✓		
USB1	✓		

Table 4-305. Clock Integration for USB

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
USB0	BUS_CLK	MAIN_SYSCLK0/2		functional and vbusp interface clock
USB0	USB2_APB_PCLK	MAIN_SYSCLK0/4		None
USB0	USB2_REFCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_USB0_CLKSEL[0:0]=0	usb reference clock
USB0	USB2_REFCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_USB0_CLKSEL[0:0]=0	usb reference clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-305. Clock Integration for USB (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
USB0	USB2_REFCLK	MAIN_PLL0_HSDIV8_CLKOUT	WKUP_CTRL_MMR_CFG0_USB0_CLKSEL[0:0]=1	usb reference clock
USB1	ACLK_CLK	MAIN_SYSCLK0/2		None
USB1	CLK_LPM_CLK	MAIN_PLL1_HSDIV6_CLKOUT/4		None
USB1	PCLK_CLK	MAIN_SYSCLK0/4		None
USB1	PIPE_REFCLK	SERDES_10G0 (INSTANCE)		None
USB1	PIPE_RXCLK	SERDES_10G0 (INSTANCE)		None
USB1	PIPE_RXFCLK	SERDES_10G0 (INSTANCE)		None
USB1	PIPE_TXFCLK	SERDES_10G0 (INSTANCE)		None
USB1	PIPE_TXMCLK	SERDES_10G0 (INSTANCE)		None
USB1	USB2_APB_PCLK_CLK	MAIN_SYSCLK0/4		None
USB1	USB2_REFCLOCK_CLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_USB1_CLKSEL[0:0]=0	None
USB1	USB2_REFCLOCK_CLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_USB1_CLKSEL[0:0]=0	None
USB1	USB2_REFCLOCK_CLK	MAIN_PLL0_HSDIV8_CLKOUT	WKUP_CTRL_MMR_CFG0_USB1_CLKSEL[0:0]=1	None

Table 4-306. Hardware Requests for USB

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB0	USB0_a_ecc_aggr_corrected_err_level_0	ESM0_esm_lvl_event_35	ESM0	USB0 interrupt request	level
USB0	USB0_a_ecc_aggr_uncorrected_err_level_0	ESM0_esm_lvl_event_111	ESM0	USB0 interrupt request	level
USB0	USB0_host_system_error_0	ESM0_esm_lvl_event_32	ESM0	USB0 interrupt request	level
USB0	USB0_irq_0	GICSS0_spi_220	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_0	R5FSS0_CORE0_intr_220	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_0	WKUP_R5FSS0_CORE0_intr_220	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_220	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_0	C7X256V0_CLEC_gic_spi_220	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_0	C7X256V1_CLEC_gic_spi_220	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_1	GICSS0_spi_221	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_1	R5FSS0_CORE0_intr_221	R5FSS0_CORE0	USB0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-306. Hardware Requests for USB (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB0	USB0_irq_1	WKUP_R5FSS0_CORE0_intr_221	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_1	MCU_R5FSS0_CORE0_cpu0_intr_221	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_1	C7X256V0_CLEC_gic_spi_221	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_1	C7X256V1_CLEC_gic_spi_221	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_2	GICSS0_spi_222	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_2	R5FSS0_CORE0_intr_222	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_2	WKUP_R5FSS0_CORE0_intr_222	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_2	MCU_R5FSS0_CORE0_cpu0_intr_222	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_2	C7X256V0_CLEC_gic_spi_222	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_2	C7X256V1_CLEC_gic_spi_222	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_3	GICSS0_spi_223	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_3	R5FSS0_CORE0_intr_223	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_3	WKUP_R5FSS0_CORE0_intr_223	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_3	MCU_R5FSS0_CORE0_cpu0_intr_223	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_3	C7X256V0_CLEC_gic_spi_223	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_3	C7X256V1_CLEC_gic_spi_223	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_4	GICSS0_spi_224	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_4	R5FSS0_CORE0_intr_224	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_4	WKUP_R5FSS0_CORE0_intr_224	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_4	MCU_R5FSS0_CORE0_cpu0_intr_224	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_4	C7X256V0_CLEC_gic_spi_224	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_4	C7X256V1_CLEC_gic_spi_224	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_5	GICSS0_spi_225	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_5	R5FSS0_CORE0_intr_225	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_5	WKUP_R5FSS0_CORE0_intr_225	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_5	MCU_R5FSS0_CORE0_cpu0_intr_225	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_5	C7X256V0_CLEC_gic_spi_225	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_5	C7X256V1_CLEC_gic_spi_225	C7X256V1_CLEC	USB0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-306. Hardware Requests for USB (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB0	USB0_irq_6	GICSS0_spi_226	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_6	R5FSS0_CORE0_intr_226	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_6	WKUP_R5FSS0_CORE0_intr_226	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_6	MCU_R5FSS0_CORE0_cpu0_intr_226	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_6	C7X256V0_CLEC_gic_spi_226	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_6	C7X256V1_CLEC_gic_spi_226	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_irq_7	GICSS0_spi_227	GICSS0	USB0 interrupt request	level
USB0	USB0_irq_7	R5FSS0_CORE0_intr_227	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_7	WKUP_R5FSS0_CORE0_intr_227	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_7	MCU_R5FSS0_CORE0_cpu0_intr_227	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_irq_7	C7X256V0_CLEC_gic_spi_227	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_irq_7	C7X256V1_CLEC_gic_spi_227	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_misc_level_0	GICSS0_spi_228	GICSS0	USB0 interrupt request	level
USB0	USB0_misc_level_0	R5FSS0_CORE0_intr_228	R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_misc_level_0	WKUP_R5FSS0_CORE0_intr_228	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_misc_level_0	MCU_R5FSS0_CORE0_cpu0_intr_228	MCU_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_misc_level_0	C7X256V0_CLEC_gic_spi_228	C7X256V0_CLEC	USB0 interrupt request	level
USB0	USB0_misc_level_0	C7X256V1_CLEC_gic_spi_228	C7X256V1_CLEC	USB0 interrupt request	level
USB0	USB0_usb_wakeup_clkstop_wakeup_0	WKUP_R5FSS0_CORE0_intr_61	WKUP_R5FSS0_CORE0	USB0 interrupt request	level
USB0	USB0_usb_wakeup_clkstop_wakeup_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_9	WKUP_DEEPSLEEP_SOURCES0	USB0 interrupt request	level
USB1	USB1_a_ecc_aggr_corrected_err_level_0	ESM0_esm_lvl_event_146	ESM0	USB1 interrupt request	level
USB1	USB1_a_ecc_aggr_uncorrected_err_level_0	ESM0_esm_lvl_event_147	ESM0	USB1 interrupt request	level
USB1	USB1_asf_int_fatal_0	ESM0_esm_lvl_event_143	ESM0	USB1 interrupt request	level
USB1	USB1_asf_int_nonfatal_0	ESM0_esm_lvl_event_155	ESM0	USB1 interrupt request	level
USB1	USB1_host_system_error_0	ESM0_esm_lvl_event_33	ESM0	USB1 interrupt request	level
USB1	USB1_host_system_error_0	GICSS0_spi_266	GICSS0	USB1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-306. Hardware Requests for USB (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB1	USB1_host_system_error_0	R5FSS0_CORE0_intr_238	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_host_system_error_0	WKUP_R5FSS0_CORE0_intr_238	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_host_system_error_0	MCU_R5FSS0_CORE0_cpu0_intr_238	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_host_system_error_0	C7X256V0_CLEC_gic_spi_266	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_host_system_error_0	C7X256V1_CLEC_gic_spi_266	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_0	GICSS0_spi_258	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_0	R5FSS0_CORE0_intr_230	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_0	WKUP_R5FSS0_CORE0_intr_230	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_230	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_0	C7X256V0_CLEC_gic_spi_258	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_0	C7X256V1_CLEC_gic_spi_258	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_1	GICSS0_spi_259	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_1	R5FSS0_CORE0_intr_231	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_1	WKUP_R5FSS0_CORE0_intr_231	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_1	MCU_R5FSS0_CORE0_cpu0_intr_231	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_1	C7X256V0_CLEC_gic_spi_259	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_1	C7X256V1_CLEC_gic_spi_259	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_2	GICSS0_spi_260	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_2	R5FSS0_CORE0_intr_232	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_2	WKUP_R5FSS0_CORE0_intr_232	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_2	MCU_R5FSS0_CORE0_cpu0_intr_232	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_2	C7X256V0_CLEC_gic_spi_260	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_2	C7X256V1_CLEC_gic_spi_260	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_3	GICSS0_spi_261	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_3	R5FSS0_CORE0_intr_233	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_3	WKUP_R5FSS0_CORE0_intr_233	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_3	MCU_R5FSS0_CORE0_cpu0_intr_233	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_3	C7X256V0_CLEC_gic_spi_261	C7X256V0_CLEC	USB1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-306. Hardware Requests for USB (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB1	USB1_irq_3	C7X256V1_CLEC_gic_spi_261	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_4	GICSS0_spi_262	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_4	R5FSS0_CORE0_intr_234	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_4	WKUP_R5FSS0_CORE0_intr_234	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_4	MCU_R5FSS0_CORE0_cpu0_intr_234	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_4	C7X256V0_CLEC_gic_spi_262	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_4	C7X256V1_CLEC_gic_spi_262	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_5	GICSS0_spi_263	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_5	R5FSS0_CORE0_intr_235	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_5	WKUP_R5FSS0_CORE0_intr_235	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_5	MCU_R5FSS0_CORE0_cpu0_intr_235	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_5	C7X256V0_CLEC_gic_spi_263	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_5	C7X256V1_CLEC_gic_spi_263	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_6	GICSS0_spi_264	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_6	R5FSS0_CORE0_intr_236	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_6	WKUP_R5FSS0_CORE0_intr_236	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_6	MCU_R5FSS0_CORE0_cpu0_intr_236	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_6	C7X256V0_CLEC_gic_spi_264	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_6	C7X256V1_CLEC_gic_spi_264	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_irq_7	GICSS0_spi_265	GICSS0	USB1 interrupt request	level
USB1	USB1_irq_7	R5FSS0_CORE0_intr_237	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_7	WKUP_R5FSS0_CORE0_intr_237	WKUP_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_7	MCU_R5FSS0_CORE0_cpu0_intr_237	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_irq_7	C7X256V0_CLEC_gic_spi_265	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_irq_7	C7X256V1_CLEC_gic_spi_265	C7X256V1_CLEC	USB1 interrupt request	level
USB1	USB1_otgirq_0	GICSS0_spi_277	GICSS0	USB1 interrupt request	level
USB1	USB1_otgirq_0	R5FSS0_CORE0_intr_52	R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_otgirq_0	WKUP_R5FSS0_CORE0_intr_52	WKUP_R5FSS0_CORE0	USB1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-306. Hardware Requests for USB (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
USB1	USB1_otgirq_0	MCU_R5FSS0_CORE0_cpu0_intr_52	MCU_R5FSS0_CORE0	USB1 interrupt request	level
USB1	USB1_otgirq_0	C7X256V0_CLEC_gic_spi_277	C7X256V0_CLEC	USB1 interrupt request	level
USB1	USB1_otgirq_0	C7X256V1_CLEC_gic_spi_277	C7X256V1_CLEC	USB1 interrupt request	level

4.10.4 Serializer/Deserializer (SerDes)

This section contains the integration details for the SerDes module on this device. For further information, see the *Serializer/Deserializer (SerDes)* section of the *Peripherals* chapter.

4.10.4.1 SerDes Unsupported Features

The following features are not supported on this family of devices:

- SATA over SerDes
- Raw SerDes
- Display Port
- JESD204B

4.10.4.2 SERDES_10G Integration

Table 4-307. SERDES_10G Module Allocations

Instance	MAIN	MCU	WKUP
SERDES_10G0	✓		
SERDES_10G1	✓		

Table 4-308. Clock Integration for SERDES_10G

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
SERDES_10G0	CLK	MAIN_SYSCLK0/4		None
SERDES_10G0	CORE_REF_CLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_SERDES0_CLKSEL[1:0]=0	None
SERDES_10G0	CORE_REF_CLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_SERDES0_CLKSEL[1:0]=1	None
SERDES_10G0	CORE_REF_CLK	MAIN_PLL2_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_SERDES0_CLKSEL[1:0]=2	None
SERDES_10G0	CORE_REF_CLK	MAIN_PLL0_HSDIV9_CLKOUT	MAIN_CTRL_MMR_CFG0_SERDES0_CLKSEL[1:0]=3	None
SERDES_10G0	IP1_LN0_TXCLK	USB1 (INSTANCE)		None
SERDES_10G1	CLK	MAIN_SYSCLK0/4		None
SERDES_10G1	CORE_REF_CLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_SERDES1_CLKSEL[1:0]=0	None
SERDES_10G1	CORE_REF_CLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_SERDES1_CLKSEL[1:0]=1	None
SERDES_10G1	CORE_REF_CLK	MAIN_PLL2_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_SERDES1_CLKSEL[1:0]=2	None

Copyright © 2026 Texas Instruments Incorporated

Table 4-308. Clock Integration for SERDES_10G (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
SERDES_10G1	CORE_REF_CLK	MAIN_PLL0_HSDIV9_CLKOUT	MAIN_CTRL_MMR_CFG0_SERDES1_CLKSEL[1:0]=3	None
SERDES_10G1	IP1_LN0_TXCLK	PCIE0 (INSTANCE)		None

Table 4-309. Hardware Requests for SERDES_10G

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	GICSS0_spi_167	GICSS0	SERDES_10G0 interrupt request	level
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	R5FSS0_CORE0_intr_166	R5FSS0_CORE0	SERDES_10G0 interrupt request	level
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	WKUP_R5FSS0_CORE0_intr_166	WKUP_R5FSS0_CORE0	SERDES_10G0 interrupt request	level
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_166	MCU_R5FSS0_CORE0	SERDES_10G0 interrupt request	level
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	C7X256V0_CLEC_gic_spi_167	C7X256V0_CLEC	SERDES_10G0 interrupt request	level
SERDES_10G0	SERDES_10G0_phy_pwr_timeout_lvl_0	C7X256V1_CLEC_gic_spi_167	C7X256V1_CLEC	SERDES_10G0 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	GICSS0_spi_119	GICSS0	SERDES_10G1 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	R5FSS0_CORE0_intr_154	R5FSS0_CORE0	SERDES_10G1 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	WKUP_R5FSS0_CORE0_intr_53	WKUP_R5FSS0_CORE0	SERDES_10G1 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_154	MCU_R5FSS0_CORE0	SERDES_10G1 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	C7X256V0_CLEC_gic_spi_119	C7X256V0_CLEC	SERDES_10G1 interrupt request	level
SERDES_10G1	SERDES_10G1_phy_pwr_timeout_lvl_0	C7X256V1_CLEC_gic_spi_119	C7X256V1_CLEC	SERDES_10G1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11 Memory Interfaces

4.11.1 Flash Subsystem (FSS)

This section contains the integration details for the Flash Subsystem on this device. For further information, see the *Flash Subsystem (FSS)* section of the *Peripherals* chapter.

4.11.1.1 FSS Unsupported Features

For more information, see Section 4.11.2.1, *OSPI Unsupported Features*.

4.11.1.2 FSS Integration

Table 4-310. FSS Module Allocations

Instance	MAIN	MCU	WKUP
FSS0	✓		

4.11.1.3 FSS_FSAS_0 Integration

Table 4-311. FSS_FSAS_0 Module Allocations

Instance	MAIN	MCU	WKUP
FSS0_FSAS_0	✓		

Table 4-312. Clock Integration for FSS_FSAS_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
FSS0_FSAS_0	FICLK	MAIN_SYSCLK0		functional and interface clock

Table 4-313. Hardware Requests for FSS_FSAS_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
FSS0_FSAS_0	FSS0_FSAS_0_ecc_intr_err_pend_0	ESM0_esm_lvl_event_141	ESM0	FSS0_FSAS_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11.2 Octal Serial Peripheral Interface (OSPI)

This section contains the integration details for the OSPI module on this device. For further information, see the *Octal Serial Peripheral Interface (OSPI)* section of the *Peripherals* chapter.

4.11.2.1 OSPI Unsupported Features

The following features are not supported on this family of devices:

- OSPI PDMA. Use CPU-triggered block DMA
- OSPI Clock Phase/Polarity modes 1,2,3
- Reset pins OSPI0_RESET_OUT2, OSPI0_RESET_OUT3

4.11.2.2 FSS_OSPI_0 Integration

Table 4-314. FSS_OSPI_0 Module Allocations

Instance	MAIN	MCU	WKUP
FSS0_OSPI_0	✓		

Table 4-315. Clock Integration for FSS_OSPI_0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
FSS0_OSPI_0	HCLK	MAIN_SYSCLK0		data transfer clock
FSS0_OSPI_0	PCLK	MAIN_SYSCLK0		configuration clock
FSS0_OSPI_0	RFCLK	MAIN_PLL0_HSDIV1_CLKOUT	MAIN_CTRL_MMR_CFG0_OSPI0_CLKSEL[0:0]=0	functional reference clock
FSS0_OSPI_0	RFCLK	MAIN_PLL1_HSDIV5_CLKOUT	MAIN_CTRL_MMR_CFG0_OSPI0_CLKSEL[0:0]=1	functional reference clock

Table 4-316. Hardware Requests for FSS_OSPI_0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
FSS0_OSPI_0	FSS0_OSPI_0_ospi_ecc_corr_lvl_intr_0	ESM0_esm_lvl_event_11	ESM0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_ecc_uncorr_lvl_intr_0	ESM0_esm_lvl_event_74	ESM0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	GICSS0_spi_171	GICSS0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	R5FSS0_CORE0_intr_171	R5FSS0_CORE0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	WKUP_R5FSS0_CORE0_intr_171	WKUP_R5FSS0_CORE0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_171	MCU_R5FSS0_CORE0	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	C7X256V0_CLEC_gic_spi_171	C7X256V0_CLEC	FSS0_OSPI_0 interrupt request	level
FSS0_OSPI_0	FSS0_OSPI_0_ospi_lvl_intr_0	C7X256V1_CLEC_gic_spi_171	C7X256V1_CLEC	FSS0_OSPI_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11.3 General-Purpose Memory Controller (GPMC)

This section contains the integration details for the GPMC module on this device. For further information, see the *General-Purpose Memory Controller (GPMC)* section of the *Peripherals* chapter.

4.11.3.1 GPMC Unsupported Features

The following features are not supported on this family of devices:

- Pins GPMC_CS[7:4]n are not pinned out
- Pins GPMC_A[27:23] are not pinned out
- Pins GPMC_AD[31:16] are not pinned out
- Pins GPMC_WAIT[3:2] are not pinned out
- Pins GPMC_BE[3:2]n are not pinned out

4.11.3.2 GPMC Integration

Table 4-317. GPMC Module Allocations

Instance	MAIN	MCU	WKUP
GPMC0	✓		

Table 4-318. Integration Attributes for GPMC

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
GPMC0	PSC0	GP_CORE	LPSC_main_gpmc	15	OFF	YES	LPSC_main_ip

Table 4-319. Clock Integration for GPMC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
GPMC0	FICLK	GPMC_FCLK		functional and interface clock
GPMC0	ICLK	MAIN_SYSCLK0/2		None

Table 4-320. Reset Integration for GPMC

Module Instance	Source	Description
GPMC0	PSC0	GPMC0 reset

Table 4-321. Hardware Requests for GPMC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
GPMC0	GPMC0_gpmc_sdmareq_0	DMASS0_INTAGGR_0_intaggr_levi_pend_26	DMASS0_INTAGGR_0	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	GICSS0_spi_138	GICSS0	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	R5FSS0_CORE0_intr_103	R5FSS0_CORE0	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	WKUP_R5FSS0_CORE0_intr_103	WKUP_R5FSS0_CORE0	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	MCU_R5FSS0_CORE0_cpu0_intr_103	MCU_R5FSS0_CORE0	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	C7X256V0_CLEC_gic_spi_138	C7X256V0_CLEC	GPMC0 interrupt request	level
GPMC0	GPMC0_gpmc_sinterrupt_0	C7X256V1_CLEC_gic_spi_138	C7X256V1_CLEC	GPMC0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11.4 Error Location Module (ELM)

This section contains the integration details for the ELM module on this device. For further information, see the *Error Location Module (ELM)* section of the *Peripherals* chapter.

4.11.4.1 ELM Integration

Table 4-322. ELM Module Allocations

Instance	MAIN	MCU	WKUP
ELM0	✓		

Table 4-323. Integration Attributes for ELM

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
ELM0	PSC0	GP_CORE	LPSC_main_gpmc	15	OFF	YES	LPSC_main_ip

Table 4-324. Clock Integration for ELM

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
ELM0	FICLK	MAIN_SYSCLK0/4		functional and interface clock

Table 4-325. Reset Integration for ELM

Module Instance	Source	Description
ELM0	PSC0	ELM0 reset

Table 4-326. Hardware Requests for ELM

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	GICSS0_spi_164	GICSS0	ELM0 interrupt request	level
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	R5FSS0_CORE0_intr_164	R5FSS0_CORE0	ELM0 interrupt request	level
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	WKUP_R5FSS0_CORE0_intr_164	WKUP_R5FSS0_CORE0	ELM0 interrupt request	level
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_164	MCU_R5FSS0_CORE0	ELM0 interrupt request	level
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	C7X256V0_CLEC_gic_spi_164	C7X256V0_CLEC	ELM0 interrupt request	level
ELM0	ELM0_elm_porocpsinterrupt_lvl_0	C7X256V1_CLEC_gic_spi_164	C7X256V1_CLEC	ELM0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11.5 Multimedia Card Secure Digital (MMCSD)

This section contains the integration details for the MMCSD module on this device. For further information, see the *Multimedia Card Secure Digital (MMCSD)* section of the *Peripherals* chapter.

4.11.5.1 MMCSD Unsupported Features

The following features are not supported on this family of devices:

- The following apply to 4-bit MMCSD instances:
  - SD Card busy LED: Not pinned out
- The following apply to 8-bit MMCSD instances:
  - SD Card busy LED - Not pinned out
  - SD/SDIO/MMC Cards (Removable) - no support for 4-pin/8-pin muxing
  - SD Card Detect Pin - Not pinned out
  - SD Card Write Protect Pin - Not pinned out
  - Enhanced Strobe
  - IO retention
  - Muxing of LVCMOS functions with the PHY at the SoC level

4.11.5.2 MMCSD4 Integration

Table 4-327. MMCSD4 Module Allocations

Instance	MAIN	MCU	WKUP
MMCSD1	✓		
MMCSD2	✓		

Table 4-328. Integration Attributes for MMCSD4

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MMCSD1	PSC0	GP_CORE	LPSC_main_emmc4b0	21	OFF	YES	LPSC_main_ip
MMCSD2	PSC0	GP_CORE	LPSC_main_emmc4b1	22	OFF	YES	LPSC_main_ip

Table 4-329. Clock Integration for MMCSD4

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MMCSD1	ICLK	MAIN_SYSCLK0/2		interface clock
MMCSD1	FCLK	MAIN_PLL0_HSDIV5_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC1_CLKSEL[0:0]=0	functional clock
MMCSD1	FCLK	MAIN_PLL2_HSDIV2_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC1_CLKSEL[0:0]=1	functional clock
MMCSD2	ICLK	MAIN_SYSCLK0/2		interface clock
MMCSD2	FCLK	MAIN_PLL0_HSDIV5_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC2_CLKSEL[0:0]=0	functional clock
MMCSD2	FCLK	MAIN_PLL2_HSDIV2_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC2_CLKSEL[0:0]=1	functional clock

Table 4-330. Reset Integration for MMCSD4

Module Instance	Source	Description
MMCSD1	PSC0	MMCSD1 reset
MMCSD2	PSC0	MMCSD2 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-331. Hardware Requests for MMCSD4

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MMCSD1	MMCSD1_emmcsdss_intr_0	GICSS0_spi_115	GICSS0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_intr_0	R5FSS0_CORE0_intr_162	R5FSS0_CORE0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_intr_0	WKUP_R5FSS0_CORE0_intr_162	WKUP_R5FSS0_CORE0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_162	MCU_R5FSS0_CORE0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_intr_0	C7X256V0_CLEC_gic_spi_115	C7X256V0_CLEC	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_intr_0	C7X256V1_CLEC_gic_spi_115	C7X256V1_CLEC	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_rxmem_corr_err_lvl_0	ESM0_esm_lvl_event_58	ESM0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_rxmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_59	ESM0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_txmem_corr_err_lvl_0	ESM0_esm_lvl_event_60	ESM0	MMCSD1 interrupt request	level
MMCSD1	MMCSD1_emmcsdss_txmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_61	ESM0	MMCSD1 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	GICSS0_spi_114	GICSS0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	R5FSS0_CORE0_intr_163	R5FSS0_CORE0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	WKUP_R5FSS0_CORE0_intr_163	WKUP_R5FSS0_CORE0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_163	MCU_R5FSS0_CORE0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	C7X256V0_CLEC_gic_spi_114	C7X256V0_CLEC	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_intr_0	C7X256V1_CLEC_gic_spi_114	C7X256V1_CLEC	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_rxmem_corr_err_lvl_0	ESM0_esm_lvl_event_34	ESM0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_rxmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_36	ESM0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_txmem_corr_err_lvl_0	ESM0_esm_lvl_event_49	ESM0	MMCSD2 interrupt request	level
MMCSD2	MMCSD2_emmcsdss_txmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_65	ESM0	MMCSD2 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.11.5.3 MMCSD8 Integration

Table 4-332. MMCSD8 Module Allocations

Instance	MAIN	MCU	WKUP
MMCSD0	✓		

Table 4-333. Integration Attributes for MMCSD8

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MMCSD0	PSC0	GP_CORE	LPSC_main_emmc8b	20	OFF	YES	LPSC_main_ip

Table 4-334. Clock Integration for MMCSD8

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MMCSD0	ICLK	MAIN_SYSCLK0/2		interface clock
MMCSD0	FCLK	MAIN_PLL0_HSDIV5_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC0_CLKSEL[0:0]=0	functional clock
MMCSD0	FCLK	MAIN_PLL2_HSDIV2_CLKOUT	MAIN_CTRL_MMR_CFG0_EMMC0_CLKSEL[0:0]=1	functional clock

Table 4-335. Reset Integration for MMCSD8

Module Instance	Source	Description
MMCSD0	PSC0	MMCSD0 reset

Table 4-336. Hardware Requests for MMCSD8

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MMCSD0	MMCSD0_emmcss_intr_0	GICSS0_spi_165	GICSS0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_intr_0	R5FSS0_CORE0_intr_161	R5FSS0_CORE0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_intr_0	WKUP_R5FSS0_CORE0_intr_161	WKUP_R5FSS0_CORE0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_161	MCU_R5FSS0_CORE0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_intr_0	C7X256V0_CLEC_gic_spi_165	C7X256V0_CLEC	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_intr_0	C7X256V1_CLEC_gic_spi_165	C7X256V1_CLEC	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_rxmem_corr_err_lvl_0	ESM0_esm_lvl_event_54	ESM0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_rxmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_55	ESM0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_txmem_corr_err_lvl_0	ESM0_esm_lvl_event_56	ESM0	MMCSD0 interrupt request	level
MMCSD0	MMCSD0_emmcss_txmem_uncorr_err_lvl_0	ESM0_esm_lvl_event_57	ESM0	MMCSD0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.12 Industrial and Control Interfaces

4.12.1 Modular Controller Area Network (MCAN)

This section contains the integration details for the MCAN module on this device. For further information, see the *Modular Controller Area Network (MCAN)* section of the *Peripherals* chapter.

4.12.1.1 MCAN Unsupported Features

The following features are not supported on this family of devices:

- Debug DMAs: DMA Ack is not supported by PDMA architecture. Debug messages can be traced through the RX FIFO.
- TX DMA channels 3-31 are not supported by the Main domain instance/s. Only TX_DMA[2:0] have associated PDMA channels.
- DMA is not supported by the MCU domain instance/s

4.12.1.2 MCAN Integration

Table 4-337. MCAN Module Allocations

Instance	MAIN	MCU	WKUP
MCAN0	✓		
MCAN1	✓		
MCU_MCAN0		✓	
MCU_MCAN1		✓	

Table 4-338. Integration Attributes for MCAN

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCAN0	PSC0	GP_CORE	LPSC_main_mcanss0	35	OFF	YES	LPSC_main_ip
MCAN1	PSC0	GP_CORE	LPSC_main_mcanss1	16	OFF	YES	LPSC_main_ip
MCU_MCAN0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_mcanss_0	7	OFF	YES	LPSC_MCU_common
MCU_MCAN1	WKUP_PSC0	PD_MCUSS	LPSC_MCU_mcanss_1	8	OFF	YES	LPSC_MCU_common

Table 4-339. Clock Integration for MCAN

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCAN0	FCLK	MAIN_PLL0_HSDIV4_CLKOUT	MAIN_CTRL_MMR_CFG0_MCAN0_CLKSEL[1:0]=0	functional clock
MCAN0	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_MCAN0_CLKSEL[1:0]=1	functional clock
MCAN0	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCAN0_CLKSEL[1:0]=2	functional clock
MCAN0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCAN0_CLKSEL[1:0]=3	functional clock
MCAN0	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCAN0_CLKSEL[1:0]=3	functional clock
MCAN0	ICLK	MAIN_SYSCLK0/4		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-339. Clock Integration for MCAN (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCAN1	FCLK	MAIN_PLL0_HSDIV4_CLKOUT	MAIN_CTRL_MMR_CFG0_MCAN1_CLKSEL[1:0]=0	functional clock
MCAN1	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_MCAN1_CLKSEL[1:0]=1	functional clock
MCAN1	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_MCAN1_CLKSEL[1:0]=2	functional clock
MCAN1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_MCAN1_CLKSEL[1:0]=3	functional clock
MCAN1	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_MCAN1_CLKSEL[1:0]=3	functional clock
MCAN1	ICLK	MAIN_SYSCLK0/4		interface clock
MCU_MCAN0	FCLK	MCU_PLL0_HSDIV4_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=0	functional clock
MCU_MCAN0	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=1	functional clock
MCU_MCAN0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=2	functional clock
MCU_MCAN0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=2	functional clock
MCU_MCAN0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=3	functional clock
MCU_MCAN0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_MCAN0_CLKSEL[1:0]=3	functional clock
MCU_MCAN0	ICLK	MCU_SYSCLK0/2		interface clock
MCU_MCAN1	FCLK	MCU_PLL0_HSDIV4_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=0	functional clock
MCU_MCAN1	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=1	functional clock
MCU_MCAN1	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=2	functional clock
MCU_MCAN1	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=2	functional clock
MCU_MCAN1	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=3	functional clock
MCU_MCAN1	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_MCAN1_CLKSEL[1:0]=3	functional clock
MCU_MCAN1	ICLK	MCU_SYSCLK0/2		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-340. Reset Integration for MCAN

Module Instance	Source	Description
MCAN0	PSC0	MCAN0 reset
MCAN1	PSC0	MCAN1 reset
MCU_MCAN0	WKUP_PSC0	MCU_MCAN0 reset
MCU_MCAN1	WKUP_PSC0	MCU_MCAN1 reset

Table 4-341. Hardware Requests for MCAN

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCAN0	MCAN0_mcanss_ecc_corr_lvl_int_0	ESM0_esm_lvl_event_16	ESM0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ecc_uncorr_lvl_int_0	ESM0_esm_lvl_event_78	ESM0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	GICSS0_spi_186	GICSS0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	R5FSS0_CORE0_intr_186	R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	WKUP_R5FSS0_CORE0_intr_186	WKUP_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_186	MCU_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	C7X256V0_CLEC_gic_spi_186	C7X256V0_CLEC	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_ext_ts_rollover_lvl_int_0	C7X256V1_CLEC_gic_spi_186	C7X256V1_CLEC	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_fe_0	PDMA0_mcanss_main_0_fe_0	PDMA0	MCAN0 interrupt request	pulse
MCAN0	MCAN0_mcanss_fe_1	PDMA0_mcanss_main_0_fe_1	PDMA0	MCAN0 interrupt request	pulse
MCAN0	MCAN0_mcanss_fe_2	PDMA0_mcanss_main_0_fe_2	PDMA0	MCAN0 interrupt request	pulse
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	GICSS0_spi_187	GICSS0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	R5FSS0_CORE0_intr_187	R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	WKUP_R5FSS0_CORE0_intr_187	WKUP_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_187	MCU_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	C7X256V0_CLEC_gic_spi_187	C7X256V0_CLEC	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_0	C7X256V1_CLEC_gic_spi_187	C7X256V1_CLEC	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	GICSS0_spi_188	GICSS0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	R5FSS0_CORE0_intr_188	R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	WKUP_R5FSS0_CORE0_intr_188	WKUP_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	MCU_R5FSS0_CORE0_cpu0_intr_188	MCU_R5FSS0_CORE0	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	C7X256V0_CLEC_gic_spi_188	C7X256V0_CLEC	MCAN0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-341. Hardware Requests for MCAN (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCAN0	MCAN0_mcanss_mcan_lvl_int_1	C7X256V1_CLEC_gic_spi_188	C7X256V1_CLEC	MCAN0 interrupt request	level
MCAN0	MCAN0_mcanss_tx_dma_0	PDMA0_mcanss_main_0_tx_0	PDMA0	MCAN0 interrupt request	pulse
MCAN0	MCAN0_mcanss_tx_dma_1	PDMA0_mcanss_main_0_tx_1	PDMA0	MCAN0 interrupt request	pulse
MCAN0	MCAN0_mcanss_tx_dma_2	PDMA0_mcanss_main_0_tx_2	PDMA0	MCAN0 interrupt request	pulse
MCAN1	MCAN1_mcanss_ecc_corr_lvl_int_0	ESM0_esm_lvl_event_68	ESM0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ecc_uncorr_lvl_int_0	ESM0_esm_lvl_event_69	ESM0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	GICSS0_spi_199	GICSS0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	R5FSS0_CORE0_intr_229	R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	WKUP_R5FSS0_CORE0_intr_229	WKUP_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_229	MCU_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	C7X256V0_CLEC_gic_spi_199	C7X256V0_CLEC	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_ext_ts_rollover_lvl_int_0	C7X256V1_CLEC_gic_spi_199	C7X256V1_CLEC	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_fe_0	PDMA0_mcanss_main_1_fe_0	PDMA0	MCAN1 interrupt request	pulse
MCAN1	MCAN1_mcanss_fe_1	PDMA0_mcanss_main_1_fe_1	PDMA0	MCAN1 interrupt request	pulse
MCAN1	MCAN1_mcanss_fe_2	PDMA0_mcanss_main_1_fe_2	PDMA0	MCAN1 interrupt request	pulse
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	GICSS0_spi_245	GICSS0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	R5FSS0_CORE0_intr_110	R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	WKUP_R5FSS0_CORE0_intr_110	WKUP_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_110	MCU_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	C7X256V0_CLEC_gic_spi_245	C7X256V0_CLEC	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_0	C7X256V1_CLEC_gic_spi_245	C7X256V1_CLEC	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	GICSS0_spi_246	GICSS0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	R5FSS0_CORE0_intr_63	R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	WKUP_R5FSS0_CORE0_intr_63	WKUP_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	MCU_R5FSS0_CORE0_cpu0_intr_63	MCU_R5FSS0_CORE0	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	C7X256V0_CLEC_gic_spi_246	C7X256V0_CLEC	MCAN1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-341. Hardware Requests for MCAN (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCAN1	MCAN1_mcanss_mcan_lvl_int_1	C7X256V1_CLEC_gic_spi_246	C7X256V1_CLEC	MCAN1 interrupt request	level
MCAN1	MCAN1_mcanss_tx_dma_0	PDMA0_mcanss_main_1_tx_0	PDMA0	MCAN1 interrupt request	pulse
MCAN1	MCAN1_mcanss_tx_dma_1	PDMA0_mcanss_main_1_tx_1	PDMA0	MCAN1 interrupt request	pulse
MCAN1	MCAN1_mcanss_tx_dma_2	PDMA0_mcanss_main_1_tx_2	PDMA0	MCAN1 interrupt request	pulse
MCU_MCAN0	MCU_MCAN0_mcanss_ecc_corr_lvl_int_0	WKUP_ESM0_esm_lvl_event_16	WKUP_ESM0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_ecc_uncorr_lvl_int_0	WKUP_ESM0_esm_lvl_event_17	WKUP_ESM0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_ext_ts_rollover_lvl_int_0	R5FSS0_CORE0_intr_42	R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_ext_ts_rollover_lvl_int_0	WKUP_R5FSS0_CORE0_intr_198	WKUP_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_ext_ts_rollover_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_42	MCU_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	GICSS0_spi_58	GICSS0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	R5FSS0_CORE0_intr_43	R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	WKUP_R5FSS0_CORE0_intr_199	WKUP_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_43	MCU_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	C7X256V0_CLEC_gic_spi_58	C7X256V0_CLEC	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_0	C7X256V1_CLEC_gic_spi_58	C7X256V1_CLEC	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	GICSS0_spi_59	GICSS0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	R5FSS0_CORE0_intr_44	R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	WKUP_R5FSS0_CORE0_intr_200	WKUP_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	MCU_R5FSS0_CORE0_cpu0_intr_44	MCU_R5FSS0_CORE0	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	C7X256V0_CLEC_gic_spi_59	C7X256V0_CLEC	MCU_MCAN0 interrupt request	level
MCU_MCAN0	MCU_MCAN0_mcanss_mcan_lvl_int_1	C7X256V1_CLEC_gic_spi_59	C7X256V1_CLEC	MCU_MCAN0 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_ecc_corr_lvl_int_0	WKUP_ESM0_esm_lvl_event_18	WKUP_ESM0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_ecc_uncorr_lvl_int_0	WKUP_ESM0_esm_lvl_event_19	WKUP_ESM0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_ext_ts_rollover_lvl_int_0	R5FSS0_CORE0_intr_45	R5FSS0_CORE0	MCU_MCAN1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-341. Hardware Requests for MCAN (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_MCAN1	MCU_MCAN1_mcanss_ext_ts_rollover_lvl_int_0	WKUP_R5FSS0_CORE0_intr_239	WKUP_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_ext_ts_rollover_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_45	MCU_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	GICSS0_spi_60	GICSS0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	R5FSS0_CORE0_intr_46	R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	WKUP_R5FSS0_CORE0_intr_247	WKUP_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	MCU_R5FSS0_CORE0_cpu0_intr_46	MCU_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	C7X256V0_CLEC_gic_spi_60	C7X256V0_CLEC	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_0	C7X256V1_CLEC_gic_spi_60	C7X256V1_CLEC	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	GICSS0_spi_61	GICSS0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	R5FSS0_CORE0_intr_47	R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	WKUP_R5FSS0_CORE0_intr_248	WKUP_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	MCU_R5FSS0_CORE0_cpu0_intr_47	MCU_R5FSS0_CORE0	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	C7X256V0_CLEC_gic_spi_61	C7X256V0_CLEC	MCU_MCAN1 interrupt request	level
MCU_MCAN1	MCU_MCAN1_mcanss_mcan_lvl_int_1	C7X256V1_CLEC_gic_spi_61	C7X256V1_CLEC	MCU_MCAN1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.12.2 Enhanced Capture (ECAP)

This section contains the integration details for the ECAP module on this device. For further information, see the *Enhanced Capture (ECAP)* section of the *Peripherals* chapter.

4.12.2.1 ECAP Integration

Table 4-342. ECAP Module Allocations

Instance	MAIN	MCU	WKUP
ECAP0	✓		
ECAP1	✓		
ECAP2	✓		

Table 4-343. Integration Attributes for ECAP

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
ECAP0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
ECAP1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
ECAP2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso

Table 4-344. Clock Integration for ECAP

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
ECAP0	FICLK	MAIN_SYSCLK0/4		functional and interface clock
ECAP1	FICLK	MAIN_SYSCLK0/4		functional and interface clock
ECAP2	FICLK	MAIN_SYSCLK0/4		functional and interface clock

Table 4-345. Reset Integration for ECAP

Module Instance	Source	Description
ECAP0	PSC0	ECAP0 reset
ECAP1	PSC0	ECAP1 reset
ECAP2	PSC0	ECAP2 reset

Table 4-346. Hardware Requests for ECAP

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ECAP0	ECAP0_ecap_int_0	GICSS0_spi_145	GICSS0	ECAP0 interrupt request	pulse
ECAP0	ECAP0_ecap_int_0	R5FSS0_CORE0_intr_83	R5FSS0_CORE0	ECAP0 interrupt request	pulse
ECAP0	ECAP0_ecap_int_0	MCU_R5FSS0_CORE0_cpu0_intr_83	MCU_R5FSS0_CORE0	ECAP0 interrupt request	pulse
ECAP0	ECAP0_ecap_int_0	C7X256V0_CLEC_gic_spi_145	C7X256V0_CLEC	ECAP0 interrupt request	pulse
ECAP0	ECAP0_ecap_int_0	C7X256V1_CLEC_gic_spi_145	C7X256V1_CLEC	ECAP0 interrupt request	pulse
ECAP1	ECAP1_ecap_int_0	GICSS0_spi_146	GICSS0	ECAP1 interrupt request	pulse
ECAP1	ECAP1_ecap_int_0	R5FSS0_CORE0_intr_84	R5FSS0_CORE0	ECAP1 interrupt request	pulse
ECAP1	ECAP1_ecap_int_0	MCU_R5FSS0_CORE0_cpu0_intr_84	MCU_R5FSS0_CORE0	ECAP1 interrupt request	pulse
ECAP1	ECAP1_ecap_int_0	C7X256V0_CLEC_gic_spi_146	C7X256V0_CLEC	ECAP1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-346. Hardware Requests for ECAP (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ECAP1	ECAP1_ecap_int_0	C7X256V1_CLEC_gic_spi_146	C7X256V1_CLEC	ECAP1 interrupt request	pulse
ECAP2	ECAP2_ecap_int_0	GICSS0_spi_147	GICSS0	ECAP2 interrupt request	pulse
ECAP2	ECAP2_ecap_int_0	R5FSS0_CORE0_intr_85	R5FSS0_CORE0	ECAP2 interrupt request	pulse
ECAP2	ECAP2_ecap_int_0	MCU_R5FSS0_CORE0_cpu0_intr_85	MCU_R5FSS0_CORE0	ECAP2 interrupt request	pulse
ECAP2	ECAP2_ecap_int_0	C7X256V0_CLEC_gic_spi_147	C7X256V0_CLEC	ECAP2 interrupt request	pulse
ECAP2	ECAP2_ecap_int_0	C7X256V1_CLEC_gic_spi_147	C7X256V1_CLEC	ECAP2 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.12.3 Enhanced Pulse Width Modulation (EPWM)

This section contains the integration details for the EPWM module on this device. For further information, see the *Enhanced Pulse Width Modulation (EPWM)* section of the *Peripherals* chapter.

4.12.3.1 EPWM Unsupported Features

The following features are not supported on this family of devices:

- EPWM digital comparators
- High resolution extension (delay-line based)

4.12.3.2 EPWM Integration

Table 4-347. EPWM Module Allocations

Instance	MAIN	MCU	WKUP
EPWM0	✓		
EPWM1	✓		
EPWM2	✓		

Table 4-348. Integration Attributes for EPWM

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
EPWM0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
EPWM1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
EPWM2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso

Table 4-349. Clock Integration for EPWM

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
EPWM0	FICLK	MAIN_SYSCLK0/2		functional and interface clock
EPWM1	FICLK	MAIN_SYSCLK0/2		functional and interface clock
EPWM2	FICLK	MAIN_SYSCLK0/2		functional and interface clock

Table 4-350. Reset Integration for EPWM

Module Instance	Source	Description
EPWM0	PSC0	EPWM0 reset
EPWM1	PSC0	EPWM1 reset
EPWM2	PSC0	EPWM2 reset

Table 4-351. Hardware Requests for EPWM

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EPWM0	EPWM0_epwm_etint_0	GICSS0_spi_229	GICSS0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_etint_0	R5FSS0_CORE0_intr_36	R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_etint_0	WKUP_R5FSS0_CORE0_intr_123	WKUP_R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_etint_0	MCU_R5FSS0_CORE0_cpu0_intr_36	MCU_R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_etint_0	C7X256V0_CLEC_gic_spi_229	C7X256V0_CLEC	EPWM0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-351. Hardware Requests for EPWM (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EPWM0	EPWM0_epwm_etint_0	C7X256V1_CLEC_gic_spi_229	C7X256V1_CLEC	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_synco_o_0	TIMESYNC_EVENT_INTROUTER0_in_8	TIMESYNC_EVENT_INTROUTER0	EPWM0 interrupt request	level
EPWM0	EPWM0_epwm_syncout_0	EPWM1_epwm_syncin_0	EPWM1	EPWM0 interrupt request	level
EPWM0	EPWM0_epwm_tripzint_0	GICSS0_spi_230	GICSS0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_tripzint_0	R5FSS0_CORE0_intr_80	R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_tripzint_0	WKUP_R5FSS0_CORE0_intr_124	WKUP_R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_tripzint_0	MCU_R5FSS0_CORE0_cpu0_intr_80	MCU_R5FSS0_CORE0	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_tripzint_0	C7X256V0_CLEC_gic_spi_230	C7X256V0_CLEC	EPWM0 interrupt request	pulse
EPWM0	EPWM0_epwm_tripzint_0	C7X256V1_CLEC_gic_spi_230	C7X256V1_CLEC	EPWM0 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	GICSS0_spi_231	GICSS0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	R5FSS0_CORE0_intr_37	R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	WKUP_R5FSS0_CORE0_intr_125	WKUP_R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	MCU_R5FSS0_CORE0_cpu0_intr_37	MCU_R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	C7X256V0_CLEC_gic_spi_231	C7X256V0_CLEC	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_etint_0	C7X256V1_CLEC_gic_spi_231	C7X256V1_CLEC	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_syncout_0	EPWM2_epwm_syncin_0	EPWM2	EPWM1 interrupt request	level
EPWM1	EPWM1_epwm_tripzint_0	GICSS0_spi_233	GICSS0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_tripzint_0	R5FSS0_CORE0_intr_81	R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_tripzint_0	WKUP_R5FSS0_CORE0_intr_126	WKUP_R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_tripzint_0	MCU_R5FSS0_CORE0_cpu0_intr_81	MCU_R5FSS0_CORE0	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_tripzint_0	C7X256V0_CLEC_gic_spi_233	C7X256V0_CLEC	EPWM1 interrupt request	pulse
EPWM1	EPWM1_epwm_tripzint_0	C7X256V1_CLEC_gic_spi_233	C7X256V1_CLEC	EPWM1 interrupt request	pulse
EPWM2	EPWM2_epwm_etint_0	GICSS0_spi_234	GICSS0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_etint_0	R5FSS0_CORE0_intr_38	R5FSS0_CORE0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_etint_0	WKUP_R5FSS0_CORE0_intr_127	WKUP_R5FSS0_CORE0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_etint_0	MCU_R5FSS0_CORE0_cpu0_intr_38	MCU_R5FSS0_CORE0	EPWM2 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-351. Hardware Requests for EPWM (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EPWM2	EPWM2_epwm_etint_0	C7X256V0_CLEC_gic_spi_234	C7X256V0_CLEC	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_etint_0	C7X256V1_CLEC_gic_spi_234	C7X256V1_CLEC	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	GICSS0_spi_235	GICSS0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	R5FSS0_CORE0_intr_82	R5FSS0_CORE0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	WKUP_R5FSS0_CORE0_intr_148	WKUP_R5FSS0_CORE0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	MCU_R5FSS0_CORE0_cpu0_intr_82	MCU_R5FSS0_CORE0	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	C7X256V0_CLEC_gic_spi_235	C7X256V0_CLEC	EPWM2 interrupt request	pulse
EPWM2	EPWM2_epwm_tripzint_0	C7X256V1_CLEC_gic_spi_235	C7X256V1_CLEC	EPWM2 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.12.4 Enhanced Quadrature Encoder Pulse (EQEP)

This section contains the integration details for the EQEP module on this device. For further information, see the *Enhanced Quadrature Encoder Pulse (EQEP)* section of the *Peripherals* chapter.

4.12.4.1 EQEP Unsupported Features

The following features are not supported on this family of devices:

- EQEPA_i[15:1], EQEPB_i[15:1] are not pinned out.

4.12.4.2 EQEP Integration

Table 4-352. EQEP Module Allocations

Instance	MAIN	MCU	WKUP
EQEP0	✓		
EQEP1	✓		
EQEP2	✓		

Table 4-353. Integration Attributes for EQEP

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
EQEP0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
EQEP1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
EQEP2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso

Table 4-354. Clock Integration for EQEP

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
EQEP0	FICLK	MAIN_SYSCLK0/4		functional and interface clock
EQEP1	FICLK	MAIN_SYSCLK0/4		functional and interface clock
EQEP2	FICLK	MAIN_SYSCLK0/4		functional and interface clock

Table 4-355. Reset Integration for EQEP

Module Instance	Source	Description
EQEP0	PSC0	EQEP0 reset
EQEP1	PSC0	EQEP1 reset
EQEP2	PSC0	EQEP2 reset

Table 4-356. Hardware Requests for EQEP

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EQEP0	EQEP0_eqep_int_0	GICSS0_spi_148	GICSS0	EQEP0 interrupt request	pulse
EQEP0	EQEP0_eqep_int_0	R5FSS0_CORE0_intr_86	R5FSS0_CORE0	EQEP0 interrupt request	pulse
EQEP0	EQEP0_eqep_int_0	WKUP_R5FSS0_CORE0_intr_244	WKUP_R5FSS0_CORE0	EQEP0 interrupt request	pulse
EQEP0	EQEP0_eqep_int_0	MCU_R5FSS0_CORE0_cpu0_intr_86	MCU_R5FSS0_CORE0	EQEP0 interrupt request	pulse
EQEP0	EQEP0_eqep_int_0	C7X256V0_CLEC_gic_spi_148	C7X256V0_CLEC	EQEP0 interrupt request	pulse
EQEP0	EQEP0_eqep_int_0	C7X256V1_CLEC_gic_spi_148	C7X256V1_CLEC	EQEP0 interrupt request	pulse
EQEP1	EQEP1_eqep_int_0	GICSS0_spi_149	GICSS0	EQEP1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-356. Hardware Requests for EQEP (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
EQEP1	EQEP1_eqep_int_0	R5FSS0_CORE0_intr_87	R5FSS0_CORE0	EQEP1 interrupt request	pulse
EQEP1	EQEP1_eqep_int_0	WKUP_R5FSS0_CORE0_intr_245	WKUP_R5FSS0_CORE0	EQEP1 interrupt request	pulse
EQEP1	EQEP1_eqep_int_0	MCU_R5FSS0_CORE0_cpu0_intr_87	MCU_R5FSS0_CORE0	EQEP1 interrupt request	pulse
EQEP1	EQEP1_eqep_int_0	C7X256V0_CLEC_gic_spi_149	C7X256V0_CLEC	EQEP1 interrupt request	pulse
EQEP1	EQEP1_eqep_int_0	C7X256V1_CLEC_gic_spi_149	C7X256V1_CLEC	EQEP1 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	GICSS0_spi_150	GICSS0	EQEP2 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	R5FSS0_CORE0_intr_88	R5FSS0_CORE0	EQEP2 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	WKUP_R5FSS0_CORE0_intr_246	WKUP_R5FSS0_CORE0	EQEP2 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	MCU_R5FSS0_CORE0_cpu0_intr_88	MCU_R5FSS0_CORE0	EQEP2 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	C7X256V0_CLEC_gic_spi_150	C7X256V0_CLEC	EQEP2 interrupt request	pulse
EQEP2	EQEP2_eqep_int_0	C7X256V1_CLEC_gic_spi_150	C7X256V1_CLEC	EQEP2 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.13 Camera Subsystem

4.13.1 Camera Serial Interface Receiver (CSI_RX_IF)

This section contains the integration details for the CSI_RX_IF module on this device. For further information, see the *Camera Serial Interface Receiver (CSI_RX_IF)* section of the *Peripherals* chapter.

4.13.1.1 CSI_RX_IF Unsupported Features

The following features are not supported on this family of devices:

- Line Count Error interrupt (Streams 1-3)
- Frame Mismatch Error interrupt (Streams 1-3)
- Frame Count Error interrupt (Streams 1-3)
- FCC Stop interrupt (Streams 1-3)
- FCC Start interrupt (Streams 1-3)
- Line/Byte interrupt (Streams 1-3)
- Timer / Timer interrupt (Streams 1-3)

4.13.1.2 csi_rx_if Integration

Table 4-357. CSI_RX_IF Module Allocations

Instance	MAIN	MCU	WKUP
csi_rx_if0	✓		
csi_rx_if1	✓		
csi_rx_if2	✓		
csi_rx_if3	✓		

Table 4-358. Integration Attributes for csi_rx_if

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
csi_rx_if0	PSC0	GP_CORE	LPSC_main_csi_rx0	25	OFF	YES	LPSC_main_dphy_rx0
csi_rx_if1	PSC0	GP_CORE	LPSC_main_csi_rx1	9	OFF	YES	LPSC_main_dphy_rx1
csi_rx_if2	PSC0	GP_CORE	LPSC_main_csi_rx2	48	OFF	YES	LPSC_main_csi_dphy_rx2
csi_rx_if3	PSC0	GP_CORE	LPSC_main_csi_rx3	44	OFF	YES	LPSC_main_dphy_rx3

Table 4-359. Clock Integration for csi_rx_if

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
csi_rx_if0	CSI_RX_MAIN_CLK	MAIN_SYSCLK0		main_clk clock runs the csi2 core and psil dma.
csi_rx_if0	CSI_RX_BYTE_CLK	DPHY_RX0 (INSTANCE)		byte_clk is the clock supplied by the dphy_rx pll and is divided down to byte clock. the dphy_rx is designed for max of 10gbps. this translates to a max byte clock of 312.5mhz. the clock is inactive when dphy_rx is not in hs operation
csi_rx_if0	CSI_RX_VBUS_CLK	MAIN_SYSCLK0/2		interface configuration clock that runs at half the speed of the main_clk
csi_rx_if0	CSI_RX_VP_CLK	MAIN_PLL5_HSDIV0_CLKOUT		video port clock
csi_rx_if1	CSI_RX_MAIN_CLK	MAIN_SYSCLK0		main_clk clock runs the csi2 core and psil dma.

Copyright © 2026 Texas Instruments Incorporated

Table 4-359. Clock Integration for csi_rx_if (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
csi_rx_if1	CSI_RX_BYTE_CLK	DPHY_RX1 (INSTANCE)		byte_clk is the clock supplied by the dphy_rx pll and is divided down to byte clock. the dphy_rx is designed for max of 10gbps. this translates to a max byte clock of 312.5mhz. the clock is inactive when dphy_rx is not in hs operation
csi_rx_if1	CSI_RX_VBUS_CLK	MAIN_SYSCLK0/2		interface configuration clock that runs at half the speed of the main_clk
csi_rx_if1	CSI_RX_VP_CLK	MAIN_PLL5_HSDIV0_CLKOUT		video port clock
csi_rx_if2	CSI_RX_MAIN_CLK	MAIN_SYSCLK0		main_clk clock runs the csi2 core and psil dma.
csi_rx_if2	CSI_RX_BYTE_CLK	DPHY_RX2 (INSTANCE)		byte_clk is the clock supplied by the dphy_rx pll and is divided down to byte clock. the dphy_rx is designed for max of 10gbps. this translates to a max byte clock of 312.5mhz. the clock is inactive when dphy_rx is not in hs operation
csi_rx_if2	CSI_RX_VBUS_CLK	MAIN_SYSCLK0/2		interface configuration clock that runs at half the speed of the main_clk
csi_rx_if2	CSI_RX_VP_CLK	MAIN_PLL5_HSDIV0_CLKOUT		video port clock
csi_rx_if3	CSI_RX_MAIN_CLK	MAIN_SYSCLK0		main_clk clock runs the csi2 core and psil dma.
csi_rx_if3	CSI_RX_BYTE_CLK	DPHY_RX3 (INSTANCE)		byte_clk is the clock supplied by the dphy_rx pll and is divided down to byte clock. the dphy_rx is designed for max of 10gbps. this translates to a max byte clock of 312.5mhz. the clock is inactive when dphy_rx is not in hs operation
csi_rx_if3	CSI_RX_VBUS_CLK	MAIN_SYSCLK0/2		interface configuration clock that runs at half the speed of the main_clk
csi_rx_if3	CSI_RX_VP_CLK	MAIN_PLL5_HSDIV0_CLKOUT		video port clock

Table 4-360. Reset Integration for csi_rx_if

Module Instance	Source	Description
csi_rx_if0	PSC0	csi_rx_if0 reset
csi_rx_if1	PSC0	csi_rx_if1 reset
csi_rx_if2	PSC0	csi_rx_if2 reset
csi_rx_if3	PSC0	csi_rx_if3 reset

Table 4-361. Hardware Requests for csi_rx_if

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
csi_rx_if0	csi_rx_if0_corr_level_0	ESM0_esm_lvl_event_66	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	ESM0_esm_lvl_event_0	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	GICSS0_spi_175	GICSS0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	R5FSS0_CORE0_intr_170	R5FSS0_CORE0	csi_rx_if0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-361. Hardware Requests for csi_rx_if (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
csi_rx_if0	csi_rx_if0_csi_err_irq_0	WKUP_R5FSS0_CORE0_intr_170	WKUP_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_170	MCU_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	C7X256V0_CLEC_gic_spi_175	C7X256V0_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_err_irq_0	C7X256V1_CLEC_gic_spi_175	C7X256V1_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_fatal_0	ESM0_esm_lvl_event_70	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	GICSS0_spi_173	GICSS0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	R5FSS0_CORE0_intr_173	R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	WKUP_R5FSS0_CORE0_intr_173	WKUP_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_173	MCU_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	C7X256V0_CLEC_gic_spi_173	C7X256V0_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_irq_0	C7X256V1_CLEC_gic_spi_173	C7X256V1_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	ESM0_esm_lvl_event_72	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	GICSS0_spi_174	GICSS0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	R5FSS0_CORE0_intr_174	R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	WKUP_R5FSS0_CORE0_intr_174	WKUP_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	MCU_R5FSS0_CORE0_cpu0_intr_174	MCU_R5FSS0_CORE0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	C7X256V0_CLEC_gic_spi_174	C7X256V0_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_level_0	C7X256V1_CLEC_gic_spi_174	C7X256V1_CLEC	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_csi_nonfatal_0	ESM0_esm_lvl_event_71	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if0	csi_rx_if0_uncorr_level_0	ESM0_esm_lvl_event_77	ESM0	csi_rx_if0 interrupt request	level
csi_rx_if1	csi_rx_if1_corr_level_0	ESM0_esm_lvl_event_191	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	ESM0_esm_lvl_event_190	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	GICSS0_spi_178	GICSS0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	R5FSS0_CORE0_intr_172	R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	WKUP_R5FSS0_CORE0_intr_172	WKUP_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_172	MCU_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_err_irq_0	C7X256V0_CLEC_gic_spi_178	C7X256V0_CLEC	csi_rx_if1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-361. Hardware Requests for csi_rx_if (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
csi_rx_if1	csi_rx_if1_csi_err_irq_0	C7X256V1_CLEC_gic_spi_178	C7X256V1_CLEC	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_fatal_0	ESM0_esm_lvl_event_189	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	GICSS0_spi_179	GICSS0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	R5FSS0_CORE0_intr_176	R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	WKUP_R5FSS0_CORE0_intr_176	WKUP_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_176	MCU_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	C7X256V0_CLEC_gic_spi_179	C7X256V0_CLEC	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_irq_0	C7X256V1_CLEC_gic_spi_179	C7X256V1_CLEC	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	ESM0_esm_lvl_event_188	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	GICSS0_spi_200	GICSS0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	R5FSS0_CORE0_intr_209	R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	WKUP_R5FSS0_CORE0_intr_209	WKUP_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	MCU_R5FSS0_CORE0_cpu0_intr_209	MCU_R5FSS0_CORE0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	C7X256V0_CLEC_gic_spi_200	C7X256V0_CLEC	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_level_0	C7X256V1_CLEC_gic_spi_200	C7X256V1_CLEC	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_csi_nonfatal_0	ESM0_esm_lvl_event_187	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if1	csi_rx_if1_uncorr_level_0	ESM0_esm_lvl_event_122	ESM0	csi_rx_if1 interrupt request	level
csi_rx_if2	csi_rx_if2_corr_level_0	ESM0_esm_lvl_event_214	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	ESM0_esm_lvl_event_206	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	GICSS0_spi_219	GICSS0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	R5FSS0_CORE0_intr_198	R5FSS0_CORE0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_198	MCU_R5FSS0_CORE0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	C7X256V0_CLEC_gic_spi_219	C7X256V0_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_err_irq_0	C7X256V1_CLEC_gic_spi_219	C7X256V1_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_fatal_0	ESM0_esm_lvl_event_200	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_irq_0	GICSS0_spi_232	GICSS0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_irq_0	R5FSS0_CORE0_intr_199	R5FSS0_CORE0	csi_rx_if2 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-361. Hardware Requests for csi_rx_if (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
csi_rx_if2	csi_rx_if2_csi_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_199	MCU_R5FSS0_CORE0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_irq_0	C7X256V0_CLEC_gic_spi_232	C7X256V0_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_irq_0	C7X256V1_CLEC_gic_spi_232	C7X256V1_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	ESM0_esm_lvl_event_31	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	GICSS0_spi_236	GICSS0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	R5FSS0_CORE0_intr_200	R5FSS0_CORE0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	MCU_R5FSS0_CORE0_cpu0_intr_200	MCU_R5FSS0_CORE0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	C7X256V0_CLEC_gic_spi_236	C7X256V0_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_level_0	C7X256V1_CLEC_gic_spi_236	C7X256V1_CLEC	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_csi_nonfatal_0	ESM0_esm_lvl_event_201	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if2	csi_rx_if2_uncorr_level_0	ESM0_esm_lvl_event_215	ESM0	csi_rx_if2 interrupt request	level
csi_rx_if3	csi_rx_if3_corr_level_0	ESM0_esm_lvl_event_216	ESM0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	ESM0_esm_lvl_event_222	ESM0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	GICSS0_spi_249	GICSS0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	R5FSS0_CORE0_intr_155	R5FSS0_CORE0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_155	MCU_R5FSS0_CORE0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	C7X256V0_CLEC_gic_spi_249	C7X256V0_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_err_irq_0	C7X256V1_CLEC_gic_spi_249	C7X256V1_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_fatal_0	ESM0_esm_lvl_event_204	ESM0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_irq_0	GICSS0_spi_250	GICSS0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_irq_0	R5FSS0_CORE0_intr_156	R5FSS0_CORE0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_irq_0	MCU_R5FSS0_CORE0_cpu0_intr_156	MCU_R5FSS0_CORE0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_irq_0	C7X256V0_CLEC_gic_spi_250	C7X256V0_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_irq_0	C7X256V1_CLEC_gic_spi_250	C7X256V1_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_level_0	ESM0_esm_lvl_event_41	ESM0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_level_0	GICSS0_spi_251	GICSS0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_level_0	R5FSS0_CORE0_intr_157	R5FSS0_CORE0	csi_rx_if3 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-361. Hardware Requests for csi_rx_if (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
csi_rx_if3	csi_rx_if3_csi_level_0	MCU_R5FSS0_CORE0_cpu0_intr_157	MCU_R5FSS0_CORE0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_level_0	C7X256V0_CLEC_gic_spi_251	C7X256V0_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_level_0	C7X256V1_CLEC_gic_spi_251	C7X256V1_CLEC	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_csi_nonfatal_0	ESM0_esm_lvl_event_205	ESM0	csi_rx_if3 interrupt request	level
csi_rx_if3	csi_rx_if3_uncorr_level_0	ESM0_esm_lvl_event_217	ESM0	csi_rx_if3 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.13.2 MIPI D-PHY Receiver (DPHY_RX)

This section contains the integration details for the DPHY_RX module on this device. For further information, see the *MIPI D-PHY Receiver (DPHY_RX)* section of the *Peripherals* chapter.

4.13.2.1 DPHY_RX Unsupported Features

The following features are not supported on this family of devices:

- Swapping of clock and data lanes

4.13.2.2 dphy_rx Integration

Table 4-362. DPHY_RX Module Allocations

Instance	MAIN	MCU	WKUP
DPHY_RX0	✓		
DPHY_RX1	✓		
DPHY_RX2	✓		
DPHY_RX3	✓		

Table 4-363. Integration Attributes for dphy_rx

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
DPHY_RX0	PSC0	GP_CORE	LPSC_main_dphy_rx0	26	OFF	YES	LPSC_main_ip
DPHY_RX1	PSC0	GP_CORE	LPSC_main_dphy_rx1	10	OFF	YES	LPSC_main_ip
DPHY_RX2	PSC0	GP_CORE	LPSC_main_csi_dphy_rx2	49	OFF	YES	LPSC_main_ip
DPHY_RX3	PSC0	GP_CORE	LPSC_main_dphy_rx3	45	OFF	YES	LPSC_main_ip

Table 4-364. Clock Integration for dphy_rx

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DPHY_RX0	RXCLKN	CSI0_RXCLKN (PIN)		clock lane differential pair in
DPHY_RX0	RXCLKP	CSI0_RXCLKP (PIN)		clock lane differential pair in
DPHY_RX0	MAIN_CLK	MAIN_SYSCLK0/4		clock for lane module state machine and analog reference clock
DPHY_RX1	RXCLKN	CSI1_RXCLKN (PIN)		clock lane differential pair in
DPHY_RX1	RXCLKP	CSI1_RXCLKP (PIN)		clock lane differential pair in
DPHY_RX1	MAIN_CLK	MAIN_SYSCLK0/4		clock for lane module state machine and analog reference clock
DPHY_RX2	RXCLKN	CSI2_RXCLKN (PIN)		clock lane differential pair in
DPHY_RX2	RXCLKP	CSI2_RXCLKP (PIN)		clock lane differential pair in
DPHY_RX2	MAIN_CLK	MAIN_SYSCLK0/4		clock for lane module state machine and analog reference clock
DPHY_RX3	RXCLKN	CSI3_RXCLKN (PIN)		clock lane differential pair in
DPHY_RX3	RXCLKP	CSI3_RXCLKP (PIN)		clock lane differential pair in
DPHY_RX3	MAIN_CLK	MAIN_SYSCLK0/4		clock for lane module state machine and analog reference clock

Table 4-365. Reset Integration for dphy_rx

Module Instance	Source	Description
DPHY_RX0	PSC0	DPHY_RX0 reset
DPHY_RX1	PSC0	DPHY_RX1 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-365. Reset Integration for dphy_rx (continued)

Module Instance	Source	Description
DPHY_RX2	PSC0	DPHY_RX2 reset
DPHY_RX3	PSC0	DPHY_RX3 reset

Copyright © 2026 Texas Instruments Incorporated

4.13.3 MIPI D-PHY Transmitter (DPHY_TX)

This section contains the integration details for the DPHY_TX module on this device. For further information, see the *MIPI D-PHY Transmitter (DPHY_TX)* section of the *Peripherals* chapter.

4.13.3.1 DPHY_TX Integration

Table 4-366. DPHY_TX Module Allocations

Instance	MAIN	MCU	WKUP
DPHY_TX0	✓		

Table 4-367. Clock Integration for DPHY_TX

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DPHY_TX0	APB_PCLK	MAIN_SYSCLK0/4		None
DPHY_TX0	DPHY_REF_CLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_DPHY0_CLKSEL[0:0]=0	None
DPHY_TX0	DPHY_REF_CLK	MAIN_PLL0_HSDIV9_CLKOUT	MAIN_CTRL_MMR_CFG0_DPHY0_CLKSEL[0:0]=1	None
DPHY_TX0	IP1_PPI_M_TXCLKESC_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None
DPHY_TX0	IP2_PPI_M_TXCLKESC_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None
DPHY_TX0	IP3_PPI_M_TXCLKESC_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None
DPHY_TX0	IP4_PPI_M_TXCLKESC_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None
DPHY_TX0	PSM_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None

4.13.4 Camera Streaming Interface Transmitter (CSI_TX_IF)

This section contains the integration details for the CSI_TX_IF module on this device. For further information, see the *Camera Streaming Interface Transmitter (CSI_TX_IF)* section of the *Peripherals* chapter.

4.13.4.1 csi_tx_if Integration

Table 4-368. CSI_TX_IF Module Allocations

Instance	MAIN	MCU	WKUP
CSI_TX_IF0	✓		

Table 4-369. Integration Attributes for csi_tx_if

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
CSI_TX_IF0	PSC0	GP_CORE	LPSC_main_csi_tx0	11	OFF	YES	LPSC_main_dphy_tx0

Table 4-370. Clock Integration for csi_tx_if

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CSI_TX_IF0	DPHY_TXBYTECLKHS_CL_CLK	DPHY_TX0 (INSTANCE)		None
CSI_TX_IF0	ESC_CLK_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		None
CSI_TX_IF0	MAIN_CLK_CLK	MAIN_SYSCLK0		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-370. Clock Integration for csi_tx_if (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CSI_TX_IF0	VBUS_CLK_CLK	MAIN_SYSCLK0/2		None

Table 4-371. Reset Integration for csi_tx_if

Module Instance	Source	Description
CSI_TX_IF0	PSC0	CSI_TX_IF0 reset

Table 4-372. Hardware Requests for csi_tx_if

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
CSI_TX_IF0	CSI_TX_IF0_cdns_ram_corr_level_0	ESM0_esm_lvl_event_123	ESM0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_cdns_ram_uncorr_level_0	ESM0_esm_lvl_event_125	ESM0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_corr_level_0	ESM0_esm_lvl_event_126	ESM0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_fatal_0	ESM0_esm_lvl_event_186	ESM0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	GICSS0_spi_278	GICSS0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	R5FSS0_CORE0_intr_50	R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	WKUP_R5FSS0_CORE0_intr_50	WKUP_R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	MCU_R5FSS0_CORE0_cpu0_intr_50	MCU_R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	C7X256V0_CLEC_gic_spi_278	C7X256V0_CLEC	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_interrupt_0	C7X256V1_CLEC_gic_spi_278	C7X256V1_CLEC	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	GICSS0_spi_279	GICSS0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	R5FSS0_CORE0_intr_51	R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	WKUP_R5FSS0_CORE0_intr_51	WKUP_R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	MCU_R5FSS0_CORE0_cpu0_intr_51	MCU_R5FSS0_CORE0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	C7X256V0_CLEC_gic_spi_279	C7X256V0_CLEC	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_level_0	C7X256V1_CLEC_gic_spi_279	C7X256V1_CLEC	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_csi_nonfatal_0	ESM0_esm_lvl_event_185	ESM0	CSI_TX_IF0 interrupt request	level
CSI_TX_IF0	CSI_TX_IF0_uncorr_level_0	ESM0_esm_lvl_event_127	ESM0	CSI_TX_IF0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.14 Timer Modules

4.14.1 Global Timebase Counter (GTC)

This section contains the integration details for the GTC module on this device. For further information, see the *Global Timebase Counter (GTC)* section of the *Peripherals* chapter.

4.14.1.1 GTC Integration

Table 4-373. GTC Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_GTC0			✓

Table 4-374. Clock Integration for GTC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_GTC0	FCLK	MAIN_PLL2_HSDIV5_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=0	None
WKUP_GTC0	FCLK	MAIN_PLL0_HSDIV6_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=1	None
WKUP_GTC0	FCLK	CP_GEMAC_CPTS_REF_CLK	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=2	None
WKUP_GTC0	FCLK	MCU_EXT_REFCLK0	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=4	None
WKUP_GTC0	FCLK	EXT_REFCLK1	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=5	None
WKUP_GTC0	FCLK	MCU_SYSCLK0	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=6	None
WKUP_GTC0	FCLK	MAIN_SYSCLK0	WKUP_CTRL_MMR_CFG0_WKUP_GTC_CLKSEL[2:0]=7	None
WKUP_GTC0	ICLK	DM_CLK/4		None

Table 4-375. Hardware Requests for GTC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_GTC0	WKUP_GTC0_gtc_push_event_0	TIMESYNC_EVENT_INTROUTER0_in_11	TIMESYNC_EVENT_INTROUTER0	WKUP_GTC0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.14.2 Real Time Interrupt (RTI)

This section contains the integration details for the RTI module on this device. For further information, see the *Real Time Interrupt (RTI)* section of the *Peripherals* chapter.

4.14.2.1 RTI Not Supported Features

The following features are not supported on this family of devices:

- Analog watchdog timer
- External clock supervision
- Periodic interrupt
- Timestamp (Capture registers)

Note

The integration only supports one feature - use of the RTI as a Digital Windowed Watchdog. All other features (captures, compares, interrupts, network time unit) are not supported.

Note

The domain instances of the RTI module are intended to function as a Digital Windowed Watchdog for the CPU core that they are associated with (by instance name).

4.14.2.2 RTI Integration

Table 4-376. RTI Module Allocations

Instance	MAIN	MCU	WKUP
SMS0_RTI_0	✓		
SMS0_RTI_1	✓		

4.14.2.3 RTI_CFG1 Integration

Table 4-377. RTI_CFG1 Module Allocations

Instance	MAIN	MCU	WKUP
RTI4	✓		
RTI5	✓		
RTI15	✓		
RTI0	✓		
RTI1	✓		
RTI2	✓		
RTI3	✓		
RTI8	✓		
MCU_RTI0		✓	
WKUP_RTI0			✓

Table 4-378. Integration Attributes for RTI_CFG1

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
RTI4	PSC0	PD_C7DSP0	LPSC_main_c7dsp0_core	67	OFF	YES	LPSC_main_c7dsp0_common
RTI5	PSC0	PD_C7DSP1	LPSC_main_c7dsp1_core	76	OFF	YES	LPSC_main_c7dsp1_common
RTI15	PSC0	PD_GPU_CTRL	LPSC_main_gpu_ctrl	60	OFF	YES	LPSC_main_ip
RTI0	PSC0	PD_MPU_CLST0_CORE0	LPSC_main_mpu_clst0_core0	56	OFF	YES	LPSC_main_mpu_clst0
RTI1	PSC0	PD_MPU_CLST0_CORE1	LPSC_main_mpu_clst0_core1	57	OFF	YES	LPSC_main_mpu_clst0

Copyright © 2026 Texas Instruments Incorporated

Table 4-378. Integration Attributes for RTI_CFG1 (continued)

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
RTI2	PSC0	PD_MPU_CLST0_CORE2	LPSC_main_mpu_clst0_core2	58	OFF	YES	LPSC_main_mpu_clst0
RTI3	PSC0	PD_MPU_CLST0_CORE3	LPSC_main_mpu_clst0_core3	59	OFF	YES	LPSC_main_mpu_clst0
RTI8	PSC0	PD_MAIN_MCUSS0	LPSC_main_mcuss0_core0	79	OFF	YES	LPSC_main_ip
MCU_RTI0	WKUP_PSC0	PD_MCUSS	LPSC_mcu_r5	6	OFF	YES	LPSC_MCU_common
WKUP_RTI0	PSC0	GP_CORE	LPSC_main_dm	1	OFF	YES	LPSC_main_alwayson

Table 4-379. Clock Integration for RTI_CFG1

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=0	functional clock
RTI4	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=0	functional clock
RTI4	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI4	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI4	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI4	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=2	functional clock
RTI4	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD4_CLKSEL[1:0]=3	functional clock
RTI4	ICLK	MAIN_SYSCLK0/4		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=0	functional clock
RTI5	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=0	functional clock
RTI5	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI5	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI5	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI5	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=2	functional clock
RTI5	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD5_CLKSEL[1:0]=3	functional clock
RTI5	ICLK	MAIN_SYSCLK0/4		interface clock
RTI15	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=0	functional clock
RTI15	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=0	functional clock
RTI15	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI15	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI15	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI15	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI15	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI15	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=2	functional clock
RTI15	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD15_CLKSEL[1:0]=3	functional clock
RTI15	ICLK	MAIN_SYSCLK0/4		interface clock
RTI0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=0	functional clock
RTI0	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=0	functional clock
RTI0	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI0	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI0	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI0	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=2	functional clock
RTI0	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD0_CLKSEL[1:0]=3	functional clock
RTI0	ICLK	MAIN_SYSCLK0/4		interface clock
RTI1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=0	functional clock
RTI1	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=0	functional clock
RTI1	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI1	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI1	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI1	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI1	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD1_CLKSEL[1:0]=3	functional clock
RTI1	ICLK	MAIN_SYSCLK0/4		interface clock
RTI2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=0	functional clock
RTI2	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=0	functional clock
RTI2	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI2	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI2	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI2	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=2	functional clock
RTI2	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD2_CLKSEL[1:0]=3	functional clock
RTI2	ICLK	MAIN_SYSCLK0/4		interface clock
RTI3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=0	functional clock
RTI3	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI3	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI3	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI3	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI3	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=2	functional clock
RTI3	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD3_CLKSEL[1:0]=3	functional clock
RTI3	ICLK	MAIN_SYSCLK0/4		interface clock
RTI8	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=0	functional clock
RTI8	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=0	functional clock
RTI8	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
RTI8	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
RTI8	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
RTI8	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
RTI8	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
RTI8	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=2	functional clock
RTI8	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_WWD8_CLKSEL[1:0]=3	functional clock
RTI8	ICLK	MAIN_SYSCLK0/4		interface clock
MCU_RTI0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=0	functional clock
MCU_RTI0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=0	functional clock
MCU_RTI0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
MCU_RTI0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_RTI0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_RTI0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-379. Clock Integration for RTI_CFG1 (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_RTI0	FCLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
MCU_RTI0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=2	functional clock
MCU_RTI0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_WWD0_CLKSEL[1:0]=3	functional clock
MCU_RTI0	ICLK	MCU_SYSCLK0/4		interface clock
WKUP_RTI0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=0	functional clock
WKUP_RTI0	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=0	functional clock
WKUP_RTI0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
WKUP_RTI0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_RTI0	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_RTI0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
WKUP_RTI0	FCLK	LFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
WKUP_RTI0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=2	functional clock
WKUP_RTI0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_WWD0_CLKSEL[1:0]=3	functional clock
WKUP_RTI0	ICLK	DM_CLK/4		interface clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-380. Reset Integration for RTI_CFG1

Module Instance	Source	Description
RTI4	PSC0	RTI4 reset
RTI5	PSC0	RTI5 reset
RTI15	PSC0	RTI15 reset
RTI0	PSC0	RTI0 reset
RTI1	PSC0	RTI1 reset
RTI2	PSC0	RTI2 reset
RTI3	PSC0	RTI3 reset
RTI8	PSC0	RTI8 reset
MCU_RTI0	WKUP_PSC0	MCU_RTI0 reset
WKUP_RTI0	PSC0	WKUP_RTI0 reset

Table 4-381. Hardware Requests for RTI_CFG1

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
RTI4	RTI4_intr_wwd_0	ESM0_esm_pls_event 0_236	ESM0	RTI4 interrupt request	pulse
RTI4	RTI4_intr_wwd_0	ESM0_esm_pls_event 1_236	ESM0	RTI4 interrupt request	pulse
RTI4	RTI4_intr_wwd_0	ESM0_esm_pls_event 2_236	ESM0	RTI4 interrupt request	pulse
RTI4	RTI4_intr_wwd_0	C7X256V0_CLEC_soc _events_in_4	C7X256V0_CLEC	RTI4 interrupt request	pulse
RTI5	RTI5_intr_wwd_0	ESM0_esm_pls_event 0_239	ESM0	RTI5 interrupt request	pulse
RTI5	RTI5_intr_wwd_0	ESM0_esm_pls_event 1_239	ESM0	RTI5 interrupt request	pulse
RTI5	RTI5_intr_wwd_0	ESM0_esm_pls_event 2_239	ESM0	RTI5 interrupt request	pulse
RTI5	RTI5_intr_wwd_0	C7X256V1_CLEC_soc _events_in_4	C7X256V1_CLEC	RTI5 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	ESM0_esm_pls_event 0_248	ESM0	RTI15 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	ESM0_esm_pls_event 1_248	ESM0	RTI15 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	ESM0_esm_pls_event 2_248	ESM0	RTI15 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	GICSS0_spi_207	GICSS0	RTI15 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	C7X256V0_CLEC_gic _spi_207	C7X256V0_CLEC	RTI15 interrupt request	pulse
RTI15	RTI15_intr_wwd_0	C7X256V1_CLEC_gic _spi_207	C7X256V1_CLEC	RTI15 interrupt request	pulse
RTI0	RTI0_intr_wwd_0	ESM0_esm_pls_event 0_224	ESM0	RTI0 interrupt request	pulse
RTI0	RTI0_intr_wwd_0	ESM0_esm_pls_event 1_224	ESM0	RTI0 interrupt request	pulse
RTI0	RTI0_intr_wwd_0	ESM0_esm_pls_event 2_224	ESM0	RTI0 interrupt request	pulse
RTI0	RTI0_intr_wwd_0	GICSS0_spi_252	GICSS0	RTI0 interrupt request	pulse
RTI0	RTI0_intr_wwd_0	C7X256V0_CLEC_gic _spi_252	C7X256V0_CLEC	RTI0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-381. Hardware Requests for RTI_CFG1 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
RTI0	RTI0_intr_wwd_0	C7X256V1_CLEC_gic_spi_252	C7X256V1_CLEC	RTI0 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	ESM0_esm_pls_event0_225	ESM0	RTI1 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	ESM0_esm_pls_event1_225	ESM0	RTI1 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	ESM0_esm_pls_event2_225	ESM0	RTI1 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	GICSS0_spi_253	GICSS0	RTI1 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	C7X256V0_CLEC_gic_spi_253	C7X256V0_CLEC	RTI1 interrupt request	pulse
RTI1	RTI1_intr_wwd_0	C7X256V1_CLEC_gic_spi_253	C7X256V1_CLEC	RTI1 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	ESM0_esm_pls_event0_241	ESM0	RTI2 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	ESM0_esm_pls_event1_241	ESM0	RTI2 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	ESM0_esm_pls_event2_241	ESM0	RTI2 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	GICSS0_spi_254	GICSS0	RTI2 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	C7X256V0_CLEC_gic_spi_254	C7X256V0_CLEC	RTI2 interrupt request	pulse
RTI2	RTI2_intr_wwd_0	C7X256V1_CLEC_gic_spi_254	C7X256V1_CLEC	RTI2 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	ESM0_esm_pls_event0_242	ESM0	RTI3 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	ESM0_esm_pls_event1_242	ESM0	RTI3 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	ESM0_esm_pls_event2_242	ESM0	RTI3 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	GICSS0_spi_255	GICSS0	RTI3 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	C7X256V0_CLEC_gic_spi_255	C7X256V0_CLEC	RTI3 interrupt request	pulse
RTI3	RTI3_intr_wwd_0	C7X256V1_CLEC_gic_spi_255	C7X256V1_CLEC	RTI3 interrupt request	pulse
RTI8	RTI8_intr_wwd_0	ESM0_esm_pls_event0_249	ESM0	RTI8 interrupt request	pulse
RTI8	RTI8_intr_wwd_0	ESM0_esm_pls_event1_249	ESM0	RTI8 interrupt request	pulse
RTI8	RTI8_intr_wwd_0	ESM0_esm_pls_event2_249	ESM0	RTI8 interrupt request	pulse
RTI8	RTI8_intr_wwd_0	R5FSS0_CORE0_intr_30	R5FSS0_CORE0	RTI8 interrupt request	pulse
MCU_RTI0	MCU_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event0_85	WKUP_ESM0	MCU_RTI0 interrupt request	pulse
MCU_RTI0	MCU_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event1_85	WKUP_ESM0	MCU_RTI0 interrupt request	pulse
MCU_RTI0	MCU_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event2_85	WKUP_ESM0	MCU_RTI0 interrupt request	pulse
MCU_RTI0	MCU_RTI0_intr_wwd_0	MCU_R5FSS0_CORE0_cpu0_intr_30	MCU_R5FSS0_CORE0	MCU_RTI0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-381. Hardware Requests for RTI_CFG1 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event0_86	WKUP_ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event1_86	WKUP_ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	WKUP_ESM0_esm_pls_event2_86	WKUP_ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	ESM0_esm_pls_event0_227	ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	ESM0_esm_pls_event1_227	ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	ESM0_esm_pls_event2_227	ESM0	WKUP_RTI0 interrupt request	pulse
WKUP_RTI0	WKUP_RTI0_intr_wwd_0	WKUP_R5FSS0_CORE0_intr_30	WKUP_R5FSS0_CORE0	WKUP_RTI0 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.14.3 Real-Time Clock (RTC)

This section contains the integration details for the RTC module on this device. For further information, see the *Real-Time Clock (RTC)* section of the *Peripherals* chapter.

4.14.3.1 RTC Unsupported Features

The following features are not supported on this family of devices:

- PMIC enable support
- Analog IP support
- External IO
- Two digital voltage domains

4.14.3.2 rtcss Integration

Table 4-382. RTCSS Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_rtcss0			✓

Table 4-383. Integration Attributes for rtcss

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_rtcss0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-384. Clock Integration for rtcss

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_rtcss0	ANA_OSC32K	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=0 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	None
WKUP_rtcss0	ANA_OSC32K	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=0 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	None
WKUP_rtcss0	ANA_OSC32K	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=0 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	None
WKUP_rtcss0	ANA_OSC32K	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=0 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	None
WKUP_rtcss0	ANA_OSC32K	LFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=0 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	None
WKUP_rtcss0	ANA_OSC32K	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_RTC_CLKSEL[0:0]=1	None

Copyright © 2026 Texas Instruments Incorporated

Table 4-384. Clock Integration for rtcss (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_rtcss0	ICLK	DM_CLK/8		None

Table 4-385. Reset Integration for rtcss

Module Instance	Source	Description
WKUP_rtcss0	PSC0	WKUP_rtcss0 reset

Table 4-386. Hardware Requests for rtcss

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	GICSS0_spi_132	GICSS0	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	R5FSS0_CORE0_intr_97	R5FSS0_CORE0	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	WKUP_R5FSS0_CORE0_intr_97	WKUP_R5FSS0_CORE0	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_97	MCU_R5FSS0_CORE0	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_7	WKUP_DEEPSLEEP_SOURCES0	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	C7X256V0_CLEC_gic_spi_132	C7X256V0_CLEC	WKUP_rtcss0 interrupt request	level
WKUP_rtcss0	WKUP_rtcss0_rtc_event_pend_0	C7X256V1_CLEC_gic_spi_132	C7X256V1_CLEC	WKUP_rtcss0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.14.4 Timer

This section contains the integration details for the Timer module on this device. For further information, see the *Timer* section of the *Peripherals* chapter.

4.14.4.1 Timer Unsupported Features

The following features are not supported on this family of devices:

- Cascading of timer instances

4.14.4.2 TIMER Integration

Table 4-387. TIMER Module Allocations

Instance	MAIN	MCU	WKUP
TIMER0	✓		
TIMER1	✓		
TIMER2	✓		
TIMER3	✓		
TIMER4	✓		
TIMER5	✓		
TIMER6	✓		
TIMER7	✓		
MCU_TIMER0		✓	
MCU_TIMER1		✓	
MCU_TIMER2		✓	
MCU_TIMER3		✓	
WKUP_TIMER0			✓
WKUP_TIMER1			✓

Table 4-388. Integration Attributes for TIMER

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
TIMER0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER1	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER2	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER3	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER4	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER5	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER6	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
TIMER7	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCU_TIMER0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
MCU_TIMER1	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
MCU_TIMER2	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO

Copyright © 2026 Texas Instruments Incorporated

Table 4-388. Integration Attributes for TIMER (continued)

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCU_TIMER3	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO
WKUP_TIMER0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
WKUP_TIMER1	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-389. Clock Integration for TIMER

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER0	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=0	functional clock
TIMER0	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=0	functional clock
TIMER0	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER0	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER0	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER0	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER0	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=10	functional clock
TIMER0	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=11	functional clock
TIMER0	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER0	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=3	functional clock
TIMER0	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=4	functional clock
TIMER0	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=5	functional clock
TIMER0	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=7	functional clock
TIMER0	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=8	functional clock
TIMER0	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER0_CLKSEL[3:0]=9	functional clock
TIMER1	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=0	functional clock
TIMER1	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=0	functional clock
TIMER1	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER1	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER1	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER1	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER1	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=10	functional clock
TIMER1	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=11	functional clock
TIMER1	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=2	functional clock
TIMER1	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=3	functional clock
TIMER1	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=4	functional clock
TIMER1	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=5	functional clock
TIMER1	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=7	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER1	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=8	functional clock
TIMER1	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER1_CLKSEL[3:0]=9	functional clock
TIMER1	FCLK	TIMER0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER1_CTRL[8:8]=1	functional clock
TIMER2	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=0	functional clock
TIMER2	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=0	functional clock
TIMER2	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER2	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER2	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER2	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER2	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=10	functional clock
TIMER2	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=11	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER2	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=2	functional clock
TIMER2	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=3	functional clock
TIMER2	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=4	functional clock
TIMER2	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=5	functional clock
TIMER2	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=7	functional clock
TIMER2	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=8	functional clock
TIMER2	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER2_CLKSEL[3:0]=9	functional clock
TIMER3	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=0	functional clock
TIMER3	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=0	functional clock
TIMER3	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER3	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER3	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER3	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER3	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=10	functional clock
TIMER3	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=11	functional clock
TIMER3	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=2	functional clock
TIMER3	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=3	functional clock
TIMER3	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=4	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER3	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=5	functional clock
TIMER3	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=7	functional clock
TIMER3	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=8	functional clock
TIMER3	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER3_CLKSEL[3:0]=9	functional clock
TIMER3	FCLK	TIMER2 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER3_CTRL[8:8]=1	functional clock
TIMER4	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=0	functional clock
TIMER4	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=0	functional clock
TIMER4	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER4	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER4	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER4	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER4	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=10	functional clock
TIMER4	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=11	functional clock
TIMER4	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=2	functional clock
TIMER4	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=3	functional clock
TIMER4	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=4	functional clock
TIMER4	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=5	functional clock
TIMER4	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=7	functional clock
TIMER4	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=8	functional clock
TIMER4	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER4_CLKSEL[3:0]=9	functional clock
TIMER5	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=0	functional clock
TIMER5	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=0	functional clock
TIMER5	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER5	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER5	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER5	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER5	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=10	functional clock
TIMER5	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=11	functional clock
TIMER5	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=2	functional clock
TIMER5	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=3	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER5	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=4	functional clock
TIMER5	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=5	functional clock
TIMER5	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=7	functional clock
TIMER5	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=8	functional clock
TIMER5	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER5_CLKSEL[3:0]=9	functional clock
TIMER5	FCLK	TIMER4 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER5_CTRL[8:8]=1	functional clock
TIMER6	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER6	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=0	functional clock
TIMER6	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=0	functional clock
TIMER6	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER6	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER6	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER6	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER6	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER6	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=10	functional clock
TIMER6	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=11	functional clock
TIMER6	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=2	functional clock
TIMER6	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=3	functional clock
TIMER6	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=4	functional clock
TIMER6	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=5	functional clock
TIMER6	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=7	functional clock
TIMER6	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=8	functional clock
TIMER6	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER6_CLKSEL[3:0]=9	functional clock
TIMER7	ICLK	MAIN_SYSCLK0/4		interface clock
TIMER7	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=0	functional clock
TIMER7	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER7	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
TIMER7	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER7	FCLK	HFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
TIMER7	FCLK	CLK_32K_RC	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
TIMER7	FCLK	LFOSC0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=1 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
TIMER7	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=10	functional clock
TIMER7	FCLK	CPSW0 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=11	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
TIMER7	FCLK	MAIN_PLL0_HSDIV7_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=2	functional clock
TIMER7	FCLK	CLK_12M_RC	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=3	functional clock
TIMER7	FCLK	MCU_EXT_REFCLK0	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=4	functional clock
TIMER7	FCLK	EXT_REFCLK1	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=5	functional clock
TIMER7	FCLK	CP_GEMAC_CPTS_REF_CLK	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=7	functional clock
TIMER7	FCLK	MAIN_PLL1_HSDIV3_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=8	functional clock
TIMER7	FCLK	MAIN_PLL2_HSDIV6_CLKOUT	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=0 MAIN_CTRL_MMR_CFG0_TIMER7_CLKSEL[3:0]=9	functional clock
TIMER7	FCLK	TIMER6 (INSTANCE)	MAIN_CTRL_MMR_CFG0_TIMER7_CTRL[8:8]=1	functional clock
MCU_TIMER0	ICLK	MCU_SYSCLK0/4		interface clock
MCU_TIMER0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=0	functional clock
MCU_TIMER0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=0	functional clock
MCU_TIMER0	FCLK	MCU_SYSCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=1	functional clock
MCU_TIMER0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER0	FCLK	MCU_PLL0_HSDIV5_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=3	functional clock
MCU_TIMER0	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=4	functional clock
MCU_TIMER0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
MCU_TIMER0	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER0	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
MCU_TIMER0	FCLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
MCU_TIMER0	FCLK	CPSW0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=6	functional clock
MCU_TIMER0	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER0_CLKSEL[2:0]=7	functional clock
MCU_TIMER1	ICLK	MCU_SYSCLK0/4		interface clock
MCU_TIMER1	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=0	functional clock
MCU_TIMER1	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER1	FCLK	MCU_SYSCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=1	functional clock
MCU_TIMER1	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=2	functional clock
MCU_TIMER1	FCLK	MCU_PLL0_HSDIV5_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=3	functional clock
MCU_TIMER1	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=4	functional clock
MCU_TIMER1	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
MCU_TIMER1	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER1	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER1	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER1	FCLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
MCU_TIMER1	FCLK	CPSW0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=6	functional clock
MCU_TIMER1	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER1_CLKSEL[2:0]=7	functional clock
MCU_TIMER1	FCLK	MCU_TIMER0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER1_CTRL[8:8]=1	functional clock
MCU_TIMER2	ICLK	MCU_SYSCLK0/4		interface clock
MCU_TIMER2	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=0	functional clock
MCU_TIMER2	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=0	functional clock
MCU_TIMER2	FCLK	MCU_SYSCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=1	functional clock
MCU_TIMER2	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=2	functional clock
MCU_TIMER2	FCLK	MCU_PLL0_HSDIV5_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=3	functional clock
MCU_TIMER2	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=4	functional clock
MCU_TIMER2	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
MCU_TIMER2	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER2	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER2	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
MCU_TIMER2	FCLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
MCU_TIMER2	FCLK	CPSW0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=6	functional clock
MCU_TIMER2	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER2_CLKSEL[2:0]=7	functional clock
MCU_TIMER3	ICLK	MCU_SYSCLK0/4		interface clock
MCU_TIMER3	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=0	functional clock
MCU_TIMER3	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=0	functional clock
MCU_TIMER3	FCLK	MCU_SYSCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=1	functional clock
MCU_TIMER3	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=2	functional clock
MCU_TIMER3	FCLK	MCU_PLL0_HSDIV5_CLKOUT	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=3	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER3	FCLK	MCU_EXT_REFCLK0	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=4	functional clock
MCU_TIMER3	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
MCU_TIMER3	FCLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER3	FCLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
MCU_TIMER3	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
MCU_TIMER3	FCLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
MCU_TIMER3	FCLK	CPSW0 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=6	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_TIMER3	FCLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=0 MCU_CTRL_MMR_CFG0_MCU_TIMER3_CLKSEL[2:0]=7	functional clock
MCU_TIMER3	FCLK	MCU_TIMER2 (INSTANCE)	MCU_CTRL_MMR_CFG0_MCU_TIMER3_CTRL[8:8]=1	functional clock
WKUP_TIMER0	ICLK	DM_CLK/2		interface clock
WKUP_TIMER0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=0	functional clock
WKUP_TIMER0	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=0	functional clock
WKUP_TIMER0	FCLK	DM_CLK	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=1	functional clock
WKUP_TIMER0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=2	functional clock
WKUP_TIMER0	FCLK	MCU_PLL0_HSDIV5_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=3	functional clock
WKUP_TIMER0	FCLK	MCU_EXT_REFCLK0	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=4	functional clock
WKUP_TIMER0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock
WKUP_TIMER0	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_TIMER0	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_TIMER0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
WKUP_TIMER0	FCLK	LFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_TIMER0	FCLK	CPSW0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=6	functional clock
WKUP_TIMER0	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER0_CLKSEL[2:0]=7	functional clock
WKUP_TIMER1	ICLK	DM_CLK/2		interface clock
WKUP_TIMER1	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=0	functional clock
WKUP_TIMER1	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=0	functional clock
WKUP_TIMER1	FCLK	DM_CLK	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=1	functional clock
WKUP_TIMER1	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=2	functional clock
WKUP_TIMER1	FCLK	MCU_PLL0_HSDIV5_CLKOUT	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=3	functional clock
WKUP_TIMER1	FCLK	MCU_EXT_REFCLK0	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=4	functional clock
WKUP_TIMER1	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	functional clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-389. Clock Integration for TIMER (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_TIMER1	FCLK	CLK_12M_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_TIMER1	FCLK	HFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	functional clock
WKUP_TIMER1	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	functional clock
WKUP_TIMER1	FCLK	LFOSC0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=5 MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	functional clock
WKUP_TIMER1	FCLK	CPSW0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=6	functional clock
WKUP_TIMER1	FCLK	CLK_32K_RC	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=0 WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CLKSEL[2:0]=7	functional clock
WKUP_TIMER1	FCLK	WKUP_TIMER0 (INSTANCE)	WKUP_CTRL_MMR_CFG0_WKUP_TIMER1_CTRL[8:8]=1	functional clock

Table 4-390. Reset Integration for TIMER

Module Instance	Source	Description
TIMER0	PSC0	TIMER0 reset
TIMER1	PSC0	TIMER1 reset
TIMER2	PSC0	TIMER2 reset
TIMER3	PSC0	TIMER3 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-390. Reset Integration for TIMER (continued)

Module Instance	Source	Description
TIMER4	PSC0	TIMER4 reset
TIMER5	PSC0	TIMER5 reset
TIMER6	PSC0	TIMER6 reset
TIMER7	PSC0	TIMER7 reset
MCU_TIMER0	WKUP_PSC0	MCU_TIMER0 reset
MCU_TIMER1	WKUP_PSC0	MCU_TIMER1 reset
MCU_TIMER2	WKUP_PSC0	MCU_TIMER2 reset
MCU_TIMER3	WKUP_PSC0	MCU_TIMER3 reset
WKUP_TIMER0	PSC0	WKUP_TIMER0 reset
WKUP_TIMER1	PSC0	WKUP_TIMER1 reset

Table 4-391. Hardware Requests for TIMER

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMER0	TIMER0_intr_pend_0	GICSS0_spi_152	GICSS0	TIMER0 interrupt request	level
TIMER0	TIMER0_intr_pend_0	R5FSS0_CORE0_intr_24	R5FSS0_CORE0	TIMER0 interrupt request	level
TIMER0	TIMER0_intr_pend_0	C7X256V0_CLEC_gic_spi_152	C7X256V0_CLEC	TIMER0 interrupt request	level
TIMER0	TIMER0_intr_pend_0	C7X256V1_CLEC_gic_spi_152	C7X256V1_CLEC	TIMER0 interrupt request	level
TIMER0	TIMER0_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_162	MAIN_GPIOMUX_INTROUTER0	TIMER0 interrupt request	pulse
TIMER0	TIMER0_timer_pwm_0	TIMESYNC_EVENT_INTROUTER0_in_0	TIMESYNC_EVENT_INTROUTER0	TIMER0 interrupt request	pulse
TIMER1	TIMER1_intr_pend_0	GICSS0_spi_153	GICSS0	TIMER1 interrupt request	level
TIMER1	TIMER1_intr_pend_0	R5FSS0_CORE0_intr_25	R5FSS0_CORE0	TIMER1 interrupt request	level
TIMER1	TIMER1_intr_pend_0	C7X256V0_CLEC_gic_spi_153	C7X256V0_CLEC	TIMER1 interrupt request	level
TIMER1	TIMER1_intr_pend_0	C7X256V1_CLEC_gic_spi_153	C7X256V1_CLEC	TIMER1 interrupt request	level
TIMER1	TIMER1_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_163	MAIN_GPIOMUX_INTROUTER0	TIMER1 interrupt request	pulse
TIMER1	TIMER1_timer_pwm_0	TIMESYNC_EVENT_INTROUTER0_in_1	TIMESYNC_EVENT_INTROUTER0	TIMER1 interrupt request	pulse
TIMER2	TIMER2_intr_pend_0	GICSS0_spi_154	GICSS0	TIMER2 interrupt request	level
TIMER2	TIMER2_intr_pend_0	R5FSS0_CORE0_intr_26	R5FSS0_CORE0	TIMER2 interrupt request	level
TIMER2	TIMER2_intr_pend_0	C7X256V0_CLEC_gic_spi_154	C7X256V0_CLEC	TIMER2 interrupt request	level
TIMER2	TIMER2_intr_pend_0	C7X256V1_CLEC_gic_spi_154	C7X256V1_CLEC	TIMER2 interrupt request	level
TIMER2	TIMER2_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_164	MAIN_GPIOMUX_INTROUTER0	TIMER2 interrupt request	pulse
TIMER2	TIMER2_timer_pwm_0	TIMESYNC_EVENT_INTROUTER0_in_2	TIMESYNC_EVENT_INTROUTER0	TIMER2 interrupt request	pulse
TIMER3	TIMER3_intr_pend_0	GICSS0_spi_155	GICSS0	TIMER3 interrupt request	level
TIMER3	TIMER3_intr_pend_0	R5FSS0_CORE0_intr_27	R5FSS0_CORE0	TIMER3 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-391. Hardware Requests for TIMER (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
TIMER3	TIMER3_intr_pend_0	C7X256V0_CLEC_gic_spi_155	C7X256V0_CLEC	TIMER3 interrupt request	level
TIMER3	TIMER3_intr_pend_0	C7X256V1_CLEC_gic_spi_155	C7X256V1_CLEC	TIMER3 interrupt request	level
TIMER3	TIMER3_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_165	MAIN_GPIOMUX_INTROUTER0	TIMER3 interrupt request	pulse
TIMER3	TIMER3_timer_pwm_0	TIMESYNC_EVENT_INTROUTER0_in_3	TIMESYNC_EVENT_INTROUTER0	TIMER3 interrupt request	pulse
TIMER4	TIMER4_intr_pend_0	GICSS0_spi_156	GICSS0	TIMER4 interrupt request	level
TIMER4	TIMER4_intr_pend_0	R5FSS0_CORE0_intr_28	R5FSS0_CORE0	TIMER4 interrupt request	level
TIMER4	TIMER4_intr_pend_0	C7X256V0_CLEC_gic_spi_156	C7X256V0_CLEC	TIMER4 interrupt request	level
TIMER4	TIMER4_intr_pend_0	C7X256V1_CLEC_gic_spi_156	C7X256V1_CLEC	TIMER4 interrupt request	level
TIMER4	TIMER4_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_166	MAIN_GPIOMUX_INTROUTER0	TIMER4 interrupt request	pulse
TIMER5	TIMER5_intr_pend_0	GICSS0_spi_157	GICSS0	TIMER5 interrupt request	level
TIMER5	TIMER5_intr_pend_0	R5FSS0_CORE0_intr_29	R5FSS0_CORE0	TIMER5 interrupt request	level
TIMER5	TIMER5_intr_pend_0	C7X256V0_CLEC_gic_spi_157	C7X256V0_CLEC	TIMER5 interrupt request	level
TIMER5	TIMER5_intr_pend_0	C7X256V1_CLEC_gic_spi_157	C7X256V1_CLEC	TIMER5 interrupt request	level
TIMER5	TIMER5_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_167	MAIN_GPIOMUX_INTROUTER0	TIMER5 interrupt request	pulse
TIMER6	TIMER6_intr_pend_0	GICSS0_spi_158	GICSS0	TIMER6 interrupt request	level
TIMER6	TIMER6_intr_pend_0	R5FSS0_CORE0_intr_34	R5FSS0_CORE0	TIMER6 interrupt request	level
TIMER6	TIMER6_intr_pend_0	C7X256V0_CLEC_gic_spi_158	C7X256V0_CLEC	TIMER6 interrupt request	level
TIMER6	TIMER6_intr_pend_0	C7X256V1_CLEC_gic_spi_158	C7X256V1_CLEC	TIMER6 interrupt request	level
TIMER6	TIMER6_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_168	MAIN_GPIOMUX_INTROUTER0	TIMER6 interrupt request	pulse
TIMER7	TIMER7_intr_pend_0	GICSS0_spi_159	GICSS0	TIMER7 interrupt request	level
TIMER7	TIMER7_intr_pend_0	R5FSS0_CORE0_intr_35	R5FSS0_CORE0	TIMER7 interrupt request	level
TIMER7	TIMER7_intr_pend_0	C7X256V0_CLEC_gic_spi_159	C7X256V0_CLEC	TIMER7 interrupt request	level
TIMER7	TIMER7_intr_pend_0	C7X256V1_CLEC_gic_spi_159	C7X256V1_CLEC	TIMER7 interrupt request	level
TIMER7	TIMER7_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_169	MAIN_GPIOMUX_INTROUTER0	TIMER7 interrupt request	pulse
MCU_TIMER0	MCU_TIMER0_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_28	MCU_R5FSS0_CORE0	MCU_TIMER0 interrupt request	level
MCU_TIMER0	MCU_TIMER0_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_170	MAIN_GPIOMUX_INTROUTER0	MCU_TIMER0 interrupt request	pulse
MCU_TIMER1	MCU_TIMER1_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_29	MCU_R5FSS0_CORE0	MCU_TIMER1 interrupt request	level
MCU_TIMER1	MCU_TIMER1_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_171	MAIN_GPIOMUX_INTROUTER0	MCU_TIMER1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-391. Hardware Requests for TIMER (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_TIMER2	MCU_TIMER2_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_138	MCU_R5FSS0_CORE0	MCU_TIMER2 interrupt request	level
MCU_TIMER2	MCU_TIMER2_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_172	MAIN_GPIOMUX_INTROUTER0	MCU_TIMER2 interrupt request	pulse
MCU_TIMER3	MCU_TIMER3_intr_pend_0	MCU_R5FSS0_CORE0_cpu0_intr_139	MCU_R5FSS0_CORE0	MCU_TIMER3 interrupt request	level
MCU_TIMER3	MCU_TIMER3_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_173	MAIN_GPIOMUX_INTROUTER0	MCU_TIMER3 interrupt request	pulse
WKUP_TIMER0	WKUP_TIMER0_intr_pend_0	WKUP_R5FSS0_CORE0_intr_138	WKUP_R5FSS0_CORE0	WKUP_TIMER0 interrupt request	level
WKUP_TIMER0	WKUP_TIMER0_timer_clkstop_wakeup_0	WKUP_R5FSS0_CORE0_intr_28	WKUP_R5FSS0_CORE0	WKUP_TIMER0 interrupt request	level
WKUP_TIMER0	WKUP_TIMER0_timer_clkstop_wakeup_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_5	WKUP_DEEPSLEEP_SOURCES0	WKUP_TIMER0 interrupt request	level
WKUP_TIMER0	WKUP_TIMER0_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_174	MAIN_GPIOMUX_INTROUTER0	WKUP_TIMER0 interrupt request	pulse
WKUP_TIMER1	WKUP_TIMER1_intr_pend_0	WKUP_R5FSS0_CORE0_intr_139	WKUP_R5FSS0_CORE0	WKUP_TIMER1 interrupt request	level
WKUP_TIMER1	WKUP_TIMER1_timer_clkstop_wakeup_0	WKUP_R5FSS0_CORE0_intr_29	WKUP_R5FSS0_CORE0	WKUP_TIMER1 interrupt request	level
WKUP_TIMER1	WKUP_TIMER1_timer_clkstop_wakeup_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_6	WKUP_DEEPSLEEP_SOURCES0	WKUP_TIMER1 interrupt request	level
WKUP_TIMER1	WKUP_TIMER1_timer_pwm_0	MAIN_GPIOMUX_INTROUTER0_in_175	MAIN_GPIOMUX_INTROUTER0	WKUP_TIMER1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

4.15 Internal Diagnostic Modules

4.15.1 Dual Clock Comparator (DCC)

This section contains the integration details for the DCC module on this device. For further information, see the *Dual Clock Comparator (DCC)* section of the *Peripherals* chapter.

4.15.1.1 DCC Integration

Table 4-392. DCC Module Allocations

Instance	MAIN	MCU	WKUP
DCC0	✓		
DCC1	✓		
DCC2	✓		
DCC3	✓		
DCC4	✓		
DCC5	✓		
DCC6	✓		
DCC7	✓		
DCC8	✓		
MCU_DCC0		✓	
MCU_DCC1		✓	

Table 4-393. Integration Attributes for DCC

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
DCC0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC1	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC2	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC3	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC4	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC5	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC6	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC7	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
DCC8	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	
MCU_DCC0	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	
MCU_DCC1	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	

Table 4-394. Clock Integration for DCC

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DCC0	DCC_CLKSRC0_CLK	MAIN_PLL0_HSDIV1_CLKOUT		counter1 clock source

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DCC0	DCC_CLKSRC1_CLK	MAIN_PLL0_HSDIV2_CLKOUT		counter1 clock source
DCC0	DCC_CLKSRC2_CLK	MAIN_PLL0_HSDIV3_CLKOUT		counter1 clock source
DCC0	DCC_CLKSRC3_CLK	MAIN_PLL0_HSDIV4_CLKOUT		counter1 clock source
DCC0	DCC_CLKSRC4_CLK	CLK_12M_RC		counter1 clock source
DCC0	DCC_CLKSRC4_CLK	HFOSC0 (INSTANCE)		counter1 clock source
DCC0	DCC_CLKSRC5_CLK	EXT_REFCLK1		counter1 clock source
DCC0	DCC_CLKSRC6_CLK	MAIN_SYSCLK0		counter1 clock source
DCC0	DCC_CLKSRC7_CLK	MAIN_PLL2_HSDIV8_CLKOUT		counter1 clock source
DCC0	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC0	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC0	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC0	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC0	DCC_INPUT10_CLK	MAIN_SYSCLK0/2		secondary oscillator clock
DCC0	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC1	DCC_CLKSRC0_CLK	MAIN_PLL0_HSDIV5_CLKOUT		counter1 clock source
DCC1	DCC_CLKSRC1_CLK	MAIN_PLL0_HSDIV6_CLKOUT		counter1 clock source
DCC1	DCC_CLKSRC2_CLK	MAIN_PLL0_HSDIV7_CLKOUT		counter1 clock source
DCC1	DCC_CLKSRC3_CLK	MAIN_PLL1_HSDIV1_CLKOUT		counter1 clock source
DCC1	DCC_CLKSRC4_CLK	MAIN_PLL15_HSDIV2_CLKOUT/4		counter1 clock source
DCC1	DCC_CLKSRC5_CLK	MAIN_PLL1_HSDIV0_CLKOUT		counter1 clock source
DCC1	DCC_CLKSRC6_CLK	CLK_12M_RC		counter1 clock source
DCC1	DCC_CLKSRC7_CLK	MAIN_PLL1_HSDIV2_CLKOUT		counter1 clock source
DCC1	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC1	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC1	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC1	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC1	DCC_INPUT10_CLK	MAIN_SYSCLK0/4		secondary oscillator clock
DCC1	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC2	DCC_CLKSRC0_CLK	MAIN_PLL1_HSDIV3_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC1_CLK	MAIN_PLL15_HSDIV0_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC2_CLK	MAIN_PLL1_HSDIV5_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC3_CLK	MAIN_PLL1_HSDIV6_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC4_CLK	MAIN_PLL5_HSDIV1_CLKOUT		counter1 clock source

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DCC2	DCC_CLKSRC5_CLK	MAIN_PLL15_HSDIV1_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC6_CLK	MAIN_PLL2_HSDIV2_CLKOUT		counter1 clock source
DCC2	DCC_CLKSRC7_CLK	RMII2_REF_CLK (PIN)		counter1 clock source
DCC2	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC2	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC2	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC2	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC2	DCC_INPUT10_CLK	MAIN_SYSCLK0/4		secondary oscillator clock
DCC2	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC3	DCC_CLKSRC0_CLK	MAIN_PLL1_HSDIV0_CLKOUT		counter1 clock source
DCC3	DCC_CLKSRC1_CLK	MAIN_PLL2_HSDIV5_CLKOUT		counter1 clock source
DCC3	DCC_CLKSRC2_CLK	C7X256V0_CLK (INSTANCE)		counter1 clock source
DCC3	DCC_CLKSRC3_CLK	MAIN_PLL2_HSDIV7_CLKOUT		counter1 clock source
DCC3	DCC_CLKSRC4_CLK	MAIN_PLL2_HSDIV6_CLKOUT		counter1 clock source
DCC3	DCC_CLKSRC5_CLK	MAIN_PLL2_HSDIV9_CLKOUT		counter1 clock source
DCC3	DCC_CLKSRC6_CLK	A53SS0 (INSTANCE)/4		counter1 clock source
DCC3	DCC_CLKSRC7_CLK	DDR32SS0 (INSTANCE)		counter1 clock source
DCC3	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC3	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC3	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC3	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC3	DCC_INPUT10_CLK	MAIN_SYSCLK0/4		secondary oscillator clock
DCC3	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC4	DCC_CLKSRC1_CLK	CP_GEMAC_CPTS_REF_CLK		counter1 clock source
DCC4	DCC_CLKSRC2_CLK	AUDIO_EXT_REFCLK1 (PIN)		counter1 clock source
DCC4	DCC_CLKSRC3_CLK	DPHY_RX0 (INSTANCE)		counter1 clock source
DCC4	DCC_CLKSRC4_CLK	MCU_EXT_REFCLK0		counter1 clock source
DCC4	DCC_CLKSRC5_CLK	RMII1_REF_CLK (PIN)/4		counter1 clock source
DCC4	DCC_CLKSRC7_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	counter1 clock source
DCC4	DCC_CLKSRC7_CLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source
DCC4	DCC_CLKSRC7_CLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DCC4	DCC_CLKSRC7_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	counter1 clock source
DCC4	DCC_CLKSRC7_CLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	counter1 clock source
DCC4	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC4	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC4	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC4	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC4	DCC_INPUT10_CLK	MAIN_SYSCLK0/2		secondary oscillator clock
DCC4	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC5	DCC_CLKSRC0_CLK	MAIN_PLL0_HSDIV8_CLKOUT		counter1 clock source
DCC5	DCC_CLKSRC1_CLK	AUDIO_EXT_REFCLK2 (PIN)		counter1 clock source
DCC5	DCC_CLKSRC2_CLK	MAIN_PLL2_HSDIV1_CLKOUT		counter1 clock source
DCC5	DCC_CLKSRC3_CLK	MAIN_PLL2_HSDIV3_CLKOUT		counter1 clock source
DCC5	DCC_CLKSRC4_CLK	MAIN_PLL2_HSDIV4_CLKOUT		counter1 clock source
DCC5	DCC_CLKSRC5_CLK	MAIN_PLL5_HSDIV0_CLKOUT/2		counter1 clock source
DCC5	DCC_CLKSRC6_CLK	MAIN_PLL17_HSDIV0_CLKOUT		counter1 clock source
DCC5	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC5	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC5	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC5	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC5	DCC_INPUT10_CLK	MAIN_SYSCLK0		secondary oscillator clock
DCC5	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC6	DCC_CLKSRC0_CLK	VOUT0_EXTPCLKIN (PIN)		counter1 clock source
DCC6	DCC_CLKSRC1_CLK	MCASP0_ACLKX (PIN)		counter1 clock source
DCC6	DCC_CLKSRC2_CLK	MCASP0_ACLKR (PIN)		counter1 clock source
DCC6	DCC_CLKSRC3_CLK	MCASP1_ACLKX (PIN)		counter1 clock source
DCC6	DCC_CLKSRC4_CLK	MCASP1_ACLKR (PIN)		counter1 clock source
DCC6	DCC_CLKSRC5_CLK	MCASP2_ACLKX (PIN)		counter1 clock source
DCC6	DCC_CLKSRC6_CLK	MCASP2_ACLKR (PIN)		counter1 clock source
DCC6	DCC_CLKSRC7_CLK	AUDIO_EXT_REFCLK0 (PIN)		counter1 clock source
DCC6	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC6	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DCC6	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC6	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC6	DCC_INPUT10_CLK	MAIN_SYSCLK0		secondary oscillator clock
DCC6	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC7	DCC_CLKSRC0_CLK	MAIN_PLL5_HSDIV2_CLKOUT/2		counter1 clock source
DCC7	DCC_CLKSRC1_CLK	GPU0 (INSTANCE)/4		counter1 clock source
DCC7	DCC_CLKSRC2_CLK	MAIN_PLL16_HSDIV0_CLKOUT/8		counter1 clock source
DCC7	DCC_CLKSRC3_CLK	DPHY_RX1 (INSTANCE)		counter1 clock source
DCC7	DCC_CLKSRC4_CLK	C7X256V1_CLK (INSTANCE)		counter1 clock source
DCC7	DCC_CLKSRC5_CLK	MAIN_PLL0_HSDIV9_CLKOUT		counter1 clock source
DCC7	DCC_CLKSRC6_CLK	MAIN_PLL1_HSDIV4_CLKOUT		counter1 clock source
DCC7	DCC_CLKSRC7_CLK	MAIN_PLL2_HSDIV0_CLKOUT		counter1 clock source
DCC7	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC7	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC7	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC7	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC7	DCC_INPUT10_CLK	MAIN_SYSCLK0		secondary oscillator clock
DCC7	FICLK	MAIN_SYSCLK0/4		functional and interface clock
DCC8	DCC_CLKSRC0_CLK	MAIN_PLL15_HSDIV3_CLKOUT/4		counter1 clock source
DCC8	DCC_CLKSRC1_CLK	MAIN_PLL18_HSDIV0_CLKOUT/2		counter1 clock source
DCC8	DCC_CLKSRC2_CLK	DPHY_RX2 (INSTANCE)		counter1 clock source
DCC8	DCC_CLKSRC3_CLK	DPHY_RX3 (INSTANCE)		counter1 clock source
DCC8	DCC_CLKSRC4_CLK	MCASP3_ACLKX (PIN)		counter1 clock source
DCC8	DCC_CLKSRC5_CLK	MCASP3_ACLKR (PIN)		counter1 clock source
DCC8	DCC_CLKSRC6_CLK	MCASP4_ACLKX (PIN)		counter1 clock source
DCC8	DCC_CLKSRC7_CLK	MCASP4_ACLKR (PIN)		counter1 clock source
DCC8	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
DCC8	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
DCC8	DCC_INPUT01_CLK	EXT_REFCLK1		primary oscillator clock
DCC8	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
DCC8	DCC_INPUT10_CLK	MAIN_SYSCLK0		secondary oscillator clock
DCC8	FICLK	MAIN_SYSCLK0/4		functional and interface clock
MCU_DCC0	DCC_CLKSRC0_CLK	MCU_PLL0_HSDIV0_CLKOUT		counter1 clock source

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_DCC0	DCC_CLKSRC1_CLK	MCU_PLL0_HSDIV1_CLKOUT		counter1 clock source
MCU_DCC0	DCC_CLKSRC2_CLK	MCU_PLL0_HSDIV2_CLKOUT		counter1 clock source
MCU_DCC0	DCC_CLKSRC3_CLK	MCU_PLL0_HSDIV3_CLKOUT/4		counter1 clock source
MCU_DCC0	DCC_CLKSRC4_CLK	MCU_PLL0_HSDIV4_CLKOUT		counter1 clock source
MCU_DCC0	DCC_CLKSRC5_CLK	CLK_32K_RC		counter1 clock source
MCU_DCC0	DCC_CLKSRC6_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	counter1 clock source
MCU_DCC0	DCC_CLKSRC6_CLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source
MCU_DCC0	DCC_CLKSRC6_CLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source
MCU_DCC0	DCC_CLKSRC6_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	counter1 clock source
MCU_DCC0	DCC_CLKSRC6_CLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	counter1 clock source
MCU_DCC0	DCC_CLKSRC7_CLK	MCU_EXT_REFCLK0		counter1 clock source
MCU_DCC0	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
MCU_DCC0	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
MCU_DCC0	DCC_INPUT01_CLK	CLK_32K_RC		primary oscillator clock
MCU_DCC0	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
MCU_DCC0	DCC_INPUT10_CLK	MCU_SYSCLK0/2		secondary oscillator clock
MCU_DCC0	FICLK	MCU_SYSCLK0/4		functional and interface clock
MCU_DCC1	DCC_CLKSRC0_CLK	MCU_PLL0_HSDIV5_CLKOUT		counter1 clock source
MCU_DCC1	DCC_CLKSRC1_CLK	MCU_PLL0_HSDIV6_CLKOUT		counter1 clock source
MCU_DCC1	DCC_CLKSRC5_CLK	CLK_32K_RC		counter1 clock source
MCU_DCC1	DCC_CLKSRC6_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=0	counter1 clock source
MCU_DCC1	DCC_CLKSRC6_CLK	CLK_12M_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source
MCU_DCC1	DCC_CLKSRC6_CLK	HFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=1	counter1 clock source
MCU_DCC1	DCC_CLKSRC6_CLK	CLK_32K_RC	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=2	counter1 clock source
MCU_DCC1	DCC_CLKSRC6_CLK	LFOSC0 (INSTANCE)	MCU_CTRL_MMR_CFG0_DEVICE_CLKOUT_32K_CTRL[1:0]=3	counter1 clock source
MCU_DCC1	DCC_CLKSRC7_CLK	MCU_EXT_REFCLK0		counter1 clock source

Copyright © 2026 Texas Instruments Incorporated

Table 4-394. Clock Integration for DCC (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_DCC1	DCC_INPUT00_CLK	CLK_12M_RC		primary oscillator clock
MCU_DCC1	DCC_INPUT00_CLK	HFOSC0 (INSTANCE)		primary oscillator clock
MCU_DCC1	DCC_INPUT01_CLK	CLK_32K_RC		primary oscillator clock
MCU_DCC1	DCC_INPUT02_CLK	CLK_12M_RC		primary oscillator clock
MCU_DCC1	DCC_INPUT10_CLK	MCU_SYSCLK0/2		secondary oscillator clock
MCU_DCC1	FICLK	MCU_SYSCLK0/4		functional and interface clock

Table 4-395. Reset Integration for DCC

Module Instance	Source	Description
DCC0	PSC0	DCC0 reset
DCC1	PSC0	DCC1 reset
DCC2	PSC0	DCC2 reset
DCC3	PSC0	DCC3 reset
DCC4	PSC0	DCC4 reset
DCC5	PSC0	DCC5 reset
DCC6	PSC0	DCC6 reset
DCC7	PSC0	DCC7 reset
DCC8	PSC0	DCC8 reset
MCU_DCC0	WKUP_PSC0	MCU_DCC0 reset
MCU_DCC1	WKUP_PSC0	MCU_DCC1 reset

Table 4-396. Hardware Requests for DCC

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DCC0	DCC0_intr_done_level_0	GICSS0_spi_128	GICSS0	DCC0 interrupt request	level
DCC0	DCC0_intr_done_level_0	R5FSS0_CORE0_intr_109	R5FSS0_CORE0	DCC0 interrupt request	level
DCC0	DCC0_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_109	WKUP_R5FSS0_CORE0	DCC0 interrupt request	level
DCC0	DCC0_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_109	MCU_R5FSS0_CORE0	DCC0 interrupt request	level
DCC0	DCC0_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC0 interrupt request	level
DCC0	DCC0_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC0 interrupt request	level
DCC0	DCC0_intr_err_level_0	ESM0_esm_lvl_event_112	ESM0	DCC0 interrupt request	level
DCC1	DCC1_intr_done_level_0	GICSS0_spi_128	GICSS0	DCC1 interrupt request	level
DCC1	DCC1_intr_done_level_0	R5FSS0_CORE0_intr_109	R5FSS0_CORE0	DCC1 interrupt request	level
DCC1	DCC1_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_109	WKUP_R5FSS0_CORE0	DCC1 interrupt request	level
DCC1	DCC1_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_109	MCU_R5FSS0_CORE0	DCC1 interrupt request	level
DCC1	DCC1_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC1 interrupt request	level
DCC1	DCC1_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-396. Hardware Requests for DCC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DCC1	DCC1_intr_err_level_0	ESM0_esm_lvl_event _113	ESM0	DCC1 interrupt request	level
DCC2	DCC2_intr_done_level _0	GICSS0_spi_128	GICSS0	DCC2 interrupt request	level
DCC2	DCC2_intr_done_level _0	R5FSS0_CORE0_intr _109	R5FSS0_CORE0	DCC2 interrupt request	level
DCC2	DCC2_intr_done_level _0	WKUP_R5FSS0_COR E0_intr_109	WKUP_R5FSS0_COR E0	DCC2 interrupt request	level
DCC2	DCC2_intr_done_level _0	MCU_R5FSS0_CORE 0_cpu0_intr_109	MCU_R5FSS0_CORE 0	DCC2 interrupt request	level
DCC2	DCC2_intr_done_level _0	C7X256V0_CLEC_gic _spi_128	C7X256V0_CLEC	DCC2 interrupt request	level
DCC2	DCC2_intr_done_level _0	C7X256V1_CLEC_gic _spi_128	C7X256V1_CLEC	DCC2 interrupt request	level
DCC2	DCC2_intr_err_level_0	ESM0_esm_lvl_event _114	ESM0	DCC2 interrupt request	level
DCC3	DCC3_intr_done_level _0	GICSS0_spi_128	GICSS0	DCC3 interrupt request	level
DCC3	DCC3_intr_done_level _0	R5FSS0_CORE0_intr _109	R5FSS0_CORE0	DCC3 interrupt request	level
DCC3	DCC3_intr_done_level _0	WKUP_R5FSS0_COR E0_intr_109	WKUP_R5FSS0_COR E0	DCC3 interrupt request	level
DCC3	DCC3_intr_done_level _0	MCU_R5FSS0_CORE 0_cpu0_intr_109	MCU_R5FSS0_CORE 0	DCC3 interrupt request	level
DCC3	DCC3_intr_done_level _0	C7X256V0_CLEC_gic _spi_128	C7X256V0_CLEC	DCC3 interrupt request	level
DCC3	DCC3_intr_done_level _0	C7X256V1_CLEC_gic _spi_128	C7X256V1_CLEC	DCC3 interrupt request	level
DCC3	DCC3_intr_err_level_0	ESM0_esm_lvl_event _115	ESM0	DCC3 interrupt request	level
DCC4	DCC4_intr_done_level _0	GICSS0_spi_128	GICSS0	DCC4 interrupt request	level
DCC4	DCC4_intr_done_level _0	R5FSS0_CORE0_intr _109	R5FSS0_CORE0	DCC4 interrupt request	level
DCC4	DCC4_intr_done_level _0	WKUP_R5FSS0_COR E0_intr_109	WKUP_R5FSS0_COR E0	DCC4 interrupt request	level
DCC4	DCC4_intr_done_level _0	MCU_R5FSS0_CORE 0_cpu0_intr_109	MCU_R5FSS0_CORE 0	DCC4 interrupt request	level
DCC4	DCC4_intr_done_level _0	C7X256V0_CLEC_gic _spi_128	C7X256V0_CLEC	DCC4 interrupt request	level
DCC4	DCC4_intr_done_level _0	C7X256V1_CLEC_gic _spi_128	C7X256V1_CLEC	DCC4 interrupt request	level
DCC4	DCC4_intr_err_level_0	ESM0_esm_lvl_event _116	ESM0	DCC4 interrupt request	level
DCC5	DCC5_intr_done_level _0	GICSS0_spi_128	GICSS0	DCC5 interrupt request	level
DCC5	DCC5_intr_done_level _0	R5FSS0_CORE0_intr _109	R5FSS0_CORE0	DCC5 interrupt request	level
DCC5	DCC5_intr_done_level _0	WKUP_R5FSS0_COR E0_intr_109	WKUP_R5FSS0_COR E0	DCC5 interrupt request	level
DCC5	DCC5_intr_done_level _0	MCU_R5FSS0_CORE 0_cpu0_intr_109	MCU_R5FSS0_CORE 0	DCC5 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-396. Hardware Requests for DCC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DCC5	DCC5_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC5 interrupt request	level
DCC5	DCC5_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC5 interrupt request	level
DCC5	DCC5_intr_err_level_0	ESM0_esm_lvl_event_117	ESM0	DCC5 interrupt request	level
DCC6	DCC6_intr_done_level_0	GICSS0_spi_128	GICSS0	DCC6 interrupt request	level
DCC6	DCC6_intr_done_level_0	R5FSS0_CORE0_intr_109	R5FSS0_CORE0	DCC6 interrupt request	level
DCC6	DCC6_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_109	WKUP_R5FSS0_CORE0	DCC6 interrupt request	level
DCC6	DCC6_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_109	MCU_R5FSS0_CORE0	DCC6 interrupt request	level
DCC6	DCC6_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC6 interrupt request	level
DCC6	DCC6_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC6 interrupt request	level
DCC6	DCC6_intr_err_level_0	ESM0_esm_lvl_event_79	ESM0	DCC6 interrupt request	level
DCC7	DCC7_intr_done_level_0	GICSS0_spi_128	GICSS0	DCC7 interrupt request	level
DCC7	DCC7_intr_done_level_0	R5FSS0_CORE0_intr_109	R5FSS0_CORE0	DCC7 interrupt request	level
DCC7	DCC7_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_109	WKUP_R5FSS0_CORE0	DCC7 interrupt request	level
DCC7	DCC7_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_109	MCU_R5FSS0_CORE0	DCC7 interrupt request	level
DCC7	DCC7_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC7 interrupt request	level
DCC7	DCC7_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC7 interrupt request	level
DCC7	DCC7_intr_err_level_0	ESM0_esm_lvl_event_73	ESM0	DCC7 interrupt request	level
DCC8	DCC8_intr_done_level_0	GICSS0_spi_128	GICSS0	DCC8 interrupt request	level
DCC8	DCC8_intr_done_level_0	R5FSS0_CORE0_intr_109	R5FSS0_CORE0	DCC8 interrupt request	level
DCC8	DCC8_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_109	WKUP_R5FSS0_CORE0	DCC8 interrupt request	level
DCC8	DCC8_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_109	MCU_R5FSS0_CORE0	DCC8 interrupt request	level
DCC8	DCC8_intr_done_level_0	C7X256V0_CLEC_gic_spi_128	C7X256V0_CLEC	DCC8 interrupt request	level
DCC8	DCC8_intr_done_level_0	C7X256V1_CLEC_gic_spi_128	C7X256V1_CLEC	DCC8 interrupt request	level
DCC8	DCC8_intr_err_level_0	ESM0_esm_lvl_event_223	ESM0	DCC8 interrupt request	level
MCU_DCC0	MCU_DCC0_intr_done_level_0	R5FSS0_CORE0_intr_108	R5FSS0_CORE0	MCU_DCC0 interrupt request	level
MCU_DCC0	MCU_DCC0_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_108	WKUP_R5FSS0_CORE0	MCU_DCC0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-396. Hardware Requests for DCC (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_DCC0	MCU_DCC0_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_108	MCU_R5FSS0_CORE0	MCU_DCC0 interrupt request	level
MCU_DCC0	MCU_DCC0_intr_err_level_0	WKUP_ESM0_esm_lvl_event_37	WKUP_ESM0	MCU_DCC0 interrupt request	level
MCU_DCC1	MCU_DCC1_intr_done_level_0	R5FSS0_CORE0_intr_137	R5FSS0_CORE0	MCU_DCC1 interrupt request	level
MCU_DCC1	MCU_DCC1_intr_done_level_0	WKUP_R5FSS0_CORE0_intr_137	WKUP_R5FSS0_CORE0	MCU_DCC1 interrupt request	level
MCU_DCC1	MCU_DCC1_intr_done_level_0	MCU_R5FSS0_CORE0_cpu0_intr_137	MCU_R5FSS0_CORE0	MCU_DCC1 interrupt request	level
MCU_DCC1	MCU_DCC1_intr_err_level_0	WKUP_ESM0_esm_lvl_event_36	WKUP_ESM0	MCU_DCC1 interrupt request	level

4.15.1.2 DCC Input Source Clock Mapping

- DCC0 Input Source Clock Mapping
- DCC1 Input Source Clock Mapping
- DCC2 Input Source Clock Mapping
- DCC3 Input Source Clock Mapping
- DCC4 Input Source Clock Mapping
- DCC5 Input Source Clock Mapping
- DCC6 Input Source Clock Mapping
- DCC7 Input Source Clock Mapping
- DCC8 Input Source Clock Mapping
- MCU_DCC0 Input Source Clock Mapping
- MCU_DCC1 Input Source Clock Mapping

Table 4-397. DCC0 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC0	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC0	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC0	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC0	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC0	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC0	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0/2
DCC0	DCC_CLKSRC0_CLK	1	1	MAIN_PLL0_HSDIV1_CLKOUT
DCC0	DCC_CLKSRC1_CLK	1	2	MAIN_PLL0_HSDIV2_CLKOUT
DCC0	DCC_CLKSRC2_CLK	1	3	MAIN_PLL0_HSDIV3_CLKOUT
DCC0	DCC_CLKSRC3_CLK	1	4	MAIN_PLL0_HSDIV4_CLKOUT
DCC0	DCC_CLKSRC4_CLK	1	5	CLK_12M_RC
DCC0	DCC_CLKSRC4_CLK	1	5	HFOSC0 (INSTANCE)
DCC0	DCC_CLKSRC5_CLK	1	6	EXT_REFCLK1
DCC0	DCC_CLKSRC6_CLK	1	7	MAIN_SYSCLK0
DCC0	DCC_CLKSRC7_CLK	1	8	MAIN_PLL2_HSDIV8_CLKOUT
DCC0	FICLK	1	9	MAIN_SYSCLK0/4

Copyright © 2026 Texas Instruments Incorporated

Table 4-398. DCC1 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC1	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC1	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC1	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC1	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC1	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC1	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0/4
DCC1	DCC_CLKSRC0_CLK	1	1	MAIN_PLL0_HSDIV5_CLKOUT
DCC1	DCC_CLKSRC1_CLK	1	2	MAIN_PLL0_HSDIV6_CLKOUT
DCC1	DCC_CLKSRC2_CLK	1	3	MAIN_PLL0_HSDIV7_CLKOUT
DCC1	DCC_CLKSRC3_CLK	1	4	MAIN_PLL1_HSDIV1_CLKOUT
DCC1	DCC_CLKSRC4_CLK	1	5	MAIN_PLL15_HSDIV2_CLKOUT/4
DCC1	DCC_CLKSRC5_CLK	1	6	MAIN_PLL1_HSDIV0_CLKOUT
DCC1	DCC_CLKSRC6_CLK	1	7	CLK_12M_RC
DCC1	DCC_CLKSRC7_CLK	1	8	MAIN_PLL1_HSDIV2_CLKOUT
DCC1	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-399. DCC2 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC2	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC2	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC2	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC2	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC2	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC2	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0/4
DCC2	DCC_CLKSRC0_CLK	1	1	MAIN_PLL1_HSDIV3_CLKOUT
DCC2	DCC_CLKSRC1_CLK	1	2	MAIN_PLL15_HSDIV0_CLKOUT
DCC2	DCC_CLKSRC2_CLK	1	3	MAIN_PLL1_HSDIV5_CLKOUT
DCC2	DCC_CLKSRC3_CLK	1	4	MAIN_PLL1_HSDIV6_CLKOUT
DCC2	DCC_CLKSRC4_CLK	1	5	MAIN_PLL5_HSDIV1_CLKOUT
DCC2	DCC_CLKSRC5_CLK	1	6	MAIN_PLL15_HSDIV1_CLKOUT
DCC2	DCC_CLKSRC6_CLK	1	7	MAIN_PLL2_HSDIV2_CLKOUT
DCC2	DCC_CLKSRC7_CLK	1	8	RMII2_REF_CLK (PIN)
DCC2	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-400. DCC3 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC3	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC3	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC3	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC3	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC3	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC3	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0/4

Copyright © 2026 Texas Instruments Incorporated

Table 4-400. DCC3 Input Source Clock Mapping (continued)

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC3	DCC_CLKSRC0_CLK	1	1	MAIN_PLL1_HSDIV0_CLKOUT
DCC3	DCC_CLKSRC1_CLK	1	2	MAIN_PLL2_HSDIV5_CLKOUT
DCC3	DCC_CLKSRC2_CLK	1	3	C7X256V0_CLK (INSTANCE)
DCC3	DCC_CLKSRC3_CLK	1	4	MAIN_PLL2_HSDIV7_CLKOUT
DCC3	DCC_CLKSRC4_CLK	1	5	MAIN_PLL2_HSDIV6_CLKOUT
DCC3	DCC_CLKSRC5_CLK	1	6	MAIN_PLL2_HSDIV9_CLKOUT
DCC3	DCC_CLKSRC6_CLK	1	7	A53SS0 (INSTANCE)/4
DCC3	DCC_CLKSRC7_CLK	1	8	DDR32SS0 (INSTANCE)
DCC3	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-401. DCC4 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC4	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC4	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC4	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC4	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC4	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC4	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0/2
DCC4	DCC_CLKSRC1_CLK	1	2	CP_GEMAC_CPTS_REF_CLK
DCC4	DCC_CLKSRC2_CLK	1	3	AUDIO_EXT_REFCLK1 (PIN)
DCC4	DCC_CLKSRC3_CLK	1	4	DPHY_RX0 (INSTANCE)
DCC4	DCC_CLKSRC4_CLK	1	5	MCU_EXT_REFCLK0
DCC4	DCC_CLKSRC5_CLK	1	6	RMII1_REF_CLK (PIN)/4
DCC4	DCC_CLKSRC7_CLK	1	8	CLK_32K_RC
DCC4	DCC_CLKSRC7_CLK	1	8	CLK_12M_RC
DCC4	DCC_CLKSRC7_CLK	1	8	HFOSC0 (INSTANCE)
DCC4	DCC_CLKSRC7_CLK	1	8	CLK_32K_RC
DCC4	DCC_CLKSRC7_CLK	1	8	LFOSC0 (INSTANCE)
DCC4	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-402. DCC5 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC5	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC5	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC5	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC5	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC5	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC5	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0
DCC5	DCC_CLKSRC0_CLK	1	1	MAIN_PLL0_HSDIV8_CLKOUT
DCC5	DCC_CLKSRC1_CLK	1	2	AUDIO_EXT_REFCLK2 (PIN)
DCC5	DCC_CLKSRC2_CLK	1	3	MAIN_PLL2_HSDIV1_CLKOUT
DCC5	DCC_CLKSRC3_CLK	1	4	MAIN_PLL2_HSDIV3_CLKOUT

Copyright © 2026 Texas Instruments Incorporated

Table 4-402. DCC5 Input Source Clock Mapping (continued)

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC5	DCC_CLKSRC4_CLK	1	5	MAIN_PLL2_HSDIV4_CLKOUT
DCC5	DCC_CLKSRC5_CLK	1	6	MAIN_PLL5_HSDIV0_CLKOUT/2
DCC5	DCC_CLKSRC6_CLK	1	7	MAIN_PLL17_HSDIV0_CLKOUT
DCC5	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-403. DCC6 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC6	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC6	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC6	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC6	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC6	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC6	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0
DCC6	DCC_CLKSRC0_CLK	1	1	VOUT0_EXTPCLKIN (PIN)
DCC6	DCC_CLKSRC1_CLK	1	2	MCASP0_ACLKX (PIN)
DCC6	DCC_CLKSRC2_CLK	1	3	MCASP0_ACLKR (PIN)
DCC6	DCC_CLKSRC3_CLK	1	4	MCASP1_ACLKX (PIN)
DCC6	DCC_CLKSRC4_CLK	1	5	MCASP1_ACLKR (PIN)
DCC6	DCC_CLKSRC5_CLK	1	6	MCASP2_ACLKX (PIN)
DCC6	DCC_CLKSRC6_CLK	1	7	MCASP2_ACLKR (PIN)
DCC6	DCC_CLKSRC7_CLK	1	8	AUDIO_EXT_REFCLK0 (PIN)
DCC6	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-404. DCC7 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC7	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC7	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC7	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC7	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC7	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC7	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0
DCC7	DCC_CLKSRC0_CLK	1	1	MAIN_PLL5_HSDIV2_CLKOUT/2
DCC7	DCC_CLKSRC1_CLK	1	2	GPU0 (INSTANCE)/4
DCC7	DCC_CLKSRC2_CLK	1	3	MAIN_PLL16_HSDIV0_CLKOUT/8
DCC7	DCC_CLKSRC3_CLK	1	4	DPHY_RX1 (INSTANCE)
DCC7	DCC_CLKSRC4_CLK	1	5	C7X256V1_CLK (INSTANCE)
DCC7	DCC_CLKSRC5_CLK	1	6	MAIN_PLL0_HSDIV9_CLKOUT
DCC7	DCC_CLKSRC6_CLK	1	7	MAIN_PLL1_HSDIV4_CLKOUT
DCC7	DCC_CLKSRC7_CLK	1	8	MAIN_PLL2_HSDIV0_CLKOUT
DCC7	FICLK	1	9	MAIN_SYSCLK0/4

Copyright © 2026 Texas Instruments Incorporated

Table 4-405. DCC8 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
DCC8	DCC_INPUT00_CLK	0	0	CLK_12M_RC
DCC8	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
DCC8	DCC_INPUT01_CLK	0	1	EXT_REFCLK1
DCC8	DCC_INPUT02_CLK	0	2	CLK_12M_RC
DCC8	DCC_INPUT03_CLK	0	3	MAIN_SYSCLK0/4
DCC8	DCC_INPUT10_CLK	1	0	MAIN_SYSCLK0
DCC8	DCC_CLKSRC0_CLK	1	1	MAIN_PLL15_HSDIV3_CLKOUT/4
DCC8	DCC_CLKSRC1_CLK	1	2	MAIN_PLL18_HSDIV0_CLKOUT/2
DCC8	DCC_CLKSRC2_CLK	1	3	DPHY_RX2 (INSTANCE)
DCC8	DCC_CLKSRC3_CLK	1	4	DPHY_RX3 (INSTANCE)
DCC8	DCC_CLKSRC4_CLK	1	5	MCASP3_ACLKX (PIN)
DCC8	DCC_CLKSRC5_CLK	1	6	MCASP3_ACLKR (PIN)
DCC8	DCC_CLKSRC6_CLK	1	7	MCASP4_ACLKX (PIN)
DCC8	DCC_CLKSRC7_CLK	1	8	MCASP4_ACLKR (PIN)
DCC8	FICLK	1	9	MAIN_SYSCLK0/4

Table 4-406. MCU_DCC0 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
MCU_DCC0	DCC_INPUT00_CLK	0	0	CLK_12M_RC
MCU_DCC0	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)
MCU_DCC0	DCC_INPUT01_CLK	0	1	CLK_32K_RC
MCU_DCC0	DCC_INPUT02_CLK	0	2	CLK_12M_RC
MCU_DCC0	DCC_INPUT03_CLK	0	3	MCU_SYSCLK0/4
MCU_DCC0	DCC_INPUT10_CLK	1	0	MCU_SYSCLK0/2
MCU_DCC0	DCC_CLKSRC0_CLK	1	1	MCU_PLL0_HSDIV0_CLKOUT
MCU_DCC0	DCC_CLKSRC1_CLK	1	2	MCU_PLL0_HSDIV1_CLKOUT
MCU_DCC0	DCC_CLKSRC2_CLK	1	3	MCU_PLL0_HSDIV2_CLKOUT
MCU_DCC0	DCC_CLKSRC3_CLK	1	4	MCU_PLL0_HSDIV3_CLKOUT/4
MCU_DCC0	DCC_CLKSRC4_CLK	1	5	MCU_PLL0_HSDIV4_CLKOUT
MCU_DCC0	DCC_CLKSRC5_CLK	1	6	CLK_32K_RC
MCU_DCC0	DCC_CLKSRC6_CLK	1	7	CLK_32K_RC
MCU_DCC0	DCC_CLKSRC6_CLK	1	7	CLK_12M_RC
MCU_DCC0	DCC_CLKSRC6_CLK	1	7	HFOSC0 (INSTANCE)
MCU_DCC0	DCC_CLKSRC6_CLK	1	7	CLK_32K_RC
MCU_DCC0	DCC_CLKSRC6_CLK	1	7	LFOSC0 (INSTANCE)
MCU_DCC0	DCC_CLKSRC7_CLK	1	8	MCU_EXT_REFCLK0
MCU_DCC0	FICLK	1	9	MCU_SYSCLK0/4

Table 4-407. MCU_DCC1 Input Source Clock Mapping

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
MCU_DCC1	DCC_INPUT00_CLK	0	0	CLK_12M_RC
MCU_DCC1	DCC_INPUT00_CLK	0	0	HFOSC0 (INSTANCE)

Copyright © 2026 Texas Instruments Incorporated

Table 4-407. MCU_DCC1 Input Source Clock Mapping (continued)

Domain Instance	Domain Input	Input/ MUX	DCCCLKSRC0/ DCCCLKSRC1 Value	Clock Source
MCU_DCC1	DCC_INPUT01_CLK	0	1	CLK_32K_RC
MCU_DCC1	DCC_INPUT02_CLK	0	2	CLK_12M_RC
MCU_DCC1	DCC_INPUT03_CLK	0	3	MCU_SYSCLK0/4
MCU_DCC1	DCC_INPUT10_CLK	1	0	MCU_SYSCLK0/2
MCU_DCC1	DCC_CLKSRC0_CLK	1	1	MCU_PLL0_HSDIV5_CLKOUT
MCU_DCC1	DCC_CLKSRC1_CLK	1	2	MCU_PLL0_HSDIV6_CLKOUT
MCU_DCC1	DCC_CLKSRC5_CLK	1	6	CLK_32K_RC
MCU_DCC1	DCC_CLKSRC6_CLK	1	7	CLK_32K_RC
MCU_DCC1	DCC_CLKSRC6_CLK	1	7	CLK_12M_RC
MCU_DCC1	DCC_CLKSRC6_CLK	1	7	HFOSC0 (INSTANCE)
MCU_DCC1	DCC_CLKSRC6_CLK	1	7	CLK_32K_RC
MCU_DCC1	DCC_CLKSRC6_CLK	1	7	LFOSC0 (INSTANCE)
MCU_DCC1	DCC_CLKSRC7_CLK	1	8	MCU_EXT_REFCLK0
MCU_DCC1	FICLK	1	9	MCU_SYSCLK0/4

Copyright © 2026 Texas Instruments Incorporated

4.15.2 Error Signaling Module (ESM)

This section contains the integration details for the ESM module on this device. For further information, see the *Error Signaling Module (ESM)* section of the *Peripherals* chapter.

4.15.2.1 ESM Unsupported Features

The following features are not supported on this family of devices:

- No Dedicated ERROR Pin for MAIN domain ESM. MAIN ESM error interrupts are routed to the MCU ESM - the MCU ESM drives the MCU_ERROR pin and can factor in errors from MAIN ESM if programmed to do so.

4.15.2.2 ESM Integration

Table 4-408. ESM Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_ESM0			✓
ESM0	✓		

Table 4-409. Integration Attributes for ESM

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
WKUP_ESM0	WKUP_PSC0	GP_core_CTL_MCU	LPSC_mcu_alwayson	0	ON	NO	
ESM0	PSC0	GP_CORE	LPSC_main_alwayson	0	ON	NO	

Table 4-410. Clock Integration for ESM

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_ESM0	FICLK	MCU_SYSCLK0/4		functional and interface clock
ESM0	CLK	MAIN_SYSCLK0/4		None

Table 4-411. Reset Integration for ESM

Module Instance	Source	Description
WKUP_ESM0	WKUP_PSC0	WKUP_ESM0 reset
ESM0	PSC0	ESM0 reset

Table 4-412. Hardware Requests for ESM

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ESM0	WKUP_ESM0_esm_int_cfg_lvl_0	ESM0_esm_lvl_event_37	ESM0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_cfg_lvl_0	R5FSS0_CORE0_intr_140	R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_cfg_lvl_0	WKUP_R5FSS0_CORE0_intr_140	WKUP_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_cfg_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_140	MCU_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_hi_lvl_0	ESM0_esm_lvl_event_38	ESM0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_hi_lvl_0	R5FSS0_CORE0_intr_141	R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_hi_lvl_0	WKUP_R5FSS0_CORE0_intr_141	WKUP_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_hi_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_141	MCU_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-412. Hardware Requests for ESM (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ESM0	WKUP_ESM0_esm_int_low_lvl_0	ESM0_esm_lvl_event_39	ESM0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_low_lvl_0	R5FSS0_CORE0_intr_142	R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_low_lvl_0	WKUP_R5FSS0_CORE0_intr_142	WKUP_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
WKUP_ESM0	WKUP_ESM0_esm_int_low_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_142	MCU_R5FSS0_CORE0	WKUP_ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	WKUP_ESM0_esm_lvl_event_0	WKUP_ESM0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	GICSS0_spi_180	GICSS0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	R5FSS0_CORE0_intr_167	R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	WKUP_R5FSS0_CORE0_intr_167	WKUP_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_167	MCU_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	C7X256V0_CLEC_gic_spi_180	C7X256V0_CLEC	ESM0 interrupt request	level
ESM0	ESM0_esm_int_cfg_lvl_0	C7X256V1_CLEC_gic_spi_180	C7X256V1_CLEC	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	WKUP_ESM0_esm_lvl_event_1	WKUP_ESM0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	GICSS0_spi_181	GICSS0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	R5FSS0_CORE0_intr_168	R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	WKUP_R5FSS0_CORE0_intr_168	WKUP_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_168	MCU_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	C7X256V0_CLEC_gic_spi_181	C7X256V0_CLEC	ESM0 interrupt request	level
ESM0	ESM0_esm_int_hi_lvl_0	C7X256V1_CLEC_gic_spi_181	C7X256V1_CLEC	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	WKUP_ESM0_esm_lvl_event_2	WKUP_ESM0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	GICSS0_spi_182	GICSS0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	R5FSS0_CORE0_intr_169	R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	WKUP_R5FSS0_CORE0_intr_169	WKUP_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	MCU_R5FSS0_CORE0_cpu0_intr_169	MCU_R5FSS0_CORE0	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	C7X256V0_CLEC_gic_spi_182	C7X256V0_CLEC	ESM0 interrupt request	level
ESM0	ESM0_esm_int_low_lvl_0	C7X256V1_CLEC_gic_spi_182	C7X256V1_CLEC	ESM0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.15.3 Memory Cyclic Redundancy Check (MCRC64)

This section contains the integration details for the MCRC64 module on this device. For further information, see the *Memory Cyclic Redundancy Check (MCRC)* section of the *Peripherals* chapter.

4.15.3.1 MCRC64 Unsupported Features

The following features are not supported on this family of devices:

- Data Trace Mode (Automatic PSA on CPU Instruction and Data TCM busses) - Wrapper only supports VBUSM Signature Analysis. ITCM, DTCM designed for Cortex R5F
- DMA is not supported by the MCU instances

4.15.3.2 MCRC64 Integration

Table 4-413. MCRC64 Module Allocations

Instance	MAIN	MCU	WKUP
MCRC64_0	✓		
MCU_MCRC64_0		✓	

Table 4-414. Integration Attributes for MCRC64

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
MCRC64_0	PSC0	GP_CORE	LPSC_main_ip	34	ON	YES	LPSC_main_dm2main_infra_iso
MCU_MCRC64_0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_common	9	ON	YES	LPSC_DM2safe_ISO

Table 4-415. Clock Integration for MCRC64

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCRC64_0	FICLK	MAIN_SYSCLK0/2		None
MCU_MCRC64_0	FICLK	MCU_SYSCLK0		None

Table 4-416. Reset Integration for MCRC64

Module Instance	Source	Description
MCRC64_0	PSC0	MCRC64_0 reset
MCU_MCRC64_0	WKUP_PSC0	MCU_MCRC64_0 reset

Table 4-417. Hardware Requests for MCRC64

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCRC64_0	MCRC64_0_dma_event_0	DMASS0_INTAGGR_0_intaggr_levi_pend_28	DMASS0_INTAGGR_0	MCRC64_0 interrupt request	pulse
MCRC64_0	MCRC64_0_dma_event_1	DMASS0_INTAGGR_0_intaggr_levi_pend_29	DMASS0_INTAGGR_0	MCRC64_0 interrupt request	pulse
MCRC64_0	MCRC64_0_dma_event_2	DMASS0_INTAGGR_0_intaggr_levi_pend_30	DMASS0_INTAGGR_0	MCRC64_0 interrupt request	pulse
MCRC64_0	MCRC64_0_dma_event_3	DMASS0_INTAGGR_0_intaggr_levi_pend_31	DMASS0_INTAGGR_0	MCRC64_0 interrupt request	pulse
MCRC64_0	MCRC64_0_int_mcrc_0	DMASS0_INTAGGR_0_intaggr_levi_pend_7	DMASS0_INTAGGR_0	MCRC64_0 interrupt request	level
MCRC64_0	MCRC64_0_int_mcrc_0	GICSS0_spi_166	GICSS0	MCRC64_0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-417. Hardware Requests for MCRC64 (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCRC64_0	MCRC64_0_int_mcrc_0	R5FSS0_CORE0_intr_119	R5FSS0_CORE0	MCRC64_0 interrupt request	level
MCRC64_0	MCRC64_0_int_mcrc_0	WKUP_R5FSS0_CORE0_intr_119	WKUP_R5FSS0_CORE0	MCRC64_0 interrupt request	level
MCRC64_0	MCRC64_0_int_mcrc_0	MCU_R5FSS0_CORE0_cpu0_intr_119	MCU_R5FSS0_CORE0	MCRC64_0 interrupt request	level
MCRC64_0	MCRC64_0_int_mcrc_0	C7X256V0_CLEC_gic_spi_166	C7X256V0_CLEC	MCRC64_0 interrupt request	level
MCRC64_0	MCRC64_0_int_mcrc_0	C7X256V1_CLEC_gic_spi_166	C7X256V1_CLEC	MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	GICSS0_spi_192	GICSS0	MCU_MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	R5FSS0_CORE0_intr_192	R5FSS0_CORE0	MCU_MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	WKUP_R5FSS0_CORE0_intr_192	WKUP_R5FSS0_CORE0	MCU_MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	MCU_R5FSS0_CORE0_cpu0_intr_192	MCU_R5FSS0_CORE0	MCU_MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	C7X256V0_CLEC_gic_spi_192	C7X256V0_CLEC	MCU_MCRC64_0 interrupt request	level
MCU_MCRC64_0	MCU_MCRC64_0_int_mcrc_0	C7X256V1_CLEC_gic_spi_192	C7X256V1_CLEC	MCU_MCRC64_0 interrupt request	level

4.15.4 Programmable Built-In Self-Test (PBIST)

This section contains the integration details for the PBIST module on this device.

4.15.4.1 PBIST Integration

Table 4-418. PBIST Module Allocations

Instance	MAIN	MCU	WKUP
PBIST0	✓		
PBIST1	✓		
WKUP_PBIST0			✓
WKUP_PBIST1			✓
PBIST2	✓		
MCU_PBIST0		✓	

Table 4-419. Integration Attributes for PBIST

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
PBIST0	PSC0	GP_CORE	LPSC_main_pbist0	37	ON	YES	LPSC_main_ip
PBIST1	PSC0	GP_CORE	LPSC_main_pbist1	46	ON	YES	LPSC_main_ip
PBIST2	PSC0	PD_MAIN_MCUSS0	LPSC_main_mcuss0_pbist	80	OFF	YES	LPSC_main_ip
MCU_PBIST0	WKUP_PSC0	PD_MCUSS	LPSC_MCU_PBIST	10	ON	YES	LPSC_MCU_common

Table 4-420. Reset Integration for PBIST

Module Instance	Source	Description
PBIST0	PSC0	PBIST0 reset
PBIST1	PSC0	PBIST1 reset

Copyright © 2026 Texas Instruments Incorporated

Table 4-420. Reset Integration for PBIST (continued)

Module Instance	Source	Description
PBIST2	PSC0	PBIST2 reset
MCU_PBIST0	WKUP_PSC0	MCU_PBIST0 reset

Table 4-421. Hardware Requests for PBIST

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
PBIST0	PBIST0_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event0_228	ESM0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event1_228	ESM0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event2_228	ESM0	PBIST0 interrupt request	pulse
PBIST0	PBIST0_dft_pbist_safety_error_0	ESM0_esm_lvl_event_157	ESM0	PBIST0 interrupt request	level
PBIST1	PBIST1_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event0_229	ESM0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event1_229	ESM0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event2_229	ESM0	PBIST1 interrupt request	pulse
PBIST1	PBIST1_dft_pbist_safety_error_0	ESM0_esm_lvl_event_207	ESM0	PBIST1 interrupt request	level
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	R5FSS0_CORE0_intr_114	R5FSS0_CORE0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_114	WKUP_R5FSS0_CORE0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_114	MCU_R5FSS0_CORE0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event0_234	ESM0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event1_234	ESM0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event2_234	ESM0	WKUP_PBIST0 interrupt request	pulse
WKUP_PBIST0	WKUP_PBIST0_dft_pbist_safety_error_0	ESM0_esm_lvl_event_158	ESM0	WKUP_PBIST0 interrupt request	level
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	R5FSS0_CORE0_intr_114	R5FSS0_CORE0	WKUP_PBIST1 interrupt request	pulse
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_114	WKUP_R5FSS0_CORE0	WKUP_PBIST1 interrupt request	pulse

Copyright © 2026 Texas Instruments Incorporated

Table 4-421. Hardware Requests for PBIST (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_114	MCU_R5FSS0_CORE0	WKUP_PBIST1 interrupt request	pulse
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event0_233	ESM0	WKUP_PBIST1 interrupt request	pulse
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event1_233	ESM0	WKUP_PBIST1 interrupt request	pulse
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_cpu_0	ESM0_esm_pls_event2_233	ESM0	WKUP_PBIST1 interrupt request	pulse
WKUP_PBIST1	WKUP_PBIST1_dft_pbist_safety_error_0	ESM0_esm_lvl_event_159	ESM0	WKUP_PBIST1 interrupt request	level
PBIST2	PBIST2_dft_pbist_cpu_0	R5FSS0_CORE0_intr_113	R5FSS0_CORE0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_113	WKUP_R5FSS0_CORE0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_113	MCU_R5FSS0_CORE0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_cpu_0	ESM0_esm_pls_event0_243	ESM0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_cpu_0	ESM0_esm_pls_event1_243	ESM0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_cpu_0	ESM0_esm_pls_event2_243	ESM0	PBIST2 interrupt request	pulse
PBIST2	PBIST2_dft_pbist_safety_error_0	ESM0_esm_lvl_event_213	ESM0	PBIST2 interrupt request	level
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event0_235	ESM0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event1_235	ESM0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	ESM0_esm_pls_event2_235	ESM0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	R5FSS0_CORE0_intr_149	R5FSS0_CORE0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	WKUP_R5FSS0_CORE0_intr_149	WKUP_R5FSS0_CORE0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_cpu_0	MCU_R5FSS0_CORE0_cpu0_intr_149	MCU_R5FSS0_CORE0	MCU_PBIST0 interrupt request	pulse
MCU_PBIST0	MCU_PBIST0_dft_pbist_safety_error_0	ESM0_esm_lvl_event_152	ESM0	MCU_PBIST0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.15.5 ECC Aggregator (ECC_AGGR)

This section contains the integration details for the ECC_AGGR module on this device. For further information, see the *ECC Aggregator (ECC_AGGR)* section of the *Peripherals* chapter.

4.15.5.1 ECC_AGGR Integration

Table 4-422. ECC_AGGR Module Allocations

Instance	MAIN	MCU	WKUP
ECC_AGGR0	✓		

Table 4-423. Clock Integration for ECC_AGGR

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
ECC_AGGR0	CLK	MAIN_PLL15_HSDIV0_CLKOUT/2		None

Table 4-424. Hardware Requests for ECC_AGGR

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
ECC_AGGR0	ECC_AGGR0_corr_level_0	ESM0_esm_lvl_event_2	ESM0	ECC_AGGR0 interrupt request	level
ECC_AGGR0	ECC_AGGR0_uncorr_level_0	ESM0_esm_lvl_event_1	ESM0	ECC_AGGR0 interrupt request	level

4.15.5.2 MCU_ECC_AGGR0 Integration

Table 4-425. MCU_ECC_AGGR0 Module Allocations

Instance	MAIN	MCU	WKUP
MCU_ECC_AGGR0		✓	

Table 4-426. Clock Integration for MCU_ECC_AGGR0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_ECC_AGGR0	CLK	MCU_SYSCLK0/2		None

Table 4-427. Hardware Requests for MCU_ECC_AGGR0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_ECC_AGGR0	MCU_ECC_AGGR0_corr_level_0	WKUP_ESM0_esm_lvl_event_14	WKUP_ESM0	MCU_ECC_AGGR0 interrupt request	level
MCU_ECC_AGGR0	MCU_ECC_AGGR0_uncorr_level_0	WKUP_ESM0_esm_lvl_event_15	WKUP_ESM0	MCU_ECC_AGGR0 interrupt request	level

4.15.5.3 MCU_ECC_AGGR1 Integration

Table 4-428. MCU_ECC_AGGR1 Module Allocations

Instance	MAIN	MCU	WKUP
MCU_ECC_AGGR1		✓	

Table 4-429. Clock Integration for MCU_ECC_AGGR1

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
MCU_ECC_AGGR1	CLK	MCU_PLL0_HSDIV3_CLKOUT		None

Copyright © 2026 Texas Instruments Incorporated

Table 4-430. Hardware Requests for MCU_ECC_AGGR1

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
MCU_ECC_AGGR1	MCU_ECC_AGGR1_corr_level_0	WKUP_ESM0_esm_lvl_event_40	WKUP_ESM0	MCU_ECC_AGGR1 interrupt request	level
MCU_ECC_AGGR1	MCU_ECC_AGGR1_uncorr_level_0	WKUP_ESM0_esm_lvl_event_41	WKUP_ESM0	MCU_ECC_AGGR1 interrupt request	level

4.15.5.4 WKUP_ECC_AGGR0 Integration

Table 4-431. WKUP_ECC_AGGR0 Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_ECC_AGGR0			✓

Table 4-432. Clock Integration for WKUP_ECC_AGGR0

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_ECC_AGGR0	CLK	DM_CLK/2		None

Table 4-433. Hardware Requests for WKUP_ECC_AGGR0

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ECC_AGGR0	WKUP_ECC_AGGR0_corr_level_0	WKUP_ESM0_esm_lvl_event_23	WKUP_ESM0	WKUP_ECC_AGGR0 interrupt request	level
WKUP_ECC_AGGR0	WKUP_ECC_AGGR0_corr_level_0	ESM0_esm_lvl_event_20	ESM0	WKUP_ECC_AGGR0 interrupt request	level
WKUP_ECC_AGGR0	WKUP_ECC_AGGR0_uncorr_level_0	WKUP_ESM0_esm_lvl_event_24	WKUP_ESM0	WKUP_ECC_AGGR0 interrupt request	level
WKUP_ECC_AGGR0	WKUP_ECC_AGGR0_uncorr_level_0	ESM0_esm_lvl_event_21	ESM0	WKUP_ECC_AGGR0 interrupt request	level

4.15.5.5 WKUP_ECC_AGGR1 Integration

Table 4-434. WKUP_ECC_AGGR1 Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_ECC_AGGR1			✓

Table 4-435. Clock Integration for WKUP_ECC_AGGR1

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_ECC_AGGR1	CLK	MCU_SYSCLK0/4		None

Table 4-436. Hardware Requests for WKUP_ECC_AGGR1

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ECC_AGGR1	WKUP_ECC_AGGR1_corr_level_0	WKUP_ESM0_esm_lvl_event_38	WKUP_ESM0	WKUP_ECC_AGGR1 interrupt request	level
WKUP_ECC_AGGR1	WKUP_ECC_AGGR1_corr_level_0	ESM0_esm_lvl_event_104	ESM0	WKUP_ECC_AGGR1 interrupt request	level
WKUP_ECC_AGGR1	WKUP_ECC_AGGR1_uncorr_level_0	WKUP_ESM0_esm_lvl_event_39	WKUP_ESM0	WKUP_ECC_AGGR1 interrupt request	level
WKUP_ECC_AGGR1	WKUP_ECC_AGGR1_uncorr_level_0	ESM0_esm_lvl_event_105	ESM0	WKUP_ECC_AGGR1 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.15.5.6 WKUP_ECC_AGGR2 Integration

Table 4-437. WKUP_ECC_AGGR2 Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_ECC_AGGR2			✓

Table 4-438. Clock Integration for WKUP_ECC_AGGR2

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
WKUP_ECC_AGGR2	CLK	MCU_SYSCLK0/4		None

Table 4-439. Hardware Requests for WKUP_ECC_AGGR2

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_ECC_AGGR2	WKUP_ECC_AGGR2_corr_level_0	WKUP_ESM0_esm_lvl_event_20	WKUP_ESM0	WKUP_ECC_AGGR2 interrupt request	level
WKUP_ECC_AGGR2	WKUP_ECC_AGGR2_uncorr_level_0	WKUP_ESM0_esm_lvl_event_21	WKUP_ESM0	WKUP_ECC_AGGR2 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

4.16 Display Subsystem (DSS)

This section contains the integration details for the DSS module on this device. For further information, see the *Display Subsystem (DSS)* section of the *Peripherals* chapter.

4.16.1 DSS_UL Unsupported Features

The following features are not supported on this family of devices:

- Simultaneous output to both DPI pipe and either OLDI or DSI from the same DSS pipeline
- Fragmented frame buffers

4.16.2 DSS Integration

Table 4-440. DSS Module Allocations

Instance	MAIN	MCU	WKUP
DSS0	✓		
DSS1	✓		

Table 4-441. Integration Attributes for DSS

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
DSS0	PSC0	PD_DSS	LPSC_main_dss0	91	OFF	YES	LPSC_main_oldi1 LPSC_main_oldi0 LPSC_main_ip
DSS1	PSC0	PD_DSS	LPSC_main_dss1	92	OFF	YES	LPSC_main_oldi1 LPSC_main_ip LPSC_main_dss_dsi0

Table 4-442. Clock Integration for DSS

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DSS0	DPI_0_IN_CLK	MAIN_PLL16_HSDIV0_CLKOUT		pixel clock intput to video pipeline
DSS0	DPI_0_IN_CLK	MAIN_PLL16_HSDIV0_CLKOUT		pixel clock intput to video pipeline
DSS0	DPI_1_IN_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS0_DISPC0_CLKSEL[1:1]=0	pixel clock input to video lite pipeline
DSS0	DPI_1_IN_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_DSS0_DISPC0_CLKSEL[1:1]=1	pixel clock input to video lite pipeline
DSS0	DSS_FUNC_CLK	MAIN_PLL1_HSDIV4_CLKOUT		dss main functional clock for the controller and peripheral interface ports to system interconnect, and all of the display controller (dispc) logic.
DSS1	DPI_0_IN_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=0	pixel clock intput to video pipeline
DSS1	DPI_0_IN_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=1	pixel clock intput to video pipeline

Copyright © 2026 Texas Instruments Incorporated

Table 4-442. Clock Integration for DSS (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DSS1	DPI_0_IN_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=1	pixel clock intput to video pipeline
DSS1	DPI_0_IN_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=0	pixel clock intput to video pipeline
DSS1	DPI_0_IN_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=1	pixel clock intput to video pipeline
DSS1	DPI_0_IN_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=1	pixel clock intput to video pipeline
DSS1	DPI_1_IN_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[1:1]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[18:18]=0	pixel clock input to video lite pipeline
DSS1	DPI_1_IN_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[1:1]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[18:18]=1	pixel clock input to video lite pipeline
DSS1	DPI_1_IN_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[1:1]=1	pixel clock input to video lite pipeline
DSS1	DSS_FUNC_CLK	MAIN_PLL1_HSDIV4_CLKOUT		dss main functional clock for the controller and peripheral interface ports to system interconnect, and all of the display controller (dispc) logic.

Table 4-443. Reset Integration for DSS

Module Instance	Source	Description
DSS0	PSC0	DSS0 reset
DSS1	PSC0	DSS1 reset

Table 4-444. Hardware Requests for DSS

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DSS0	DSS0_dispc_intr_req_0_0	DMASS0_INTAGGR_0_intaggr_levi_pend_2	DMASS0_INTAGGR_0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_0_0	GICSS0_spi_116	GICSS0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_0_0	R5FSS0_CORE0_intr_40	R5FSS0_CORE0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_0_0	WKUP_R5FSS0_CORE0_intr_24	WKUP_R5FSS0_CORE0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_0_0	MCU_R5FSS0_CORE0_cpu0_intr_40	MCU_R5FSS0_CORE0	DSS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-444. Hardware Requests for DSS (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DSS0	DSS0_dispc_intr_req_0_0	C7X256V0_CLEC_gic_spi_116	C7X256V0_CLEC	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_0_0	C7X256V1_CLEC_gic_spi_116	C7X256V1_CLEC	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	DMASS0_INTAGGR_0_intaggr_levi_pend_3	DMASS0_INTAGGR_0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	GICSS0_spi_117	GICSS0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	R5FSS0_CORE0_intr_41	R5FSS0_CORE0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	WKUP_R5FSS0_CORE0_intr_25	WKUP_R5FSS0_CORE0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	MCU_R5FSS0_CORE0_cpu0_intr_41	MCU_R5FSS0_CORE0	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	C7X256V0_CLEC_gic_spi_117	C7X256V0_CLEC	DSS0 interrupt request	level
DSS0	DSS0_dispc_intr_req_1_0	C7X256V1_CLEC_gic_spi_117	C7X256V1_CLEC	DSS0 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	DMASS0_INTAGGR_0_intaggr_levi_pend_4	DMASS0_INTAGGR_0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	GICSS0_spi_62	GICSS0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	R5FSS0_CORE0_intr_92	R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	WKUP_R5FSS0_CORE0_intr_117	WKUP_R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	MCU_R5FSS0_CORE0_cpu0_intr_92	MCU_R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	C7X256V0_CLEC_gic_spi_62	C7X256V0_CLEC	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_0_0	C7X256V1_CLEC_gic_spi_62	C7X256V1_CLEC	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	DMASS0_INTAGGR_0_intaggr_levi_pend_5	DMASS0_INTAGGR_0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	GICSS0_spi_63	GICSS0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	R5FSS0_CORE0_intr_93	R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	WKUP_R5FSS0_CORE0_intr_118	WKUP_R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	MCU_R5FSS0_CORE0_cpu0_intr_93	MCU_R5FSS0_CORE0	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	C7X256V0_CLEC_gic_spi_63	C7X256V0_CLEC	DSS1 interrupt request	level
DSS1	DSS1_dispc_intr_req_1_0	C7X256V1_CLEC_gic_spi_63	C7X256V1_CLEC	DSS1 interrupt request	level

4.16.3 DSS_DSI Integration

Table 4-445. DSS_DSI Module Allocations

Instance	MAIN	MCU	WKUP
DSS_DSI0	✓		

Copyright © 2026 Texas Instruments Incorporated

Table 4-446. Integration Attributes for DSS_DSI

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
DSS_DSI0	PSC0	PD_DSS	LPSC_main_dss_dsi0	93	OFF	YES	LPSC_main_dphy_tx0

Table 4-447. Clock Integration for DSS_DSI

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DSS_DSI0	DPHY_0_RX_ESC_CLK	DPHY_TX0 (INSTANCE)		rx escape clock
DSS_DSI0	DPHY_0_TX_ESC_CLK	MAIN_PLL1_HSDIV6_CLKOUT/6		tx escape clock
DSS_DSI0	DPI_0_CLK	DSS1 (INSTANCE)		dss video port 1 (vp1) pixel clock for the output main_pll17_hsdiv0_c interface
DSS_DSI0	PLL_CTRL_CLK	MAIN_SYSCLK0		None
DSS_DSI0	DPHY_0_TX_BYTE_HS_CLK	DPHY_TX0 (INSTANCE)		dphy ppi byte clock frequency set by the dphy and its high speed input clock (tx_byte_hs_clk = dphy bit rate /8) maximum limit due to risk of underrun

Table 4-448. Reset Integration for DSS_DSI

Module Instance	Source	Description
DSS_DSI0	PSC0	DSS_DSI0 reset

Table 4-449. Hardware Requests for DSS_DSI

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	GICSS0_spi_118	GICSS0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	R5FSS0_CORE0_intr_218	R5FSS0_CORE0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	WKUP_R5FSS0_CORE0_intr_218	WKUP_R5FSS0_CORE0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	MCU_R5FSS0_CORE0_cpu0_intr_218	MCU_R5FSS0_CORE0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	C7X256V0_CLEC_gic_spi_118	C7X256V0_CLEC	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_func_intr_0	C7X256V1_CLEC_gic_spi_118	C7X256V1_CLEC	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_safety_error_fatal_intr_0	ESM0_esm_lvl_event_178	ESM0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_dsi_0_safety_error_nonfatal_intr_0	ESM0_esm_lvl_event_179	ESM0	DSS_DSI0 interrupt request	level
DSS_DSI0	DSS_DSI0_ecc_intr_uncorr_level_sys_0	ESM0_esm_lvl_event_142	ESM0	DSS_DSI0 interrupt request	level

4.16.4 OLDI_TX_CORE

This section contains the integration details for the OLDI_TX_CORE module on this device. For further information, see the *OLDITX Functional Description* in *Display Subsystem (DSS)* section of the *Peripherals* chapter.

Copyright © 2026 Texas Instruments Incorporated

4.16.4.1 oldi_tx_core Integration

Table 4-450. OLDI_TX_CORE Module Allocations

Instance	MAIN	MCU	WKUP
oldi_tx_core0	✓		
oldi_tx_core1	✓		

Table 4-451. Clock Integration for oldi_tx_core

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
oldi_tx_core0	OLDI_0_FWD_P_CLK	MAIN_PLL16_HSDIV0_CLKOUT		pixel clock looped back at soc
oldi_tx_core0	OLDI_0_FWD_P_CLK	MAIN_PLL16_HSDIV0_CLKOUT		pixel clock looped back at soc
oldi_tx_core0	OLDI_PLL_CLK	MAIN_PLL16_HSDIV0_CLKOUT		main clock from pll
oldi_tx_core0	OLDI_PLL_CLK	MAIN_PLL16_HSDIV0_CLKOUT		main clock from pll
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL16_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=0	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL16_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=0	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=0	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=1	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=1	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=0	pixel clock looped back at soc

Copyright © 2026 Texas Instruments Incorporated

Table 4-451. Clock Integration for oldi_tx_core (continued)

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
oldi_tx_core1	OLDI_0_FWD_P_CLK	MAIN_PLL17_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=0 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[16:16]=1	pixel clock looped back at soc
oldi_tx_core1	OLDI_0_FWD_P_CLK	VOUT0_EXTPCLKIN (PIN)	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1 MAIN_CTRL_MMR_CFG0_DSS1_DISPC0_CLKSEL[0:0]=1	pixel clock looped back at soc
oldi_tx_core1	OLDI_PLL_CLK	MAIN_PLL16_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=0	main clock from pll
oldi_tx_core1	OLDI_PLL_CLK	MAIN_PLL16_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=0	main clock from pll
oldi_tx_core1	OLDI_PLL_CLK	MAIN_PLL18_HSDIV0_CLKOUT	MAIN_CTRL_MMR_CFG0_OLDI1_CLKSEL[0:0]=1	main clock from pll

4.17 On-Chip Debug

This section contains the integration details for the On-Chip Debug module on this device. For further information, see the *On-Chip Debug* chapter.

4.17.1 DEBUGSS_WRAP Integration

Table 4-452. DEBUGSS_WRAP Module Allocations

Instance	MAIN	MCU	WKUP
DEBUGSS_WRAP0	✓		

Table 4-453. Clock Integration for DEBUGSS_WRAP

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
DEBUGSS_WRAP0	ATB_CLK	MAIN_SYSCLK0/2		atb clock

4.17.2 debugss Integration

Table 4-454. DEBUGSS Module Allocations

Instance	MAIN	MCU	WKUP
DEBUGSS0	✓		

Table 4-455. Hardware Requests for debugss

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	GICSS0_spi_201	GICSS0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	R5FSS0_CORE0_intr_201	R5FSS0_CORE0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	WKUP_R5FSS0_CORE0_intr_201	WKUP_R5FSS0_CORE0	DEBUGSS0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-455. Hardware Requests for debugss (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	MCU_R5FSS0_CORE0_cpu0_intr_201	MCU_R5FSS0_CORE0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	C7X256V0_CLEC_gic_spi_201	C7X256V0_CLEC	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_aqcmpintr_level_0	C7X256V1_CLEC_gic_spi_201	C7X256V1_CLEC	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	GICSS0_spi_202	GICSS0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	R5FSS0_CORE0_intr_202	R5FSS0_CORE0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	WKUP_R5FSS0_CORE0_intr_202	WKUP_R5FSS0_CORE0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	MCU_R5FSS0_CORE0_cpu0_intr_202	MCU_R5FSS0_CORE0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	C7X256V0_CLEC_gic_spi_202	C7X256V0_CLEC	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_ctm_level_0	C7X256V1_CLEC_gic_spi_202	C7X256V1_CLEC	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_davdma_level_0	DMASS0_INTAGGR_0_intaggr_levi_pend_27	DMASS0_INTAGGR_0	DEBUGSS0 interrupt request	level
DEBUGSS0	DEBUGSS0_davdma_level_0	USB0_trace_inep_pkt_buff_avail_0	USB0	DEBUGSS0 interrupt request	level

4.17.3 dbgsuspendrouter Integration

Table 4-456. DBGSUSPENDROUTER Module Allocations

Instance	MAIN	MCU	WKUP
DBGSUSPENDROUTER0	✓		

Table 4-457. Integration Attributes for dbgsuspendrouter

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
DBGSUSPENDROUTER0	PSC0	GP_CORE	LPSC_main_debugss	33	ON	YES	LPSC_main_ip

Table 4-458. Reset Integration for dbgsuspendrouter

Module Instance	Source	Description
DBGSUSPENDROUTER0	PSC0	DBGSUSPENDROUTER0 reset

4.17.4 icemelter Integration

Table 4-459. ICEMELTER Module Allocations

Instance	MAIN	MCU	WKUP
WKUP_icemelter0			✓

Table 4-460. Hardware Requests for icemelter

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_icemelter0	WKUP_icemelter0_psc_force_power_on_0	WKUP_R5FSS0_CORE0_intr_23	WKUP_R5FSS0_CORE0	WKUP_icemelter0 interrupt request	level

Copyright © 2026 Texas Instruments Incorporated

Table 4-460. Hardware Requests for icemelter (continued)

Module Instance	Module Interrupt Signal	Destination Interrupt Input	Destination	Description	Type
WKUP_icemelter0	WKUP_icemelter0_psc_force_power_on_0	WKUP_DEEPSLEEP_SOURCES0_lsam62_dm_wakeup_deepsleep_sources_3	WKUP_DEEPSLEEP_SOURCES0	WKUP_icemelter0 interrupt request	level

4.17.5 STM Integration

Table 4-461. STM Module Allocations

Instance	MAIN	MCU	WKUP
STM0	✓		

Table 4-462. Integration Attributes for STM

Module Instance	Power Sleep Controller	Power Domain	Module Domain	Index	Default	Controllable	Dependencies
STM0	PSC0	GP_CORE	LPSC_main_debugss	33	ON	YES	LPSC_main_ip

Table 4-463. Reset Integration for STM

Module Instance	Source	Description
STM0	PSC0	STM0 reset

4.17.6 CTI Integration

Table 4-464. CTI Module Allocations

Instance	MAIN	MCU	WKUP
CTI0	✓		
CTI1	✓		

Table 4-465. Clock Integration for CTI

Module Instance	Module Clock Input	Source Clock	Source Control Register	Description
CTI0	DBG_CLK	MAIN_SYSCLK0/4		None
CTI1	DBG_CLK	MAIN_SYSCLK0/4		None

4.17.7 k3_arm_atb_funnel_3_32 Integration

Table 4-466. K3_ARM_ATB_FUNNEL_3_32 Module Allocations

Instance	MAIN	MCU	WKUP
k3_arm_atb_funnel_3_32_0	✓		
k3_arm_atb_funnel_3_32_1	✓		

4.17.8 sam67_debug_funnel_2_32 Integration

Table 4-467. SAM67_DEBUG_FUNNEL_2_32 Module Allocations

Instance	MAIN	MCU	WKUP
sam67_debug_funnel_2_32_0	✓		

4.17.9 sam67_debug_funnel_2_64 Integration

Table 4-468. SAM67_DEBUG_FUNNEL_2_64 Module Allocations

Instance	MAIN	MCU	WKUP
sam67_debug_funnel_2_64_0	✓		

Copyright © 2026 Texas Instruments Incorporated

Chapter 5

Initialization

This chapter describes the steps for non-secure device initialization.

5.1 Initialization Overview 353
5.2 Boot Process 356
5.3 Boot Mode Pins 361
5.4 Boot Modes 363
5.5 PLL Configuration 393
5.6 Boot Parameter Tables 395
5.7 Boot Image Format 405
5.8 Boot Memory Maps 412

Copyright © 2026 Texas Instruments Incorporated

5.1 Initialization Overview

Figure 5-1 is an overview of the initialization process and its steps:

- **Preinitialization:** Power, clock, and control connections must be present, and the boot configuration pins must be held at the desired logical levels.
- **Power, clock, reset ramp sequence:** Specific sequence that is applied by the power-management chip(s)
- **ROM code:** Responsible for finding, for downloading, and for executing the initial software (SBL)
- **Initial software:** Software that loads, prepares, and passes control to application software or to the high-level operating system (HLOS)
- **High-Level Operating System** or bare-metal application which runs on main processor(s)



Figure 5-1. Initialization Process

The first two steps in the initialization process are hardware-oriented; however, they require an understanding of the process of configuring these system interface pins (balls on the device), which have software-configurable functionality. This configuration is an essential part of the chip configuration and is application-dependent. This chapter discusses these system-interface pins, the associated configuration registers, and memory structures that are vital to the correct initialization of the device.

5.1.1 ROM Code Overview

ROM bootloader (or ROM Code) is a software that resides in a on-chip read-only memory (ROM) to assist the customer in transferring and executing their application code. The device has two ROM codes operating in tandem – the Public ROM code, and the M4 ROM code.

In order to accommodate various system scenarios, the ROM Code supports several boot modes. These boot modes can be broadly classified as:

- Host boot modes
- Memory boot modes.

During a host boot, the device is configured to receive code from a host via the selected interface. Either the host writes the application code directly into internal memory or the ROM Code receives the application code on the selected interface and stores it in internal memory.

During a memory boot, the device transfers code from non-volatile memory to internal memory for execution.

In all boot modes, the entire boot operation can be partitioned into two sections:

1. Hardware initialization phase
2. Boot process.

During initialization, the ROM Code configures the device resources (PLLs, peripherals, pins) as needed to support the boot process. The resources used depend on the boot mode requirements.

During the boot process the boot image can be loaded into device memory and executed, or executed in place, depending on the boot peripheral. M4 will perform code verification and allow, or forbid, the image execution.

Main configuration source for boot after power-up are the BOOTMODE pins sampled automatically after reset release and stored in device status registers. At ROM Code startup, these pin values are read from the registers to create the boot peripheral list and the boot configuration tables used later to initialize and startup the PLLs and boot peripherals.

Copyright © 2026 Texas Instruments Incorporated

5.1.2 Bootloader Modes

Table 5-1 shows the boot modes supported by ROM code.

Table 5-1. ROM Code Boot Modes

Boot Mode	Boot Media/Host	SoC Peripheral	Can be a Backup Mode?(1)	Notes
No-boot/Dev-boot	No media or host	None	N	No boot or development boot – debug modes
OSPI	OSPI flash	FSS0_OSPI0	N	On OSPI port
QSPI	QSPI flash	FSS0_OSPI0	N	On OSPI port
SPI	SPI flash	FSS0_OSPI0	Y	On OSPI port
Ethernet	External host	CPSW0	Y	In BOOTP mode. RGMII or RMII PHY
I2C	I^2C EEPROM	I2C0	Y	I2C target boot is not supported
UART	External host	UART0	Y	XMODEM protocol
MMCSD	MMCSD card	MMCSD0 (8bit) or MMCSD1 (4bit)	Y	Boot from User Data Area (UDA) or file system
eMMC	eMMC flash	MMCSD0 (8bit)	N	Boot from boot partition
USB - target	USB boot from external host	USB0	Y	USB device mode boot using DFU (device firmware upgrade), Boot is running on USB2.0 speeds.
USB - host	USB mass storage	USB0	Y	USB2.0 host mode, boot from FAT32 filesystem
Serial Flash	Serial Flash	FSS0_OSPI0	N	On OSPI port
xSPI	xSPI flash	FSS0_OSPI0	N	On OSPI port
Fast-xSPI	Fast-xSPI	FSS0_OSPI0	N	On OSPI port
GPMC	NOR flash, NAND flash	GPMC0	N	CSn0 connected, 8 bit NAND flash only, 16-bit non-mux NOR flash only

(1) The peripheral can be selected also as a backup boot mode. A backup mode is tried if primary boot mode fails.

Note

Because different devices support different sets of peripherals, see the *device-specific Datasheet* to obtain the list of peripherals supported in your device.

5.1.3 Terminology

- **Boot Mode Pins:** Boot mode pins provide vital information to ROM code for boot. These pins must be properly set up before power ramp.
- **Bootstrap:** Initial software launched by the ROM code during the memory booting phase.
- **Boot Header:** Optional structure that precedes the initial software and allows the redefinition of the ROM code default settings.
- **Downloaded software:** Initial software downloaded into on-chip RAM by the ROM code during the peripheral booting phase.
- **eFuse:** A one-time programmable memory location usually set at the factory.
- **Flash loader:** Downloaded software launched by the ROM code during the preflashing stage. It also programs an image in external memories.
- **GP device:** General-purpose device (SoC) or a non-secure device.
- **Initial software:** Software executed by any of the ROM code mechanisms (memory booting or peripheral booting). Initial software is a generic term for bootstrap and downloaded software. This can be the SBL (secondary bootloader) responsible for loading an OS.
- **Memory booting:** ROM code mechanism that consists of executing initial software from external memory.

Copyright © 2026 Texas Instruments Incorporated

- **Controller CPU:** The Arm® Cortex® CPU for which CPU-ID is 0. It configures the multicore platform and starts the ROM code to ensure device booting from a mass storage memory (memory booting) or a peripheral interface (peripheral booting).
- **Peripheral booting:** ROM code mechanism that consists of polling selected interfaces, downloading, and executing initial software (in this case, downloaded software) in the internal RAM.
- **Preflashing:** A specific case of peripheral booting where the ROM code mechanism is used to program the external flash memory.
- **ROM Code:** or ROM bootloader (RBL), the on-chip software in device ROM that executes first and implements booting.
- **ROM Code-controlled Boot Phase:** This phase covers the sequence operations from the time the platform releases the reset to the time first user- or customer-owned software starts execution. This phase is fully controlled by the device ROM code.
- **Booting Parameter Table:** A logical structure stored in the on-chip RAM memory and contains information for the boot, such as the boot file name or an address to boot from.

Copyright © 2026 Texas Instruments Incorporated

5.2 Boot Process

5.2.1 Public ROM Code Architecture

The Public ROM code has the following components (see also Public ROM Code Architecture):

- Main
- Buffer manager
- X.509
- Log and Trace
- System
- Protocol
- Driver

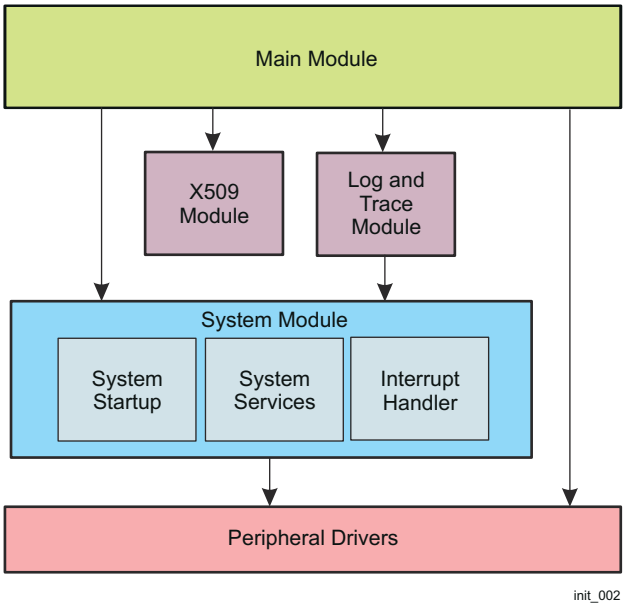


Figure 5-2. Public ROM Code Architecture

5.2.1.1 Main Module

The Main module contains the top level execution loop. This loop repeats until a boot image has been received or directed to sleep by the M4. The main loop has three different execution sub-paths based on the boot peripheral.

- **Image Path** This path is used by the OSPI, QSPI, SPI, and xSPI boot modes. In these cases the image data can be directly read by both the R5 and the M4 in place.
- **Block Path** This path is used by the I2C, USB-DFU, UART, eMMC, Ethernet, eMMC/SD cards in raw mode. In this mode data is received from the peripheral in blocks. Blocks are accumulated in the boot buffer until a full X.509 certificate header has been received, at which point this full certificate and any subsequent blocks are passed to the M4 as they arrive.
- **Filesystem Path** This path is used by the USB host (MSC) mode and eMMC/SD cards in filesystem mode. This mode executes exactly like in the block path, except that the boot image location is defined by a filesystem.

The main level is able to detect if a received boot image is in the correct format and reject non-conforming images. In some cases, detection of an invalid image is done after the initial data buffer has been sent to the M4. In that case the R5 must inform the M4 that the image was in fact invalid and it will be looking for the next image and try again.

Copyright © 2026 Texas Instruments Incorporated

5.2.1.2 X509 Module

The X509 module parses the boot header. The boot header is an X.509 certificate as defined in RFC5280. Extensions specific to boot are described in Section 5.7.2, *X.509 Certificate*.

This module also includes an OID decoder as well as defines OID values as C #define constants for any values that can be used during boot.

5.2.1.3 Buffer Manager Module

The buffer manager module is used to allocate buffers to hold boot data. The R5 code allocates a buffer when it is ready to read a block of data from the boot peripheral. Once the data is read the buffer ownership is passed to the M4. When the M4 has completed processing the data in the buffer ownership is returned to the R5, which then frees the buffer.

5.2.1.4 Log and Trace Module

The Log and Trace modules are not integral to the boot process. Instead, they provide ability to record operation of the ROM code and track unexpected occurrences.

Log and trace function is currently TI internal.

5.2.1.5 System Module

The system module provides services to other modules. These services are not directly related to boot drivers. The system module is the only module that operates in the supervisor mode of the R5 (a few services will run in user). The system module supports the following functions:

- Interrupt enable, disable, and service
- IPC
- Power
- Pinmux
- Clock
- Task switch

The main level is able to detect if a received boot image is in the correct format and reject non-conforming images.

5.2.1.6 Protocol Module

The protocol modules provide implementation of high-level data transfer protocols. The BOOTP/TFTP and XMODEM protocols reside in this layer. These modules provide services as defined in well-known standards.

5.2.1.7 Driver Module

The driver module implements the low level peripheral drivers.

5.2.2 M4 ROM Description

In a general-purpose (GP) device, M4 ROM performs the following functions:

- Device management
- Configures the boot vectors (in BOOT_CFG) and controls reset release of R5 core. That is, M4 is the boot controller of R5 core.
- IPC configuration via Main DMSS rings and Secure Proxy
- PLL configuration (R5 and SA2UL)
- X509 certificate parsing
- SA2UL configuration to SHA512 for image integrity checks
- M4 firmware loading

5.2.3 Boot Process Flow

The R5 boot process flow is shown in Figure 5-3.

Copyright © 2026 Texas Instruments Incorporated

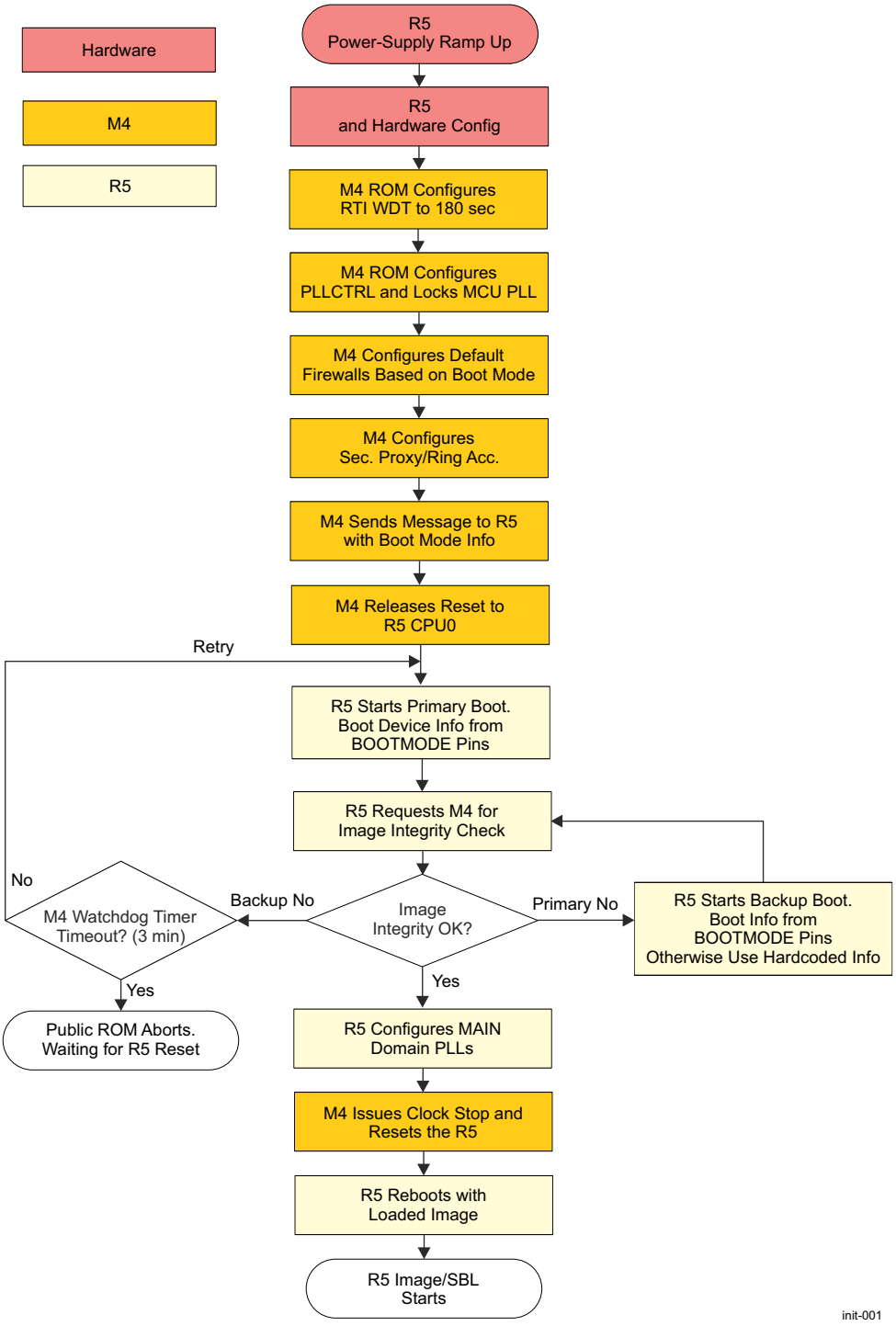


Figure 5-3. Boot Process

The values of BOOTMODE[15:0] pins are latched into the Device Status register CTRLMMR_MAIN_DEVSTAT[15:0] by hardware as the device comes out of global cold reset, sampled after MCU_PORz deassertion. For more information how to set BOOTMODE pins, see Section 5.3, *Boot Mode Pins*.

The M4 is the boot controller for the Public ROM. M4 performs the necessary configurations and releases R5's reset for CPU0.

Copyright © 2026 Texas Instruments Incorporated

The R5 checks the boot mode pins and then configures the apropriate peripheral interface to get access to a boot image. A cursory check of the image is made, and the image is passed to M4. M4 ROM then will perform code verification and route the boot image to the on-chip RAM. Once the image has been received, R5 enters a clean state and idles. M4 ROM code will assert reset to the R5, redirect the boot vector to the newly loaded image, and release the reset. This restarts the R5 with the Public ROM code fully disconnected.

The Public ROM code executes after a cold or warm reset.

Note

M4 ROM sets up a 3-minute watchdog timer (RTI0) timeout. During this time, the R5 boot needs to get completed, otherwise a WDT reset will occur. Once the R5 image is loaded (SBL/SPL), M4 ROM will restart the watchdog timer for additional 3 minutes upon entering the R5 SBL. The customer-provided R5 image needs to load and install the TI-provided SYSFW image into the M4, which will manage the watchdog timer during run time.

Note

The following system conditions must be met at POR to perform device boot:

- USB cable plug must be inserted
- Ethernet PHY is powered up and out of reset
- The SD card cage must be powered before entering the SD card boot mode. A SD/MMC card with pre-loaded image must be inserted
- Memory devices must be up and ready (power ramped up and reset completed) at device startup:
  - eMMC
  - xSPI/OSPI/QSPI/SPI and Serial flash
  - I2C EEPROM
  - NOR/NAND flash

Failing to meet these requirements may result in boot fail and performing a backup boot (if available for that mode).

Figure 5-4 describes the external bootloader (SBL) typical tasks.

Copyright © 2026 Texas Instruments Incorporated

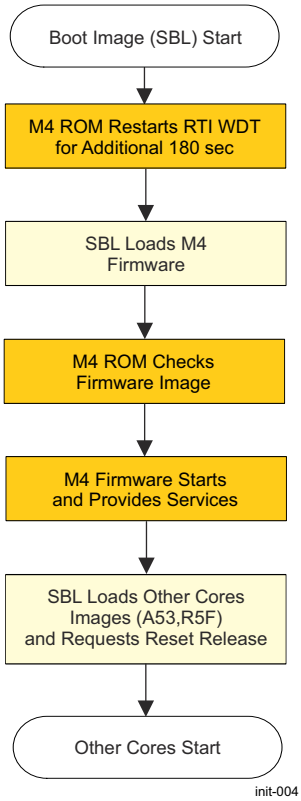


Figure 5-4. External Bootloader Tasks

Upon R5 reset and SBL execution start, M4 ROM restarts the RTI watchdog timer for additional 180 seconds of timeout. During that time, SBL must load the M4 firmware provided by TI otherwise an R5 reset will occur as a preventive measure against software misbehavior.

One of the SBL's main tasks is to load the M4 firmware. Only after this task is performed, SBL can load the other processors' image and request a reset release from M4 firmware for those cores.

Copyright © 2026 Texas Instruments Incorporated

5.3 Boot Mode Pins

Boot Mode pins provide means to select the boot mode and options before the device is powered up. After every POR, they are the main source to populate the Boot Parameter Tables. See Section 5.6, *Boot Parameter Tables* for table list and description.

Boot mode pins can be divided into the following categories:

- **BOOTMODE[02:00]** – Denote system clock frequency (MCU_OSC0_XI/XO) to ROM code for PLL configuration.
- **BOOTMODE[06:03]** – Select the requested boot (primary) mode after POR, that is, the peripheral/memory to boot from.
- **BOOTMODE[09:07]** – These pins provide optional configurations for primary boot and are used in conjunction with the boot mode selected.
- **BOOTMODE[12:10]** – Select the backup boot mode, that is, the peripheral/memory to boot from, if primary boot device failed.
- **BOOTMODE[13]** – This pin provides optional configurations for the backup boot devices.
- **BOOTMODE[15:14]** – Reserved pins.

Note

It is user's responsibility to set the boot mode pins (via pullups or pulldowns, and jumpers/switches) depending on the desired boot scenario.

Copyright © 2026 Texas Instruments Incorporated

5.3.1 BOOTMODE Pin Mapping

The ROM execution is directed through the main boot mode pins. This provides more flexibility and more booting peripherals to boot from. The Main domain must be powered and functional.

Main boot mode pins are shown in Table 5-2.

Table 5-2. BOOTMODE Pin Mapping

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0	0	Backup Boot Mode Config	Backup Boot Mode			Primary Boot Mode Config			Primary Boot Mode				PLL Config		

Table 5-3 describes the BOOTMODE pins that need to be set according to the system clock provided to the device.

The ROM Code will configure any PLLs required during the boot process.

Note

Reference clock selections may be limited by the device. Please consult the device specific Datasheet to determine which system clock frequencies are valid.

Table 5-3. PLL Reference Clock Selection

PLL Config Pins			Ref Clock (MHz)
B2	B1	B0	
0	1	0	Reserved
0	1	1	25
1	0	0	Reserved

5.3.1.1 Primary Boot Mode Selection and Configuration

The primary boot mode is the first mode attempted after reset. Table 5-4 lists all possible primary boot modes.

Note

All BOOTMODE[15:00] signals must be pulled high through a resistor to VDDSHV3, or pulled low to ground, including Reserved signals. Reserved BOOTMODE signals are recommended to be pulled low.

Table 5-4. Primary Boot Mode Selection

Primary Boot Mode Config							Primary Boot Mode
B9	B8	B7	B6	B5	B4	B3	
Reserved	Read Mode 2	Read Mode 1	0	0	0	0	Serial NAND
Reserved	Reserved	Csel	0	0	0	1	OSPI
Reserved	Reserved	Csel	0	0	1	0	QSPI
Reserved	Mode	Csel	0	0	1	1	SPI
0	0	Link Info	0	1	0	0	Ethernet RGMII
Clkout	Clk src	0	0	1	0	1	Ethernet RMII
Bus reset	Reserved	Addr	0	1	1	0	I2C
Reserved	Reserved	0	0	1	1	1	UART
Port	Reserved	Fs/raw	1	0	0	0	MMCSD Boot (SD Card Boot or)
Reserved	Reserved	Reserved	1	0	0	1	
0	Mode	Lane Swap	1	0	1	0	USB
Reserved	Reserved	Reserved	1	0	1	1	GPMC NAND

Copyright © 2026 Texas Instruments Incorporated

Table 5-4. Primary Boot Mode Selection (continued)

Primary Boot Mode Config							Primary Boot Mode
B9	B8	B7	B6	B5	B4	B3	
Reserved	Reserved	Reserved	1	1	0	0	GPMC NOR
Reserved	Reserved	Reserved	1	1	0	1	Fast-xSPI
SFDP	Read Cmd	Mode	1	1	1	0	xSPI
Reserved	ARM/Thumb	No/Dev	1	1	1	1	No-boot/Dev boot

5.3.1.2 Backup Boot Mode Selection and Configuration

The backup boot mode is selected via pins within the main BOOTMODE map. Table 5-5 lists all possible backup boot modes.

Table 5-5. Backup Mode Selection

Backup Boot Config	Backup Boot Mode Selection			Backup Boot Mode Selected
B13	B12	B11	B10	
Reserved	0	0	0	None
Mode	0	0	1	USB
Reserved	0	1	0	Reserved
0	0	1	1	UART
IF	1	0	0	Ethernet
Port	1	0	1	MMCSD Boot (SD Card Boot or eMMC Boot)
Reserved	1	1	0	SPI
Reserved	1	1	1	I2C

5.4 Boot Modes

Copyright © 2026 Texas Instruments Incorporated

5.4.1 Boot with OSPI Controller

The following apply to all or multiple boot modes that are SPI related.

- Octal SPI flash memories support various protocols, however, the OSPI boot mode of the device will only support a specific protocol defined in OSPI Bootloader Operation. If the flash memory is complaint with JEDEC xSPI standards JESD251 and JESD216D, then xSPI boot mode is additionally supported. Please refer to xSPI boot mode description for further details.
- Command protocols follow the JEDEC spec definition and indicate the number of active pins used for the instruction, address, and data, and also the data rate used for each (S = Single Data Rate, D= Double Data Rate). For example, 1S-1S-8S describes a protocol which uses 1 signal (D0) for command in SDR mode, 1 signal (D0) for address in SDR mode, and 8 signals (D7:D0) for data in SDR mode
- When using a OSPI\xSPI\QSPI\SPI flash device greater than 128Mb which supports 3-byte and 4-byte addressing modes, a flash device package with a RESET signal must be used. The reason is that the ROM only uses 3 byte addressing mode (address is 24bits). To address the full memory address range, software will typically switch to 4-byte addressing mode. If a reset to the processor occurs (eg, due to a warm reset), the ROM will execute expecting 3-byte addressing mode, but the flash will have been left in 4-byte addressing mode. In order for the flash device to return to 3-byte addressing mode, it must be reset using this signal. This typically can be achieved by using the RESET signal on the flash memory device. The ROM does not issue a software reset command.

Copyright © 2026 Texas Instruments Incorporated

5.4.1.1 OSPI Boot

Table 5-6 shows configuration pins assignment to functions when boot mode is the Octal SPI using the OSPI module.

Table 5-6. OSPI Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
7	Csel	0	Boot Flash is on CS 0
		1	Boot Flash is on CS 1

Table 5-7 summarizes the OSPI pin configuration done by ROM code for OSPI boot device.

Table 5-7. OSPI Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
OSPI0_CLK	OSPI0_CLK	Disable	NA	0	Disable	0	PADCONFIG0
OSPI0_DQS	OSPI0_DQS	Disable	NA	0	Enable	0	PADCONFIG2
OSPI0_D0	OSPI0_D0	Disable	NA	0	Enable	0	PADCONFIG3
OSPI0_D1	OSPI0_D1	Disable	NA	0	Enable	0	PADCONFIG4
OSPI0_D2	OSPI0_D2	Disable	NA	0	Enable	0	PADCONFIG5
OSPI0_D3	OSPI0_D3	Disable	NA	0	Enable	0	PADCONFIG6
OSPI0_D4	OSPI0_D4	Disable	NA	0	Enable	0	PADCONFIG7
OSPI0_D5	OSPI0_D5	Disable	NA	0	Enable	0	PADCONFIG8
OSPI0_D6	OSPI0_D6	Disable	NA	0	Enable	0	PADCONFIG9
OSPI0_D7	OSPI0_D7	Disable	NA	0	Enable	0	PADCONFIG10
OSPI0_CSn0(1)	OSPI0_CSn0	Disable	NA	0	Disable	0	PADCONFIG11
OSPI0_CSn1(1)	OSPI0_CSn1	Disable	NA	0	Disable	0	PADCONFIG12

(1) OSPI0_CSn0 and OSPI0_CSn1 are configured as chip select outputs from the OSPI0 host controller, but OSPI boot only uses the chip select output selected via BOOTMODE[7].

Note

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.1.1.1 OSPI Bootloader Operation

Please refer to Section 5.4.1 for more information that applies to OSPI boot mode

The OSPI boot mode supports 1S-1S-8S mode only. The ROM will issue a Read Command (0x8B) followed by a 24 bit (3 byte) address (the starting address is all zeros), followed by 8 dummy cycles. The flash device should then respond with 8-bit data. The frequency of operation during the read portion is 50 MHz.

5.4.1.1.1.1 OSPI Initialization Process

In the OSPI boot mode, the ROM Code initializes the OSPI module and the image is read from the OSPI flash connected to the selected chip-select. If the image fails to be read correctly from offset 0x0 of the flash memory, the ROM will attempt to obtain the image at offset 0x400000. This is the only redundant image location supported by the ROM. See Section 5.4.1.1, *OSPI Boot Device Configuration* for OSPI port settings.

A detailed summary of the OSPI boot parameter table and the boot configuration definitions are listed in Section 5.6.3, *OSPI Boot Parameter Table*.

5.4.1.1.1.2 OSPI Loading Process

OSPI boot mode is not eXecute-In-Place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

Copyright © 2026 Texas Instruments Incorporated

5.4.1.2 xSPI Boot

Table 5-8 shows configuration pins assignment to functions when boot mode is xSPI using the OSPI module.

Table 5-8. xSPI Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
9	SFDP	0	SFDP disabled
		1	SFDP enabled
8	Read cmd	0	0x0B Read Command
		1	0xEE Read Command
7	Mode	0	1S-1S-1S mode @ 50MHz
		1	8D-8D-8D mode @ 25MHz

Table 5-9 summarizes the OSPI pin configuration done by ROM code for xSPI boot device on port 0.

Table 5-9. xSPI Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
OSPI0_CLK	OSPI0_CLK	Disable	NA	0	Disable	0	PADCONFIG0
OSPI0_DQS	OSPI0_DQS	Disable	NA	0	Enable	0	PADCONFIG2
OSPI0_D0	OSPI0_D0	Disable	NA	0	Enable	0	PADCONFIG3
OSPI0_D1	OSPI0_D1	Disable	NA	0	Enable	0	PADCONFIG4
OSPI0_D2	OSPI0_D2	Disable	NA	0	Enable	0	PADCONFIG5
OSPI0_D3	OSPI0_D3	Disable	NA	0	Enable	0	PADCONFIG6
OSPI0_D4	OSPI0_D4	Disable	NA	0	Enable	0	PADCONFIG7
OSPI0_D5	OSPI0_D5	Disable	NA	0	Enable	0	PADCONFIG8
OSPI0_D6	OSPI0_D6	Disable	NA	0	Enable	0	PADCONFIG9
OSPI0_D7	OSPI0_D7	Disable	NA	0	Enable	0	PADCONFIG10
OSPI0_CSn0(1)	OSPI0_CSn0	Disable	NA	0	Disable	0	PADCONFIG11
OSPI0_CSn1(1)	OSPI0_CSn1	Disable	NA	0	Disable	0	PADCONFIG12

(1) Both OSPI0_CSn0 and OSPI0_CSn1 are configured as chip select outputs from the OSPI0 host controller, but xSPI boot only uses OSPI0_CSn0. It is not possible to select OSPI0_CSn1 as a chip select option for xSPI boot.

Note

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.1.2.1 xSPI Bootloader Operation

Please refer to Section 5.4.1 for more information that applies to xSPI boot mode

The xSPI protocol defines 1S-1S-1S mode for general backwards compatibility, and 8D-8D-8D for maximum throughput.

For 1S-1S-1S mode of operation, the ROM will issue a Fast Read Output command (0x0B) followed by a 24 bit (3 byte) address (the starting address is all zeros), followed by 8 dummy cycles. The frequency of operation is 50 MHz.

For 8D-8D-8D mode of operation, the ROM will issue a Fast Read command (0x0B or 0xEE, depending on BOOTMODE signal), followed by a 32 bit (4 byte) address (the starting address is all zeros). The frequency of operation is 25 MHz.

When SFDP is enabled using a BOOTMODE signal, the ROM starts operation in 1S-1S-1S mode reads SFDP header from flash memory to get 8D-8D-8D switching sequence, Read Command, CMD Extension and Byte

Copyright © 2026 Texas Instruments Incorporated

Order. SFDP parsing of ROM is described below and on successful parsing ROM will issue 8D-8D-8D command switching sequence and then will read the boot image in 8D-8D-8D mode with read command specified in SFDP header

Copyright © 2026 Texas Instruments Incorporated

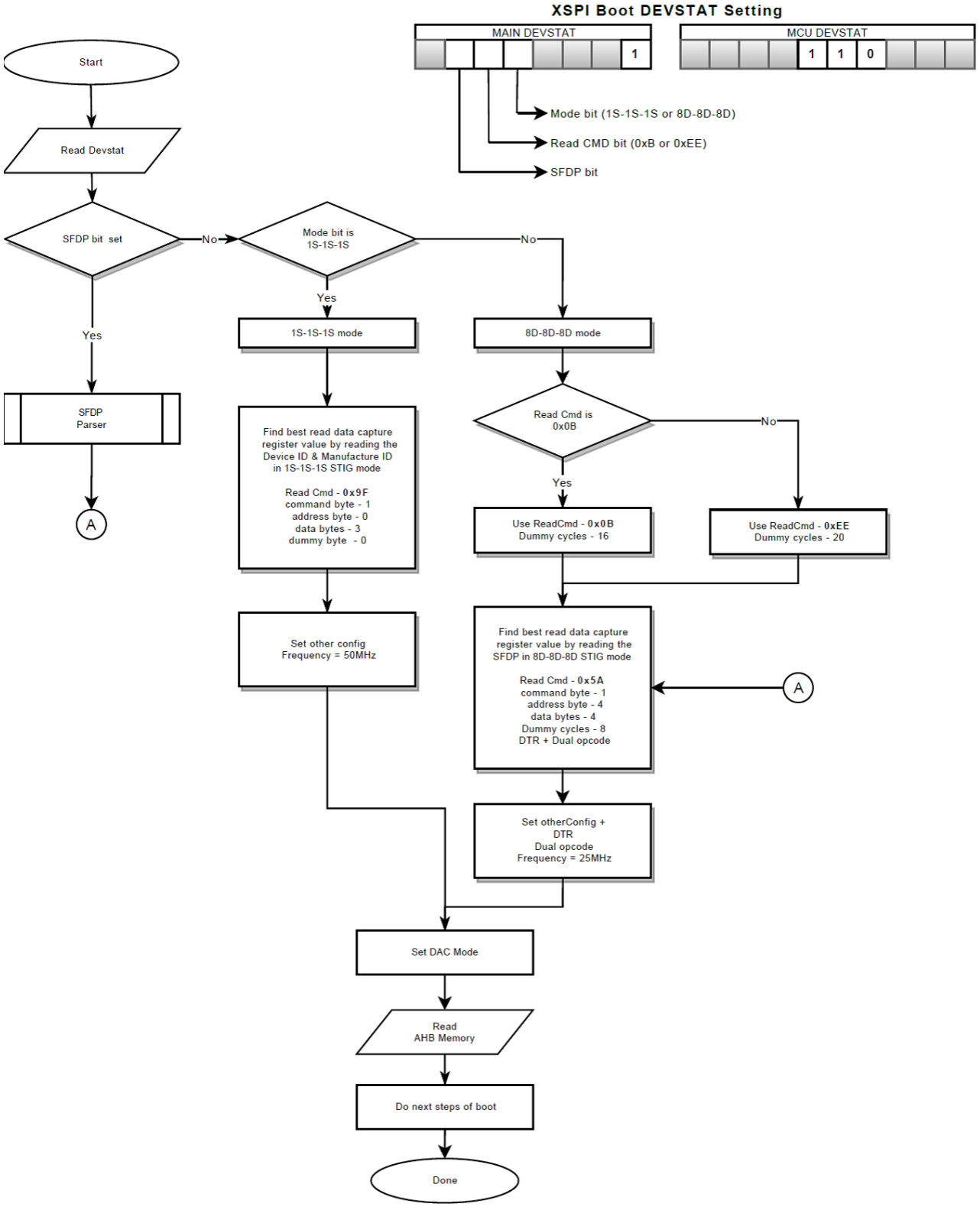


Figure 5-5. xSPI Flow Chart

Copyright © 2026 Texas Instruments Incorporated

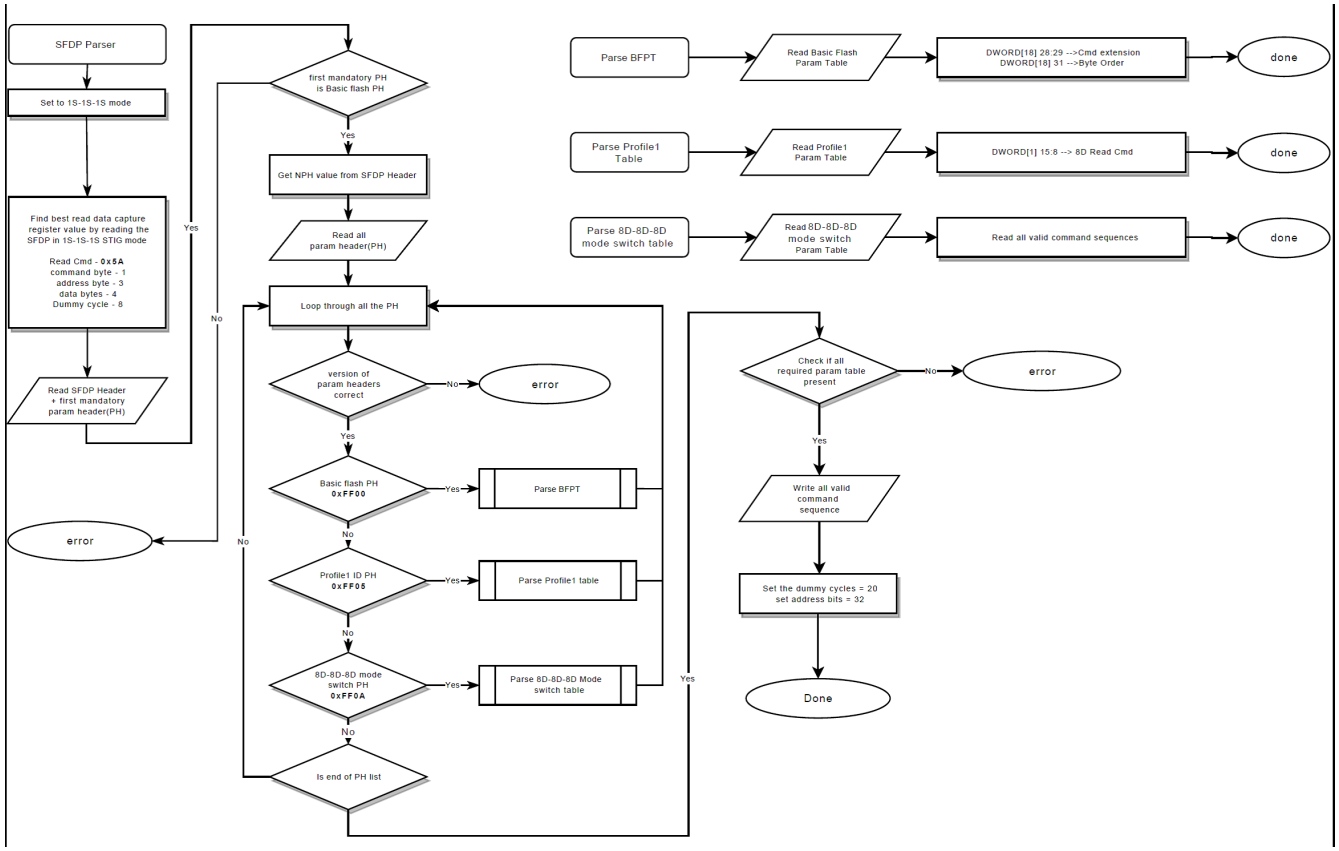


Figure 5-6. ROM SFDP Parser Flow

5.4.1.3 Fast-xSPI Boot Mode Configuration

The Fast-xSPI boot mode is an extension of the xSPI boot mode where the OSPI PHY will be tuned in order to set the frequency of operation to 100MHz for 8D-8D-8D mode. Flash needs to support SFDP.

To support Fast-xSPI boot, ROM expects the tuning configuration in the last sector of flash which is determined by the flash device configuration. Absence of the configuration will cause ROM to switch the non-PHY based/ standard xSPI boot.

ROM will get the flash memory density (flash memory size **S**) andlargest erase sector size (**E** bytes) from the BFPT table and calculates the beginning of last sector as E bytes from the end of flash. We call this location as start of the configuration data (**C0 = S - E**)

Supported Configurations:

1. ROM will support online PHY tuning as specified by the search bounds in the configuration data
2. ROM will support offline PHY tuning as specified by the delays in the configuration data

Selection between the two configurations is through magic words defined by ROM.

Copyright © 2026 Texas Instruments Incorporated

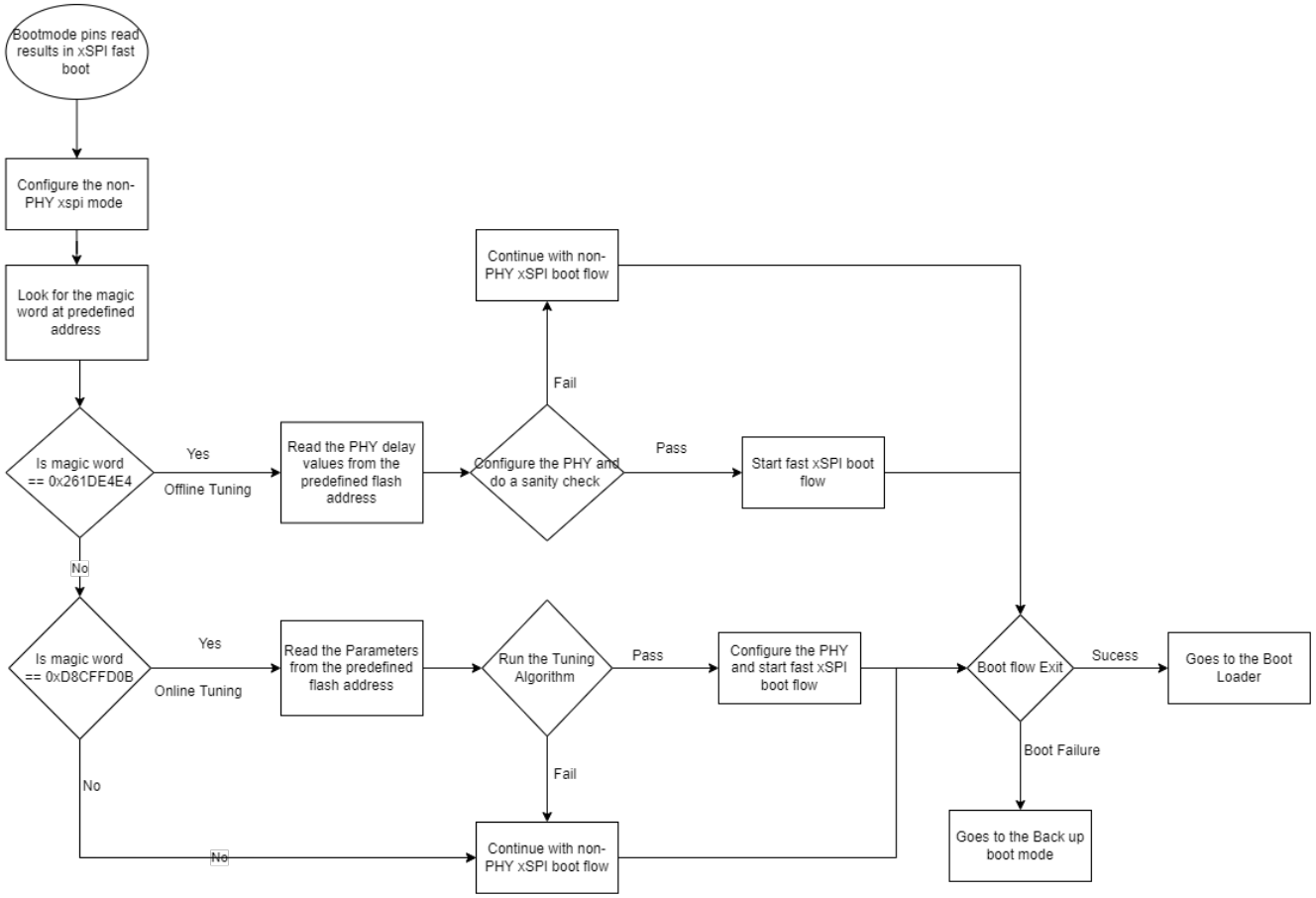


Figure 5-7. Fast-xSPI Control Flow

Table 5-10. Configuration Data for Fast-xSPI Boot

Offset from C0	Bitfields	Description
0 to 127	All	128 byte calibration pattern
128 to 131	All	4 byte Magic Word (defined by ROM). There are two distinct magic words to indicate • Offline tuning - 0x261DE4E4 – byte 128 - E4, 129 - E4, 130 - 1D, 131 - 26 • Online tuning - 0xD8CFFD0B – byte 128 - 0B, 129 - FD, 130 - CF, 131 - D8 The magic word is stored in little endian format. Absence of the magic word leads the ROM use the non-PHY mode.
132 to 135	All	4 byte unsigned int for clock frequency. Only 100Mhz is currently supported. ROM operates at 20 dummy cycles, so the flash should support the chosen frequency at 20 dummy cycles. The clock frequency is stored in little endian format. Any unknown clk freq leads ROM to the non-PHY mode. eg: • 100Mhz - 100000000 - 0x5F5E100 byte 132 - 00, 133 - E1, 134 - F5, 135 – 05

Copyright © 2026 Texas Instruments Incorporated

Table 5-10. Configuration Data for Fast-xSPI Boot (continued)

Offset from C0	Bitfields	Description
136	7:0	Flash Parameter Table version 0x00 or 0xFF - Parameter Table version 0 (Not supported for Fast-xSPI boot) 0x01 - Parameter Table version 1 (Latest version supported for Fast-xSPI boot)
137 to 139	All	3 bytes are reserved must be set to 0xFF
140 to 143		3 bytes for Tx_delay, Rx_delay, Rd_delay and one byte for padding. (Only used in case of offline tuning) Byte 143 - Padding must be set to 0xFF.
144 to 158		15 parameters (one byte each) (Only used in case of online tuning)
159	0	Determines if the master delay line locks on a full cycle or half cycle of delay. 0 - Full cycle of Delay 1 - Half cycle of Delay
159	1	DLL bypass mode control Controls the bypass mode of the master and slave DLLs 0 - Master Operation mode 1 - Bypass mode
159	4:2	DLL Phase Detect Selector for sampling clock generation to handle the clock domain crossing between the reference clock and sampling clock. Selects the number of delay elements to be inserted between the phase detect flip-flops: 3'b000 = One delay element 3'b001 = Two delay element 3'b010 = Three delay element 3'b011 = Four delay element 3'b100 = Five delay element 3'b101 = Six delay element 3'b110 = Seven delay element 3'b111 = Eight delay element
159	7:5	Reserved
160	7:0	Secondary search offset: To ensure the search point is valid, we try to do a secondary search. Value provided in this field will be used as an offset which is added to the primary search point TxDLL and then Secondary search is performed from offseted TxDLL point. Supports value from 0 to 127
161		DLL Lock Timeout flag For non-zero flag value, calculated values of DllLockErrCnt and LoopbackLockErrCnt are ignored and 0 is returned instead.
162 to 163		DLL Lock Timeout value in µs.

Table 5-11. Offline Tuning Parameters

Parameters	Offset from C0
Tx_delay	byte 140
Rx_delay	byte 141
Rd_delay	byte 142

Copyright © 2026 Texas Instruments Incorporated

Table 5-11. Offline Tuning Parameters (continued)

Parameters	Offset from C0
0xFF (Padding)	byte 143
DLL Config parameter	byte 159
DLL lock timeout flag	byte 161
DLL lock timeout value in µs	byte 162 and 163

Table 5-12. Online Tuning Parameters

Parameters	Offset from C0
Tx_Low_window_start	byte 144
Tx_Low_window_end	byte 145
Tx_High_window_start	byte 146
Tx_High_window_end	byte 147
Rx_min_bound_start	byte 148
Rx_min_bound_end	byte 149
Rx_max_bound_start	byte 150
Rx_max_bound_end	byte 151
Tx_min_bound_start	byte 152
Tx_min_bound_end	byte 153
Tx_max_bound_start	byte 154
Tx_max_bound_end	byte 155
Rd_Delay_init	byte 156
Rd_Delay_max	byte 157
Tx_offset_tuning_point	byte 158
DLL Config parameter	byte 159
Secondary Tx Search Offset	byte 160
DLL lock timeout flag	byte 161
DLL lock timeout value in µs	byte 162 and 163

The boot mode pin configuration and corresponding pin usage and mux configuration are shown below.

Table 5-13. MCU_BOOTMODE Pin Map xSPI

9	8	7	6	5	4	3	2	1	0
Rsvd (not for boot use)		OVRD	MCU Only	Primary Boot Mode A			PLL Config		
X	X	X	X	0	1	1	X	X	X

Table 5-14. BOOTMODE Pin Map xSPI

7	6	5	4	3	2	1	0
Backup Boot Mode Config	Primary Boot Mode Config			Backup Boot Mode			Primary Boot B
X	X	X	X	X	X	X	1

Table 5-15. xSPI Pin Usage

Package Name	Function Name	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
MCU_OSPI0_CLK	MCU_OSPI0_CLK	Disable	n/a	0	Disable	0	WKUP_PADCONFIG_0
MCU_OSPI0_LBCLKO	MCU_OSPI0_LBCLKO	Disable	n/a	0	Disable	0	WKUP_PADCONFIG_1
MCU_OSPI0_DQS	MCU_OSPI0_DQS	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_2

Copyright © 2026 Texas Instruments Incorporated

Table 5-15. xSPI Pin Usage (continued)

Package Name	Function Name	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
MCU_OSPI0_D0	MCU_OSPI0_D0	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_3
MCU_OSPI0_D1	MCU_OSPI0_D1	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_4
MCU_OSPI0_D2	MCU_OSPI0_D2	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_5
MCU_OSPI0_D3	MCU_OSPI0_D3	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_6
MCU_OSPI0_D4	MCU_OSPI0_D4	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_7
MCU_OSPI0_D5	MCU_OSPI0_D5	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_8
MCU_OSPI0_D6	MCU_OSPI0_D6	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_9
MCU_OSPI0_D7	MCU_OSPI0_D7	Disable	n/a	0	Enable	0	WKUP_PADCONFIG_10
MCU_OSPI0_CSn0	MCU_OSPI0_CSn0	Disable	n/a	0	Disable	0	WKUP_PADCONFIG_11
MCU_OSPI0_CSn1	MCU_OSPI0_CSn1	Disable	n/a	0	Disable	0	WKUP_PADCONFIG_12

Copyright © 2026 Texas Instruments Incorporated

5.4.1.4 QSPI Boot

Table 5-16 shows configuration pins assignment to functions when boot mode is QSPI using the OSPI module.

Table 5-16. QSPI Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
7	Csel	0	Boot Flash is on CS 0
		1	Boot Flash is on CS 1

Table 5-17 summarizes the OSPI pin configuration done by ROM code for QSPI boot device on port 0.

Table 5-17. QSPI Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Strength Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
OSPI0_CLK	OSPI0_CLK	Disable	NA	0	Disable	0	PADCONFIG0
OSPI0_D0	OSPI0_D0	Disable	NA	0	Enable	0	PADCONFIG3
OSPI0_D1	OSPI0_D1	Disable	NA	0	Enable	0	PADCONFIG4
OSPI0_D2	OSPI0_D2	Disable	NA	0	Enable	0	PADCONFIG5
OSPI0_D3	OSPI0_D3	Disable	NA	0	Enable	0	PADCONFIG6
OSPI0_CSn0(1)	OSPI0_CSn0	Disable	NA	0	Disable	0	PADCONFIG11
OSPI0_CSn1(1)	OSPI0_CSn1	Disable	NA	0	Disable	0	PADCONFIG12

(1) OSPI0_CSn0 and OSPI0_CSn1 are configured as chip select outputs from the OSPI0 host controller, but QSPI boot only uses the chip select output selected via BOOTMODE[7].

Note

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.1.4.1 QSPI Bootloader Operation

Please refer to Section 5.4.1 for more information that applies to QSPI boot mode

The QSPI boot mode supports the 1S-1S-4S mode only. The ROM will issue a Fast Read Quad Output command (0x6B) followed by a 24 bit (3 byte) address (the starting address is all zeros), followed by 8 dummy cycles. The flash device should then respond with 4-bit data. The frequency of operation during the quad read portion is 50 MHz.

5.4.1.4.1.1 QSPI Initialization Process

In the QSPI boot mode, the ROM Code initializes the OSPI module and the image is read from the QSPI flash connected to the selected chip-select. If the image fails to be read correctly from offset 0x0 of the flash memory, the ROM will attempt to obtain the image at offset 0x400000. This is the only redundant image location supported by the ROM. See Section 5.4.1.4, *QSPI Boot Device Configuration* for OSPI port settings.

A detailed summary of the QSPI boot parameter table and the boot configuration definitions are listed in Section 5.6.3, *QSPI Boot Parameter Table*.

5.4.1.4.1.2 QSPI Loading Process

QSPI boot mode is not eXecute-In-Place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

Copyright © 2026 Texas Instruments Incorporated

5.4.1.5 SPI Boot

Table 5-18 shows configuration pins assignment to functions when boot mode is SPI using the OSPI module.

Table 5-18. SPI Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
8	Mode	0	SPI Mode 0
		1	SPI Mode 3
7	Csel	0	Boot Flash is on CS 0
		1	Boot Flash is on CS 1

Table 5-19 summarizes the OSPI pin configuration done by ROM code for SPI boot device on port 0.

Table 5-19. SPI Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
OSPI0_CLK	OSPI0_CLK	Disable	NA	0	Disable	0	PADCONFIG0
OSPI0_D0	OSPI0_D0	Disable	NA	0	Enable	0	PADCONFIG3
OSPI0_D1	OSPI0_D1	Disable	NA	0	Enable	0	PADCONFIG4
OSPI0_CSn0(1)	OSPI0_CSn0	Disable	NA	0	Disable	0	PADCONFIG11
OSPI0_CSn1(1)	OSPI0_CSn1	Disable	NA	0	Disable	0	PADCONFIG12

(1) OSPI0_CSn0 and OSPI0_CSn1 are configured as chip select outputs from the OSPI0 host controller, but SPI boot only uses the chip select output selected via BOOTMODE[7].

Note

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.1.5.1 SPI Bootloader Operation

Please refer to Section 5.4.1 for more information that applies to SPI boot mode

The SPI boot mode supports the 1S-1S-1S mode only. The Read Command is 8-bits (0x03) followed by a 24 bit (3 byte) address. There are no dummy cycles issued after the read command. The frequency of operation supported is 6.250 MHz. OSPI0_D0 will have data transfers **FROM** the processor **TO** the flash device, and OSPI0_D1 will have data transfers **TO** the processor **FROM** the flash device

5.4.1.5.1.1 SPI Initialization Process

In the SPI boot mode, the ROM Code initializes the OSPI peripheral to SPI mode and the image is read from the flash memory connected to the corresponding OSPI port and chip-select. A detailed summary of the SPI boot parameter table and the boot configuration definitions are listed in Section 5.6.3, *OSPI/QSPI/SPI Boot Parameter Table*. If the image fails to be read correctly from offset 0x0 of the flash memory, the ROM will attempt to obtain the image at offset 0x400000. This is the only redundant image location supported by the ROM.

5.4.1.5.1.2 SPI Loading Process

SPI boot mode is not eXecute-In-Place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

Copyright © 2026 Texas Instruments Incorporated

5.4.2 I2C Boot

Table 5-20 shows configuration pins assignment to functions when boot mode is the I2C mode.

Table 5-20. I2C Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
9	Bus reset	0	Hung bus reset attempt after 1 ms
		1	No hung bus reset attempted
7	Address	0	EEPROM's address is 0x50
		1	EEPROM's address is 0x51

The I^{2}C bus is considered inactive if the data line is low and clock remains high for the specified timeout time. Recovery consists of driving the clock a stop condition is detected. A stop condition is a transition on the data line from 0 to 1 while the clock line is high. If the clock line is stuck low there is no way to take control of the bus.

Table 5-21 summarizes the I2C pin configuration done by ROM code for I^{2}C boot device.

Table 5-21. I2C Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
I2C0_SCL	I2C0_SCL	Disable	NA	0	Enable	0	PADCONFIG120
I2C0_SDA	I2C0_SDA	Disable	NA	0	Enable	0	PADCONFIG121

5.4.2.1 I2C Bootloader Operation

5.4.2.1.1 I2C Initialization Process

In the I2C boot mode, the ROM Code configures the Main Domain I2C0 in controller mode.

The boot controller drives the I^{2}C target device where the image is stored. The image is copied to internal RAM, and is executed from there. If the image is not recognized, the ROM will attempt to read the image at offset 0x8000. This is the only redundant image supported by the ROM.

A detailed summary of the I^{2}C boot parameter table and the BOOTMODE pins definitions are listed in Section 5.6.5, *I2C Boot Parameter Table* and Section 5.4.2, *I2C Boot Device Configuration*.

5.4.2.1.1.1 Block Size

ROM code will read 0x800 bytes before processing the data. The last block must be padded to the next 0x800 byte boundary and it must be padded with zeros.

5.4.2.1.1.2 Addressing

The boot code does not support byte address to bus address wrapping. So the maximum image size that can be access is 64 kbytes. For example, if the read address is 0xFF00, the read size is 0x200 and the I^{2}C bus address is 0x50, then 0x100 bytes will be read from 0xFF00 at bus address 0x50, followed by 0x100 bytes from 0x0000 at bus address 0x50. The bus address does not increment when the address rolls over.

5.4.2.1.2 I2C Loading Process

5.4.2.1.2.1 Loading a Boot Image From EEPROM

In this mode, the Main Domain I2C0 peripheral is configured as I^{2}C controller.

ROM Code will start reading from the I^{2}C EEPROM at the specified I^{2}C bus address. This read will be done beginning at the specified base address offset. Data will be read in 2-KB chunks. The data will be stored at the address specified in the boot header. It will continue reading image data until a complete image has been read. When the complete image has been read, the ROM Code will branch to the start address of the image.

Copyright © 2026 Texas Instruments Incorporated

5.4.3 SD Card Boot

Table 5-22 shows configuration pins assignment to functions when boot mode is the SD card boot mode.

Table 5-22. SD Card Boot Configuration Fields

SD Card Boot Configuration Fields			
BOOTMODE Pins	**Field**	**Value**	**Description**
9 (13 [1])	Port	0	Reserved
		1	MMC Port 1 (4 bit width). This bit must be set to 1
7	FS/Raw	0	Filesystem mode
		1	Raw Mode

(1) When MMCSD is the backup mode.

Table 5-23 summarizes the MMCSD pin configuration done by ROM code for SD Card boot on port1 (MMCSD1). SD Card boot on Port 0 (MMCSD0) is not supported.

Table 5-23. SD Card (MMCSD1) Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
MMC1_DAT3	MMC1_DAT3	Disable	NA	0	Enable	0	PADCONFIG137
MMC1_DAT2	MMC1_DAT2	Disable	NA	0	Enable	0	PADCONFIG138
MMC1_DAT1	MMC1_DAT1	Disable	NA	0	Enable	0	PADCONFIG139
MMC1_DAT0	MMC1_DAT0	Disable	NA	0	Enable	0	PADCONFIG140
MMC1_CLK	MMC1_CLK	Disable	NA	0	Enable	0	PADCONFIG141
MMC1_CLKLB	MMC1_CLKLB	Disable	NA	0	Enable	0	PADCONFIG142
MMC1_CMD	MMC1_CMD	Disable	NA	0	Enable	0	PADCONFIG143
MMC1_SDCD	MMC1_SDCD	Disable	NA	0	Enable	0	PADCONFIG144

Note

MMC1_CLKLB signal is not pinned out on the device. The pinmux configuration enables the input buffer of the internal loopback clock.

Note

MMC1_SDWP is not configured by ROM since the ROM never writes to the SD card.

5.4.3.1 SD Card Bootloader Operation

SD Card boot is only available on Port 1 of the MMCSD controller (MMCSD1). The ROM will boot from SD cards using one of these methods:

- User Data Area (UDA) in raw mode
- User Data Area (UDA) in filesystem mode

The ROM is capable of booting from any size SD card because the MMCSD controller is responsible for addressing. Only single data rate with backward compatible interface timing is supported.

The MMCSD module depends on proper voltage level on the SD card detect signal (SDCD). If SDCD=1, the ROM will assume the card is not present and will fail the boot mode.The SDCD must be 0 for the ROM to continue to attempt to boot. Card detect using DAT3 is not supported. If the card is detected, the initial discovery phase is performed:

1. Send CMD 0. (GO IDLE, both for MMC and SD)

Copyright © 2026 Texas Instruments Incorporated

2. Send CMD 55. If the card responds then the card type is assumed to be SD. On timeout the code sends CMD 1. If there is a response then the card type is MMC.

The ROM Code will start reading from the MMCSD memory boot sector, or filesystem, as specified by the BOOTMODE pin (FS/Raw). Only FAT32 and FAT16 formats are supported in filesystem mode. It will continue reading data from the memory and storing it in internal RAM until a complete image has been read. In RAW mode, the ROM supports a redundant image at offset 0x400000 in case the initial image fails to be recognized. When the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

When SD Card boot is used as a backup boot option, only User Data Area (UDA) in filesystem mode is supported. Raw mode is not supported. Additionally, boot will only occur in 1-bit mode during backup booting.

Copyright © 2026 Texas Instruments Incorporated

5.4.4 eMMC Boot

Table 5-24 shows configuration pins assignment to functions when boot mode is the eMMC Boot using a UDA boot mode (BOOTMODE[6:3]=1000b). Note that eMMC Boot using alternate eMMC Boot (BOOTMODE[6:3]=1001b) does not have any extra bootmode configuration fields. See Section 5.4.4.1 for more information.

Table 5-24. eMMC Boot Configuration Fields (UDA mode)

eMMC Boot Configuration Fields			
BOOTMODE Pins	**Field**	**Value**	**Description**
9 (13 [(1)])	Port	0	MMCSD Port 0 (8 bit width). This bit must be set to 0
		1	Reserved
7	FS/Raw	0	Filesystem mode
		1	Raw Mode

(1) When MMCSD Boot is the backup mode.

MMC Port 0 has no pin mux because the pins are dedicated.

5.4.4.1 eMMC Bootloader Operation

Booting from an eMMC device is only available on Port 0 of the MMCSD controller (MMCSD0). Booting from an eMMC device is not available on Port1 (MMCSD1)

The ROM will boot from eMMC devices using one of these methods:

- User Data Area (UDA) in raw mode
- User Data Area (UDA) in filesystem mode
- eMMC bootmode (alternate)

To boot from UDA in either raw or filesystem mode, choose bootmode "MMCSD Boot" (see Table 5-4).

A special alternate boot mode is available with eMMC devices which allows the ROM to boot from an image that is in a separate boot partition in the eMMC. To boot from this mode, choose "eMMC Boot" (see Table 5-4).

The ROM Code will start reading from the memory boot sector, or filesystem, as specified by the BOOTMODE pin (FS/Raw). Only FAT32 and FAT16 formats are supported in filesystem mode. It will continue reading data from the memory and storing it in internal RAM until a complete image has been read. In RAW mode, the ROM supports a redundant image at offset 0x400000 in case the initial image fails to be recognized. When the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

When eMMC boot is used as a backup boot option, only User Data Area (UDA) in filesystem mode is supported. Raw mode is not supported. Additionally, boot will only occur in 1-bit mode during backup booting.

Copyright © 2026 Texas Instruments Incorporated

5.4.5 Ethernet Boot

Table 5-25 shows configuration pins assignment to functions when boot mode is the Ethernet RGMII mode.

Table 5-25. Ethernet RGMII Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
9	Clkout	0	Must be set to 0 to choose external clock
		1	Reserved
8	Delay	0	Must be set to 0 for RGMII with internal Tx delay
		1	Reserved
7	Link info	0	MDIO PHY scan used for link parameters.
		1	Link parameters programmed by the ROM

Table 5-26 shows configuration pins assignment to functions when boot mode is the Ethernet RMII mode.

Table 5-26. Ethernet RMII Boot Configuration Fields

BOOTMODE Pins	Field	Value	Description
9	Clkout	0	50 MHz clock not generated on CLKOUT0
		1	50 MHz clock generated on CLKOUT0
8	Clk src	0	External clock source for RMII1_REF_CLK
		1	Internal clock source for RMII1_REF_CLK
7	RMII	0	This bit must be set to 0
		1	Reserved

Table 5-27. Ethernet RMII Clocking

BOOTMODE Pin 9 (Clk out)	BOOTMODE Pin 8 (Clk src)	Description
0	0	50MHz external source to RMII_REF_CLK and to external Ethernet PHY input clock (CLKOUT0 is unused) These are the recommended settings
0	1	Not a valid configuration
1	0	CLKOUT0 is configured to 50MHz and connect to both RMII1_REF_CLK and to external Ethernet PHY input clock
1	1	Not a valid configuration

Table 5-28 shows configuration pins assignment to functions when the backup boot mode Ethernet. The Interface configuration field chooses which interface will be used (RGMII or RMII)

Table 5-28. Ethernet Backup Boot Configuration Field

BOOTMODE Pins	Field	Value	Description
13	Interface	0	RGMII with internal TX delay
		1	RMII with external clock source

Table 5-29 summarizes the RGMII pin configuration done by ROM code for Ethernet boot device on RGMII port.

Table 5-29. RGMII Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
RGMII1_TX_CTL	RGMII1_TX_CTL	Disable	NA	0	Disable	0	PADCONFIG75
RGMII1_TXC	RGMII1_TXC	Disable	NA	0	Disable	0	PADCONFIG76
RGMII1_TD0	RGMII1_TD0	Disable	NA	0	Disable	0	PADCONFIG77

Copyright © 2026 Texas Instruments Incorporated

Table 5-29. RGMII Pin Usage (continued)

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
RGMII1_TD1	RGMII1_TD1	Disable	NA	0	Disable	0	PADCONFIG78
RGMII1_TD2	RGMII1_TD2	Disable	NA	0	Disable	0	PADCONFIG79
RGMII1_TD3	RGMII1_TD3	Disable	NA	0	Disable	0	PADCONFIG80
RGMII1_RX_CTL	RGMII1_RX_CTL	Disable	NA	0	Enable	0	PADCONFIG81
RGMII1_RXC	RGMII1_RXC	Disable	NA	0	Enable	0	PADCONFIG82
RGMII1_RD0	RGMII1_RD0	Disable	NA	0	Enable	0	PADCONFIG83
RGMII1_RD1	RGMII1_RD1	Disable	NA	0	Enable	0	PADCONFIG84
RGMII1_RD2	RGMII1_RD2	Disable	NA	0	Enable	0	PADCONFIG85
RGMII1_RD3	RGMII1_RD3	Disable	NA	0	Enable	0	PADCONFIG86
MDIO0_MDIO	MDIO0_MDIO	Disable	NA	0	Enable	0	PADCONFIG87
MDIO0_MDC	MDIO0_MDC	Disable	NA	0	Disable	0	PADCONFIG88

Table 5-30 summarizes the RMII pin configuration done by ROM code for Ethernet boot device on RMII port.

Table 5-30. RMII Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
RGMII1_RX_CTL	RMII1_RX_ER	Disable	NA	0	Enable	1	PADCONFIG81
RGMII1_RXC	RMII1_REF_CLK	Disable	NA	0	Enable	1	PADCONFIG82
RGMII1_RD0	RMII1_RXD0	Disable	NA	0	Enable	1	PADCONFIG83
RGMII1_RD1	RMII1_RXD1	Disable	NA	0	Enable	1	PADCONFIG84
RGMII1_TD0	RMII1_TXD0	Disable	NA	0	Disable	1	PADCONFIG77
RGMII1_TD1	RMII1_TXD1	Disable	NA	0	Disable	1	PADCONFIG78
RGMII1_TX_CTL	RMII1_TX_EN	Disable	NA	0	Disable	1	PADCONFIG75
RGMII1_TXC	RMII1_CRS_DV	Disable	NA	0	Enable	1	PADCONFIG76
RGMII1_TD3	CLKOUT0	Disable	NA	0	Disable	1	PADCONFIG_80
MDIO0_MDIO	MDIO0_MDIO	Disable	NA	0	Enable	0	PADCONFIG87
MDIO0_MDC	MDIO0_MDC	Disable	NA	0	Disable	0	PADCONFIG88

5.4.5.1 Ethernet Bootloader Operation

5.4.5.1.1 Ethernet Initialization Process

When the device is set to boot through the Ethernet mode, the ROM Code configures the Ethernet module and the interface mode (RMII, RGMII) according the the BOOTMODE pin settings, see Section 5.4.5, *Ethernet Boot Device Configuration*. Also consult the boot parameter table for the Ethernet boot, see Section 5.6.7, *Ethernet Boot Parameter Table*.

When Link Info = 0, the link parameters are read using MDIO scan. This is the typical setting when using an external PHY. With these parameters, the ROM identifies the PHY and establishes link with the supported speed and duplex mode.

Copyright © 2026 Texas Instruments Incorporated

When Link Info = 1, no MDIO scan is performed, and the link parameters are programmed by the ROM based on the RGMII status register. This is the typical setting when configured as a MAC to MAC RGMII interface whereby RGMII pins of two devices can be connected diretly and establish link in force mode. The chosen configuration will be Gigabit, full duplex mode.

Note that booting from RGMII requires that an attached PHY be configured for RGMII-ID Mode immediately upon exiting reset. The ROM will not make this configuration change to the PHY.

The ROM will setup RGMII mode enabling internal delay mode.

5.4.5.1.2 Ethernet Loading Process

After device configuration, the bootloader performs a standard BOOTP/TFTP boot. The device sends a BOOTP request with its MAC address to a host TFTP server to be assigned an IP from a pool of addresses. The timeout for each BOOTP packet is 4 seconds, and the ROM will attempt 10 BOOTP retries, after which the boot mode will fail. If the connection is established, the device initiates a TFTP download and is able to receive image data encapsulated in Ethernet packets, see Section 5.4.5.1.2.1. There is a timeout of 1 second to receive a response for the READ request, and the ROM will retry the READ request 10 times, after which the boot mode will fail. If TFTP download is successful, data received is stripped of its network headers and the boot data is stored in internal RAM. When the transfer completes and the image is found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

5.4.5.1.2.1 Ethernet Boot Data Formats

Ethernet boot uses the BOOTP/TFTP protocol for downloads. Only IPv4 is supported.

5.4.5.1.2.1.1 Limitations

- Received packets cannot be IP fragmented (should not be a problem in most systems since the BOOTP/TFTP packets have fixed lengths of small size)
- Only DIX Ethernet headers are supported.
  - 802.3 with SNAP/LLC not supported
  - DIX Ethernet with VLAN not supported
  - 802.3 with VLAN and SNAP/LLC not supported

5.4.5.1.2.1.2 BOOTP Request

5.4.5.1.2.1.2.1 MAC Header (DIX)

- Destination MAC = value from parameter table (default is broadcast)
- Source MAC = value from parameter table (default is e-fuse value)
- Type = IPv4 (0x0800)

5.4.5.1.2.1.2.2 IPv4 Header

- Version = 4
- Header length = 0
- TOS = 0
- Len = computed during operation
- ID = 0x0001
- Flags + Fragment offset = 0
- TTL = 0x10
- Protocol = UDP (17)
- Header checksum = Computed during operation
- SRC IP = 0.0.0.0
- Dest IP = 255.255.255.255

5.4.5.1.2.1.2.3 UDP Header

- Source port = BOOTP client (68 decimal)
- Destination Port = BOOTP server (67 decimal)
- Length = computed during operation

Copyright © 2026 Texas Instruments Incorporated

- Checksum = computed during operation

5.4.5.1.2.1.2.4 BOOTP Payload

- Opcode = Request (1)
- HW Type = Ethernet (1)
- HW Addr Len = 6
- Hop Count = 0
- Transaction ID = 1
- Number of seconds = 0
- Client IP = 0.0.0.0
- Your IP = 0.0.0.0
- Server IP = 0.0.0.0
- Gateway IP = 0.0.0.0
- Client HW Address = Device MAC address (from parameter table)
- Server hostname = NULL
- Filename = NULL
- Option 60, Vendor ID string, from parameter table
- Option 61, Client ID string, from parameter table

5.4.5.1.2.1.2.5 TFTP

There are no ROM code specific TFTP configurations.

5.4.5.1.3 Ethernet Hand Over Process

Once the ROM Code receives the valid packet, it decodes it to get the image sections and loads them in the appropriate memory location. After the image is loaded and validated, the ROM Code starts the boot image execution.

Copyright © 2026 Texas Instruments Incorporated

5.4.6 USB Boot

Table 5-31 shows configuration pins assignment to functions when boot mode is the USB mode.

Table 5-31. USB Boot Configuration Fields

USB Configuration Fields for Primary Boot Mode			
BOOTMODE Pins	**Field**	**Value**	**Description**
9	CoreVoltage	0	Must be set
		1	Reserved
8 (13[1])	Mode	0	DFU (USB device firmware upgrade)
		1	Host (MSC boot)
7	Lane Swap	0	D+/D- lines are not swapped
		1	D+/D- lines are swapped

(1) When USB is the backup mode.

Table 5-32 summarizes the USB pin configuration done by ROM code for USB boot device on port 0.

Table 5-32. USB Port 0 Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
USB0_DRVVBUS	USB0_DRVVBUS	Disable	NA	0	Disable	0	PADCONFIG149

Note

Note that other USB pins do not have pin mux options. USB_DRVVBUS is configured in DFU mode even though it is not used.

Note on USB backup boot:

- Lane Swap (D+/D- lines) are not allowed

Configurability (pins) of these options do not exist in the USB backup boot mode.

CAUTION

A USB Type-AB connector should not be connected to another host when the device is booting as a USB Host. This will prevent two USB power sources from being connected together.

5.4.6.1 USB Bootloader Operation

The USB boot mode is used to read the boot image from external USB host.

See Section 5.4.6, *USB Boot Device Configuration* and Section 5.6.10, *USB Boot Parameter Table* for the available configuration options.

More information about USB DFU protocol can be found at http://www.usb.org/sites/default/files/DFU_1.1.pdf.

In DFU mode, the ROM will attempt an enumeration for 60 seconds, after which the boot mode will fail. If a successful enumeration is achieved, the ROM Code will start reading from the external host as specified by the BOOTMODE pins. It will continue reading data from the memory and storing it in internal RAM until a complete image has been read. When the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

5.4.6.1.1 USB-Specific Attributes

5.4.6.1.1.1 DFU Device Mode

- Vendor ID = 0x451
- Product ID = 0x6165
- Device ID = 0

Copyright © 2026 Texas Instruments Incorporated

5.4.7 UART Boot

ROM Code always configures the UART port to 115200 kbaud, 8-n-1 mode, and the XMODEM protocol is used to transfer the boot data.

Table 5-33 summarizes the UART pin configuration done by ROM code for UART host on port 0.

Table 5-33. UART Port 0 Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
UART0_TXD	UART0_TXD	Disable	NA	0	Disable	0	PADCONFIG115
UART0_RXD	UART0_RXD	Disable	NA	0	Enable	0	PADCONFIG114

5.4.7.1 UART Bootloader Operation

5.4.7.1.1 Initialization Process

In the UART boot mode, the selected UART module (port) is the only peripheral configured. The baud rate, data, parity, and stop bits are configured based on the information in the UART boot parameter table. The boot parameter table definitions and the boot configuration values that can be set are in Section 5.4.7, *UART Boot Device Configuration* and Section 5.6.4, *UART Boot Parameter Table*.

Once the ROM Code configures the UART, it sends the UART pings for few seconds, which can be seen in the host. The pings consist of an ASCII capital *C* character. The UART boot mode supports only the CRC mode of XMODEM and does not support CHECKSUM mode. Both 128 and 1024 byte block sizes are supported.

5.4.7.1.2 UART Loading Process

Before the ping from the device stops, load the boot image from the host using the XMODEM protocol. Until UART boot data is sent from the host, the ROM code will continue to send pings for 120 seconds before aborting and moving to the secondary boot mode (if applicable).

Note

The device also implements a separate 180 watchdog timer. If UART boot is the primary boot mode, and no secondary boot mode is selected, the subsequent UART boot will timeout after 60sec due to the watchdog timer.

5.4.7.1.2.1 UART XMODEM

The XMODEM protocol is used to transfer boot data. Only CRC mode is supported (not checksum), with both 128- and 1024-byte block sizes. The general, format of received frames is shown in Table 5-34 and Table 5-35.

Table 5-34. XMODEM 1024- and 128-byte Data Frames

STX	Block Num	Inv Block Num	1024 data bytes		CRC	CRC
SOH	Block Num	Inv Block Num	128 data bytes	CRC	CRC	

Table 5-35. XMODEM Data Frame Fields

Field	Value	Description
STX	0x02	The start character for 1024-byte CRC data blocks
SOH	0x01	The start character for 128-byte CRC data block
Block Num	0x01-0xFF – 0x00	The block number. The first block has value 1, and the block number wraps around 0xFF to 0
Inv Block Num	0xFE-0x00	The inverse block number (bit inverse of the block number)
CRC	Calculated	The 16-bit CRC generated from the polynomial 0x1021

The XMODEM protocol is implemented as a half-duplex protocol as shown in Table 5-36.

Copyright © 2026 Texas Instruments Incorporated

Table 5-36. Example of XMODEM Transfer protocol

Transmitter Sends		Receiver Sends
	←	Ping (‘C’)
Frame 1	→	
	←	ACK (or NACK)
Frame 2	→	
	←	ACK (or NACK)
EOT	→	
	←	ACK (or NACK)

5.4.7.1.3 UART Hand-Over Process

Once the complete image has been read and found in good integrity, the ROM Code will branch to the address defined in the Boot Info field of the boot header.

Copyright © 2026 Texas Instruments Incorporated

5.4.8 GPMC NOR Boot

There are no configuration fields for this boot mode. GPMC NOR boot only supports 16-bit non-mux memory.

Table 5-37 summarizes the GPMC pin configuration done by ROM code for GPMC NOR boot.

Table 5-37. GPMC NOR Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
GPMC0_AD0	GPMC0_AD0	Disable	NA	0	Enable	0	PADCONFIG15
GPMC0_AD1	GPMC0_AD1	Disable	NA	0	Enable	0	PADCONFIG16
GPMC0_AD2	GPMC0_AD2	Disable	NA	0	Enable	0	PADCONFIG17
GPMC0_AD3	GPMC0_AD3	Disable	NA	0	Enable	0	PADCONFIG18
GPMC0_AD4	GPMC0_AD4	Disable	NA	0	Enable	0	PADCONFIG19
GPMC0_AD5	GPMC0_AD5	Disable	NA	0	Enable	0	PADCONFIG20
GPMC0_AD6	GPMC0_AD6	Disable	NA	0	Enable	0	PADCONFIG21
GPMC0_AD7	GPMC0_AD7	Disable	NA	0	Enable	0	PADCONFIG22
GPMC0_AD8	GPMC0_AD8	Disable	NA	0	Enable	0	PADCONFIG23
GPMC0_AD9	GPMC0_AD9	Disable	NA	0	Enable	0	PADCONFIG24
GPMC0_AD10	GPMC0_AD10	Disable	NA	0	Enable	0	PADCONFIG25
GPMC0_AD11	GPMC0_AD11	Disable	NA	0	Enable	0	PADCONFIG26
GPMC0_AD12	GPMC0_AD12	Disable	NA	0	Enable	0	PADCONFIG27
GPMC0_AD13	GPMC0_AD13	Disable	NA	0	Enable	0	PADCONFIG28
GPMC0_AD14	GPMC0_AD14	Disable	NA	0	Enable	0	PADCONFIG29
GPMC0_AD15	GPMC0_AD15	Disable	NA	0	Enable	0	PADCONFIG30
VOUT0_DATA0	GPMC_A0	Disable	NA	0	Disable	1	PADCONFIG46
VOUT0_DATA1	GPMC_A1	Disable	NA	0	Disable	1	PADCONFIG47
VOUT0_DATA2	GPMC_A2	Disable	NA	0	Disable	1	PADCONFIG48
VOUT0_DATA3	GPMC_A3	Disable	NA	0	Disable	1	PADCONFIG49
VOUT0_DATA4	GPMC_A4	Disable	NA	0	Disable	1	PADCONFIG50
VOUT0_DATA5	GPMC_A5	Disable	NA	0	Disable	1	PADCONFIG51
VOUT0_DATA6	GPMC_A6	Disable	NA	0	Disable	1	PADCONFIG52
VOUT0_DATA7	GPMC_A7	Disable	NA	0	Disable	1	PADCONFIG53
VOUT0_DATA8	GPMC_A8	Disable	NA	0	Disable	1	PADCONFIG54
VOUT0_DATA9	GPMC_A9	Disable	NA	0	Disable	1	PADCONFIG55
VOUT0_DATA10	GPMC_A10	Disable	NA	0	Disable	1	PADCONFIG56
VOUT0_DATA11	GPMC_A11	Disable	NA	0	Disable	1	PADCONFIG57
VOUT0_DATA12	GPMC_A12	Disable	NA	0	Disable	1	PADCONFIG58
VOUT0_DATA13	GPMC_A13	Disable	NA	0	Disable	1	PADCONFIG59
VOUT0_DATA14	GPMC_A14	Disable	NA	0	Disable	1	PADCONFIG60
VOUT0_DATA15	GPMC_A15	Disable	NA	0	Disable	1	PADCONFIG61
VOUT0_HSYNC	GPMC_A16	Disable	NA	0	Disable	1	PADCONFIG62
VOUT0_DE	GPMC_A17	Disable	NA	0	Disable	1	PADCONFIG63
VOUT0_VSYNC	GPMC_A18	Disable	NA	0	Disable	1	PADCONFIG64
VOUT0_PCLK	GPMC_A19	Disable	NA	0	Disable	1	PADCONFIG65
GPMC0_CSn3	GPMC_A20	Disable	NA	0	Disable	2	PADCONFIG45
GPMC0_ADVn_ALE	GPMC0_ADVn_ALE	Disable	NA	0	Disable	0	PADCONFIG33
GPMC0_OEn_REN	GPMC0_OEn_Ren	Disable	NA	0	Disable	0	PADCONFIG34
GPMC0_WEN	GPMC0_WEN	Disable	NA	0	Disable	0	PADCONFIG35

Copyright © 2026 Texas Instruments Incorporated

Table 5-37. GPMC NOR Boot Pin Usage (continued)

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
GPMC0_BE0n_CLE	GPMC0_BEOn_CLE	Disable	NA	0	Disable	0	PADCONFIG36
GPMC0_BE1n	GPMC0_BE1n	Disable	NA	0	Disable	0	PADCONFIG37
GPMC0_CSn0	GPMC0_CSn0	Disable	NA	0	Disable	0	PADCONFIG42

Note

Only 21 address lines (GPMC0_A0 – GPMC0_A20) are used because the GPMC0_A21 and GPMC0_A22 lines are muxed with GPMC0_WAIT1 and GPMC0_WPn respectively.

GPMC0_A20 is muxed with GPMC0_CSn3, and ROM uses GPMC0_A20 for this address line. Thus, no CSn3 support when using GPMC NOR boot.

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.8.1 GPMC NOR Bootloader Operation

In this mode, the ROM Code configures the GPMC interface based on the configuration parameters specified in the boot parameter table for the GPMC NOR boot mode Section 5.6.11, see*GPMC NOR Boot Parameter Table*.

Timing registers are programmed as follows for NOR flash boot:

Table 5-38. GPMC NOR Timing Configuration

GPMC register	Value
GPMC_CONFIG1	0x00001010
GPMC_CONFIG2	0x00101c01
GPMC_CONFIG3	0x23060917
GPMC_CONFIG4	0x1005bc1a
GPMC_CONFIG5	0x011b111e
GPMC_CONFIG6	0x8f070000
GPMC_CONFIG7	0x00000c50

GPMC NOR boot mode is not executable-in-place (XIP). ROM code first copies boot image into on-chip RAM and then executes it. Only non-muxed memory is supported. If the initial image at offset 0x0 is not recognized, the ROM will attempt to read a redundant image from offset 0x100000. This is the only redundant image supported by the ROM.

Copyright © 2026 Texas Instruments Incorporated

5.4.9 GPMC NAND Boot

Table 5-39 summarizes the GPMC pin configuration done by ROM code for NAND boot.

Table 5-39. GPMC NAND Boot Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
GPMC0_AD0	GPMC0_AD0	Disable	NA	0	Enable	0	PADCONFIG15
GPMC0_AD1	GPMC0_AD1	Disable	NA	0	Enable	0	PADCONFIG16
GPMC0_AD2	GPMC0_AD2	Disable	NA	0	Enable	0	PADCONFIG17
GPMC0_AD3	GPMC0_AD3	Disable	NA	0	Enable	0	PADCONFIG18
GPMC0_AD4	GPMC0_AD4	Disable	NA	0	Enable	0	PADCONFIG19
GPMC0_AD5	GPMC0_AD5	Disable	NA	0	Enable	0	PADCONFIG20
GPMC0_AD6	GPMC0_AD6	Disable	NA	0	Enable	0	PADCONFIG21
GPMC0_AD7	GPMC0_AD7	Disable	NA	0	Enable	0	PADCONFIG22
GPMC0_ADVn_ALE	GPMC0_ADVn_ALE	Disable	NA	0	Disable	0	PADCONFIG33
GPMC0_OEn_Ren	GPMC0_OEn_Ren	Disable	NA	0	Disable	0	PADCONFIG34
GPMC0_Wen	GPMC0_Wen	Disable	NA	0	Disable	0	PADCONFIG35
GPMC0_BEOn_CLE	GPMC0_BEOn_CLE	Disable	NA	0	Disable	0	PADCONFIG36
GPMC0_WAIT0	GPMC0_WAIT0	Disable	NA	0	Enable	0	PADCONFIG38
GPMC0_CSn0	GPMC0_CSn0	Disable	NA	0	Disable	0	PADCONFIG42

5.4.9.1 GPMC NAND Bootloader Operation

Timing registers are programmed as follows for NAND flash boot:

Table 5-40. GPMC NAND Timing Configuration

GPMC register	Value
GPMC_CONFIG1	0x00000812
GPMC_CONFIG2	0x00080b00
GPMC_CONFIG3	0x22080810
GPMC_CONFIG4	0x05006890
GPMC_CONFIG5	0x0107080b
GPMC_CONFIG6	0x80000180
GPMC_CONFIG7	0x00000f50

GPMC NAND boot only supports boot from ONFI 1.0 compatible 8 bit parallel NAND memory up to 2Gbytes in size connected to GPMC CS0 with the following geometries:

- 2Kbyte page and spare area of at least 64 bytes or
- 4Kbyte page size and spare area of at least 128 bytes.
- Non-ECC part only:
  - ROM uses ELM to handle ECC
  - ECC is BCH8 using D[7:0] for data
  - The param page CRC is checked and in case of failure the redundant page is used

GPMC is setup to comply with mode 0 timing for NAND flash.

Copyright © 2026 Texas Instruments Incorporated

5.4.10 Serial NAND Boot

Serial NAND Configuration Fields table shows configuration pins assignment to functions when boot mode is Serial NAND.

Table 5-41. Serial NAND Configuration Fields

BOOTMODE Pins	Field	Value	Description
8	Read Mode 2	0	Reserved (Read mode is taken from Read Mode 1
		1	SPI/ 1-1-1 mode (Read mode is taken from Read Mode 2 and Read Mode 1 is ignored)
7	Read Mode 1	0	OSPI/ 1-1-8 Mode (valid only when Read Mode 2 is 0)
		1	OSPI/ 1-1-4 Mode (valid only when Read Mode 2 is 0)

Serial NAND Pin Usage table summarizes the pin configuration done by ROM code for the Serial NAND device.

Table 5-42. Serial NAND Pin Usage

Device Pin	Module Signal	Pull Enable	Pull Direction	Driver Index	Rx En/Dis	Pinmux Sel	Pad Configuration Register
OSPI0_CLK	OSPI0_CLK	Disable	NA	0	Disable	0	PADCONFIG0
OSPI0_DQS	OSPI0_DQS	Disable	NA	0	Enable	0	PADCONFIG2
OSPI0_D0	OSPI0_D0	Disable	NA	0	Enable	0	PADCONFIG3
OSPI0_D1	OSPI0_D1	Disable	NA	0	Enable	0	PADCONFIG4
OSPI0_D2	OSPI0_D2	Disable	NA	0	Enable	0	PADCONFIG5
OSPI0_D3	OSPI0_D3	Disable	NA	0	Enable	0	PADCONFIG6
OSPI0_D4	OSPI0_D4	Disable	NA	0	Enable	0	PADCONFIG7
OSPI0_D5	OSPI0_D5	Disable	NA	0	Enable	0	PADCONFIG8
OSPI0_D6	OSPI0_D6	Disable	NA	0	Enable	0	PADCONFIG9
OSPI0_D7	OSPI0_D7	Disable	NA	0	Enable	0	PADCONFIG10
OSPI0_CSn0(1)	OSPI0_CSn0	Disable	NA	0	Disable	0	PADCONFIG11
OSPI0_CSn1(1)	OSPI0_CSn1	Disable	NA	0	Disable	0	PADCONFIG12

(1) Both OSPI0_CSn0 and OSPI0_CSn1 are configured as chip select outputs from the OSPI0 host controller, but Serial NAND boot only uses OSPI0_CSn0. It is not possible to select OSPI0_CSn1 as a chip select option for Serial NAND boot.

Note

All signals in the table will be configured even though some may not be used by this particular boot mode.

5.4.10.1 Serial NAND Bootloader Operation

Serial NAND defines 1S-1S-1S mode for general backwards compatibility, and 1S-1S-8S for maximum throughput (S here means *S*ingle Data rate). Serial NAND memory array is organized into pages of size 2KB/4KB. Read is a two-step process where a complete page is first read into flash's internal buffer/cache using Page read command and then the host controller reads from internal buffer in 1- or 4- or 8-bit mode using the read commands. Page read command that is issued is 0x13, followed by 24 address bits. The frequency of operation supported is 50 MHz.

For 1S-1S-1S mode of operation (Bit-width =1, Single Data Rate). The Command and Address issued are 8 bits and 16 bits respectively. The Read Command that is issued is 0x0B, followed by address bits and 8 dummy cycles.

For 1S-1S-8S mode of operation (Bit-width =8, Single Data Rate). The Command and Address issued are 8 bits and 16 bits respectively. The Read Command that is issued is 0x8B, followed by address bits and 8 dummy

Copyright © 2026 Texas Instruments Incorporated

cycles. Additionally, flash is configured in 8-bit mode after POR through volatile configuration register if the manufacturer is Winbond.

For 1S-1S-4S mode of operation (Bit-width =4, Single Data Rate). The Command and Address issued are 8 bits and 16 bits respectively. The Read Command that is issued is 0x6B, followed by address bits and 8 dummy cycles. Note that in 8-bit mode pin mux is done for all 8 OSPI data lines and in 4-bit/1-bit mode pin mux is done for 4 OSPI data lines. This is done to disable the HOLD functionality feature in 1-bit mode.

Serial NAND boot expects ECC to be auto-managed by the flash. Most of the flashes have the ECC enabled by default and can do 1-bit correction and 2-bit detection for ECC errors. ROM checks for 2-bit ECC error via status register 3 (address 0xC0) bit 5 after every page load. In case of 2-bit ECC error the boot will fail and ROM will take the fallback option.

Serial NAND boot also manages bad blocks that can be present in the flash at time of shipment or develop during the lifetime. Bad block marker is a non-FFh data byte stored at Byte 0 of spare area of Page 0 for each bad block and ROM checks for the same while reading the first page of a memory block. ROM will skip the particular block if it is marked as bad and move to the next one.

The Serial NAND driver in ROM does not support devices that have multiple planes as they require special handling to read even numbered blocks

5.4.10.2 Serial NAND Initialization Process

In the OSPI boot mode, the ROM Code initializes the OSPI module and the image is read from the OSPI flash connected to the selected chip-select. If the image fails to be read correctly from offset 0x0 of the flash memory, the ROM will attempt to obtain the image at offset 0x400000. This is the only redundant image location supported by the ROM. See Section 5.4.1.1, *OSPI Boot Device Configuration* for OSPI port settings.

A detailed summary of the OSPI boot parameter table and the boot configuration definitions are listed in Section 5.6.3, *OSPI Boot Parameter Table*.

5.4.10.3 Serial NAND Loading Process

OSPI boot mode is not eXecute-In-Place (XIP). ROM code first copies boot image into on-chip RAM and then executes it.

Copyright © 2026 Texas Instruments Incorporated

5.4.11 No boot/Development boot

Table 5-43 shows configuration pins assignment to functions when boot mode is the No-boot mode.

Table 5-43. No-boot/Dev-boot Configuration Fields

Field	Value	Description
No/Dev	0	Development Boot
	1	No boot
ARM/Thumb	0	ARM mode
	1	Thumb mode

These boot modes are useful for debugging purposes to preclude the execution of the ROM.

During the Development boot (BOOTMODE[7] = 0), the SMS M4F ROM code will act as if boot of the primary image has completed, and the SMS M4F ROM then will be waiting for a firmware load message from DM R5. Thus the user can load a standard u-boot/SPL image to the DM R5 RAM. U-boot/SPL will then load the SMS firmware and complete the full boot.

In No-boot (BOOTMODE[7] = 1), both the SMS M4F and DM R5 ROMs are bypassed and both CPUs are held in a dummy branch-to-self loop. No-boot is the most minimal device touch state by the ROM - only minimal hardware configurations are done and none of the PLLs is locked/configured. No-boot is suitable if user wants to load his own PLL, Pad config, and other basic settings.

Copyright © 2026 Texas Instruments Incorporated

5.5 PLL Configuration

ROM code must be aware of the reference clock provided to PLLs. That is, the speed of the quartz crystal, or the clock supplied by an external clock oscillator. On how to indicate the PLL reference clock, see PLL Reference Clock Selection

ROM code configures only PLLs which are required during boot. Therefore, if a PLL is required for the backup boot mode but not the primary boot mode, and if the backup boot mode never executes, then the PLLs required for backup boot are not enabled.

The following tables show the HSDIV values that are programmed by the R5 ROM if a boot mode uses it. A value of NA means that the ROM does not program that HSDIV

Table 5-44. MAIN_PLL0 (MAIN PLL) (2000MHz)

PLL POSTDIV	HSDIV	Value	Frequency(MHz)	Boot Peripheral/IP
0	HSDIV0	4	500	MAIN SYSCLK0
0	HSDIV1	10	200	OSPI
0	HSDIV2	0	NA	WKUP_CLKOUT
0	HSDIV3	15	133	GPMC NOR/GPMC NAND
0	HSDIV4	0	NA	MCAN
1	HSDIV5	5	200	MMCSD0
1	HSDIV6	0	NA	CPTS
1	HSDIV7	4	250	TIMER
1	HSDIV8	0	NA	USB0
1	HSDIV9	0	NA	Reserved

Table 5-45. MAIN_PLL1 (PER0 PLL) (1920MHz)

PLL POSTDIV	HSDIV	Value	Frequency(MHz)	Boot Peripheral/IP
0	HSDIV0	10	192	UART
0	HSDIV1	12	160	Configured but not used
0	HSDIV2	0	NA	WKUP_CLKOUT
0	HSDIV3	0	NA	TIMER
0	HSDIV4	0	NA	Reserved
1	HSDIV5	0	NA	OSPI
1	HSDIV6	0	NA	McASP

Table 5-46. MAIN_PLL2 (PER1 PLL) (2000MHz)

PLL POSTDIV	HSDIV	Value	Frequency(MHz)	Boot Peripheral/IP
0	HSDIV0	0	NA	Reserved
0	HSDIV1	8	250	CPSW0
0	HSDIV2	10	200	MMCSD1
0	HSDIV3	0	NA	DebugSS
0	HSDIV4	0	NA	Video Encoder/Decoder
1	HSDIV5	0	NA	MCSPI
1	HSDIV6	0	NA	TIMER
1	HSDIV7	0	NA	GPMC
1	HSDIV8	0	NA	McASP
1	HSDIV9	0	NA	WKUP_CLKOUT

Copyright © 2026 Texas Instruments Incorporated

Table 5-47. MAIN_PLL15 (2400MHz)

HSDIV	Value	Frequency(MHz)	Boot Peripheral/IP
HSDIV0	5	400	HSM/SMS
HSDIV1	5	400	SA3_UL PKA
HSDIV2	3	800	DM

Note

All other PLLs are not used or programmed by the ROM

Note

The bringup and configuration of bootmode-specific PLLs by ROM code will result in a bootmode-specific device clocking setup which for example has an impact on which peripheral modules can readily be clocked and used by SBL/SPL prior to loading and bringing up System Firmware (SYSFW)

Copyright © 2026 Texas Instruments Incorporated

5.6 Boot Parameter Tables

The boot parameter tables direct the main module boot process. On cold boot the tables are created based on pin strapped values (see Section 5.3, *Boot Mode Pins*) and built-in data. The ROM Code supports two parameter tables stored as an array in a fixed memory address in the MSRAM, each of size 512 bytes. The ROM will attempt to boot using the primary table. On boot failure, ROM Code will retry using the second table.

Using two tables handles two cases.

- The first is in initial board manufacture where the primary boot table specifies boot from a flash device, and the flash is blank. ROM Code would then switch to the secondary boot mode which would receive the image externally (Ethernet, USB, UART) and this image would flash the boot image.
- The second case is failure due to total flash corruption. In all flash parameter tables there exist backup addresses within the primary boot mode. This covers the problem of a flash update failure with a backup image present on the same flash device.

The boot tables reside at a fixed location in memory which is described in Section 5.8.2, *Global Memory Addresses Used by ROM Code*.

5.6.1 Common Header

These boot parameter tables have certain parameters common across all the boot modes, while the rest of the parameters are unique to the boot modes. The common entries in the boot parameter table are shown in Table 5-48.

Table 5-48. Boot Parameter Table Common Header

Byte Offset	Size (bytes)	Name	Description
0	2	Length	The length of the table
2	2	Checksum	Ones complement checksum over length bytes in the table. If 0 the checksum is not validated.
4	2	Peripheral	Identifies the boot peripheral and format of the table after the common header. See Table 5-49
6	2	Reserved	Reserved
8	4	Timeout	Timeout for this boot mode, in milliseconds
12	4	Magic	Magic value 0x01AD0911
16	40	PLL Config 0	PLL Configuration 0. See Table 5-50
56	40	PLL Config 1	PLL Configuration 1
96	40	PLL Config 2	PLL Configuraiton 2
136	40	PLL Config 3	PLL Configuration 3
176	40	PLL Config 4	PLL Configuration 4
216	40	PLL Config 5	PLL Configuration 5

Table 5-49 lists the possible boot modes used in the boot parameter tables.

Table 5-49. Boot Peripheral Selection

Peripheral Field Value	Description
0	Sleep (No boot)
10	Ethernet Reserved
20	Ethernet BOOTP/TFTP (general)
21	Ethernet RGMII specific
22	Ethernet RMII specific
30	Reserved
31	Reserved
32	Reserved
40	I2C

Copyright © 2026 Texas Instruments Incorporated

Table 5-49. Boot Peripheral Selection (continued)

Peripheral Field Value	Description
50	SPI
60	UART
70	USB DFU
71	USB Reserved
72	USB Host MSC
80	QSPI
85	OSPI
90	Reserved
100	MMCSD Card (general)
101	eMMC (general)
102	MMC 1-bit
103	MMC 4-bit
104	MMC 8-bit
110	GPMC NOR
120	Reserved
130	xSPI
140	GPMC NAND
150	Serial NAND
160	Fast-xSPI

5.6.2 PLL Setup

Table 5-50 through Table 5-54 describe the PLL configuration fields.

Table 5-50. Boot Parameter Table PLL Configuration

Byte Offset	Size (bytes)	Name	Description
0	1	Domain/cfg	See Table 5-51
1	1	Pll number	PLL number indexed from 0.
2	1	Input source	See Table 5-53
3	1	Pll Type	This field must be 1 to indicate an SCPLL
4	4	Input Ref Clock	The PLL input clock, in Q16.16 format
8	4	Feed back divider, integer part	Integer value of feedback divider
12	4	Feed back divider, fractional part	Fractional portion of feedback divider. Total divider is the Integer part + (Fractional part / 2^{24})
16	1	Ref divider	Input clock pre-divider
17	1	Post divider 1	Output post divider 1
18	1	Post divider 2	Output post divider 2
19	1	Reserved	Reserved
20	2	Hsdiv Enable	Bit map. A set bit indicates that the corresponding hsdiv is enabled.
22	2	Reserved	Reserved
24	16	Hsdiv[16]	Array of hs divider values.

Table 5-51. PLL Domain and Enable Configuration

7	6	5	4	3	2	1	0
Reserved		Enable		Reserved			Domain

Copyright © 2026 Texas Instruments Incorporated

Table 5-52. PLL Domain and Enable Field Description

Field	Value	Description
Enable	0	PLL not configured
	1	PLL enabled only if currently disabled or in bypass
	2	PLL is unconditionally enabled. If currently enabled with a different configuration the PLL is first disabled
	3	PLL is unconditionally disabled
Domain	0	PLL is in the MCU domain
	1	PLL is in the MAIN domain

Table 5-53. PLL Reference Source Bit Fields

7	6	5	4	3	2	1	0
Source Type			Source Index				

Table 5-54. PLL Reference Source Field Description

Field	Value	Description
Source Type	0	Source is HFOSC
	1	Source is external pin
	2	Reserved
	3-7	Reserved
Source Index	0-31	Source index (HFOSC[0-31] or pin[0-31], depending on Source Type) HFOSC[0] – WKUP_HFOSC0 HFOSC[1] – HFOSC1 (in MAIN domain) PIN[1] – EXT_REFCLK1 pin (not all PLLs, see *Clocking*

5.6.3 OSPI/QSPI/SPI Boot Parameter Table

Table 5-55 shows the boot parameter table for OSPI, QSPI, or SPI boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-55. OSPI/QSPI/SPI Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode on	From Pins	If non-zero, the mode byte will be sent
258	1	Instruct Width	From Pins	Number of pins used to send instructions (1, 2, 4, 8)
259	1	Address Width	From Pins	Number of pins used to send address (1, 2, 4, 8)
260	1	Data Width	From Pins	Number of pins used to received data (1, 2, 4, 8)
261	1	Address Size	24	24, and 32 bits are the valid address sizes
262	1	Mode	0	OSPI clock polarity and phase mode
263	1	CSEL	From Pins	Chip select number (0–3)
264	1	Read Cmd	From Pins	Command used to read read data
265	1	Mode byte	0	Value used for the mode byte (when active)
266	1	Dummy Cycles	From pins	Number of dummy cycles sent after the read command
267	1	clkRecovery	From pins	Clock recovery
268	1	dqsEnable	0	Enable DQS
269	1	Reserved	0	Reserved
270	2	Module Freq	0	The OSPI module frequency after PLL enable, in kHz. If 0, ROM code uses the value from the module clock tables.
272	4	Bus Frequency	From pins	The OSPI bus frequency, in kHz
276	4	Delay	0x08080808 or 0x01010101	The chip select read delays. Default value is based on the read command

Copyright © 2026 Texas Instruments Incorporated

Table 5-55. OSPI/QSPI/SPI Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
280	4	Tap Delay	0xFFFFFFFF	The read tap selection. If 0xFFFFFFFF, the ROM code will scan the taps to find the best delay. The result will then overwrite the value in this table.
284	4	Internal Clk	From pins	0 = external (dqs) 1 = internal
288	4	notDAC	From pins	When 0, DAC mode is used
292	4	Read Index	0	Index to the active read address (0-1)
296	4	Read Addr 0	0x000000	The initial flash read address
300	4	Read Addr 1	0x400000 (0x4000 SPI)	Backup read address
304	4	Reserved	0	Reserved
308	4	Reserved	0	Reserved

5.6.4 UART Boot Parameter Table

Table 5-56 shows the boot parameter table for UART boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-56. UART Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Magic	0x49	Required Magic value
257	1	Protocol	63 (0x3F)	Specifies the transfer protocol = XMODEM
258	2	Reserved	0	Reserved
260	1	Max error Count	10	Error count resulting in boot abort
261	1	Ack timeout	3	Timeout in seconds on ack
262	1	Char timeout	20	Inter-character timeout in milliseconds
263	1	Reserved	0	Reserved
264	4	Port	From Pins	Physical port number
268	4	Mod Ref Clk	48000	Module reference clock, in kHz
272	4	Data Rate	115200	Baud rate (bps)
276	1	Parity	0	0=none, 1=odd, 2=even
277	1	Data bits	8	Only 8 data bit width is supported
278	1	Stop bits	2	Stop bits in Q7.1 format (2 = 1 stop bit)
279	1	Flow Control	0	0=none, 1= RTS/CTS
280	1	Over sample	16	Only 16× and 13× oversample are supported
281	1	Magic 2	0xB7	Required magic value

5.6.5 I2C Boot Parameter Table

Table 5-57 shows the boot parameter table for I2C boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-57. I2C Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode	From Pins	0x4E = I2C Controller, 0x72 = I2C target
258	1	Dev Addr	From Pins	I2C address when target mode (0x10 or 0x11)
259	1	Reserved	0	Reserved

Copyright © 2026 Texas Instruments Incorporated

Table 5-57. I2C Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
260	4	Mod Clock	0	I2C Module input clock. If 0, it is computed by ROM code.
264	2	Bus Freq	400	I2C Controller mode bus frequency, in kHz
266	2	Bus Addr	From Pins	I2C Controller mode storage device's address (0x50 or 0x51)
268	2	Read Index	0	Index to the active read offset (0 or 1)
270	2	Read Offset 0	0x0000	I2C Controller mode read offset
272	2	Read Offset 1	0x8000	I2C Controller mode backup read offset
274	2	Reserved	0	Reserved
276	2	Reserved	0	Reserved
280	2	Busy Timeout	From pins	Number of µs before a bus recovery is attempted. In units of microseconds in Q3 number format. Value of 0 disables bus recovery attempts.

5.6.6 MMCSD/eMMC Boot Parameter Table

MMCSD/eMMC Boot Parameter Table shows the boot parameter table for eMMC, MMC or SD card boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-58. MMCSD/eMMC Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	From Pins	Physical port number
257	1	eMMC	From Pins	0 = SD/MMC, else eMMC
258	1	bootAck	1 for eMMC 0 for SD/MMC	If 1 and in eMMC mode the controller expects a boot ack from the eMMC
259	1	Media	1 for eMMC 0 for SD/MMC	0 = auto detect card type 1 = MMC 2 = SD
260	1	busWidth	8 for eMMC From Pins for SD/MMC	Number of data pins (1, 4, or 8). If 0, max supported pins of the port are used.
261	1	bootAlt	1 for eMMC 0 for SD/MMC	If 1 and in eMMC mode, the controller uses the alt boot method (CMD1 with arg 0xFFFF_FFFA) to initiate data transfer
262	1	sigVolt	0x18 for port0 0x33 for port1	Sets the signal voltage for the controller. 0x18 = 1.8V 0x33 = 3.3V
263	1	Reserved	0	Reserved
264	4	Max Bus Freq	0	Max bus frequency in kHz. If 0, the value is determined by reading the CSD register from the card (still maxes out at 25 MHz)
268	4	refClkkHz	0	Module reference clock frequency, in kHz. If 0, the ROM code computes the value.
272	4	respTimeout	40000	The timeout period on initial card read, in milli-seconds, Q3 number format
276	128	Filename	"\tiboot3.bin"	For SD/MMC, boot filename in 16-bit Unicode characters (max 64)
404	4	Mode	0x1144D091	0x1144D091 = file system boot 0x1144C180 = raw image boot
408	4	CardIsInit	0	0 = card not yet initialized 1 = card has been initialized
412	4	Rsvd	0	Reserved
416	4	RawIndex	0	Current active read offset (0 or 1)

Copyright © 2026 Texas Instruments Incorporated

Table 5-58. MMCSD/eMMC Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
420	4	Rsvd	0	Reserved
424	4	Raw Offset 0	0x000000	Raw read offset
428	4	Raw Offset 1	0x400000	Backup raw read offset
432	4	Rsvd	0	Reserved
436	4	Rsvd	0	Reserved

5.6.7 Ethernet Boot Parameter Table

Table 5-59 is shown segmented into four sections:

1. The first section contains information required to configure the device hardware
2. The second section contains information used by the top level Ethernet module to execute the boot
3. The third section contains information for the Ethernet stack code.
4. The fourth section contains information for creating the BOOTP packet (vendor string, ID string and debug string), and holds information returned in the BOOTP response (Default route and file name)

Table 5-59 shows the boot parameter table for Ethernet boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-59. Ethernet Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
Hardware Configuration Options				
256	2	Mod Freq	0	Module clock frequency, kHz. If 0, ROM code computes the value.
258	1	Port Num	0	Physical port number
259	1	Interface	From Pins	0 = RGMII with internal delay 1 = RGMII with external delay 2 = RMII
260	1	Init Level	0	0 = Initialize only not enabled modules 1 = Full ethernet sub-system initialization
261	1	Clock out enable	From pins	0x10 = CLKOUT0 enable 0x11 = CLKOUT0 disable
262	1	Clk out freq	From pins	0x20 = CLKOUT0 25 MHz (RGMII) 0x21 = CLKOUT0 50 MHz (RMII)
263	1	RMII Clk In	From pins	0x60 = RMII internal (SoC) clock 0x61 = RMII external clock
264	1	Port Enable	0x01	Bit map. A set bit indicates that the corresponding physical port will be enabled
265	1	Phy Query	From pins	0x30 = speed/duplex determined from RGMII status register 0x31 = MDIO used to query PHY 0x32 = Use fixed speed/duplex values from offset 266/267.
266	1	Speed		0x40 = full speed (1Gbit for RGMII, 100Mbit for RMII) 0x41 = slow speed (100Mbit for RGMII, 10Mbit for RMII)
267	1	Duplex		0x50 = full duplex 0x51 = half duplex
Main Level Boot Control				
268	1	Bootp enable	1	0 = Image information already in this structure 1 = Use BOOTP to get boot image information
269	1	Reserved	0	Reserved
270	2	Bootp Timeout	4000	BOOTP timeout in milli-seconds
272	2	TFTP timeout	1000	TFTP timeout in milli-seconds
274	2	Bootp retries	10	Number of BOOTP retries before fail

Copyright © 2026 Texas Instruments Incorporated

Table 5-59. Ethernet Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
276	2	TFTP retries	10	Number of TFTP retries before fail
278	2	Reserved	0	Reserved
Network Stack Configuration (plus TFTP server ID)				
280	6	MAC Address	From E-fuse	MAC address of the device
286	2	Reserved	0	Reserved
288	4	Device IP	0	IP address of the device. Valid only if BOOTP not enabled
292	4	Net Mask	0	Net mask. Valid only if BOOTP not enabled
296	4	Tftp server IP	0	TFTP server IP. Valid only if BOOTP not enabled
BOOTP send and receive Information				
300	20	Vendor String	"TI K3 Bootp Boot"	BOOTP request vendor string. Valid only if BOOTP not enabled
320	9	Client ID	1-mac-address-0	Client ID. See RFC1700
329	1	ID len	7	Client ID length
330	45	Debug array	SOC ID up to size available	Debug array output
375	1	Debug len	Varies	The number of valid bytes in debug array
376	4	Next hop	0	Next hop IP address. Valid only if bootp not enabled
380	4	Default Route	0	IP default route IP address. Valid only if bootp not enabled
384	128	Boot filename	0	Boot filename. Valid only if bootp not enabled

5.6.8 xSPI/Fast-xSPI Boot Parameter Table

Table 5-60 shows the boot parameter table for xSPI boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-60. xSPI/Fast-xSPI Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	1	Port	0	Physical port number
257	1	Mode on	From Pins	If non-zero, the mode byte will be sent
258	1	Instruct Width	From Pins	Number of pins used to send instructions (1, 8)
259	1	Address Width	From Pins	Number of pins used to send address (1, 8)
260	1	Data Width	From Pins	Number of pins used to received data (1, 8)
261	1	Address Size	24	24 and 32 bits are the valid address sizes
262	1	Mode	0	QSPI clock polarity and phase mode
263	1	CSEL	From Pins	Chip select number (0–3)
264	1	Read Cmd	From Pins	Command used to read read data
265	1	Mode byte	0	Value used for the mode byte (when active)
266	1	Dummy Cycles	From pins	Number of dummy cycles sent after the read command
267	1	clkRecovery	From pins	Clock recovery
268	1	dqsEnable	0	Enable DQS
269	1	ddrEnable	From pins	OSPI DDR mode operation
270	2	Module Freq	0	The QSPI module frequency after PLL enable, in kHz. If 0, ROM code uses the value from the module clock tables.
272	4	Bus Frequency	From pins	The QSPI bus frequency, in kHz
276	4	Delay	0x08080808 or 0x01010101	The chip select read delays. Default value is based on the read command
280	1	SFDP	From Pins	Enables SFDP parser for 1S-1S-1S to 8D-8D-8D switching

Copyright © 2026 Texas Instruments Incorporated

Table 5-60. xSPI/Fast-xSPI Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
282	4	Tap Delay	0xFFFFFFFF	The read tap selection. If 0xFFFFFFFF, the ROM code will scan the taps to find the best delay. The result will then overwrite the value in this table.
286	4	Internal Clk	From pins	0 = external (dqs) 1 = internal
290	4	inDAC	From pins	When 0, XIP mode is used
294	4	Read Index	0	Index to the active read address
298	4	Read Addr 0	0x000000	The initial flash read address
302	4	Read Addr 1	0x400000	Backup read address

5.6.9 USB DFU Boot Parameter Table

USB DFU Boot Parameter Table shows the boot parameter table for USB DFU boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-61. USB DFU Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	Port	From pins	Physical port number. Always set to 0.
260	4	Base address 0	From pins (port)	Base address of USB subsystem (CMN)
264	4	Base address 1	From pins (port)	Base address of controller
268	4	phyBaseAddress	From pins (port)	Base address of USB PHY module
272	4	modRefClkkHz	varies	Module reference clock, in kHz
276	2	Vendor ID	0x0451	USB vendor ID. Read from control registers.
278	2	Product ID	0x6165	USB product ID. Read from control registers.
280	2	BCD Device	0x200	Binary Coded Decimal device release number
284	4	String Table addr	0x4182A76C	Pointer to the string table in RAM
288	2	Vendor String offset	0	Offset to vendor string in string table
290	2	Prod string offset	32	Offset to product string in string table
292	2	Serial num string offset	64	Offset to serial number string in string table
294	2	Timeout	5000	USB timeout in milliseconds
296	2	Mode	1	1 = DFU ≠1 = MSC Mode 1 = backup DFU, 2 = backup MSC
298	1	Lane reverse	From pins	Set to non-zero for lane reverse
299	1	CoreVoltage	From pins	USB Core Voltage 0 = 0.85V, 1 = 0.75V
300	4	Block Size	4096	DFU Block Size

5.6.10 USB MSC Boot Parameter Table

USB MSC Boot Parameter Table shows the boot parameter table for USB boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-62. USB MSC Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	Port	From pins	Physical port number. Always set to 0.
260	4	Base address 0	From pins (port)	Base address of USB subsystem (CMN)
264	4	Base address 1	From pins (port)	Base address of controller
268	4	phyBaseAddress	From pins (port)	Base address of USB PHY module
272	4	modRefClkkHz	varies	Module reference clock, in kHz

Copyright © 2026 Texas Instruments Incorporated

Table 5-62. USB MSC Boot Parameter Table (continued)

Byte Offset	Size (bytes)	Name	Default Value	Description
276	2	Vendor ID	0x0451	USB vendor ID. Read from control registers.
278	2	Product ID	0x6164	USB product ID. Read from control registers.
280	2	BCD Device	0	Binary Coded Decimal device release number
284	4	String Table addr	0x4182BCA8	Pointer to the string table in RAM
288	2	Vendor String offset	0	Offset to vendor string in string table
290	2	Prod string offset	0	Offset to product string in string table
292	2	Serial num string offset	0	Offset to serial number string in string table
294	2	Timeout	5000	USB timeout in milliseconds
296	2	Mode	1	1 = DFU ≠1 = MSC Mode 1 = backup DFU, 2 = backup MSC
298	1	Lane reverse	From pins	Set to non-zero for lane reverse
299	1	CoreVoltage	From pins	USB Core Voltage 0 = 0.85V, 1 = 0.75V
300	64	FileName	tiboot3.bin	Boot file name

5.6.11 GPMC NOR Boot Parameter Table

Table 5-63 shows the boot parameter table for GPMC NOR boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-63. GPMC NOR Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	refClkkHz	0	The module functional clock frequency, in kHz. 0 = ROM code computes the value.
260	4	csSizeMb	64	The size of each chip-select, in MB
264	1	Csel	From pins	The chip-select to use (0-3)
265	1	adMux	From pins	The address/data multiplexing used. 0 = A/D parallel, 1 = A/A/D mux, 2 = A/D mux
266	1	Width	16	Data bus width
267	1	Reserved	0	Reserved
268	4	Read index	0	The currently active read offset (0-1)
272	4	Read offset 0	0x000000	Read address offset 0
276	4	Read offset 1	0x400000	Backup read address offset 1
280	4	Reserved	0	Reserved
284	4	Reserved	0	Reserved

5.6.12 GPMC NAND Boot Parameter Table

GPMC NAND Boot Parameter Table shows the boot parameter table for GPMC NAND boot. Must be preceded with the common boot parameters described in Table 5-48.

Table 5-64. GPMC NAND Boot Parameter Table

Byte Offset	Size (bytes)	Name	Default Value	Description
256	4	Ref Clk kHz	From pins	Specifies the module ref clock.
260	4	Page Size	0	Page size in bytes 2048 or 4096
264	1	Chip Select	0	Chip Select
265	1	Ad Mux	0	Address/data multiplexing

Copyright © 2026 Texas Instruments Incorporated

Table 5-64. GPMC NAND Boot Parameter Table (continued)

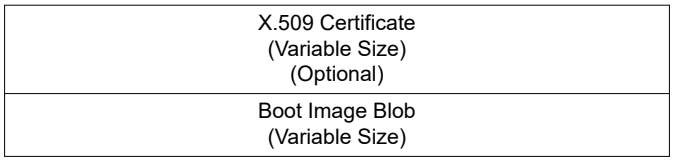
Byte Offset	Size (bytes)	Name	Default Value	Description
266	1	Data Bus Width	8	Data bus width. Valid values are 8 and 16
267	1	AD rows	0	Number of Row address
268	1	AD columns	0	Number of Column address cycles
269	1	ECC Nibbles	0	Size of BCH remainder in nibbles. 0 if no ECC, 26 for BCH8
270	2	Current Valid Block	0xFFFF	Block number of last known good block
272	2	Pages per Block	0	Number of pages in a block
274	2	Reserved	0	Reserved
276	4	Current Read Index	0	Active read index
280	4	Read Offset[0]	0	Offsets from base of GPMC NAND memory
284	4	Read Offset[1]	0x400000	Offsets from base of GPMC NAND memory

Copyright © 2026 Texas Instruments Incorporated

5.7 Boot Image Format

5.7.1 Overall Structure

The boot image consists of an X.509 Certificate, followed immediately by a boot image blob.



5.7.2 X.509 Certificate

The X.509 certificate is described in RFC5280. Section 4.1 of the specification describes the format.

The X.509 fields relevant to the public boot (taken from RFC5280) are shown below.

```
Certificate  ::=  SEQUENCE  {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING  }
   TBSCertificate  ::=  SEQUENCE  {
        version         [0]  EXPLICIT Version DEFAULT v1,
        serialNumber         CertificateSerialNumber,
        signature            AlgorithmIdentifier,
        issuer               Name,
        validity             Validity,
        subject              Name,
        subjectPublicKeyInfo SubjectPublicKeyInfo,
        issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        extensions      [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version MUST be v3
        }
  Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
   Extension  ::=  SEQUENCE  {
        extnID      OBJECT IDENTIFIER,
        critical    BOOLEAN DEFAULT FALSE,
        extnValue   OCTET STRING
                    -- contains the DER encoding of an ASN.1 value
                    -- corresponding to the extension type identified
                    -- by extnID
        }
```

In general, an X.509 certificate contains a public key which has been signed by a private key. The public ROM code does not directly use the keys. In non-secure devices, the public key value is in general a don't care condition. The exception is certificates containing a degenerate RSA public key. GP devices with a degenerate RSA key allow for integrity checking of most (but not all) of the certificate.

The public ROM only needs to extract some of information from the X.509 formatted structure:

- The total size of the X.509 Certificate
- The total size of the boot image

The total size of the X.509 Certificate is determined by reading the length of the sequence containing the certificate. The length of the image is determined by parsing the certificate to find the extension field which holds the image length.

The ROM defines several extensions that are used only by TI for boot. These are placed in the extensions field of the TBS certificate.

5.7.3 Organizational Identifier (OID)

OID (organizational identifier) values are represented as a tree structure. TI has the following node registered:

Copyright © 2026 Texas Instruments Incorporated

1.3.6.1.4.1.294: iso(1), identified-organization(3), dod(6), internet(1), private(4), enterprise(1), Texas Instruments(294)

ROM code adds the following branch after *Texas Instruments*: *device-boot(1)*. The OID values shown in this section are leaves off the device boot branch.

5.7.4 X.509 Extensions Specific to Boot

These values are not defined in any standard, but created by TI for boot.

5.7.4.1 Boot Info (OID 1.3.6.1.4.1.294.1.1)

This extension must be present on all boot images. It is from this extension that the image length is extracted.

```
bootInfo ::=  SEQUENCE {
   cert_type: INTEGER,-- identifies the certificate type
   boot_core:INTEGER,-- identifies the boot core
   core_opts:INTEGER,-- 32 or 64 bit boot core target
   load_addr:OCTET STRING,-- Global address image destination
   image_size:INTEGER,-- Image size in bytes
}
```

Table 5-65. Certificate Type Values

Value	Description
0x0000_0001	Primary boot image
0x0000_0002	Firmware image

Table 5-66. Boot Core Values

Value	Description
0x00	Firmware (M4) image
0x08	M4 certificate
0x10	R5 image
0x20	Reserved

Table 5-67. Core Options Bit Fields

31	2	1	0
Reserved		Split	Mode

Table 5-68. Core Options Field Description

Bits	Field	Value	Description
1	Split	0	Dual MCU set to lockstep (two cores in lockstep)
		1	Dual MCU set to split mode (two independent cores)
0	Mode	0	MCU starts execution in Arm® mode
		1	MCU starts execution in Thumb® mode

5.7.4.2 Image Integrity (OID 1.3.6.1.4.1.294.1.2)

```
imageIntegrity ::= SEQUENCE {
   sha_type:OID,-- Identifies the SHA type
   hash:OCTET STRING-- The SHA of the boot image
}
```

5.7.5 Extended Boot Info Extension

The ROM supports a combined boot image boot flow. In this flow, a boot binary blob has both Secondary bootloader (SBL) and System Firmware (SYS-FW) embedded in the boot image with a single X509 certificate. This method helps with the following situations:

Copyright © 2026 Texas Instruments Incorporated

- Allows ROM to load and run both the bootloader and SYS-FW in parallel without any dependency.
- Optimizes ROM boot time by minimizing different x509 certificate parsing and authentication.

To support this combined boot format, ROM employs a new X509 extension called: ext_boot_info. It supports multiple boot components with a single certificate. It allows up to 5 components as part of this extension:

- Component1: Mandatory and should point to info about SBL binary
- Component2: Optional and if present should point to SYS-FW binary in all device types
- Component3: Optional (Load section to SBL, new certType)
- Component4: Optional (Load section to SYS-FW, new certType)
- Component5:
  - HS-FS and HS-SE non Prime devices. Mandatory and should point to SYS-FW Inner certificate
  - GP and HS-SE Prime devices. Optional (Load Section to SBL or SYS-FW)

This extended boot info extension replaces boot_seq (boot_info) and image_integrity extensions from the previous sections, and these should be exclusive in any given certificate.

ROM supports other extensions, such as sw_rev and debug_info, in both formats.

ROM selects the combined image flow based on the presence of ext_boot_info extension in the certificate and skips boot_seq (boot_info) and image_integrity extensions boot flow.

Having one component in ext_boot_info is same as legacy flow (that is, flow using boot_seq and image_integrity); for two or more components, ROM starts both SBL and SYS-SW, the third and fourth components are loaded by ROM to the allowed loading memory range of SBL and SYS-FW if there is no overlap in load address with executable binary info.

Each of the components can independently specify hash value of the binary, and ROM validates the hash on HS and GP if RSA degenerate key is used for signing.

Additionally, ROM rejects the full image if hash of any single component mismatches.

5.7.5.1 Impact on HS Device

For HS devices, ROM supports encryption of images optionally when specified with the Encryption extenstion.

Encryption - Encryption of the components must be specified with a new extension called ext_boot_enc. This is an optional extension that must be specified only if any of the components are encrypted with a customer key set. Specify a component number followed by encryption extension details for ROM to decrypt the given components. This extension is used only with combined boot image format (that is, using Extended Boot Info Extension).

Prime vs Non-Prime - The main difference between HS prime and non-prime devices is the presence of the SYS-FW Inner certificate. Extended boot info extension supports both prime and non-prime devices. ROM supports SYS-FW binary as Comp#2 in all device types; for HS-SE non-prime and HS-FS devices, the SYS-FW inner certificate is expected as another optional component in #3, 4, or 5.

5.7.5.2 Extended Boot Info Details

```
1.3.6.1.4.1.294.1.9=ASN1:SEQUENCE:ext_boot_info
[ ext_boot_info ]
 extImgSize = INTEGER:470656
 numComp = INTEGER:4
 sbl=SEQUENCE:comp1
 fw=SEQUENCE:comp2
 bd1=SEQUENCE:comp3
 bd2=SEQUENCE:comp4

[ comp1 ]
 compType = INTEGER:1
 bootCore = INTEGER:16
 compOpts = INTEGER:0
 destAddr = FORMAT:HEX,OCT:41c00000
 compSize = INTEGER:237376
 shaType = OID:2.16.840.1.101.3.4.2.3
```


Copyright © 2026 Texas Instruments Incorporated

```
 shaValue =
FORMAT:HEX,OCT:6779fdb2b2c27169737c184085b97938bd77bdf698245840f166ca30c7125c29a6675139a25a0a2a3f00a
76d43d082df238c12cb6b293ec0eeb5990bcd603a23

[ comp2 ]
 compType = INTEGER:2
 bootCore = INTEGER:0
 compOpts = INTEGER:0
 destAddr = FORMAT:HEX,OCT:00040000
 compSize = INTEGER:196608
 shaType = OID:2.16.840.1.101.3.4.2.3
 shaValue =
FORMAT:HEX,OCT:47f27fb24b81927a928845a4c2b993e6a3d5bdfe5ed01c0ec4f96a7ead991c69bf4ed4b9b958fd36a75f3
3aba04d2c28602a85ca737ee75617d6a9a41f4353a3

[ comp3 ]
 compType = INTEGER:18
 bootCore = INTEGER:0
 compOpts = INTEGER:0
 destAddr = FORMAT:HEX,OCT:00072000
 compSize = INTEGER:16384
 shaType = OID:2.16.840.1.101.3.4.2.3
 shaValue =
FORMAT:HEX,OCT:2d165ec1d8a38acb977d9298e8a6a27491c62d6daa31f921db9135f9a68779b30b384573c6e4e8203de5f
0a47191c3ff9a35b52a911874e07615f10b4b2f5829

[ comp4 ]
 compType = INTEGER:17
 bootCore = INTEGER:16
 compOpts = INTEGER:0
 destAddr = FORMAT:HEX,OCT:41c40000
 compSize = INTEGER:20288
 shaType = OID:2.16.840.1.101.3.4.2.3
 shaValue =
FORMAT:HEX,OCT:12d5be5b2b9774d1b0cad21cbf1dbcdbd7c310657f4334902e3cc6228cf2d8fc844139dae4db6041c38cd
a502d6a900d57039322d360032268a13445021b04a7
```

5.7.5.3 Certificate / Component Types

ROM supports only the following component types; all other component types will be rejected.

- CompType 0x1 for SBL binary - compType = INTEGER:1
- CompType 0x2 for SYS_FW binary - compType = INTEGER:2
- CompType 0x3 for SYS_FW Inner Certificate - compType = INTEGER:3
- CompType 0x11 for SBL Memory load section - compType = INTEGER:17

Primary Image load section, that can be loaded in to SBL allowed memory range:

- CompType 0x12 for SYS_FW Memory load section - compType = INTEGER:18

SYS-FW load section, that can be loaded in to SYS-FW allowed memory range

5.7.5.4 Extended Boot Encryption Info

Extended Boot Encryption Info specifies the imageEncryption extension per component basis. This extension is not applicable for GP or HS-FS devices, and is optional for HS-SE devices if any components in the image are encrypted. In HS-FS and HS-SE non-prime devices, the SYS-FW binary encryption details are part of SYS-FW Inner certificate. This extension is applicable if either SBL, SYS-FW binary in HS-SE prime, or any binary blobs are encrypted in HS-SE devices.

```
1.3.6.1.4.1.294.1.10=ASN1:SEQUENCE:ext_enc_info

[ ext_enc_info ]
 numComp = INTEGER:2
 esbl=SEQUENCE:enc1
 efw=SEQUENCE:enc2

[ enc1 ]
 compNum = INTEGER:1
 iv = FORMAT:HEX,OCT:474bfd801866beecc7ab6d4c61490e1a
 randString = FORMAT:HEX,OCT:772fc5810fa36f516e595ad8adf19260f47a8461f193892746692fbb932727a1
 iterationCnt = INTEGER:0
```


Copyright © 2026 Texas Instruments Incorporated

```
 salt = FORMAT:HEX,OCT:42ea40851298339c8baa84f29d6b68d0

[ enc2 ]
 compNum = INTEGER:2
 iv = FORMAT:HEX,OCT:aaaaaaaaaaaabeecc7ab6d4c61490e1a
 randString = FORMAT:HEX,OCT:bbbbbbbbbbbbbbbbbe595ad8adf19260f47a8461f193892746692fbb932727a1
 iterationCnt = INTEGER:2
 salt = FORMAT:HEX,OCT:ccccccccccccccccccaa84f29d6b68d0
```

5.7.5.5 Component Ordering

ROM supports fixed Component ordering for SBL- Binary and SYS-FW Cert and Binary.

For GP and HS-SE Prime devices, Comp#1 should be SBL binary, and Comp#2 should be SYS-FW binary.

For HS-FS and HS-SE non-Prime devices, Comp#1 should be SBL binary, Comp#2 should be SYS-FW Inner Certificate, and Comp#3 should be SYS-FW binary.

5.7.5.6 Memory Load Sections Overlap with Executable Components

Memory load sections for SBL and SYS-FW (CompType 0x11 and 0x12) the destination address and size should not overlap with the corresponding load and execute sections, and should also fall in the allowed memory range by ROM.

5.7.5.7 Device Type and Extended Boot Extension

X509 Extension	Component	GP Device	HS-FS	HS-SE non-prime	HS-SE Prime Device
ext_boot_info	Comp#1	SBL Binary	SBL Binary	SBL Binary	SBL Binary
	Comp#2	SYS-FW binary	SYS-FW binary	SYS-FW binary	SYS-FW binary
	Comp#3	SBL mem load section	SBL mem load section	SBL mem load section	SBL mem load section
	Comp#4	SysFW mem load section	SysFW mem load section	SysFW mem load section	SysFW mem load section
	Comp#5	N/A	SysFW inner certificate	SysFW inner certificate	N/A
ext_enc_info	Comp#1	N/A	N/A1	SBL encryption	SBL encryption
	Comp#2			N/A	SYS FW encryption
	Comp#3			SBL memory load Encryption	SBL memory load encryption
	Comp#4			Sys-FW memory load encryption	Sys-FW memory load encryption
	Comp#5			N/A1	NA

1. SYS-FW encryption in HS-FS and HS-SE (non Prime) are handled in SYS-FW Inner certificate and they are not part of the extended boot info extensions.

5.7.6 Generating X.509 Certificates

X.509 Certificates are generated using OpenSSL and a configuration script to supply values in the extension fields.

5.7.6.1 Key Generation

The SBL must always be signed with a given OpenSSL key - Secure and GP devices. Secure must have encryption and authentication, GP device must only have authentication. The key used for authentication can be random or specific. If a random key is generated and SBL is signed with this,the ROM will copy the SBL image for authentication using memcopy. With this key, ROM code will be directed to use DMA to load the SBL for authentication which saves boot time.

5.7.6.1.1 Degenerate RSA Keys

Degenerate RSA keys are valid RSA keys with the private exponent set to 1. This results in the signature field being equal to the digest, since in RSA:

Copyright © 2026 Texas Instruments Incorporated

$$Signature = digestprivExp \bmod nprivExp \bmod n^{privExp} \quad (1)$$

Where n is the key size. Since the hash used is SHA-512 and the signature is an ASN.1 sequence containing the OID defining which has was used as well as the hash value, the degenerate RSA must have a value of n greater than the maximum digest size. Typically 1024-bit is chosen.

The following sequence is used to generate degenerate RSA keys:

1. Create a random RSA key:

```
openssl genrsa –out key.pm 1024
```

2. Convert to text:

```
openssl rsa –in key.pem –text –noout > key.txt
```

3. Create an asn1 template for the degenerate key called degenerateKey.txt. Simply copy the values for modulus, prime (listed as p in key.txt), prime 2 (listed as q), and coefficient (listed as coeff). Set the public and private key exponents to 1, as well as the values for e1 and e2. See the example below.
4. Convert the template to DER:

```
openssl asn1parse -genconf degenerateKey.txt -out degenerateKey.der
```

5. Sanity check the key:

```
 openssl rsa –in degenerateKey.der –inform der -text -check
```

6. If there are no errors create the degenerate key pem file:

```
openssl rsa –in degenerateKey.der –inform der –outform pem -out degenerateKey.pem
```

An example degenerateKey.txt file is shown.

```
asn1=SEQUENCE:rsa_key
[rsa_key]
version=INTEGER:0
modulus=INTEGER<copied from key.txt>
pubExp=INTEGER:1
privExp=INTEGER:1
p=INTEGER:<copied from key.txt>
q=INTEGER<copied from key.txt>
e1=INTEGER:1
e2=INTEGER:1
coeff=INTEGER<copied from key.txt>
```

Note that when copying the multi-byte fields from key.txt it is necessary to remove the colons, catenate the lines and add a preceding 0x.

5.7.6.2 Configuration Script

An example openssl configuration script is shown below. Not all extensions are required, but all possible are shown.

```
[ req ]
distinguished_name = req_distinguished_name
X509_extensions = v3_ca
prompt = no
dirstring_type = nobmp
[ req_distinguished_name ]
C = GB
ST = HI
L = Boston
O = Texas Instruments., Inc.
OU = DSP
CN = Bob
```


Copyright © 2026 Texas Instruments Incorporated

```
emailAddress = Bob@hou.ti.com
[ v3_ca ]
basicConstraints = CA:true
1.3.6.1.4.1.294.1.1 = ASN1:SEQUENCE:boot_seq
1.3.6.1.4.1.294.1.2 = ASN1:SEQUENCE:image_integrity
1.3.6.1.4.1.294.1.3 = ASN1:SEQUENCE:swrv
1.3.6.1.4.1.294.1.4 = ASN1:SEQUENCE:encryption
1.3.6.1.4.1.294.1.5 = ASN1:SEQUENCE:key_derivation
1.3.6.1.4.1.294.1.7 = ANSI:SEQUENCE:pllControl
1.3.6.1.4.1.294.1.8 = ANSI:SEQUENCE:debug
[ boot_seq ]
certType =INTEGER:1
bootCore = INTEGER:16
bootArchWidth = INTEGER:32
destAddr = FORMAT:HEX,OCT:bc934b00
imageSize = INTEGER:0x00004860
[ image_integrity ]
shaType = OID:1.3.14.3.2.26
shaValue = FORMAT:HEX,OCT:4cf4d59ef77b5d9ab28d2ceb3c9fe83cb52ae6d2
[ swrv ]
rollback = INTEGER:0x00010001
[ encryption ]
Iv =FORMAT:HEX,OCT:00112233445566778899aabbccddeeff
Rstring = FORMAT:HEX,OCT:00112233445566778899aabbccddeeff101112131415161718191a1b1c1d1e1f
Icount = INTEGER:1
Salt = FORMAT:HEX,OCT:00112233445566778899aabbccddeeff
[ pllControl ]
pll0_num = INTEGER:0
pll0_cfg = FORMAT:HEX,OCT:00345678900
pll1_num = INTEGER:1
pll0_cfg = FORMAT:HEX,OCT:00345678900
[ debug ]
uid = FORMAT:HEX,OCT:00345678900
type = INTEGER:1
dbgE = INTEGER:0
secDbgEn = INTEGER:0
```

The certificate is then generated using the following openssl command:

```
openssl req -new -x509 -key <private_key_pem_file> -nodes -out <output_X.509_pem_file> -config
<config_file> -sha512
```

If a delegate key is being signed, then add the option -signkey <sign_key_pem_file> to the command above.

5.7.6.3 Image Data

The image data (blob) is considered simply as a byte stream. On devices that are multiple bytes wide the image must be formatted so that all multi-byte fields match the endianness of the device. The R5 will always run in little endian mode.

Copyright © 2026 Texas Instruments Incorporated

5.8 Boot Memory Maps

5.8.1 Memory Layout/MPU

Memory Layout/MPU shows an overview of the MPU configuration. In the R5 MPU, higher numbered regions have priority, therefore in the table, where two regions overlap, the region on the right defines the memory attributes.

5.8.2 Global Memory Addresses Used by ROM Code

The ROM code uses several global memory addresses that are useful for debugging. They are shown in Global Memory Addresses.

Table 5-69. Global Memory Addresses

Group	Address	Size (bytes)	Content
SMS0_HSM_SRAM0_0	0x43c00000	0x7e000	Loadable memory space for SBL
Warning/Error logs	0x43c7e480	0x200	Warning Entries
Warning/Error logs	0x43c7e680	0x200	Severe Entries
Warning/Error logs	0x43c7e880	0x100	Critical Entries
Warning/Error logs	0x43c7eb80	0x14	Circular message buffer
Warning/Error logs	0x43c7eb98	0x50	Boot log context
Trace	0x43c7ebe8	0x18	Boot trace context
Trace	0x43c7ec00	0x400	Boot trace entry buffers
Parameter tables	0x43c7f290	0x4	Parameter tables index for R5 bootloader
Parameter tables	0x43c7f298	0x400	Boot Parameter table for R5 bootloader
Parameter tables	0x43c7f1e0	0xb0	Extended boot data for R5 bootloader

The ROM code version information is a structure shown in Table 5-70

Table 5-70. ROM Code Version

Field	Address	Size (bytes)	Value for PG1
Version Number	0x4182_FF80	0x4	TBD
Version Date	0x4182_FF84	0x8	"TBD"
Device Name	0x4182_FF8C	0xC	"TBD"
Commit ID	0x4182_FF98	0x28	"TBD"

Copyright © 2026 Texas Instruments Incorporated

Chapter 6

Device Configuration

6.1 Control Module 414
6.2 Power 418
6.3 Reset 459
6.4 Clocking 499

Copyright © 2026 Texas Instruments Incorporated

6.1 Control Module

Note

Some features may not be available. See *Module Integration* for more information.

6.1.1 Memory Mapped Control Register Modules (CTRL_MMR)

This chapter covers three categories of device level control register modules:

- General-Purpose Control Register Modules
- Pad Configuration Register Modules
- Security Control Register Modules

Each category of module is described in its own section, along with detailed register descriptions for each module.

6.1.2 General Purpose Control Register Modules Overview

There are three General Purpose Control Register Modules on the Device:

- CTRL_MMR0 for the MAIN domain
- WKUP_CTRL_MMR0 for the WKUP domain
- MCU_CTRL_MMR0 shared between the WKUP and MCU domains, with Parity Protection.

6.1.2.1 Description for General Purpose Control Register Module Register Types

The following sub-sections provide summary and description for most of the registers which are part of the CTRL_MMR0, WKUP_CTRL_MMR0 and MCU_CTRL_MMR0 modules memory spaces. The rest of the registers are described in the corresponding chapters where the functionality associated with particular register is covered in more detail.

6.1.2.1.1 Register Partitions

The CTRL_MMR0 memory space is divided into the partitions shown in Table 6-1.

Table 6-1. CTRL_MMR0 Register Partitions

Partition(1)	Description	Proxy0 Offset Range	Proxy1 Offset Range
Partition 0	General configuration and IPC regs	0000h to 1FFFh	2000h to 3FFFh
Partition 1	IP control/status regs	4000h to 5FFFh	6000h to 7FFFh
Partition 2	Clock control/status regs	8000h to 9FFFh	A000h to BFFFh
Partition 6	Power and reset control and status	1 8000h to 1 9FFFh	1 A000h to 1 BFFFh

(1) Not listed partitions are reserved and not used.

The MCU_CTRL_MMR0 memory space is divided into the partitions shown in Table 6-2.

Table 6-2. MCU_CTRL_MMR0 Register Partitions

Partition(1)	Description	Proxy0 Offset Range	Proxy1 Offset Range
Partition 0	General configuration and IPC regs	0000h to 1FFFh	2000h to 3FFFh
Partition 1	IP control/status regs	4000h to 5FFFh	6000h to 7FFFh
Partition 2	Clock control/status regs	8000h to 9FFFh	A000h to BFFFh
Partition 3	BIST control/status regs	C000h to DFFFh	E000h to FFFFh
Partition 4	Verification and test/control status regs	1 0000h to 1 1FFFh	1 2000h to 1 3FFFh
Partition 6	Power and reset control and status	1 8000h to 1 9FFFh	1 A000h to 1 BFFFh

(1) Not listed partitions are reserved and not used.

The WKUP_CTRL_MMR0 memory space is divided into the partitions shown in Table 6-3.

Copyright © 2026 Texas Instruments Incorporated

Table 6-3. WKUP_CTRL_MMR0 Register Partitions

Partition[1]	Description	Proxy0 Offset Range	Proxy1 Offset Range
Partition 0	General configuration and IPC regs	0000h to 1FFFh	2000h to 3FFFh
Partition 1	IP control/status regs	4000h to 5FFFh	6000h to 7FFFh
Partition 2	Clock control/status regs	8000h to 9FFFh	A000h to BFFFh
Partition 3	EFUSE CRC status regs	C000h to DFFFh	E000h to FFFFh
Partition 4	Verification and test control/status regs	1 0000h to 1 1FFFh	1 2000h to 1 3FFFh
Partition 5	DDR frequency Control	1 4000h to 1 5FFFh	1 6000h to 1 7FFFh
Partition 6	Power and reset control and status	1 8000h to 1 9FFFh	1 A000h to 1 BFFFh
Partition 7	Backup Data for Deep Sleep	1 C000h to 1 DFFFh	1 E000h to 1 FFFFh

(1) Not listed partitions are reserved and not used.

6.1.2.1.2 Pad Configuration Registers

The pad configuration registers are used to configure most of the device pads. Each pad configuration register is associated with one pad and has bits as described in *Pad Configuration Register Functional Description*.

Many of the device pads support pad multiplexing. This means that their function can be independently chosen from two or more options. The selection of functions available on each pad is enumerated in table “Pin Multiplexing” of the device-specific Datasheet. The desired function is selected via the MUXMODE field of the associated pad configuration register.

The PADCFG_CTRL0_MMR_CFG0_PADCONFIGx registers control the signal multiplexing of modules in the device MAIN domain.

Note

It is recommended to first configure the PADCONFIG and debounce, as well as the debounce counter values in the SoC MMR, prior to enabling the IP.

6.1.2.1.3 Kick Protection Registers

There are Kick and Proxy registers associated with each CTRL_MMR register partition. This section describes the Kick registers. For information about the Proxy registers see Proxy Addressing Registers.

The Kick registers provide a protection mechanism which prevents spurious writes from changing the values of its registers. The LOCKi_KICK0 and LOCKi_KICK1 registers are used for this purpose. A write is required first to the LOCKi_KICK0[31-1] KEY field and then to the LOCKi_KICK1[31-0] KEY field with exact data values to unlock the protection mechanism. Once released then all registers within Partition "i" having write permissions can be written to. The read only registers are still read only. An indication for unlocked Partition "i" is when the LOCKi_KICK0[0] UNLOCKED bit is set to 1h. When the protection mechanism is locked (indicated by LOCKi_KICK0[0] UNLOCKED = 0h) none of the registers within Partition "i" can be written to. They can only be read..

The key values for all partitions are:

- LOCKi_KICK0 Key: 68EF 3490h
- LOCKi_KICK1 Key: D172 BC5Ah

See *Kick Protection Registers* for Partition Unlock Values.

Writing any other data value to either of these registers locks the protection mechanism and blocks any writes to the registers that reside in Partition "i".

Note

In order to ensure that all registers from all partitions are write protected, software must always re-lock the protection mechanism after completing the register writes.

Copyright © 2026 Texas Instruments Incorporated

6.1.2.1.4 Proxy Addressing Registers

There are Kick and Proxy registers associated with each CTRL_MMR partition. This section describes the Proxy registers. For information about the Kick registers see Kick Protection Registers.

The Proxy registers provide proxy addressing, allowing each of the CTRL_MMR registers to be accessed through two different addresses - Proxy0 and Proxy1. Each partition is 16KB. The first 8KB are for Proxy0 addresses, and the second 8KB are for Proxy1 addresses. The Proxy0 address is intended as the normal read and write access address. The Proxy1 address provides exclusive register write control.

When a bit from the P**x**_CLAIM**y**[31-0] PROXY1_CLAIMED field is set to 1h, a register becomes read-only at its Proxy0 address and may only be written through its Proxy1 address. The Proxy registers themselves may always be read through their Proxy0 address but may only be written through their Proxy1 address.

The **x** (representing the partition) and **y** indices can be calculated through the following equations:

- **x** = INT(HEX2DEC(address_offset)/16384))
- **y** = INT(MOD(HEX2DEC(address_offset),16384)/128)

where:

- INT() rounds down to the nearest integer
- HEX2DEC() converts from hexadecimal to decimal
- MOD() returns the reminder of HEX2DEC(address_offset)/16384

6.1.3 I/O Debounce Control Registers

Some device pads have debounce logic. The following MCU_CTRL_MMR0 registers are used to configure the debounce period as shown in Table 6-4.

Table 6-4. Debounce period

MCU_CTRL_MMR Debounce Register	PADCONFIGx[13-11] DEBOUNCE_SEL setting
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG1	1h
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG2	2h
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG3	3h
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG4	4h
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG5	5h
MCU_CTRL_MMR_CFG0_DBOUNCE_CFG6	6h

Note

The debounce logic is not associated with all signals that can be multiplexed on a pad. Only certain signals can use it.

For information about each signal that has associated debounce logic, see Signal Descriptions in the device Datasheet.

Note

It is recommended to first configure the PADCONFIG and debounce, as well as the debounce counter values in the SoC MMR, prior to enabling the IP.

For information about the PADCONFIGx registers, see *Pad Configuration Registers*.

Table 6-5 shows the debounce period values to load in the MCU_CTRL_MMR_CFG0_DBOUNCE_CFGx[5-0] DB_CFG field.

Copyright © 2026 Texas Instruments Incorporated

Note

The debounce clock selection should happen before the consumer of the debounced signals is enabled as the clock multiplexers for the debounce logic are NOT glitch-free.

Table 6-5. Debounce Period Values

MCU_CTRL_MMR_CFG0_DBOUNCE_CFGx[5:0] DB_CFG Decimal Value	Slow Inputs (32.768kHz (1))	Fast Inputs (25MHz(1))	MCU_CTRL_MMR_CFG0_DBOUNCE_CFGx[5:0] DB_CFG Decimal Value	Slow Inputs (25MHz(1))	Fast Inputs (250MHz(1))
	Delay[ms]	Delay[µs]		Delay[µs]	Delay[ns]
0	bypassed	n/a	32	0.04	4
1	1.95	2.56	33	0.08	8
2	2.93	3.84	34	0.12	12
3	3.91	5.12	35	0.16	16
4	4.88	6.40	36	0.20	20
5	5.86	7.68	37	0.24	24
6	6.84	8.96	38	0.28	28
7	7.81	10.24	39	0.32	32
8	8.79	11.52	40	0.36	36
9	9.77	12.80	41	0.40	40
10	10.74	14.08	42	0.44	44
11	11.72	15.36	43	0.48	48
12	12.7	16.64	44	0.52	52
13	13.67	17.92	45	0.56	56
14	14.65	19.20	46	0.60	60
15	15.63	20.48	47	0.64	64
16	16.6	21.76	48	0.68	68
17	17.58	23.04	49	0.72	72
18	18.55	24.32	50	0.76	76
19	19.53	25.60	51	0.80	80
20	20.51	26.88	52	0.84	84
21	21.48	28.16	53	0.88	88
	Delay[ms]	Delay[µs]		Delay[µs]	Delay[µs]
22	15.63	20.48	54	20.48	2.048
23	31.25	40.96	55	40.96	4.096
24	46.88	61.44	56	61.44	6.144
25	62.5	81.92	57	81.92	8.192
26	78.13	102.40	58	102.40	10.24
27	93.75	122.88	59	122.88	12.288
28	109.38	143.36	60	143.36	14.336
29	125	163.84	61	163.84	16.384
30	140.63	184.32	62	184.32	18.432
31	156.25	204.80	63	204.80	20.48

(1) These are the debounce logic clock frequencies.

Copyright © 2026 Texas Instruments Incorporated

6.1.4 Security Control Register Modules

The following sections describe the CTRL_MMR0 and PADCFG_CTRL0_CFG0 modules.

6.1.4.1 Security Control Register Modules Overview

There are three Security Control Register Modules on the Device:

- MAIN_SEC_MMR0 for the MAIN domain
- MCU_SEC_MMR0 for the MCU domain
- WKUP_SEC_MMR0 for the WKUP domain

6.2 Power

This chapter describes the power-management architecture implemented in the device.

6.2.1 Power Management Overview

To provide a versatile architecture that supports multiple power-management techniques, the power management framework is built with three levels of resource management: clock, power, and voltage.

These management levels are enforced by defining the managed entities or building blocks of the power management architecture, called the clock, power, and voltage domains. A domain is a group of modules or subsections of the device that share a common entity (for example, common clock source, common voltage source, or common power switch).

To minimize device power consumption, the clocks can be gated (turned off), or the power to the modules can be switched off, when they are not in use. Independent power and clock control of sections of the device allows for turning on and off specific sections of the device without affecting other sections.

6.2.2 Power Control Modules

The Power Control Modules are divided into two sections, dependent on their functionality - Power Sleep Controller section and Device Manager section.

6.2.2.1 Power Sleep Controller (PSC)

6.2.2.1.1 PSC Architecture

Power Sleep Controller (PSC) is a control component to manage the power domain and module's clock and reset transition. Each PSC module contains multiple power domains, within each power domain, it can contain multiple LPSC. Each LPSC can be used to manage the one or multiple module's reset and clock stop status, see Figure 6-1.

Each power domain has a unique index, starting from number 0. The power domain 0 is a special power domain, which is always on. The status of power domain 0 can't be changed after the device is out of reset.

Each power domain can have one or more Local Power Sleep Control (LPSC). Each LPSC in the same PSC has a unique index value as well. LPSC 0 is special, which user can't change its status. LPSC 0 is always set to be enabled after the device is out of reset.

One LPSC can be used to control the reset and clock status for one or more modules. Some of the LPSC can be used to provide local reset function for the processors, such as A53 and R5 core. LPSC can also be used to manage the clock stop interface for the interfaces between two interconnect IPs. This is mainly used for providing isolation features required by some special features such as security isolation, functional safety isolation or reset isolation between different parts of the SoC.

Copyright © 2026 Texas Instruments Incorporated

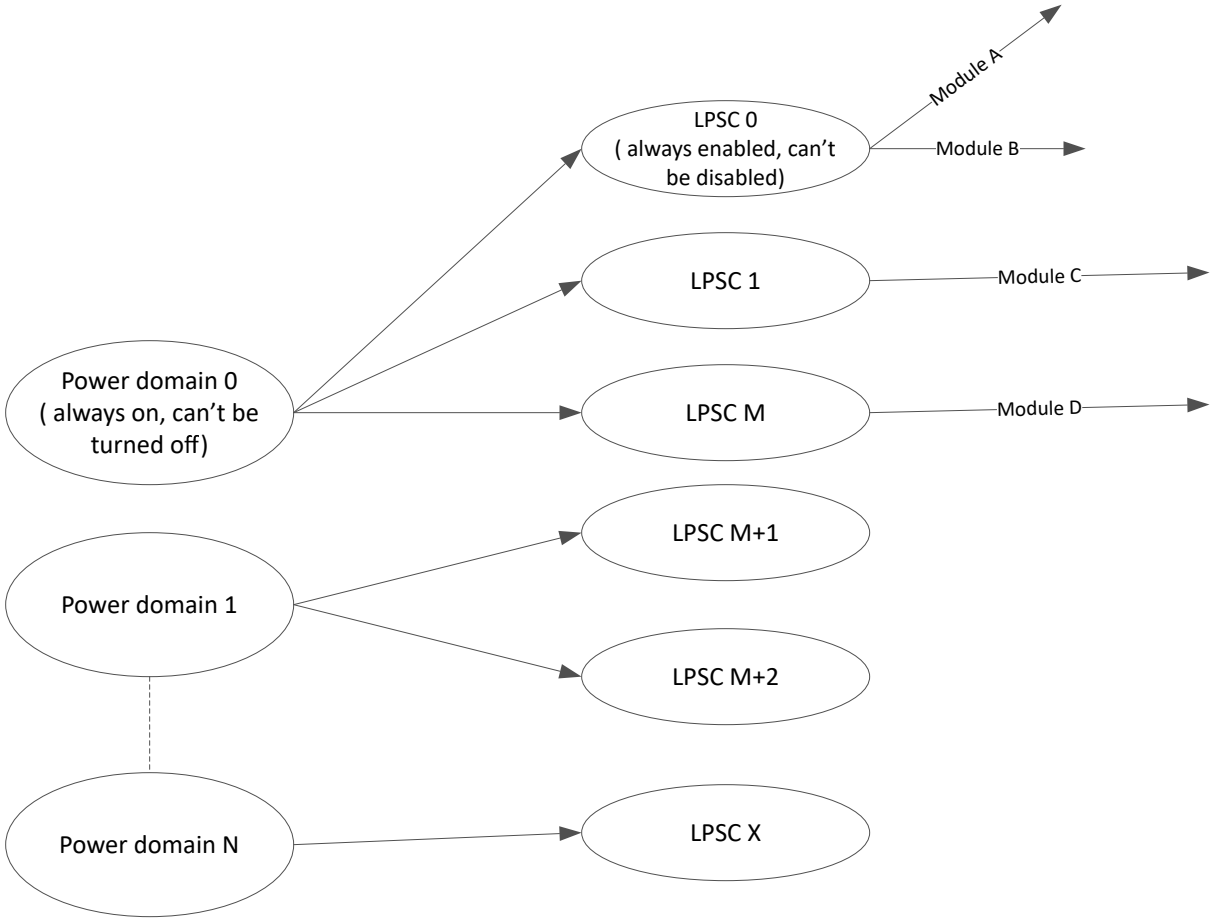


Figure 6-1. PSC Architecture

Note

PSC functions are controlled via Device Manager. For more information how to use Device Manager, see TISCI API available at ti.com.

6.2.2.1.2 PSC0 LPSC Features

Table 6-6. PSC0 LPSC Features

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	MAIN_GPIOMUX_INTROUTER0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	PLL0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	TIMESYNC_EVENT_INTROUTER0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_WKUP_SEC_MMR0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	PADCFG_CTRL0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	CPT2_AGGR1
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC1
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC2
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC3
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC4
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC5
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC6
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC7

Copyright © 2026 Texas Instruments Incorporated

Table 6-6. PSC0 LPSC Features (continued)

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DCC8
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_TIMER0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_TIMER1
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	ESM0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	GPIO0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	GPIO1
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_GTC0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	CTRL_MMR0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_CTRL_MMR0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	DDPA0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_VTM0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_I2C0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	PSRAMECC0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	PSRAMECC1
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_PSRAMECC_8K0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_ROM0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_RTCSS0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	EFUSE0
PSC0	GP_CORE	LPSC_MAIN_ALWAYSON	0	ON	NO	WKUP_UART0
PSC0	GP_CORE	LPSC_MAIN_DM	1	OFF	YES	WKUP_R5FSS0_CORE0
PSC0	GP_CORE	LPSC_MAIN_DM	1	OFF	YES	WKUP_RTI0
PSC0	GP_CORE	LPSC_MAIN_DM_PBIST0	2	ON	YES	WKUP_PBIST0
PSC0	GP_CORE	LPSC_MAIN_DM_PBIST1	8	ON	YES	WKUP_PBIST1
PSC0	GP_CORE	LPSC_MAIN_CSI_RX1	9	OFF	YES	CSI_RX_IF1
PSC0	GP_CORE	LPSC_MAIN_DPHY_RX1	10	OFF	YES	DPHY_RX1
PSC0	GP_CORE	LPSC_MAIN_CSI_TX0	11	OFF	YES	CSI_TX_IF0
PSC0	GP_CORE	LPSC_MAIN_TEST	14	ON	YES	DFTSS0
PSC0	GP_CORE	LPSC_MAIN_GPMC	15	OFF	YES	ELM0
PSC0	GP_CORE	LPSC_MAIN_GPMC	15	OFF	YES	GPMC0
PSC0	GP_CORE	LPSC_MAIN_MCANSS1	16	OFF	YES	MCAN1
PSC0	GP_CORE	LPSC_MAIN_MCASP0	17	OFF	YES	MCASP0
PSC0	GP_CORE	LPSC_MAIN_MCASP1	18	OFF	YES	MCASP1
PSC0	GP_CORE	LPSC_MAIN_MCASP2	19	OFF	YES	MCASP2
PSC0	GP_CORE	LPSC_MAIN_EMMC8B	20	OFF	YES	MMCSD0
PSC0	GP_CORE	LPSC_MAIN_EMMC4B0	21	OFF	YES	MMCSD1
PSC0	GP_CORE	LPSC_MAIN_EMMC4B1	22	OFF	YES	MMCSD2
PSC0	GP_CORE	LPSC_MAIN_USB0	23	OFF	YES	USB0
PSC0	GP_CORE	LPSC_MAIN_CSI_RX0	25	OFF	YES	CSI_RX_IF0
PSC0	GP_CORE	LPSC_MAIN_DPHY_RX0	26	OFF	YES	DPHY_RX0
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAIN_SEC_MMR0
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_0
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_1
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_2
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_3
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_4
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_5
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_6
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	MAILBOX0_MAILBOX_CLUSTER_7
PSC0	GP_CORE	LPSC_MAIN_SMS_COMMON	27	ON	YES	SPINLOCK0

Copyright © 2026 Texas Instruments Incorporated

Table 6-6. PSC0 LPSC Features (continued)

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
PSC0	GP_CORE	LPSC_MAIN_FSS_OSPI	28	ON	YES	FSS0_FSAS_0
PSC0	GP_CORE	LPSC_MAIN_FSS_OSPI	28	ON	YES	FSS0_OSPI_0
PSC0	GP_CORE	LPSC_MAIN_SA3UL	31	ON	YES	SA3_SS0_DMSS_HSM_0
PSC0	GP_CORE	LPSC_MAIN_SA3UL	31	ON	YES	SA3_SS0_SA_UL_0
PSC0	GP_CORE	LPSC_MAIN_DEBUGSS	33	ON	YES	DBGSUSPENDROUTER0
PSC0	GP_CORE	LPSC_MAIN_DEBUGSS	33	ON	YES	STM0
PSC0	GP_CORE	LPSC_MAIN_DEBUGSS	33	ON	YES	DEBUGSS_WRAP0
PSC0	GP_CORE	LPSC_MAIN_DEBUGSS	33	ON	YES	DEBUGSS0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	CPT2_AGGR0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_BCDMA_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_CBASS_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_ECC_AGGR_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_INTAGGR_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_IPCSS_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_PKTDMA_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_PSILCFG_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_PSILSS_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_RINGACC_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS0_SEC_PROXY_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS1_BCDMA_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS1_ECC_AGGR_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS1_INTAGGR_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	DMASS1_PSILCFG_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER3
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER4
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER5
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER6
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	TIMER7
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	ECAP0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	ECAP1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	ECAP2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EQEP0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EQEP1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EQEP2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EPWM0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EPWM1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	EPWM2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	MCRC64_0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	I2C0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	I2C1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	I2C2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	I2C3
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	I2C4
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	MSRAM8KX256E0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	PDMA2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	PDMA1

Copyright © 2026 Texas Instruments Incorporated

Table 6-6. PSC0 LPSC Features (continued)

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	PDMA3
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	PDMA0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	MCSPI0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	MCSPI1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	MCSPI2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART0
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART1
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART2
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART3
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART4
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART5
PSC0	GP_CORE	LPSC_MAIN_IP	34	ON	YES	UART6
PSC0	GP_CORE	LPSC_MAIN_MCANSS0	35	OFF	YES	MCAN0
PSC0	GP_CORE	LPSC_MAIN_GIC	36	ON	YES	GICSS0
PSC0	GP_CORE	LPSC_MAIN_PBIST0	37	ON	YES	PBIST0
PSC0	GP_CORE	LPSC_MAIN_DPHY_TX0	38	OFF	YES	DPHY_TX0
PSC0	GP_CORE	LPSC_MAIN_JPEG	39	OFF	YES	JPGENC0
PSC0	GP_CORE	LPSC_MAIN_SERDES0	40	OFF	YES	SERDES_10G0
PSC0	GP_CORE	LPSC_MAIN_SERDES1	41	OFF	YES	SERDES_10G1
PSC0	GP_CORE	LPSC_MAIN_CPSW	42	OFF	YES	CPSW0
PSC0	GP_CORE	LPSC_MAIN_GPCORE_RSVD2	43	OFF	YES	ATL0
PSC0	GP_CORE	LPSC_MAIN_CSI_RX3	44	OFF	YES	CSI_RX_IF3
PSC0	GP_CORE	LPSC_MAIN_DPHY_RX3	45	OFF	YES	DPHY_RX3
PSC0	GP_CORE	LPSC_MAIN_PBIST1	46	ON	YES	PBIST1
PSC0	GP_CORE	LPSC_MAIN_USB1	47	OFF	YES	USB1
PSC0	GP_CORE	LPSC_MAIN_CSI_RX2	48	OFF	YES	CSI_RX_IF2
PSC0	GP_CORE	LPSC_MAIN_CSI_DPHY_RX2	49	OFF	YES	DPHY_RX2
PSC0	GP_CORE	LPSC_MAIN_GPCORE_RSVD3	50	OFF	YES	MCASP3
PSC0	GP_CORE	LPSC_MAIN_GPCORE_RSVD3	50	OFF	YES	MCASP4
PSC0	PD_MPU_CLST0	LPSC_MAIN_MPU_CLST0	53	OFF	YES	A53SS0
PSC0	PD_MPU_CLST0	LPSC_MAIN_MPU_CLST0_PBIST0	54	OFF	YES	COMPUTE_CLUSTER0_PBIST_0
PSC0	PD_MPU_CLST0_CORE0	LPSC_MAIN_MPU_CLST0_CORE0	56	OFF	YES	RTI0
PSC0	PD_MPU_CLST0_CORE0	LPSC_MAIN_MPU_CLST0_CORE0	56	OFF	YES	A53SS0_CORE_0
PSC0	PD_MPU_CLST0_CORE1	LPSC_MAIN_MPU_CLST0_CORE1	57	OFF	YES	RTI1
PSC0	PD_MPU_CLST0_CORE1	LPSC_MAIN_MPU_CLST0_CORE1	57	OFF	YES	A53SS0_CORE_1
PSC0	PD_MPU_CLST0_CORE2	LPSC_MAIN_MPU_CLST0_CORE2	58	OFF	YES	RTI2
PSC0	PD_MPU_CLST0_CORE2	LPSC_MAIN_MPU_CLST0_CORE2	58	OFF	YES	A53SS0_CORE_2
PSC0	PD_MPU_CLST0_CORE3	LPSC_MAIN_MPU_CLST0_CORE3	59	OFF	YES	RTI3
PSC0	PD_MPU_CLST0_CORE3	LPSC_MAIN_MPU_CLST0_CORE3	59	OFF	YES	A53SS0_CORE_3
PSC0	PD_GPU_CTRL	LPSC_MAIN_GPU_CTRL	60	OFF	YES	RTI15
PSC0	PD_GPU_CTRL	LPSC_MAIN_GPU_CTRL	60	OFF	YES	GPU0
PSC0	PD_GPU_CTRL	LPSC_MAIN_GPU_PBIST	61	OFF	YES	GPU0
PSC0	PD_CODEC	LPSC_MAIN_CODEC	65	OFF	YES	CODEC0
PSC0	PD_CODEC	LPSC_MAIN_CODEC_PBIST	66	OFF	YES	PBIST3
PSC0	PD_C7DSP0	LPSC_MAIN_C7DSP0_CORE	67	OFF	YES	RTI4
PSC0	PD_C7DSP0	LPSC_MAIN_C7DSP0_CORE	67	OFF	YES	C7X256V0_C7XV_CORE_0
PSC0	PD_C7DSP0	LPSC_MAIN_C7DSP0_PBIST	68	OFF	YES	C7X256V0_PBIST
PSC0	PD_C7DSP0	LPSC_MAIN_C7DSP0_COMMON	69	OFF	YES	C7X256V0_CORE0
PSC0	PD_VPAC	LPSC_MAIN_VPAC	70	OFF	YES	VPAC0

Copyright © 2026 Texas Instruments Incorporated

Table 6-6. PSC0 LPSC Features (continued)

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
PSC0	PD_VPAC	LPSC_MAIN_VPAC_PBIST	71	OFF	YES	VPAC0
PSC0	PD_DDR	LPSC_MAIN_EMIF_LOCAL	72	OFF	YES	DDR32SS0
PSC0	PD_PCIE	LPSC_MAIN_PCIE0	75	OFF	YES	PCIE0
PSC0	PD_C7DSP1	LPSC_MAIN_C7DSP1_CORE	76	OFF	YES	RTI5
PSC0	PD_C7DSP1	LPSC_MAIN_C7DSP1_CORE	76	OFF	YES	C7X256V1_C7XV_CORE_0
PSC0	PD_C7DSP1	LPSC_MAIN_C7DSP1_PBIST	77	OFF	YES	C7X256V1_PBIST
PSC0	PD_C7DSP1	LPSC_MAIN_C7DSP1_COMMON	78	OFF	YES	C7X256V1_CORE0
PSC0	PD_MAIN_MCUSS0	LPSC_MAIN_MCUSS0_CORE0	79	OFF	YES	R5FSS0_CORE0
PSC0	PD_MAIN_MCUSS0	LPSC_MAIN_MCUSS0_CORE0	79	OFF	YES	RTI8
PSC0	PD_MAIN_MCUSS0	LPSC_MAIN_MCUSS0_PBIST	80	OFF	YES	PBIST2
PSC0	PD_RSVD0	LPSC_PDRSVD0_RSVD0	85	OFF	YES	DMPAC0
PSC0	PD_RSVD0	LPSC_PDRSVD0_RSVD1	86	OFF	YES	DMPAC0
PSC0	PD_RSVD0	LPSC_PDRSVD0_RSVD2	87	OFF	YES	DMPAC0
PSC0	PD_DSS	LPSC_MAIN_DSS0	91	OFF	YES	DSS0
PSC0	PD_DSS	LPSC_MAIN_DSS1	92	OFF	YES	DSS1
PSC0	PD_DSS	LPSC_MAIN_DSS_DSI0	93	OFF	YES	DSS_DSI0

6.2.2.1.3 WKUP_PSC0 LPSC Features

Table 6-7. WKUP_PSC0 LPSC Features

PSC	Power Domain Name	LPSC Name	LPSC Index	Default State	Software Controllable	Modules
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_CTRL_MMR0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	WKUP_MCU_GPIOMUX_INTROUTER0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	WKUP_PLL0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_MCU_SEC_MMR0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_PADCFG_CTRL0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_DCC0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_DCC1
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	WKUP_ESM0
WKUP_PSC0	GP_CORE_CTL_MCU	LPSC_MCU_ALWAYSON	0	ON	NO	MCU_GPIO0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_R5	6	OFF	YES	MCU_R5FSS0_CORE0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_R5	6	OFF	YES	MCU_RTI0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_MCANSS_0	7	OFF	YES	MCU_MCAN0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_MCANSS_1	8	OFF	YES	MCU_MCAN1
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_CPT2_AGGR0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_TIMER0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_TIMER1
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_TIMER2
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_TIMER3
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_MCRC64_0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_I2C0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_MSRAM_256K0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_MSRAM_256K1
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_MCSPI0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_MCSPI1
WKUP_PSC0	PD_MCUSS	LPSC_MCU_COMMON	9	ON	YES	MCU_UART0
WKUP_PSC0	PD_MCUSS	LPSC_MCU_PBIST	10	ON	YES	MCU_PBIST0

Copyright © 2026 Texas Instruments Incorporated

6.2.2.1.4 Software Sequence to enable an IP through PSC

In order to enable a module, user needs to program the PSC to put LPSC into enable state. There is dependence between power domain and LPSC and among different LPSC controls.

First of all, in order to enable any LPSC, the power domain which that LPSC belongs needs to be enabled first. The LPSC can't put into any state other than disable if the power domain is not enabled.

Among different LPSC, there is also a special sequence needs to be followed to make sure the device is not getting into undefined state.

Similarly, a special sequence shall be followed to disable the LPSC to prevent device getting into undefined states. Before a power domain can be turned off, all the LPSCs within that power domain shall be disabled first. Normally, user can turn off the power domain to save power. However, it is not necessary to turn off the power domain when all the LPSCs inside that power domain are disabled.

Figure 6-2 shows how to understand the dependent relationship between two LPSCs. In order to enable LPSC_2, LPSC_1 shall be enabled first.

If user wants to disable LPSC_1, LPSC_2 shall be disabled first.

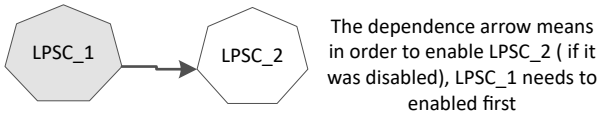


Figure 6-2. LPSC Dependence

Figure 6-3 shows the dependent relationship among all the LPSCs in the device.

Copyright © 2026 Texas Instruments Incorporated

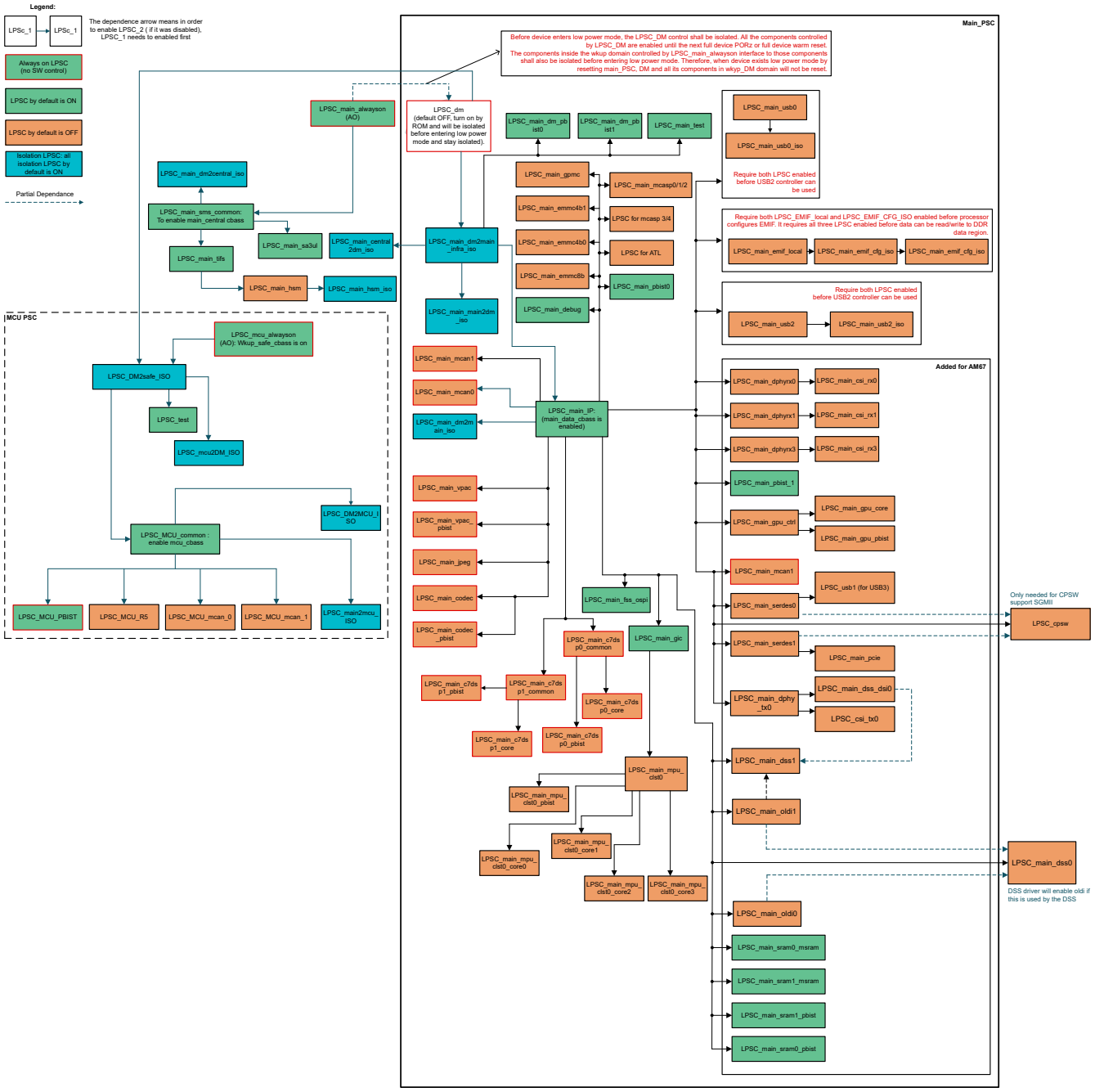


Figure 6-3. LPSC Dependency Tree

6.2.2.2 Device Manager

The device has one Device Manager located in WKUP domain. Device Manager is responsible for entry/exit to low power modes. Also, Device Manager hosts RM/PM (Resource Manager/Power Manager) SYSFW code for providing RM/PM services to HLOS. As part of boot, public ROM is executed over Device Manager.

Device Manager Features

- Single Core Cortex-R5 subsystem
  - ARMv7-R architecture with following extensions

Copyright © 2026 Texas Instruments Incorporated

    - Advanced SIMD extension for integer and floating-point vector operations
    - Vector Floating Point Version 3 (VFPv3)
  - L1 memory architecture
    - 32KB I-cache with 64-bit ECC error checking
    - 32KB D-cache with 64-bit ECC error checking
  - 64Kbyte TCM(L2) with 32-bit ECC error checking
  - 32-bit AXI3 initiator interface for accessing local peripherals
- 1x RTI/WWDT
- 2x Timer
- 1x UART
- 1x I2C
- Wake-up logic
- Public ROM
- RTC digital
- VTM
- LFOSC

Note

PSC functions are controlled via Device Manager. For more information how to use Device Manager, see TISCI API available at ti.com.

6.2.3 Power Management Techniques

The following section describes the power management techniques supported by the processor.

6.2.3.1 Dynamic Frequency Scaling (DFS)

DFS is a power management technique where the operating frequency is dynamically scaled across device Operating Performance Points (OPP). An OPP is a voltage/frequency pair that defines a specific power consumption operating condition (specific power state.) For each OPP, software controls clock frequency in order to adjust the performance and power to the optimum point. The Device supports DFS for Cortex-A53 only.

6.2.3.2 OPP Low

The device supports lower bus frequency operation as OPP Low. The OPP Low must be configured at boot time. In OPP Low, the main CBASS clock frequency is reduced in half in order to lower active power consumption with reduced performance. The performance of some peripheral modules is limited or not available in this operating condition.

Refer to the device datasheet to determine the supported features and the clock frequency of cores, modules, and peripheral I/Os in OPP Low.

Copyright © 2026 Texas Instruments Incorporated

6.2.4 Power Management Subsystems

This section provides an overview of how three Power blocks work together; the three blocks are: Power On Reset module (POR), Power OK module (POK), and the POK Reset Generator – Ping-Pong module (PRG_PP) and a PRG without Ping-Pong (PRG).

6.2.4.1 Power Subsystems Overview

6.2.4.1.1 POK Overview

POK modules are responsible for accurately detecting average voltage levels.

Two types of POK modules are implemented in this family of devices - POK and POK_SA.

POK module is capable of monitoring a range of supplies and indicating a failure within the programmable upper and lower threshold limits for the supply being monitored. POKs are used to monitor 3.3 V, 1.8 V and core supply levels with programmable threshold levels.

Figure 6-4 shows the concept of the POK. It is a comparator with a fixed 0.45V and the voltage to be monitored. The voltage-to-be-monitored is defined based upon a CTRL register (see *POK Related Registers*) and the silicon hook-up of the POK. The POK has three possible input voltage ports – VDD_MON3P3, VDD_MON1P8, and VDD_MON. The effect of the CTRL register depends upon which of these three ports is selected (defined in Table 6-12); for example, a threshold value of 0x28 in the threshold register would create a different threshold voltage depending upon:

- How the POK was hooked-up (i.e. VDD_MON3P3, VDD_MON1P8, or VDD_MON) see *POK Modules and Monitored Voltages*.
- Whether the POK is selected for undervoltage detection or overvoltage detection as determined by bit 7 in the POK CTRL register (e.g. CTRL_MMR_POR_VDDR_MCU_UV_CTRL)

The POK output triggers high (i.e. indicating a fail condition) when:

- Under-voltage: the monitored voltage (divided based upon CTRL registers) is less than 0.45V. The CTRL register tables translate the input voltage back to a monitored voltage level (see Table 6-12). If the monitored voltage falls below the trip point, the POK outputs HIGH, (i.e. FAIL).
- Over-voltage: the monitored voltage (divided based upon CTRL registers) is greater than 0.45V. The CTRL register tables translate the input voltage back to a monitored voltage level (see Table 6-12). If the monitored voltage rises above the trip point, the POK outputs HIGH, (i.e. FAIL).

When the monitored voltage threshold is changed, the new threshold settles within 9us.

Figure 6-4 also shows a mux on the output of the comparator. The mux is configured based upon the selection of under-voltage or over-voltage and is used to maintain the proper polarity of the fail signature.

Note

POK_SA (used only on VMON1_ER_VSYS) directly compares the monitored voltage to 0.45V.

Two other features of the POK diagram (i.e. Figure 6-4 and Figure 6-5) need some explanation:

- The comparator can be enabled or disabled individually; note, that once enabled, the comparator output is continuous (though as shown below in is sampled and filtered). The comparator enable settling time is 15us.
- Hysteresis is applied to the comparator input. In general, it is expected that the monitored voltage is far away from the reference and that the hysteresis is moot.

Copyright © 2026 Texas Instruments Incorporated

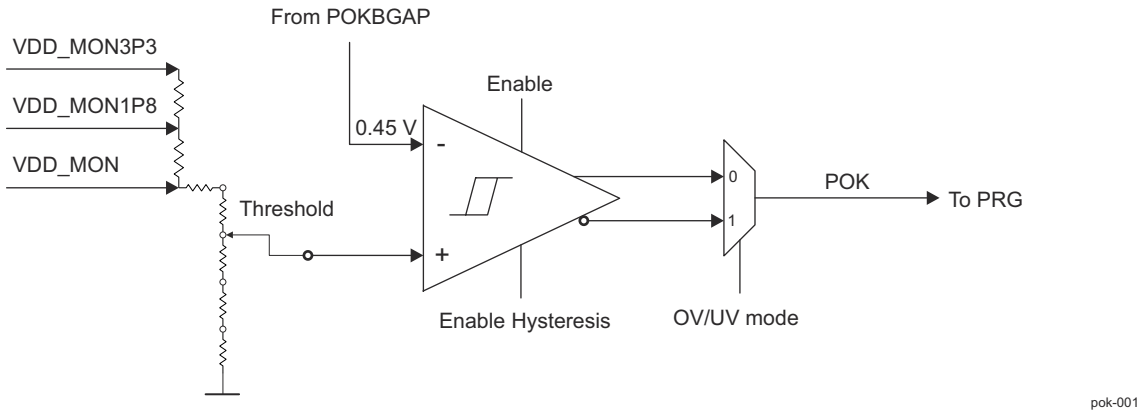


Figure 6-4. POK Block Diagram

Table 6-8. POK Programmable Features

Programmable Feature	Bitfield in the POK Dedicated Register
POK hysteresis enable	[31] HYST_EN
POK over- or under-voltage detection mode	[7] OVER_VOLT_DET
POK trim bits for voltage comparator threshold	[6-0] POK_TRIM

Like the POK, the POK_SA is a comparator with 0.45V reference; however, voltage division must occur in hardware on the PCB before the monitored voltage is applied to VMON1_ER_VSYS.

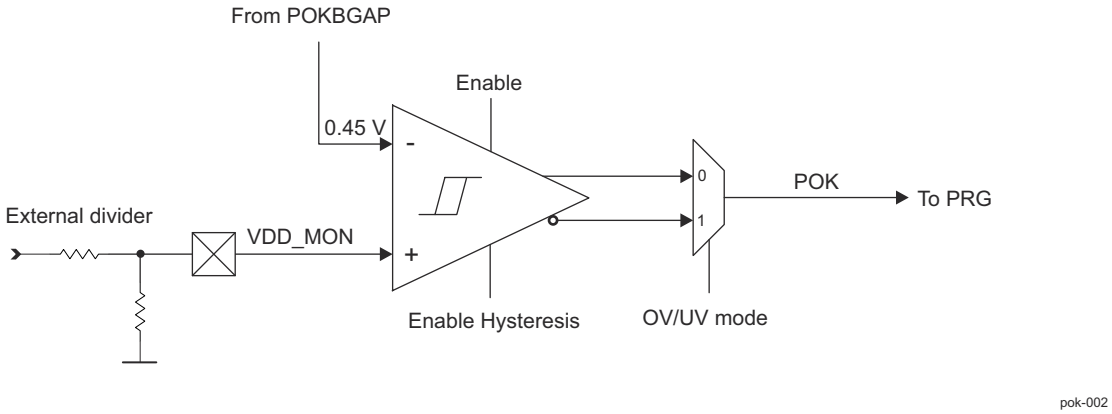


Figure 6-5. POK_SA Block Diagram

Table 6-9. POK_SA Programmable Features

Programmable Feature	Bitfield in the POK Dedicated Register
POK hysteresis enable	[31] HYST_EN
POK over- or under-voltage detection mode	[0] OVER_VOLT_DET

Note

POK configuration is not designed to find and report an instantaneous dip / rise in the voltage.

Copyright © 2026 Texas Instruments Incorporated

Table 6-10. POK Module Overview

POK Module Instance Name	PRG Index (3)	POK Index (1)	Voltage Monitored	Module Type	OV/ UV(2)	Description
IPOK_VDD_CORE_OV	PRG0	4	VDD_CORE	POK_CORE	OV	Voltage Monitor for VDD_CORE
IPOK_VDDA_PMIC_IN_UV	PRG0	3	VMON_ER_VSYS	POK_SA	UV	Voltage Monitor for PMIC_IN
IPOK_VDDS_DDRIO	PRG1	4	VDDS_DDR	POK_CORE	UV/OV	Voltage Monitors for DDRIO
IPOK_VDDR_CORE	PRG1	3	VDDR_CORE	POK_CORE	UV/OV	Voltage Monitors for VDDR_CORE
IPOK_VDDS_CAP_MCU	PRG1	2	CAP_VDDS_MCU	POK_CORE	UV/OV	Voltage Monitors for MCU mid-rail
IPOK_VDDS_MCU_1P8	PRG1	1	VMON_1P8_SOC	POK_1P8	UV/OV	Voltage Monitors for 1.8-V I/O supply in MCU
IPOK_VDDSHV_MCU_3P3	PRG1	0	VMON_3P3_SOC	POK_3P3	UV/OV	Voltage Monitors for 3.3-V I/O supply in MCU

(1) The POK Index also represents the mapping index for relevant under and overvoltage status and interrupts where applicable.
(2) UV: POK is set for undervoltage detection. OV: POK is set for overvoltage detection. UV/OV: POK monitors both OV and UV through a Ping-Pong mechanism.
(3) Ping-pong mechanism is only applicable to PRG1.

Table 6-11. POK Related Registers

Module Instance	Register
PRG_POR	
POR_POKHV	CTRL_MMR_POR_POKHV_UV_CTRL
POR_POKLVA	CTRL_MMR_POR_POKLVA_OV_CTRL
POR_POKLVB	CTRL_MMR_POR_POKLVB_UV_CTRL
IPOK_VDD_MCU_OV	CTRL_MMR_POK_VDD_MCU_OV_CTRL
IPOK_VDDA_PMIC_IN	CTRL_MMR_POK_VDDA_PMIC_IN_CTRL
PRG_PP_MCU	
IPOK_VDDR_MCU	CTRL_MMR_POK_VDDR_MCU_UV_CTRL CTRL_MMR_POK_VDDR_MCU_OV_CTRL
IPOK_VDDSHV_WKUP_GEN	CTRL_MMR_POK_VDDSHV_WKUP_GEN_UV_CTRL CTRL_MMR_POK_VDDSHV_WKUP_GEN_OV_CTRL
IPOK_VMON_CAP_VDDS _MCU_GEN	CTRL_MMR_POK_VMON_CAP_MCU_GEN_UV_CTRL CTRL_MMR_POK_VMON_CAP_MCU_GEN_OV_CTRL
PRG_PP_MAIN	
IPOK_VDD_CORE	CTRL_MMR_POK_VDD_CORE_UV_CTRL CTRL_MMR_POK_VDD_CORE_OV_CTRL
IPOK_VDDR_CORE	CTRL_MMR_POK_VDDR_CORE_UV_CTRL CTRL_MMR_POK_VDDR_CORE_OV_CTRL
IPOK_VDD_CPU	CTRL_MMR_POK_VDD_CPU_UV_CTRL CTRL_MMR_POK_VDD_CPU_OV_CTRL
IPOK_VMON_EXT	CTRL_MMR_POK_VMON_EXT_UV_CTRL CTRL_MMR_POK_VMON_EXT_OV_CTRL
IPOK_VMON_EXT_1P8	CTRL_MMR_POK_VMON_EXT_MAIN1P8_UV_CTRL CTRL_MMR_POK_VMON_EXT_MAIN1P8_OV_CTRL
IPOK_VMON_EXT_3P3	CTRL_MMR_POK_VMON_EXT_MAIN3P3_UV_CTRL CTRL_MMR_POK_VMON_EXT_MAIN3P3_OV_CTRL
IPOK_VDD_CPU1	CTRL_MMR_POK_VDD_CPU1_UV_CTRL CTRL_MMR_POK_VDD_CPU1_OV_CTRL

Copyright © 2026 Texas Instruments Incorporated

The possible values of the monitored voltage differ among the POK types, see *POK Modules and Monitored Voltages*.

Table 6-12. POK Threshold Levels

POK_TRIM [6:0]	CORE_POK UV (OVER_VOLT _DET = 0)	CORE_POK OV (OVER_VOLT _DET = 1)	1P8_POK UV (OVER_VOLT _DET = 0)	1P8_POK OV (OVER_VOLT _DET = 1)	3P3_POK UV (OVER_VOLT _DET = 0)	3P3_POK OV (OVER_VOLT _DET = 1)
0x00	0.475 V	0.725 V			1.425 V	2.175 V
0x01	0.4875 V	0x7375 V			1.4625 V	2.2125 V
0x02	0.50 V	0.75 V			1.5 V	2.25 V
…						
0xC				1.432 V		
0xD				1.452 V		
0xE				1.473 V		
…						
0x20			1.432 V			
0x21			1.452 V			
0x22			1.473 V			
…	0.475 V + POK_TRIM[7:0] *0.0125 V	0.725 V + POK_TRIM[7:0] *0.0125 V	0.7775 V + 0.02045* POK_TRIM[7:0]	1.1865 V + 0.02045* POK_TRIM[7:0]	1.425 V + POK_TRIM[7:0] *0.0375 V	2.175 V + POK_TRIM[7:0] *0.0375 V
0x2D						3.8625
0x2E				2.127V		3.9
0x2F				2.148 V		3.9375
0x30				2.168 V		
…						
0x42			2.127 V		3.9 V	
0x43			2.148 V		3.9375 V	
0x44	1.325 V		2.168 V		3.975 V	
0x45	1.3375 V					
0x46	1.35 V					
…						
0x48		1.625 V				
0x49		1.6375 V				
0x4A		1.65 V				

6.2.4.1.1.1 Configuration Registers

Configuration registers needed for POK (Supervisor Module) are given in Table 6-13. For each POK there is one comparator that is controlled through the registers. These registers decide the functionality of the POK Comparator. It decides if it is UV/OV mode and also the threshold voltages.

Comparator threshold levels and corresponding configuration register settings are given in the following tables:

- POK_CORE: Table 6-14
- POK_1P8: Table 6-15
- POK_3P3: Table 6-16

Table 6-13. POK Configuration Registers

Register Name	POK Type
MCU_CTRL_MMR_CFG0_POK_VDD_CORE_OV_CTRL[7:0]	POK_CORE
MCU_CTRL_MMR_CFG0_POK_VDDR_CORE_UV_CTRL[7:0]	POK_CORE

Copyright © 2026 Texas Instruments Incorporated

Table 6-13. POK Configuration Registers (continued)

Register Name	POK Type
MCU_CTRL_MMR_CFG0_POK_VDDR_CORE_OV_CTRL[7:0]	POK_CORE
MCU_CTRL_MMR_CFG0_POK_VMON_CAP_MCU_GENERAL_UV_CTRL[7:0]	POK_1P8
MCU_CTRL_MMR_CFG0_POK_VMON_CAP_MCU_GENERAL_OV_CTRL[7:0]	POK_1P8
MCU_CTRL_MMR_CFG0_POK_VDDSHV_MAIN_1P8_UV_CTRL[7:0]	POK_1P8
MCU_CTRL_MMR_CFG0_POK_VDDSHV_MAIN_1P8_OV_CTRL[7:0]	POK_1P8
MCU_CTRL_MMR_CFG0_POK_VDDSHV_MAIN_3P3_UV_CTRL[7:0]	POK_3P3
MCU_CTRL_MMR_CFG0_POK_VDDSHV_MAIN_3P3_OV_CTRL[7:0]	POK_3P3
MCU_CTRL_MMR_CFG0_POK_VDDS_DDRIO_UV_CTRL[7:0]	POK_CORE
MCU_CTRL_MMR_CFG0_POK_VDDS_DDRIO_OV_CTRL[7:0]	POK_CORE

Copyright © 2026 Texas Instruments Incorporated

Table 6-14. Bit Settings of POK_CORE with VMON Threshold Values

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**474mV - 1.35V**
Over Voltage Detection	**1**	**XXX XXXX**	**726mV - 1.645V**
POK_CORE VMON Selection (Under Voltage Detection) hys_en=0	0	000 0000	0.4743
	0	000 0001	0.4875
	0	000 0010	0.5000
	0	000 0011	0.5125
	0	000 0100	0.5250
	0	000 0101	0.5375
	0	000 0110	0.5513
	0	000 0111	0.5631
	0	000 1000	0.5754
	0	000 1001	0.5883
	0	000 1010	0.6017
	0	000 1011	0.6122
	0	000 1100	0.6268
	0	000 1101	0.6382
	0	000 1110	0.6500
	0	000 1111	0.6623
	0	001 0000	0.6750
	0	001 0001	0.6882
	0	001 0010	0.7020
	0	001 0011	0.7115
	0	001 0100	0.7262
	0	001 0101	0.7363
	0	001 0110	0.7521
	0	001 0111	0.7630
	0	001 1000	0.7742
	0	001 1001	0.7858
	0	001 1010	0.7977
	0	001 1011	0.8100
	0	001 1100	0.8226
	0	001 1101	0.8356
	0	001 1110	0.8491
	0	001 1111	0.8630
	0	010 0000	0.8774
	0	010 0001	0.8848
	0	010 0010	0.8999
	0	010 0011	0.9155
	0	010 0100	0.9235
	0	010 0101	0.9400
	0	010 0110	0.9485
	0	010 0111	0.9659
	0	010 1000	0.9748
	0	010 1001	0.9839

Copyright © 2026 Texas Instruments Incorporated

Table 6-14. Bit Settings of POK_CORE with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**474mV - 1.35V**
Over Voltage Detection	**1**	**XXX XXXX**	**726mV - 1.645V**
POK_CORE VMON Selection (Under Voltage Detection) hys_en=0	0	010 1010	1.0030
	0	010 1011	1.0120
	0	010 1100	1.0220
	0	010 1101	1.0320
	0	010 1110	1.0530
	0	010 1111	1.0630
	0	011 0000	1.0740
	0	011 0001	1.0850
	0	011 0010	1.0970
	0	011 0011	1.1140
	0	011 0100	1.1260
	0	011 0101	1.1380
	0	011 0110	1.1510
	0	011 0111	1.1630
	0	011 1000	1.1760
	0	011 1001	1.1900
	0	011 1010	1.1960
	0	011 1011	1.2100
	0	011 1100	1.2240
	0	011 1101	1.2380
	0	011 1110	1.2530
	0	011 1111	1.2610
	0	100 0000	1.2760
	0	100 0001	1.2920
	0	100 0010	1.3000
	0	100 0011	1.3160
	0	100 0100	1.3240
	0	100 0101	1.3410
	0	100 0110	1.3500
POK_CORE VMON Selection (Over Voltage Detection) hys_en=0	1	000 0000	0.7262
	1	000 0001	0.7363
	1	000 0010	0.7521
	1	000 0011	0.7630
	1	000 0100	0.7743
	1	000 0101	0.7858
	1	000 0110	0.7977
	1	000 0111	0.8100
	1	000 1000	0.8226
	1	000 1001	0.8357
	1	000 1010	0.8491
	1	000 1011	0.8630
	1	000 1100	0.8774

Copyright © 2026 Texas Instruments Incorporated

Table 6-14. Bit Settings of POK_CORE with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**474mV - 1.35V**
Over Voltage Detection	**1**	**XXX XXXX**	**726mV - 1.645V**
POK_CORE VMON Selection (Over Voltage Detection) hys_en=0	1	000 1101	0.8848
	1	000 1110	0.8999
	1	000 1111	0.9155
	1	001 0000	0.9236
	1	001 0001	0.9400
	1	001 0010	0.9485
	1	001 0011	0.9659
	1	001 0100	0.9748
	1	001 0101	0.9839
	1	001 0110	1.0030
	1	001 0111	1.0120
	1	001 1000	1.0220
	1	001 1001	1.0320
	1	001 1010	1.0530
	1	001 1011	1.0630
	1	001 1100	1.0740
	1	001 1101	1.0850
	1	001 1110	1.0970
	1	001 1111	1.1140
	1	010 0000	1.1260
	1	010 0001	1.1380
	1	010 0010	1.1510
	1	010 0011	1.1630
	1	010 0100	1.1760
	1	010 0101	1.1900
	1	010 0110	1.1960
	1	010 0111	1.2100
	1	010 1000	1.2240
	1	010 1001	1.2380
	1	010 1010	1.2530
POK_CORE VMON Selection (Over Voltage Detection) hys_en=0	1	010 1011	1.2610
	1	010 1100	1.2760
	1	010 1101	1.2920
	1	010 1110	1.3000
	1	010 1111	1.3160
	1	011 0000	1.3240
	1	011 0001	1.3410
	1	011 0010	1.3500
	1	011 0011	1.3670
	1	011 0100	1.3760
	1	011 0101	1.3850
	1	011 0110	1.4040

Copyright © 2026 Texas Instruments Incorporated

Table 6-14. Bit Settings of POK_CORE with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**474mV - 1.35V**
Over Voltage Detection	**1**	**XXX XXXX**	**726mV - 1.645V**
POK_CORE VMON Selection (Over Voltage Detection) hys_en=0	1	011 0111	1.4130
	1	011 1000	1.4230
	1	011 1001	1.4420
	1	011 1010	1.4520
	1	011 1011	1.4620
	1	011 1100	1.4720
	1	011 1101	1.4830
	1	011 1110	1.5040
	1	011 1111	1.5150
	1	100 0000	1.5260
	1	100 0001	1.5370
	1	100 0010	1.5480
	1	100 0011	1.5600
	1	100 0100	1.5710
	1	100 0101	1.5830
	1	100 0110	1.5950
	1	100 0111	1.6070
	1	100 1000	1.6200
	1	100 1001	1.6320
	1	100 1010	1.6450

Copyright © 2026 Texas Instruments Incorporated

Table 6-15. Bit Settings of POK_1P8 with VMON Threshold Values

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**1.44V - 2.173V**
Over Voltage Detection	**1**	**XXX XXXX**	**1.44V - 2.173V**
POK_1P8 (Under Voltage Detection) hys_en=0	0	010 0000	1.44
	0	010 0001	1.452
	0	010 0010	1.477
	0	010 0011	1.502
	0	010 0100	1.515
	0	010 0101	1.543
	0	010 0110	1.556
	0	010 0111	1.585
	0	010 1000	1.6
	0	010 1001	1.615
	0	010 1010	1.645
	0	010 1011	1.661
	0	010 1100	1.677
	0	010 1101	1.694
	0	010 1110	1.728
	0	010 1111	1.745
	0	011 0000	1.763
	0	011 0001	1.781
	0	011 0010	1.799
	0	011 0011	1.828
	0	011 0100	1.848
	0	011 0101	1.868
	0	011 0110	1.888
	0	011 0111	1.909
	0	011 1000	1.93
	0	011 1001	1.952
	0	011 1010	1.963
	0	011 1011	1.986
	0	011 1100	2.009
	0	011 1101	2.032
	0	011 1110	2.056
	0	011 1111	2.069
	0	100 0000	2.094
	0	100 0001	2.12
	0	100 0010	2.133
	0	100 0011	2.159
	0	100 0100	2.173
POK_1P8 (Under Voltage Detection) hys_en=0	1	000 1100	1.44
	1	000 1101	1.452
	1	000 1110	1.477
	1	000 1111	1.502
	1	001 0000	1.515

Copyright © 2026 Texas Instruments Incorporated

Table 6-15. Bit Settings of POK_1P8 with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**1.44V - 2.173V**
Over Voltage Detection	**1**	**XXX XXXX**	**1.44V - 2.173V**
POK_1P8 (Under Voltage Detection) hys_en=0	1	001 0001	1.543
	1	001 0010	1.556
	1	001 0011	1.585
	1	001 0100	1.6
	1	001 0101	1.615
	1	001 0110	1.645
	1	001 0111	1.661
	1	001 1000	1.677
	1	001 1001	1.694
	1	001 1010	1.728
	1	001 1011	1.745
	1	001 1100	1.763
	1	001 1101	1.781
	1	001 1110	1.799
	1	001 1111	1.828
	1	010 0000	1.848
	1	010 0001	1.868
	1	010 0010	1.888
	1	010 0011	1.909
	1	010 0100	1.93
	1	010 0101	1.952
	1	010 0110	1.963
	1	010 0111	1.986
	1	010 1000	2.009
	1	010 1001	2.032
	1	010 1010	2.056
	1	010 1011	2.069
	1	010 1100	2.094
	1	010 1101	2.12
	1	010 1110	2.133
	1	010 1111	2.159
	1	011 0000	2.173

Copyright © 2026 Texas Instruments Incorporated

Table 6-16. Bit Settings of POK_3P3 with VMON Threshold Values

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**1.427V - 3.982V**
Over Voltage Detection	**1**	**XXX XXXX**	**2.184V - 3.958V**
POK_3P3 (Under Voltage Detection) hys_en=0	0	000 0000	1.427
	0	000 0001	1.466
	0	000 0010	1.501
	0	000 0011	1.545
	0	000 0100	1.576
	0	000 0101	1.616
	0	000 0110	1.658
	0	000 0111	1.694
	0	000 1000	1.731
	0	000 1001	1.769
	0	000 1010	1.81
	0	000 1011	1.841
	0	000 1100	1.885
	0	000 1101	1.919
	0	000 1110	1.955
	0	000 1111	1.992
	0	001 0000	2.03
	0	001 0001	2.07
	0	001 0010	2.111
	0	001 0011	2.14
	0	001 0100	2.184
	0	001 0101	2.215
	0	001 0110	2.262
	0	001 0111	2.295
	0	001 1000	2.329
	0	001 1001	2.363
	0	001 1010	2.399
	0	001 1011	2.436
	0	001 1100	2.474
	0	001 1101	2.513
	0	001 1110	2.554
	0	001 1111	2.596
	0	010 0000	2.639
	0	010 0001	2.661
	0	010 0010	2.706
	0	010 0011	2.753
	0	010 0100	2.778
	0	010 0101	2.827
	0	010 0110	2.853
	0	010 0111	2.905
	0	010 1000	2.932
	0	010 1001	2.959

Copyright © 2026 Texas Instruments Incorporated

Table 6-16. Bit Settings of POK_3P3 with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**1.427V - 3.982V**
Over Voltage Detection	**1**	**XXX XXXX**	**2.184V - 3.958V**
POK_3P3 (Under Voltage Detection) hys_en=0	0	010 1010	3.015
	0	010 1011	3.044
	0	010 1100	3.074
	0	010 1101	3.104
	0	010 1110	3.166
	0	010 1111	3.198
	0	011 0000	3.231
	0	011 0001	3.264
	0	011 0010	3.298
	0	011 0011	3.35
	0	011 0100	3.386
	0	011 0101	3.423
	0	011 0110	3.46
	0	011 0111	3.498
	0	011 1000	3.538
	0	011 1001	3.577
	0	011 1010	3.598
	0	011 1011	3.639
	0	011 1100	3.681
	0	011 1101	3.724
	0	011 1110	3.769
	0	011 1111	3.791
	0	100 0000	3.837
	0	100 0001	3.884
	0	100 0010	3.908
	0	100 0011	3.957
	0	100 0100	3.982
POK_3P3 (Over Voltage Detection) hys_en=0	1	000 0000	2.184
	1	000 0001	2.215
	1	000 0010	2.262
	1	000 0011	2.295
	1	000 0100	2.329
	1	000 0101	2.363
	1	000 0110	2.399
	1	000 0111	2.436
	1	000 1000	2.474
	1	000 1001	2.513
	1	000 1010	2.554
	1	000 1011	2.596
	1	000 1100	2.639
	1	000 1101	2.661
	1	000 1110	2.707

Copyright © 2026 Texas Instruments Incorporated

Table 6-16. Bit Settings of POK_3P3 with VMON Threshold Values (continued)

POK Programmability	POK Configuration Register bits		THRESHOLD (V)
	[7]	[6:0]	
Under Voltage Detection	**0**	**XXX XXXX**	**1.427V - 3.982V**
Over Voltage Detection	**1**	**XXX XXXX**	**2.184V - 3.958V**
POK_3P3 (Over Voltage Detection) hys_en=0	1	000 1111	2.754
	1	001 0000	2.778
	1	001 0001	2.827
	1	001 0010	2.853
	1	001 0011	2.905
	1	001 0100	2.932
	1	001 0101	2.959
	1	001 0110	3.016
	1	001 0111	3.045
	1	001 1000	3.074
	1	001 1001	3.104
	1	001 1010	3.166
	1	001 1011	3.198
	1	001 1100	3.231
	1	001 1101	3.264
	1	001 1110	3.298
	1	001 1111	3.35
	1	010 0000	3.386
	1	010 0001	3.423
	1	010 0010	3.46
	1	010 0011	3.499
	1	010 0100	3.538
	1	010 0101	3.578
	1	010 0110	3.598
	1	010 0111	3.639
	1	010 1000	3.682
	1	010 1001	3.725
	1	010 1010	3.769
	1	010 1011	3.792
	1	010 1100	3.838
	1	010 1101	3.885
	1	010 1110	3.909
	1	010 1111	3.958

6.2.4.1.2 PRG / PRG_PP Overview

The POKs are both controlled by and send their output back through the PRG(_PP) module; the PRG(_PP) module, all of which reside in the WKUP domain.

1. Controls the POKs – enable / disable, over-volt / under-volt / ping-pong
2. Samples the continuous output
3. Filters the output

Copyright © 2026 Texas Instruments Incorporated

Each POK is controlled, sampled, and filtered by a specific PRG(_PP) (see *POK Modules and Monitored Voltages*) and the control register within that PRG(_PP) is where the POKs are controlled (e.g. CTRL_MMR_PRG_POR_CTRL).

1. Control of the POKs: The POK threshold voltage and hysteresis enable are set in CTRL registers defined in *POK Related Registers*. The PRG(_PP) controls how these POKs are enabled / disabled and how the threshold are applied:
   - Enable / Disable has both;
     a. A global control –POK_EN_SEL (e.g. CTRL_MMR_PRG_PP_MCU_CTRL[15]). POK_EN_SEL disables all controlled POKs when 0 and gives control to the per-POK enable when 1.
     b. A per-POK enable (e.g. CTRL_MMR_PRG_PP_MCU_CTRL[1]. POK_VDDR_MCU_EN)

     To enable a given POK, both levels of enable are required to be set.
   - The options on how thresholds are applied to the POK is the key difference between the PRG_PP and PRG modules. The PRG_PP module will be described in this bullet and then the restriction imposed by the PRG module will be addressed. The PRG_PP can be configured in any of three modes, shown in Table 6-17.
   - Sampling the POKs: The POKs are sampled by a divided-down CLK_12M_RC (WKUP_RC_OSC_12M from the internal RC oscillator). The sampling clock is nominally 12.5MHz (though because it is an RC-generated clock, the frequency is not precise). This input clock is divided by 16 so that sampling records the POK output on the rising edge every ~1.25us.
   - Filtering the output: The output is filtered by means of the DEGLITCH_SEL bitfield (e.g. CTRL_MMR_PRG_PP_MCU_CTRL[17:16]). The bitfield allows the user to select a window of 5us, 10us, 15us or 20us; the POK must be in a FAIL state for ALL samples within the filter window in order for the POK output from the PRG to be seen as FAIL.

Note

DEGLITCH_SEL defines number of samples; the precise duration of the filter is not well controlled.
- For example, the 5us deglitch window samples the POK 4 times and ALL 4 samples must FAIL before the PRG propagates FAIL to the system.
- For example, the 20us deglitch window samples the POK 16 times and ALL 16 samples must FAIL before the PRG propagates FAIL to the system.

Note

PRG in POR does not support ping-pong.

Table 6-17. PRG_PP Modes

Mode:	Example Register / Bitfield
Each POK can be configured to apply the undervoltage threshold continuously. 1. OV_SEL is cleared; the threshold voltage is taken from *UV_CTRL 2. POK_PP_EN is cleared	CTRL_MMR_ POK_VDDR_MCU_UV_CTRL CTRL_MMR_PRG_PP_MCU_CTRL[9]. POK_VDDR_MCU_OV_SEL = 0 CTRL_MMR_PRG_PP_MCU_CTRL[19] POK_PP_EN = 0
Each POK can be configured to apply the overvoltage threshold continuously. 1. OV_SEL is set; the threshold voltage is taken from *OV_CTRL 2. POK_PP_EN is cleared	CTRL_MMR_ POK_VDDR_MCU_OV_CTRL CTRL_MMR_PRG_PP_MCU_CTRL[9]. POK_VDDR_MCU_OV_SEL = 1 CTRL_MMR_PRG_PP_MCU_CTRL[19] = 0

Copyright © 2026 Texas Instruments Incorporated

Table 6-17. PRG_PP Modes (continued)

Mode:	Example Register / Bitfield
All POK can be configured to apply the undervoltage threshold then the overvoltage threshold iteratively (ping-pong) 1. All OV_SEL are cleared 2. POK_PP_EN is set; the threshold voltage is taken from *UV_CTRL in the UV phase; the threshold voltage is taken from *OV_CTRL in the OV phase	CTRL_MMR_ POK_VDDR_MCU_UV_CTRL CTRL_MMR_ POK_VDDR_MCU_OV_CTRL CTRL_MMR_PRG_PP_MCU_CTRL[10:8] = 000b CTRL_MMR_PRG_PP_MCU_CTRL[19] = 1

6.2.4.1.3 POR Overview

The POR module should be viewed as a transparent wrapper to the PRG / POKs instantiated within *except* for the following points:

- The bandgap which supplies all POKs is within the POR. The bandgap generates an untrimmed voltage / current after MCU_PORz is released. In order to apply the device-specific trim to the bandgap, the user must set CTRL_MMR_POR_CTRL[7]. TRIM_SEL. The bandgap requires a 100us settling time (during which POK / PRG configuration may occur).
- The device-specific bandgap trim is stored in CTRL_MMR_POR_BANDGAP_CTRL and it is recommended to not modify this trim value.
- The threshold controls for the POKs within the POR are also enabled when TRIM_SEL is set.
- As noted above, the PRG within the POR does not support ping-pong application of thresholds; these POKs monitor a fixed threshold.

6.2.4.1.4 POK / PRG(_PP) /POR Overview

Summarizing this section: The POK block diagram shows a comparator with:

Element of POK	Point of control
resistor divider	*POK*_CTRL registers
hysteresis enable	
comparator enable	*PRG*_CTRL
mux on the output	*POK*_CTRL registers

The output of the POR / PRG_PP / POK voltage monitoring goes to the ESM. Each undervoltage / overvoltage condition has its own bit in the ESM, and the ESM can be programmed to respond to these bits as an interrupt and/or to generate an external signal to the system.

6.2.4.1.5 Timing

As noted in above, there are three timings that are critical:

- 100-us bandgap settling
- 15-us POK enable settling can be concurrent to 100-us bandgap settling
- 9-us POK threshold settling which can be concurrent to both the 100-us bandgap settling and the 15us POK settling

Figure 6-6 shows the sampled POK output to the ESM when the POK threshold is ping-ponged.

- A POK settling time of 1.25-us clocks is fixed in the design; this provides settling for the new threshold voltage when the POK is ping-ponged
- The deglitch (filter) time can be chosen as {4, 8, 12, 16} 1.25-us clock cycles.
- One 1.25-us clock period provides an update window to the ESM.

In this case, the ESM sees an update of the POK threshold every (+ {4, 8,12,16} + 1) × 1.25[µs]. Since the ESM sees an update of a threshold every ~15us to every ~-us, the UV and OV flags are individually being updated every ~-us to every -us.

Copyright © 2026 Texas Instruments Incorporated

In the case where the PRG does not ping-pong, a fixed threshold is applied. The deglitch (filter) time can be chosen as {4, 8, 12, 16} 1.25-us clock cycles. The filter is a sliding window that evaluates the filtered output every 1.25-us based upon the last *n* samples (n = {4, 8, 12, 16}). In this case, the ESM sees an update of the POK threshold every 1.25 us.

The POKs are sampled by a divided-down CLK_12M_RC (WKUP_RC_OSC_12M from Section 5.4.4.1.2). The sampling clock is nominally 12.5 MHz (though because it is an RC-generated clock, the frequency is not precise). This input clock is divided by 16 so that sampling records the POK output on the rising edge every ~1.25 us.

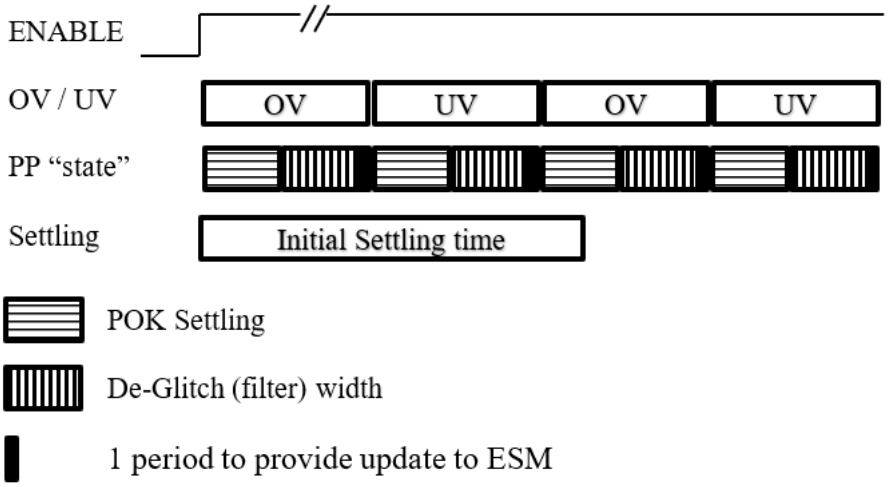


Figure 6-6. Timing for POK with Ping-Pong and Deglitch of 5us

6.2.4.1.6 Restrictions

1. POK monitoring occurs within the initial POK settling (from enable). It is therefore required that the initial state of the POK after enable should be discarded; further, the ESM should be configured for its system response after the initial results during the settling time are discarded.
2. Within a PRG, all POKs must ping-pong or all monitor a static threshold
3. It is not advised to change the threshold from UV to OV while the POK is enabled

6.2.4.2 Power System Modules

6.2.4.2.1 Power OK (POK) Modules

Table 6-18 shows POK modules allocation within device domains.

Table 6-18. POK Allocation Within Device Domains

Module Instance	Domain		
	WKUP	MCU	MAIN
IPOK_VDD_CORE	✓	-	-
IPOK_VDDR_CORE	✓	-	-
IPOK_VDD_CPU	✓	-	-
IPOK_VMON_EXT	✓	-	-
IPOK_VMON_EXT_MAIN_1P8	✓	-	-
IPOK_VMON_EXT_MAIN_3P3	✓	-	-
IPOK_VDD_MCU_OV	✓	-	-
IPOK_VDDR_MCU	✓	-	-
IPOK_VDDSHV_WKUP_GEN	✓	-	-

Copyright © 2026 Texas Instruments Incorporated

Table 6-18. POK Allocation Within Device Domains (continued)

IPOK_CAP_VDDS_MCU_GEN	✓	-	-
IPOK_VDDA_PMIC_IN	✓	-	-

Note

All POK modules are located in the WKUP domain regardless of the voltage domain they monitor.

Note

In this family of devices, the Power-on-reset (POR) module is used only for voltage monitoring, as three additional POK modules. For more information about POK functionality in POR module - voltage being monitored and type (under voltage or over voltage), see *Power on Reset (POR) Module*.

Table 6-19. Possible Monitored Voltages

POK Type	Monitored Voltage Values
POK	3.3 V, 1.8 V, or core voltages
POK_SA	0.45 V (allows for external resistor divider for a wide range of scaled voltages)

Figure 6-7 and Figure 6-8 are block diagrams of the two POK types.

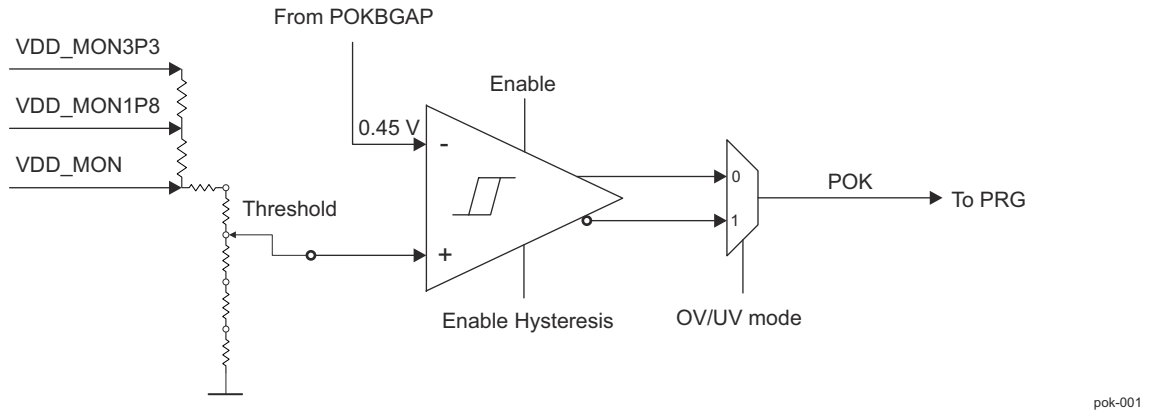


Figure 6-7. POK Block Diagram

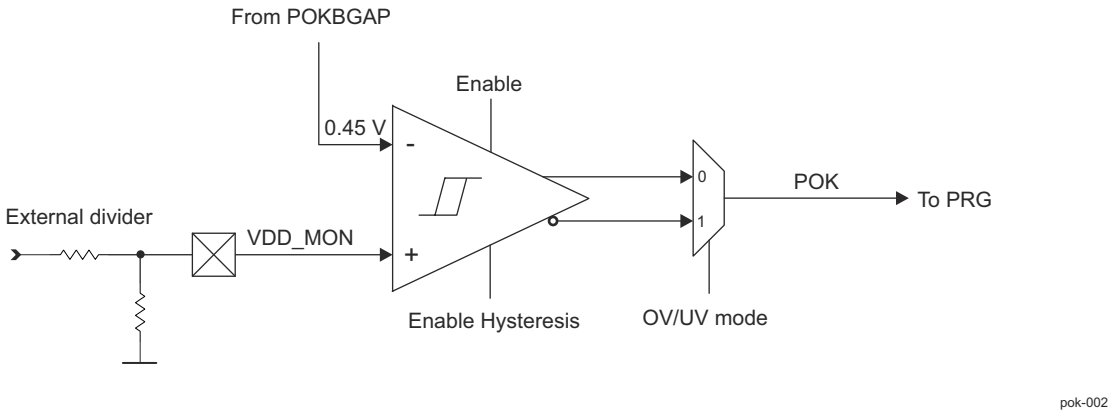


Figure 6-8. POK_SA Block Diagram

Vref for both POK and POK_SA types in Figure 6-7 and Figure 6-8 is supplied from POR module.

Copyright © 2026 Texas Instruments Incorporated

See Table 6-20 for the type of a POK and the voltage it is monitoring.

Table 6-20. POK Modules and Monitored Voltages

Module Instance	Connected to PRG	Type	Voltage Monitored	TAP	OV / UV
POR_POKHV	PRG_POR	POK	VDDA_MCU (1P8V)	VDD_MON1P8	UV
POR_POKLVA	PRG_POR	POK	VDDA_MCU (1P8V)	VDD_MON1P8	OV
POR_POKLVB	PRG_POR	POK	VDD_MCU_UV	VDD_MON	UV
IPOK_VDD_MCU_OV	PRG_POR	POK	VDD_MCU	VDD_MON	OV
IPOK_VDDA_PMIC_IN	PRG_POR	POK_SA	VMON1_ER_VSYS	n/a	UV
IPOK_VDDR_MCU	PRG_PP_MCU	POK	VDDAR_MCU	VDD_MON	OV + UV
IPOK_VDDSHV_WKUP_GEN	PRG_PP_MCU	POK	VDDSHV0_MCU	VDD_MON3P3	OV + UV
IPOK_VMON_CAP_VDDS _MCU_GEN	PRG_PP_MCU	POK	CAP_VDDS0_MCU	VDD_MON1P8	OV + UV
IPOK_VDD_CORE	PRG_PP_MAIN	POK	VDD_CORE	VDD_MON	OV + UV
IPOK_VDDR_CORE	PRG_PP_MAIN	POK	VDDAR_CORE	VDD_MON	OV + UV
IPOK_VDD_CPU	PRG_PP_MAIN	POK	VMON2_IR_VCPU	VDD_MON	OV + UV
IPOK_VMON_EXT	PRG_PP_MAIN	POK	VMON3_IR_VEXT1P8	VDD_MON1P8	OV + UV
IPOK_VMON_EXT_1P8	PRG_PP_MAIN	POK	VMON4_IR_VEXT1P8	VDD_MON1P8	OV + UV
IPOK_VMON_EXT_3P3	PRG_PP_MAIN	POK	VMON5_IR_VEXT3P3	VDD_MON3P3	OV + UV
IPOK_VDD_CPU1	PRG_PP_MAIN	POK	VMON6_IR_VEXT0P8	VDD_MON	OV + UV

6.2.4.2.1.1 POK Programming Model

Table 6-21 and Table 6-22 show the programmable features for the POK modules based on the block diagrams shown in Figure 6-7 and Figure 6-8. Furthermore, each POK module has its own dedicated register to set its programmable features. Table 6-23 lists those registers.

Table 6-21. POK Programmable Features

Programmable Feature	Bitfield in the POK Dedicated Register
POK hysteresis enable	[31] HYST_EN
POK over- or under-voltage detection mode	[7] OVER_VOLT_DET
POK trim bits for voltage comparator threshold	[6-0] POK_TRIM

Table 6-22. POK_SA Programmable Features

Programmable Feature	Bitfield in the POK Dedicated Register
POK hysteresis enable	[31] HYST_EN
POK over- or under-voltage detection mode	[0] OVER_VOLT_DET

PRG controls for the POKs include:

- ping-pong operation enable
- de-glitch period select
- POK OV or UV operation select (if not in ping-pong mode)
- POK enable

See the corresponding PRG control register in Table 6-23.

Copyright © 2026 Texas Instruments Incorporated

Table 6-23. POK Related Registers

Module Instance	Register
PRG_POR	
POR_POKHV	CTRL_MMR_POR_POKHV_UV_CTRL
POR_POKLVA	CTRL_MMR_POR_POKLVA_OV_CTRL
POR_POKLVB	CTRL_MMR_POR_POKLVB_UV_CTRL
IPOK_VDD_MCU_OV	CTRL_MMR_POK_VDD_MCU_OV_CTRL
IPOK_VDDA_PMIC_IN	CTRL_MMR_POK_VDDA_PMIC_IN_CTRL
PRG_PP_MCU	
IPOK_VDDR_MCU	CTRL_MMR_POK_VDDR_MCU_UV_CTRL CTRL_MMR_POK_VDDR_MCU_OV_CTRL
IPOK_VDDSHV_WKUP_GEN	CTRL_MMR_POK_VDDSHV_WKUP_GEN_UV_CTRL CTRL_MMR_POK_VDDSHV_WKUP_GEN_OV_CTRL
IPOK_VMON_CAP_VDDS _MCU_GEN	CTRL_MMR_POK_VMON_CAP_MCU_GEN_UV_CTRL CTRL_MMR_POK_VMON_CAP_MCU_GEN_OV_CTRL
PRG_PP_MAIN	
IPOK_VDD_CORE	CTRL_MMR_POK_VDD_CORE_UV_CTRL CTRL_MMR_POK_VDD_CORE_OV_CTRL
IPOK_VDDR_CORE	CTRL_MMR_POK_VDDR_CORE_UV_CTRL CTRL_MMR_POK_VDDR_CORE_OV_CTRL
IPOK_VDD_CPU	CTRL_MMR_POK_VDD_CPU_UV_CTRL CTRL_MMR_POK_VDD_CPU_OV_CTRL
IPOK_VMON_EXT	CTRL_MMR_POK_VMON_EXT_UV_CTRL CTRL_MMR_POK_VMON_EXT_OV_CTRL
IPOK_VMON_EXT_1P8	CTRL_MMR_POK_VMON_EXT_MAIN1P8_UV_CTRL CTRL_MMR_POK_VMON_EXT_MAIN1P8_OV_CTRL
IPOK_VMON_EXT_3P3	CTRL_MMR_POK_VMON_EXT_MAIN3P3_UV_CTRL CTRL_MMR_POK_VMON_EXT_MAIN3P3_OV_CTRL
IPOK_VDD_CPU1	CTRL_MMR_POK_VDD_CPU1_UV_CTRL CTRL_MMR_POK_VDD_CPU1_OV_CTRL

The possible values of the monitored voltage differ among the POK types, see Table 6-19.

Table 6-24. POK Threshold Levels

POK_TRIM [6:0]	CORE_POK UV (OVER_VOLT _DET = 0)	CORE_POK OV (OVER_VOLT _DET = 1)	1P8_POK UV (OVER_VOLT _DET = 0)	1P8_POK OV (OVER_VOLT _DET = 1)	3P3_POK UV (OVER_VOLT _DET = 0)	3P3_POK OV (OVER_VOLT _DET = 1)
0x00	0.475 V	0.725 V			1.425 V	2.175 V
0x01	0.4875 V	0x7375 V			1.4625 V	2.2125 V
0x02	0.50 V	0.75 V			1.5 V	2.25 V
…						
0xC				1.432 V		
0xD				1.452 V		
0xE				1.473 V		
…						
0x20			1.432 V			
0x21			1.452 V			
0x22			1.473 V			

Copyright © 2026 Texas Instruments Incorporated

Table 6-24. POK Threshold Levels (continued)

POK_TRIM [6:0]	CORE_POK UV (OVER_VOLT _DET = 0)	CORE_POK OV (OVER_VOLT _DET = 1)	1P8_POK UV (OVER_VOLT _DET = 0)	1P8_POK OV (OVER_VOLT _DET = 1)	3P3_POK UV (OVER_VOLT _DET = 0)	3P3_POK OV (OVER_VOLT _DET = 1)
…	0.475 V + POK_TRIM[7:0] *0.0125 V	0.725 V + POK_TRIM[7:0] *0.0125 V	0.7775 V + 0.02045* POK_TRIM[7:0]	1.1865 V + 0.02045* POK_TRIM[7:0]	1.425 V + POK_TRIM[7:0] *0.0375 V	2.175 V + POK_TRIM[7:0] *0.0375 V
0x2D						3.8625
0x2E				2.127V		3.9
0x2F				2.148 V		3.9375
0x30				2.168 V		
…						
0x42			2.127 V		3.9 V	
0x43			2.148 V		3.9375 V	
0x44	1.325 V		2.168 V		3.975 V	
0x45	1.3375 V					
0x46	1.35 V					
…						
0x48		1.625 V				
0x49		1.6375 V				
0x4A		1.65 V				

If the voltage values are programmable, they are set through [6-0] POK_TRIM bitfield in the corresponding POK control register, see Table 6-23.

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.2 PoR/Reset Generator (PRG_PP) Modules

6.2.4.2.2.1 PRG_PP Overview

A PRG_PP (Ping-Pong) module is responsible for implementing enable/disable controls, POK Ping-Pong operation, and event/interrupt masking logic on all POK outputs. Table 6-25 shows PRG_PP modules allocation within device domains.

Table 6-25. PRG_PP Allocation within Device Domains

Module Instance	Domain		
	WKUP	MCU	MAIN
PRG_PP_MCU	✓	-	-
PRG_POR	✓	-	-
PRG_PP_MAIN	✓	-	-

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.2.2 PRG_PP Programming Model

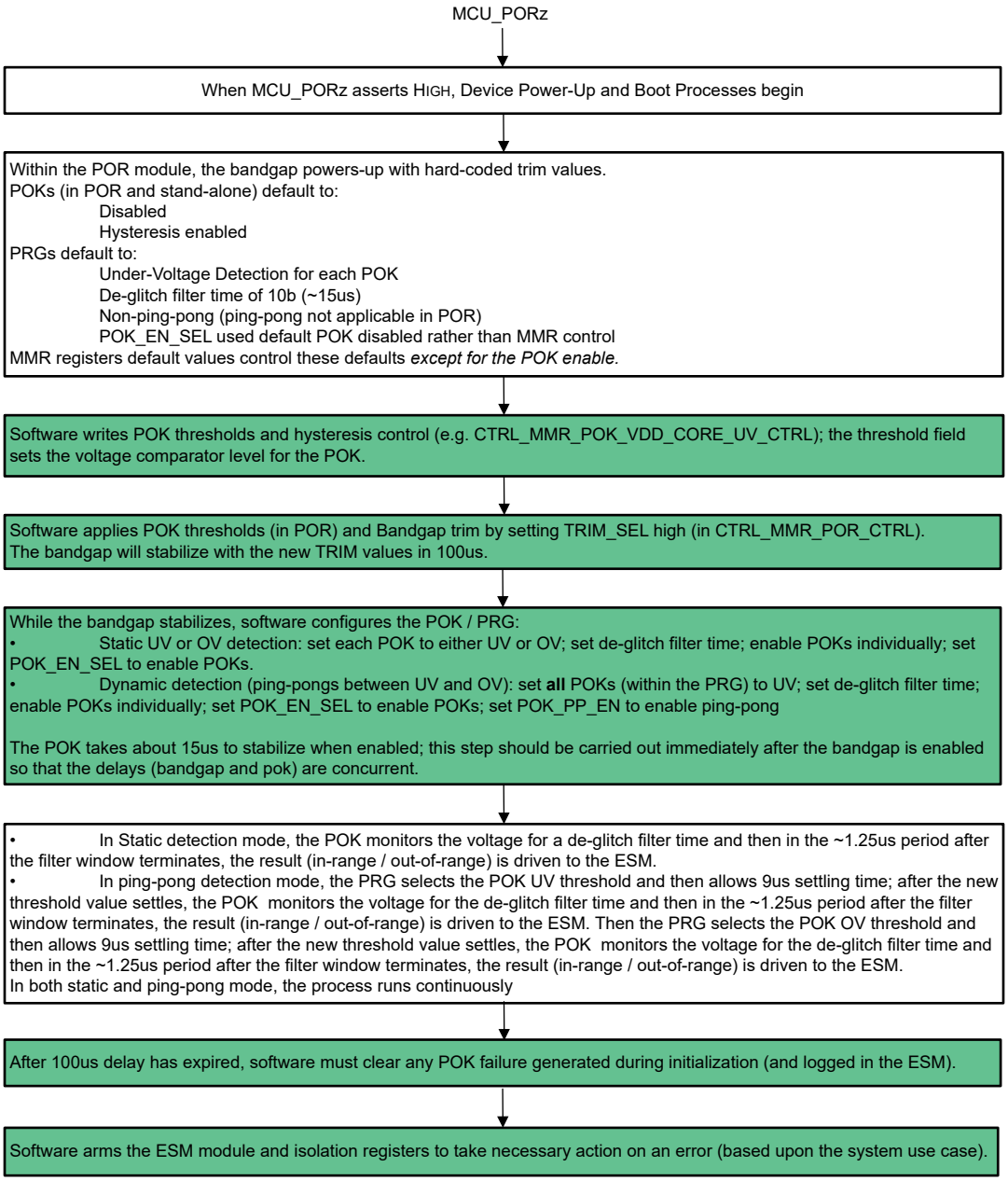


Figure 6-9. PRG_PP Programming Flow

The PRG_PP modules registers used to control and monitor the status of POK modules are:

- CTRL_MMR_PRG_POR_CTRL
- CTRL_MMR_PRG_PP_MCU_CTRL
- CTRL_MMR_PRG_PP_MAIN_CTRL

For more information about control registers, see *Control Module (CTRL_MMR)*.

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.3 Power Glitch Detect (PGD) Modules

A Power Glitch Detect (PGD) circuit is used to detect short duration “glitches” on the core and MPU power supplies. The PGD provides a low output when no glitch is detected and a high output when a glitch is detected.

Figure 6-10 shows the PGD block diagram.

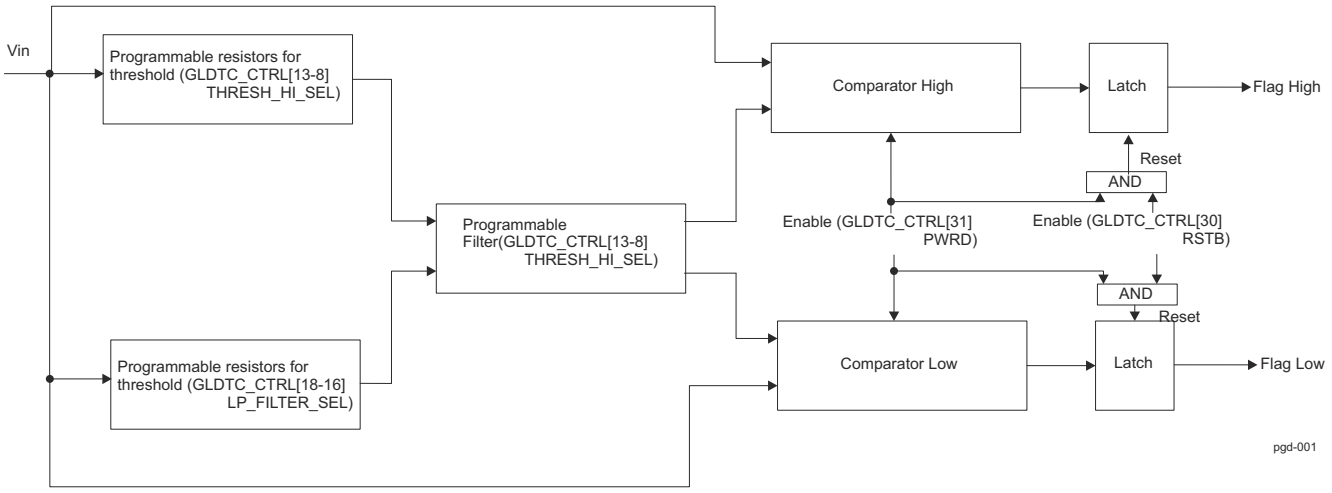


Figure 6-10. PGD Block Diagram

Table 6-26 summarize the PGD integration.

Table 6-26. PGD Integration Summary

Module Instance	Monitored Voltage	PGD Control/Status Registers
IPGD_VDD_MCU_CORE	VDD_MCU	CTRL_MMR_VDD_MCU_GLDTC_CTRL
		CTRL_MMR_VDD_MCU_GLDTC_STAT
IPGD_VDD_MCU_MEM	VDDR_MCU (SRAM)	CTRL_MMR_VDDR_MCU_GLDTC_CTRL
		CTRL_MMR_VDDR_MCU_GLDTC_STAT
IPGD_VDD_CPU_CORE	VDD_CPU	CTRL_MMR_VDD_CPU_GLDTC_CTRL
		CTRL_MMR_VDD_CPU_GLDTC_STAT
IPGD_VDD_MAIN_CORE	VDD_CORE	CTRL_MMR_VDD_CORE_GLDTC_CTRL
		CTRL_MMR_VDD_CORE_GLDTC_STAT
IPGD_VDD_CPU_MEM	VDDR_CPU (SRAM)	CTRL_MMR_VDDR_CPU_GLDTC_CTRL
		CTRL_MMR_VDDR_CPU_GLDTC_STAT
IPGD_VDD_MAIN_MEM	VDDR_CORE (SRAM)	CTRL_MMR_VDDR_CORE_GLDTC_CTRL
		CTRL_MMR_VDDR_CORE_GLDTC_STAT
IPGD_VDD_CPU1_CORE	VDD_CORE	CTRL_MMR_VDD_CPU1_GLDTC_CTRL
		CTRL_MMR_VDD_CPU1_GLDTC_STAT

Note

A flag status bit in corresponding STAT register is cleared by clearing the [30] RSTB bit in the corresponding CTRL register, see for STAT and CTRL registers for a certain PGD module.

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.4 Voltage and Thermal Manager (VTM)

6.2.4.2.4.1 VTM Overview

This section describes the Voltage and Thermal Manager (VTM) module in the device.

In the same way that PRGs control and provide output from the POKs, the VTM controls and outputs signals from temperature sensors that are spread around the die.VTM provides the control, status as well as interrupt and event generation related to integrated temperature sensors and thermal events programed by the user. VTM also contains a set of memory mapped registers that store device-specific operating voltages (AVSVNOM) set during device testing. These values may be used for the system AVS software to adjust operating voltages for the device to achieve optimal operating power.

The device supports one VTM module - WKUP_VTM0. Table 6-27 shows VTM module allocation within device domains.

Table 6-27. VTM Module Allocation within Device Domains

Module Instance	Domain		
	WKUP	MCU	MAIN
WKUP_VTM0	✓	-	-

6.2.4.2.4.1.1 VTM Features

VTM module supports the following features:

- AVS-Class0 only
- Supports up to 8 temperature monitors
- Programming of temperature-crossing thresholds
- Signals when programed thresholds are exceeded - up to 3 alerts:
  - Three full reference 10-bit temperature threshold points, THPT1, THPT2 and THPT0.
- Supports up to 8 temperature monitors
- Allows resolution of 0.5°C for temperature reading and threshold point temperature alert/interrupt generation.
- Supports PMIC/LDO set-up with Class-0 VDD-VID settings
- Support of tentative customization of OPP voltage per device (in support of multiple OPPs)
- AVS-voltage or thermal management for up to 8 voltage domains
- Maximum temperature alert
- Supports one shot sampling mode and continuous monitoring mode for the sensors

Note

The VTM of this family of devices has only 7 temperature sensors and 3 voltage domains.

The ‘j’ notation in the following registers increment from 0 to 6 to map to the 7 temp sensors on the SoC. (Refer to Table 6-32):

- TMPSENS_CTRL_j
- TMPSENS_STAT_j
- TMPSENS_TH_j
- TMPSENS_TH2_j

The ‘j’ notation in the following registers increment from 0 to 2 to map to the 3 voltage domains on the SoC (refer to Table 6-28):

- VD_EVT_SEL_SET_j
- VD_EVT_SEL_CLR_j

6.2.4.2.4.1.2 VTM Not Supported Features

- No support for AVS-Class0 with temperature compensation (AVS0 + TC)
- No integrated I2C inside VTM
- No direct hardware triggering of I2C transactions by VTM

Copyright © 2026 Texas Instruments Incorporated

- No support for voltage and thermal management of IO voltage domains
- No oversampling of the temperature reading
- No support for eFuse defaults of all register values

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.4.2 VTM Functional Description

In the same way that PRGs control and provide output from the POKs, the VTM controls and outputs signals from several temperature sensors that are spread around the die. The VTM provides three interrupts / ESM inputs to the system as well as a warm reset.

The temperature sensor is configured to run off a clock frequency between 1.15MHz and 1.25MHz. The frequency is obtained by dividing-down HFOSC0 or the internal RC clock; the expected usage is that the temperature sensors use HFOSC0 (TSENS_CLK_DIV is the divider). The temperature sensor will sample the temperature on a period defined by VTM_VTM_SAMPLE_CTRL.

When the temperature sensor is enabled, it senses the temperature and reports the temperature as a 10-bit value.

- The raw 10-bit value is reported in VTM_VTM_TMPSENS_STAT_j;
- It is also compared to pre-defined thresholds (each temperature sensor has its own set thresholds set in VTM_VTM_TMPSENS_TH_j and VTM_VTM_TMPSENS_TH2_j) ;
- It is also compared to a pre-defined global threshold set in VTM_VTM_MISC_CTRL2.

Each of these comparisons can be individually enabled (VTM_VTM_TMPSENS_CTRL_j and VTM_VTM_TMPSENS_CTRL_j within mmr_vbusp_cfg2). Additionally, each temperature sensor has status bits for these comparator outputs in VTM_VTM_TMPSENS_STAT_j.

The comparator functionality that drives the interrupts and warm reset work only when the sensor is configured for continuous mode; continuous operation becomes a de facto requirement (and is set in VTM_VTM_TMPSENS_CTRL_j[4].CONT within mmr_vbusp_cfg2 – that is, the mode is set for *each* temperature sensor).

Interrupts / ESM inputs:

The comparator thresholds shall always configured so that TH2 > TH1 > TH0. The concept is:

- TH1 is configured as an early alarm to indicate that the temperature is greater than the threshold defined in VTM_VTM_TMPSENS_TH_j[25-16] TH1_VAL.
- TH2 is configured as a warning to indicate that the temperature is greater than the threshold defined in VTM_VTM_TMPSENS_TH2_j[9-0] TH2_VAL. The concept is that TH1 indicates that the processor is getting hot; TH2 demands immediate attention.
- TH0 is configured to trigger when the sensor detects the temperature less than the threshold defined in VTM_VTM_TMPSENS_TH_j[9-0] TH0_VAL. This interrupt indicates that the thermal mitigations can be relaxed because the temperature has fallen below the original TH1 level.

LT_TH0_INT, if enabled, will get triggered always when the temperature being read is less than TH0, regardless of whether TH1 interrupt and TH2 interrupt are enabled or have ever been triggered. Therefore if TH0 interrupt is generated, then firmware/software is responsible to enable the TH0 interrupt only as part of the interrupt service routine of TH1 and TH2. Otherwise it will keep triggering when is not needed.

Warm Reset input:

Similar to the interrupts / ESM inputs, the warm reset generation is triggered off a threshold defined in VTM_VTM_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR. The reset will be released when the compared temperature drops below VTM_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0.

VTM muxing of the temperature comparisons:

The VTM is configured by voltage domains. At this level, all the interrupt / ESM inputs from each temperature sensor can be enabled or disabled; as an example, the comparator outputs from temperature sensor 0 can be ignored in voltage domain 0. The enable is configured with the VTM_VTM_VD_EVT_SEL_SET_j and VTM_VTM_VD_EVT_SEL_CLR_j registers. At the voltage domain-level, the enabled comparator results are combined by interrupt / ESM signal. The combined comparator result can be observed in VTM_VTM_VD_EVT_STAT_j. As is shown in the following table, j increments over voltage domains for these

Copyright © 2026 Texas Instruments Incorporated

registers (see Table 6-28). The registers TMPSENS_*_j map to the individual temp sensors, as shown in *VTM TEMPSENSOR Register Groups Mapping to Voltage Domains*.

Table 6-28. VTM Voltage Domains

Register Group	Voltage Domain
WKUP_VTM_*_VD_*_0	VDD_MCU
WKUP_VTM_*_VD_*_1	VDD_CORE
WKUP_VTM_*_VD_*_2	VDD_CPU_AVS
others	Unused

After the temperature sensors are combined at a voltage domain level, the outputs from each voltage domain are enabled at the top level of the VTM with VTM_VTM_GT_TH2_INT_EN_SET / VTM_VTM_GT_TH2_INT_EN_CLR, VTM_VTM_GT_TH1_INT_EN_SET / VTM_VTM_GT_TH1_INT_EN_CLR, and VTM_VTM_LT_TH0_INT_EN_SET / VTM_VTM_LT_TH0_INT_EN_CLR. At this top level of the VTM, the status of these signals is checked and cleared with VTM_VTM_GT_TH2_INT_RAW_STAT_SET / VTM_VTM_GT_TH2_INT_EN_STAT_CLR, VTM_VTM_GT_TH1_INT_RAW_STAT_SET / VTM_VTM_GT_TH1_INT_EN_STAT_CLR, and VTM_VTM_LT_TH0_INT_RAW_STAT_SET / VTM_VTM_LT_TH0_INT_EN_STAT_CLR.

VTM control of the global comparison that generates a warm reset:

The global comparison that generates a warm rest is enabled at each temperatures sensor with WKUP_VTM_TMPSENS_CTRL_j[11] within mmr_vbusp_cfg2. Additionally, there is a global enable for this signal in WKUP_VTM_MISC_CTRL[0].

AVS voltage definition:

The VTM also contains the Adaptive Voltage Scaling (AVS) voltage for the VDD_CPU domain. Since the VDD_CPU domain is defined by voltage domain 2, the registers of interest are VTM_VTM_VD_DEVINFO_2 and VTM_VTM_VD_OPPVID_2. If VTM_VTM_VD_DEVINFO_2[12] AVS0_SUP is set, then the device should set the voltage defined in VTM_VTM_VD_OPPVID_2[15-8] OPP_1. The voltage is encoded according to the table below:

Table 6-29. Output Voltage Selection for BUCK Regulators

BUCKn_ VSETn	Output Voltage [V] 20 mV steps	BUCKn_ VSETn	Output Voltage [V] 5 mV steps	BUCKn_ VSETn	Output Voltage [V] 5 mV steps	BUCKn_ VSETn	Output Voltage [V] 10 mV steps	BUCKn_ VSETn	Output Voltage [V] 20 mV steps	BUCKn_ VSETn	Output Voltage [V] 20 mV steps
0x00	0.3	0x0F	0.6	0x41	0.85	0x73	1.1	0xAB	1.66	0xD6	2.52
0x01	0.32	0x10	0.605	0x42	0.855	0x74	1.11	0xAC	1.68	0xD7	2.54
0x02	0.34	0x11	0.61	0x43	0.86	0x75	1.12	0xAD	1.7	0xD8	2.56
0x03	0.36	0x12	0.615	0x44	0.865	0x76	1.13	0xAE	1.72	0xD9	2.58
0x04	0.38	0x13	0.62	0x45	0.87	0x77	1.14	0xAF	1.74	0xDA	2.6
0x05	0.4	0x14	0.625	0x46	0.875	0x78	1.15	0xB0	1.76	0xDB	2.62
0x06	0.42	0x15	0.63	0x47	0.88	0x79	1.16	0xB1	1.78	0xDC	2.64
0x07	0.44	0x16	0.635	0x48	0.885	0x7A	1.17	0xB2	1.8	0xDD	2.66
0x08	0.46	0x17	0.64	0x49	0.89	0x7B	1.18	0xB3	1.82	0xDE	2.68
0x09	0.48	0x18	0.645	0x4A	0.895	0x7C	1.19	0xB4	1.84	0xDF	2.7
0x0A	0.5	0x19	0.65	0x4B	0.9	0x7D	1.2	0xB5	1.86	0xE0	2.72
0x0B	0.52	0x1A	0.655	0x4C	0.905	0x7E	1.21	0xB6	1.88	0xE1	2.74
0x0C	0.54	0x1B	0.66	0x4D	0.91	0x7F	1.22	0xB7	1.9	0xE2	2.76
0x0D	0.56	0x1C	0.665	0x4E	0.915	0x80	1.23	0xB8	1.92	0xE3	2.78
0x0E	0.58	0x1D	0.67	0x4F	0.92	0x81	1.24	0xB9	1.94	0xE4	2.8

Copyright © 2026 Texas Instruments Incorporated

Table 6-29. Output Voltage Selection for BUCK Regulators (continued)

BUCKn_ VSETn	Output Voltage [V] 20 mV steps	BUCKn_ VSETn	Output Voltage [V] 5 mV steps	BUCKn_ VSETn	Output Voltage [V] 5 mV steps	BUCKn_ VSETn	Output Voltage [V] 10 mV steps	BUCKn_ VSETn	Output Voltage [V] 20 mV steps	BUCKn_ VSETn	Output Voltage [V] 20 mV steps
		0x1E	0.675	0x50	0.925	0x82	1.25	0xBA	1.96	0xE5	2.82
		0x1F	0.68	0x51	0.93	0x83	1.26	0xBB	1.98	0xE6	2.84
		0x20	0.685	0x52	0.935	0x84	1.27	0xBC	2	0xE7	2.86
		0x21	0.69	0x53	0.94	0x85	1.28	0xBD	2.02	0xE8	2.88
		0x22	0.695	0x54	0.945	0x86	1.29	0xBE	2.04	0xE9	2.9
		0x23	0.7	0x55	0.95	0x87	1.3	0xBF	2.06	0xEA	2.92
		0x24	0.705	0x56	0.955	0x88	1.31	0xC0	2.08	0xEB	2.94
		0x25	0.71	0x57	0.96	0x89	1.32	0xC1	2.1	0xEC	2.96
		0x26	0.715	0x58	0.965	0x8A	1.33	0xC2	2.12	0xED	2.98
		0x27	0.72	0x59	0.97	0x8B	1.34	0xC3	2.14	0xEE	3.0
		0x28	0.725	0x5A	0.975	0x8C	1.35	0xC4	2.16	0xEF	3.02
		0x29	0.73	0x5B	0.98	0x8D	1.36	0xC5	2.18	0xF0	3.04
		0x2A	0.735	0x5C	0.985	0x8E	1.37	0xC6	2.2	0xF1	3.06
		0x2B	0.74	0x5D	0.99	0x8F	1.38	0xC7	2.22	0xF2	3.08
		0x2C	0.745	0x5E	0.995	0x90	1.39	0xC8	2.24	0xF3	3.1
		0x2D	0.75	0x5F	1.0	0x91	1.4	0xC9	2.26	0xF4	3.12
		0x2E	0.755	0x60	1.005	0x92	1.41	0xCA	2.28	0xF5	3.14
		0x2F	0.76	0x61	1.01	0x93	1.42	0xCB	2.3	0xF6	3.16
		0x30	0.765	0x62	1.015	0x94	1.43	0xCC	2.32	0xF7	3.18
		0x31	0.77	0x63	1.02	0x95	1.44	0xCD	2.34	0xF8	3.2
		0x32	0.775	0x64	1.025	0x96	1.45	0xCE	2.36	0xF9	3.22
		0x33	0.78	0x65	1.03	0x97	1.46	0xCF	2.38	0xFA	3.24
		0x34	0.785	0x66	1.035	0x98	1.47	0xD0	2.4	0xFB	3.26
		0x35	0.79	0x67	1.04	0x99	1.48	0xD1	2.42	0xFC	3.28
		0x36	0.795	0x68	1.045	0x9A	1.49	0xD2	2.44	0xFD	3.3
		0x37	0.8	0x69	1.05	0x9B	1.5	0xD3	2.46	0xFE	3.32
		0x38	0.805	0x6A	1.055	0x9C	1.51	0xD4	2.48	0xFF	3.34
		0x39	0.81	0x6B	1.06	0x9D	1.52	0xD5	2.5		
		0x3A	0.815	0x6C	1.065	0x9E	1.53				
		0x3B	0.82	0x6D	1.07	0x9F	1.54				
		0x3C	0.825	0x6E	1.075	0xA0	1.55				
		0x3D	0.83	0x6F	1.08	0xA1	1.56				
		0x3E	0.835	0x70	1.085	0xA2	1.57				
		0x3F	0.84	0x71	1.09	0xA3	1.58				
		0x40	0.845	0x72	1.095	0xA4	1.59				
						0xA5	1.6				
						0xA6	1.61				
						0xA7	1.62				
						0xA8	1.63				
						0xA9	1.64				
						0xAA	1.65				

Copyright © 2026 Texas Instruments Incorporated

6.2.4.2.4.2.1 VTM Temperature Status and Thermal Management

VTM module provides temperature reading and interrupt/alert information for thermal management.

6.2.4.2.4.2.1.1 10-bit Temperature Values Versus Temperature

Table 6-30 shows a table lookup method for translating code to temperature from temperature monitors which correspond to the temperature measured read from the VTM0_VTM_TMPSENS_STAT_J [9-0] DATA_OUT bitfields. Table 6-30 also provides the values for the temperatures coded in VTM0_VTM_MISC_CTRL2 [25-16] MAXT_OUTRG_ALERT_THR0 and VTM0_VTM_MISC_CTRL2 [9-0] MAXT_OUTRG_ALERT_THR thresholds.

Table 6-30. Temperature Translation Table

Parameter Type	Value								
Temperature [°C] (rounded)	-40	-25	0	25	50	75	100	125	150
Temperature [°C] (unrounded)	-40.03	-24.95	0.01	25.02	49.94	75.03	100.02	124.97	150.00
DATA_OUT Code	28	77	164	260	366	485	620	773	949

Table 6-31 shows a table that gives coefficients of polynomial to calculate code or temperature. The equation to calculate code or temperature is:

$y = a4 \times x^4 + a3 \times x^3 + a2 \times x^2 + a1 \times x + a0$

Table 6-31. Equation Method to Calculate Code or Temperature

Function	Unit	Coefficient				
		a4	a3	a2	a1	a0
Code = f(Temperature)	Decimal value of code	4.9847E-08	1.6972E-05	6.8759E-03	3.6509E+00	1.6407E+02
Temperature = f(Code)	°C	-9.2627E-12	6.0373E-08	-1.7058E-04	3.2512E-01	-4.9002E+01

6.2.4.2.4.2.2 VTM Temperature Driven Alerts and Interrupts

All 3 alerts from the 7 (5 shown in sketch) temperature monitors are available as inputs of the mask and alert merging logic block of each voltage domain, such that the temperature monitors that are relevant to each voltage domain can be selected as the contributors to the generation of the 3 combined interrupts in each voltage domain (5 shown in sketch; 3). This logic is presented in Figure 6-11. The THERM_MAXTEMP_OUTRANGE_ALERT is not shown in this figure. Notice that the same temperature sensor can contribute to more than one voltage domain and each voltage domain can have multiple sensors contributing to the interrupt generation in that VD, see VTM_CFG1_VD_EVT_SET_j and VTM_CFG1_VD_EVT_CLR_j registers.

Copyright © 2026 Texas Instruments Incorporated

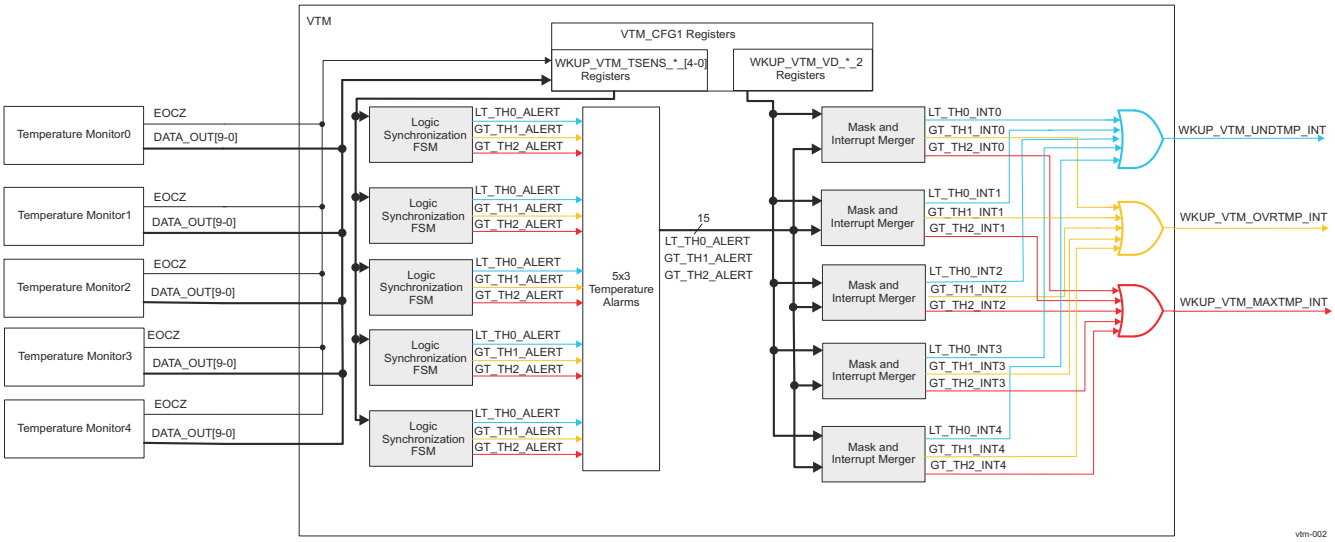


Figure 6-11. VTM Alert and Interrupt Generation

Note

The interrupts are only active when the sensor is in continuous mode. A one-shot sampling of the sensor will not trigger any interrupts.

Table 6-32 presents the connection of VTM TEMPSENSOR registers groups to voltage domains.

Table 6-32. VTM TEMPSENSOR Register Groups Mapping to Voltage Domains

Register Group	Voltage Domain
WKUP_VTM_TMPSENS_*_0	Near MCU_R5FSS in VDD_MCU
WKUP_VTM_TMPSENS_*_1	On VDD_MCU / VDD_CPU boundary near A72 (in VDD_MCU)
WKUP_VTM_TMPSENS_*_2	Near GPU in VDD_CORE
WKUP_VTM_TMPSENS_*_3	On VDD_CORE / VDD_CPU boundary near C7x / MMA (in VDD_CORE)
WKUP_VTM_TMPSENS_*_4	On VDD_CORE / VDD_CPU boundary near A72 (in VDD_CORE)
WKUP_VTM_TMPSENS_*_5	On VDD_CORE / VDD_CPU boundary near C7x / MMA (in VDD_CORE)
WKUP_VTM_TMPSENS_*_6	Near LPDDR4 in VDD_CORE (DDR)
Others	Unused

6.2.4.2.4.2.3 VTM ECC Aggregator

Note

For more information about ECC Aggregator functionality, see *ECC Aggregator*.

6.2.4.2.4.2.4 VTM Programming Model

6.2.4.2.4.2.4.1 VTM Maximum Temperature Outrange Alert

THERM_MAXTEMP_OUTRANGE_ALERT is used at the device level in order to generate device warm reset. The alert is asserted when the device overheats and exceeds the maximum temperature limit specified for the device, namely 125°C.

The following is the sequence to assert the THERM_MAXTEMP_OUTRANGE_ALERT:

Copyright © 2026 Texas Instruments Incorporated

1. For the output to go active (‘1’), the VTM has to be configured to operate with at least one of its temperature sensors enabled, and VTM_CFG1_TMPSENS_CTRL_j[11] MAXT_OUTRG_EN = 1 for the corresponding enabled sensor.
2. In addition, VTM_CFG2_MISC_CTRL[0] ANYMAXT_OUTRG_ALERT_EN has to be set to ‘1’.
3. With the previous configuration set and at least one of the temperature sensors properly configured, and enabled, if the temperature of the die ever exceeds the programmed value in VTM_CFG2_MISC_CTRL2, the VTM output will be driven ‘1’ once the sensor detects the programmed maximum temperature.
4. At the device level the VTM output, THERM_MAXTEMP_OUTRANGE_ALERT, will drive the PLL controllers into warm reset state with the PLL controllers entering warm reset condition, at the same time that all the PLLs go into clock bypass mode.
5. By the actions described in No.4, the device will considerably reduce its power consumption, which over the passing of a few seconds will cause the device to reduce its temperature.
6. The VTM sensor continues to read the temperature and once the code corresponding to the programmed value in VTM_CFG2_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 is detected the VTM drives THERM_MAXTEMP_OUTRANGE_ALERT = 0.
7. After No.6 is fulfilled, the PLL controllers no longer have the THERM_MAXTEMP_OUTRANGE_ALERT driving the warm reset request and thus the PLL controllers bring the SoC out of reset and the boot sequence re-starts. At the same time the PLLs get out of bypass mode and start with their target programmed frequencies for the boot process.

Note

THERM_MAXTEMP_OUTRANGE_ALERT output of VTM is mapped to VTM_CFG2_MISC_CTRL[0] ANY_MAXT_OUTRG_ALERT_EN.

Notice that exceeding of the programmed value in VTM_CFG2_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR boundary doesn’t result in a PORz (cold reset) event, but instead a warm reset event. Otherwise if cold reset is applied to the MCU/WKUP domains of the device, the VTM will be fully reset and the output THERM_MAXTEMP_OUTRANGE_ALERT will be cleared to ‘0’ immediately.

6.2.4.2.4.2.4.2 Sensors Programming Sequences

Sensors Control Modifications Sequence:

1. Write changes to registers.
2. Wait for 900ns to 1us (long enough for sensor period of slowest 1.15 MHz sensor clock plus some extra).
3. New changes can be written to registers.

Reset Sensors Sequence:

1. Set the WKUP_VTM_TMPSENS_CTRL_j[6] CLRZ bit to enable the sensor (so that it has the ENA input high).
2. Wait for 900ns to 1us for the sensor clock to toggle and register the updated ENA pin.
3. Clear WKUP_VTM_TMPSENS_CTRL_j[6] CLRZ to disable and reset the sensor (as it requires an ENA transition from 1 to 0 to reset initially) .
4. Wait for 900ns to 1us for the sensor clock to toggle and register the updated ENA pin.
5. Now the sensor can be configured and enabled normally.

6.2.4.2.4.2.5 AVS-Class0

Adaptive Voltage Scaling (AVS) Class 0 is a procedure for lowering the voltage on certain device power rails. AVS Class 0 attempts to normalize the power consumption across all devices. The optimal voltage for each AVS supported rail of each device is determined after analysis in the factory, based on the strength of the device during manufacturing. This value is written in the device eFuse where it can be read through dedicated registers.

AVS can be enabled by system software after primary boot. Refer to *AVS Support* on how to enable AVS based on these registers.

Copyright © 2026 Texas Instruments Incorporated

6.3 Reset

This chapter describes the device reset signals and contains details on reset management.

6.3.1 Overview

Reset Introduction

A reset is designed to bring a device or subsystem into a known state. Resets occur after power up events, as well as through software and hardware reset requests. This section introduces the device reset capabilities and functionality.

Device Resets

The Device Power-on-Reset (POR) is controlled by the external pin MCU_PORz. This pin is driven by an external (off-chip) "Power-Good" Circuit or Power Management IC (PMIC). The MCU_PORz pin should be held active LOW (0) during the entire power-up phase. The device should be held in reset until all power supplies are stable with an additional delay for the High Frequency Oscillator (HFOSC0) clock to stabilize.

The SoC is divided into two separate functional Reset Domains: MCU Reset Domain and MAIN Reset Domain (each containing specific processing cores and peripherals).

- MCU Reset Domain: Managed by the MCU Domain, PLLCTRL and PSC, this includes ARM® Cortex®- R5F MCU Processor (R5FSS).
- MAIN Reset Domain: Managed by the MAIN Domain, PLLCTRL and PSC, this includes the high performance application Arm Cortex-A53 cores (A53SS), Device Management and Low Power Management (DM R5F), Security Controller (SMS0) .

Both MCU and Main domains have independent Warm Reset MCU_RESETz & MAIN_RESETz_REQ pins. All MCU domain resets act as controller resets to the whole device, whereas Main domain resets only reset Main domain (MCU domain is reset isolated from all Main domain resets). The following section outlines the various resets coming from pins and software MMR bits and their behavior.

Figure 6-12 illustrates the high-level reset domain partition.

Copyright © 2026 Texas Instruments Incorporated

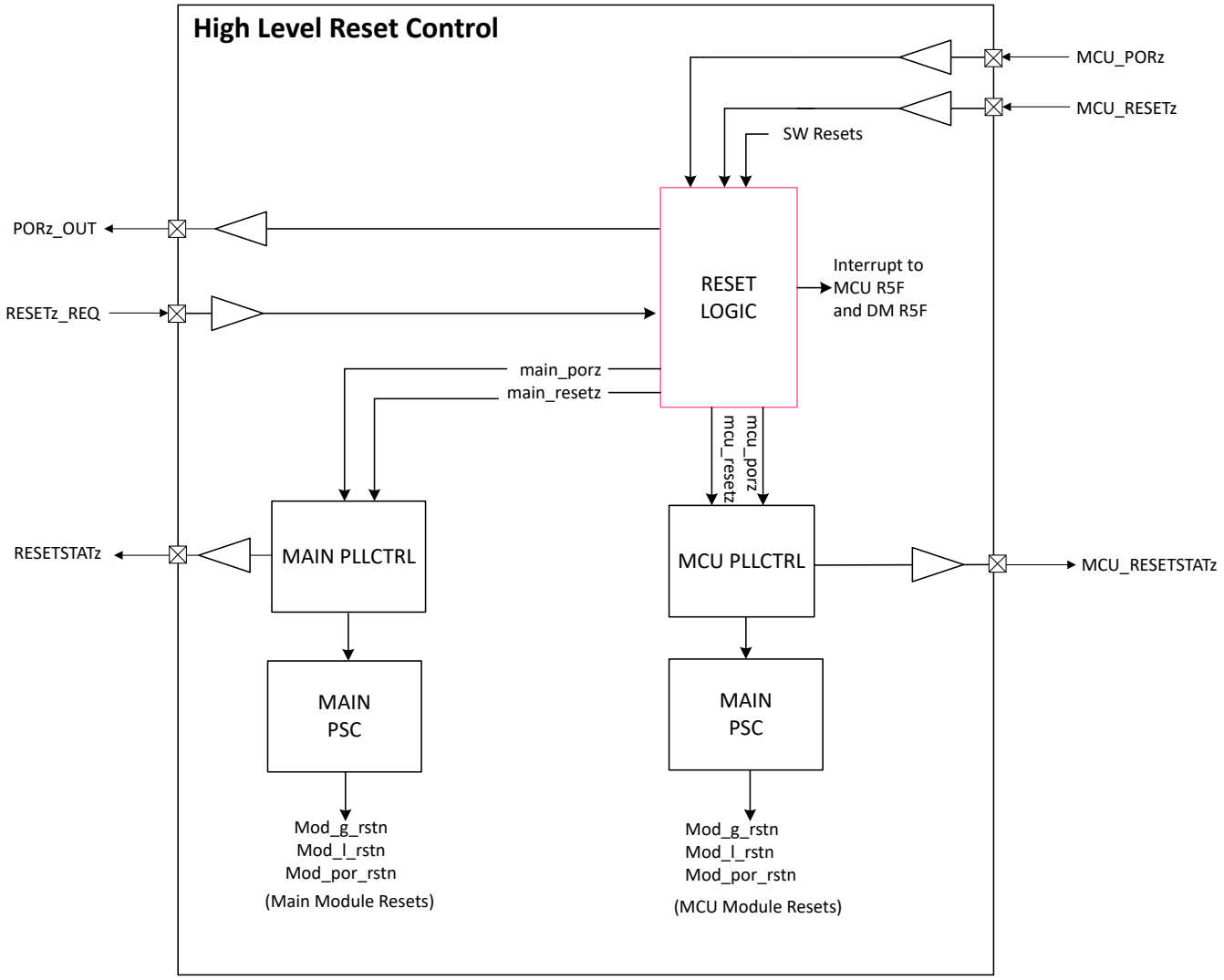


Figure 6-12. High Level Reset Control

Copyright © 2026 Texas Instruments Incorporated

6.3.1.1 MCU Domain Supported Resets

- Power-On-Resets
  - MCU_PORx device pin: This power-on-reset will reset the entire device including MCU and MAIN domains.
- Warm Resets
  - MCU_RESETz device pin
  - MCU domain software warm reset
  - MCU domain ESM error reset
  - SMS cold reset

  The warm resets listed above will reset the entire device including MCU and MAIN domains.
- Local Module Resets
  - MCU domain LPSC module local reset: This reset is generated by the MCU PSC to locally reset the R5F CPU core

6.3.1.2 MAIN Domain Supported Resets

- Power-On-Resets
  - MAIN domain software POR reset
  - MAIN domain hardware POR reset generated by MCU_PORz
- Warm Resets
  - RESETz_REQ device pin
  - MAIN domain ESM error reset
  - MAIN domain software warm reset
  - SMS cold reset
  - MCU domain warm reset will also generate a MAIN domain warm reset
- Local Module Resets
  - MAIN domain LPSC module local reset: This reset is generated by the MAIN PSC to locally reset the Processor cores (A53, SMS R5F, DM R5F, PRUSS)

The device can be configured via software such that a Main domain resets can be isolated from MCU domain. When the MCU domain is configured for reset isolation, MAIN domain resets will only reset the MAIN domain and the MCU domain will remain unaffected by all MAIN domain resets. All MCU domain resets will cause a reset to the entire device (including MAIN and MCU domains).

6.3.1.3 High-Level Reset Flow

Copyright © 2026 Texas Instruments Incorporated

High Level Reset Flow explains the high-level device power-up and reset use cases.

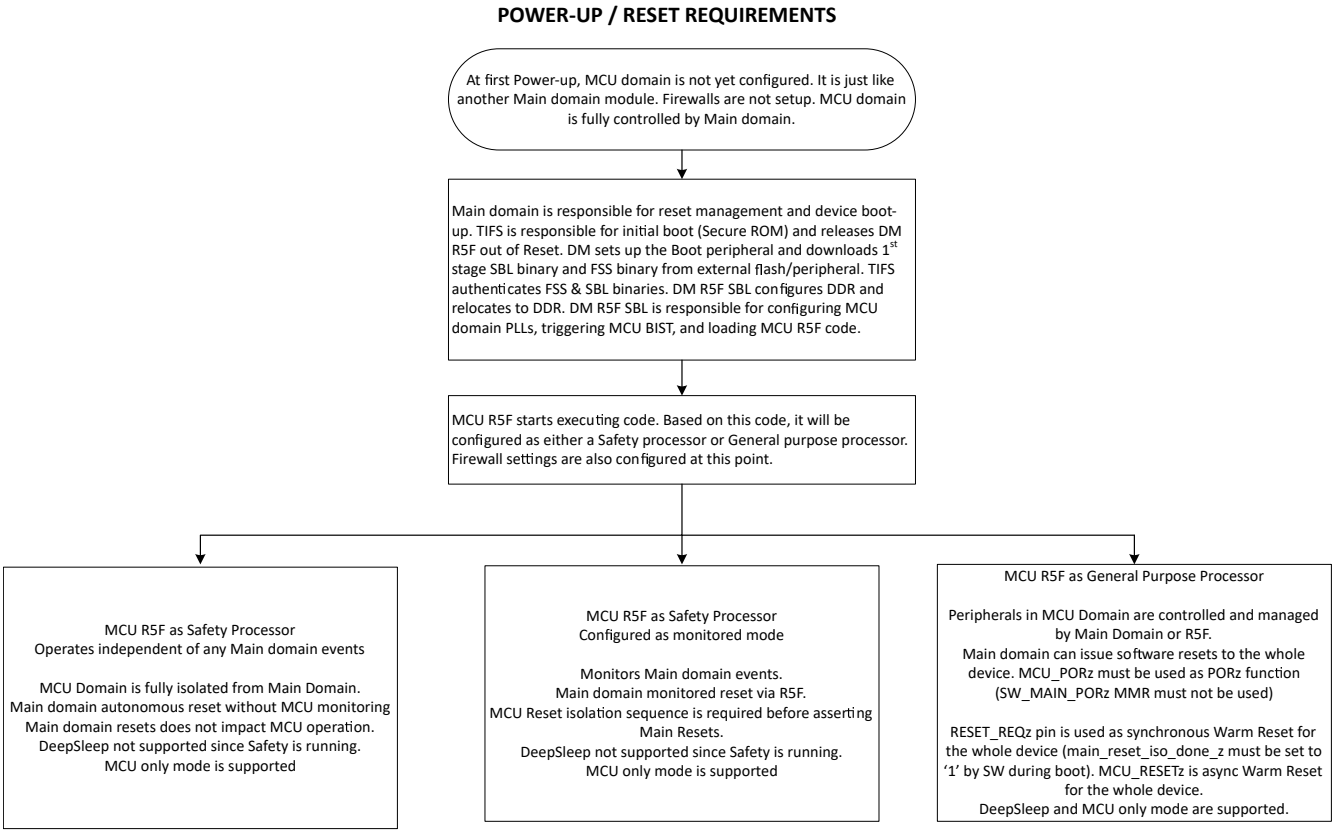


Figure 6-13. High Level Reset Flow

Copyright © 2026 Texas Instruments Incorporated

6.3.1.4 Reset Terminology

DM	Device Management (including Low Power Management)
POR	Power-On-Reset
ESM	Error Signaling Module
Reset Isolation	The module or domain is exempt from specific resets
PSC	Power Sleep Controller
LPSC	Local Power Sleep Controller
PD	Power Domain
VD	Voltage Domain
PRG	Power Reset Generator
POK	Power OK
PGD	Power Glitch Detector
PLLCTRL	PLL and Reset Controller
HHV	High Heating Value mode (IO Pin Fail-safe Control (Tri-State Output))
VTM	Voltage and Temperature Monitor
HFOSC	Internal Device High-Frequency Oscillator
SMS	Security Management Subsystem
TIFS	Texas Instruments Foundational Security
HSM	Hardware Security Module

Copyright © 2026 Texas Instruments Incorporated

6.3.1.5 Reset Architecture

Copyright © 2026 Texas Instruments Incorporated

6.3.1.5.1 Reset Architecture Block Diagram

Reset Architecture Block Diagram shows the device reset architecture block diagram. It describes the device reset sub-modules and critical internal signal interconnections.

Copyright © 2026 Texas Instruments Incorporated

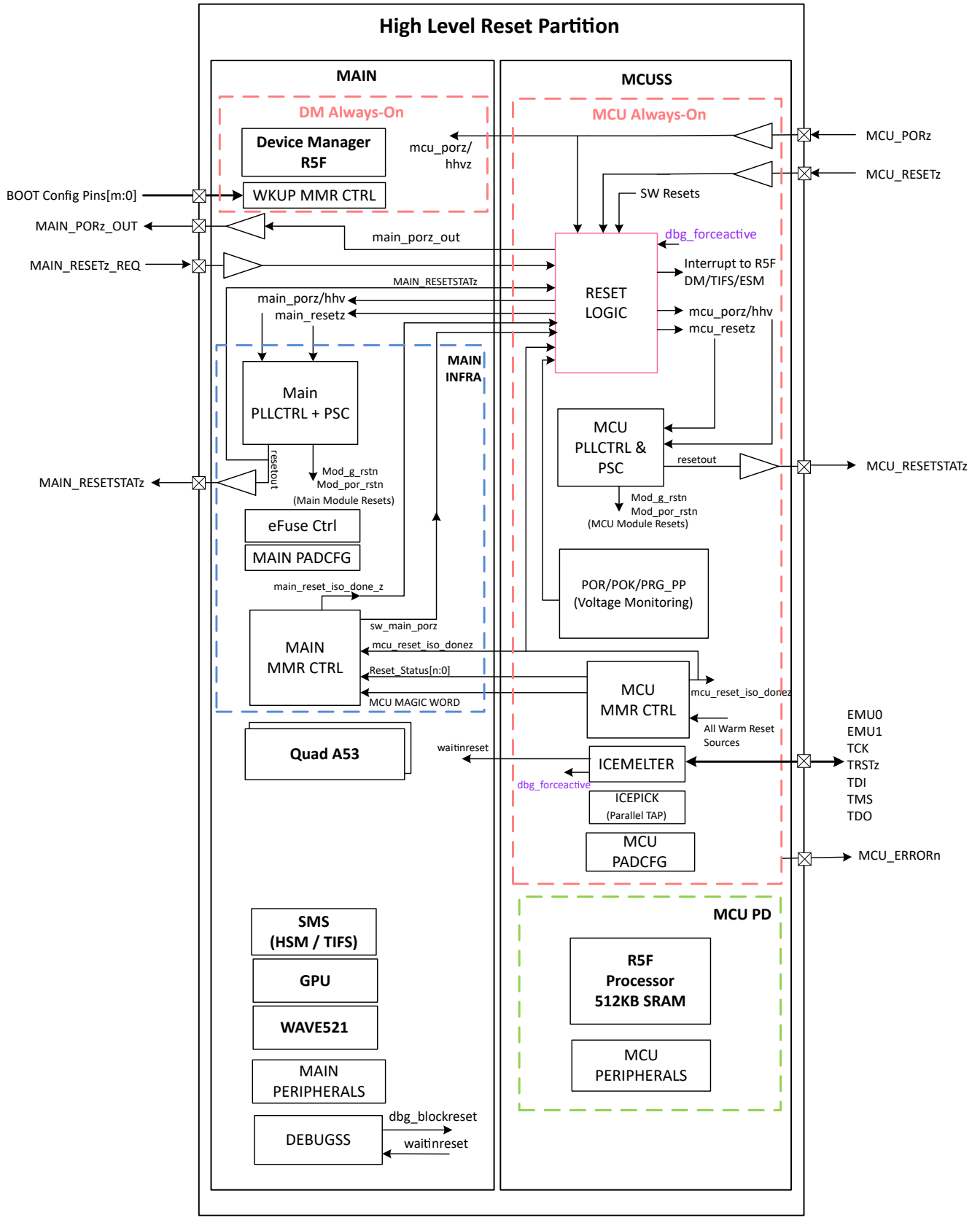


Figure 6-14. Reset Architecture Block Diagram

Copyright © 2026 Texas Instruments Incorporated

6.3.1.5.2 SoC Reset Hardware Logic Diagram

SoC Reset Hardware Logic Diagram shows the reset hardware logic diagram. It describes the MAIN and MCU reset domains and processors connections as well as the available reset sources including reset pins and CTRLMMRs.

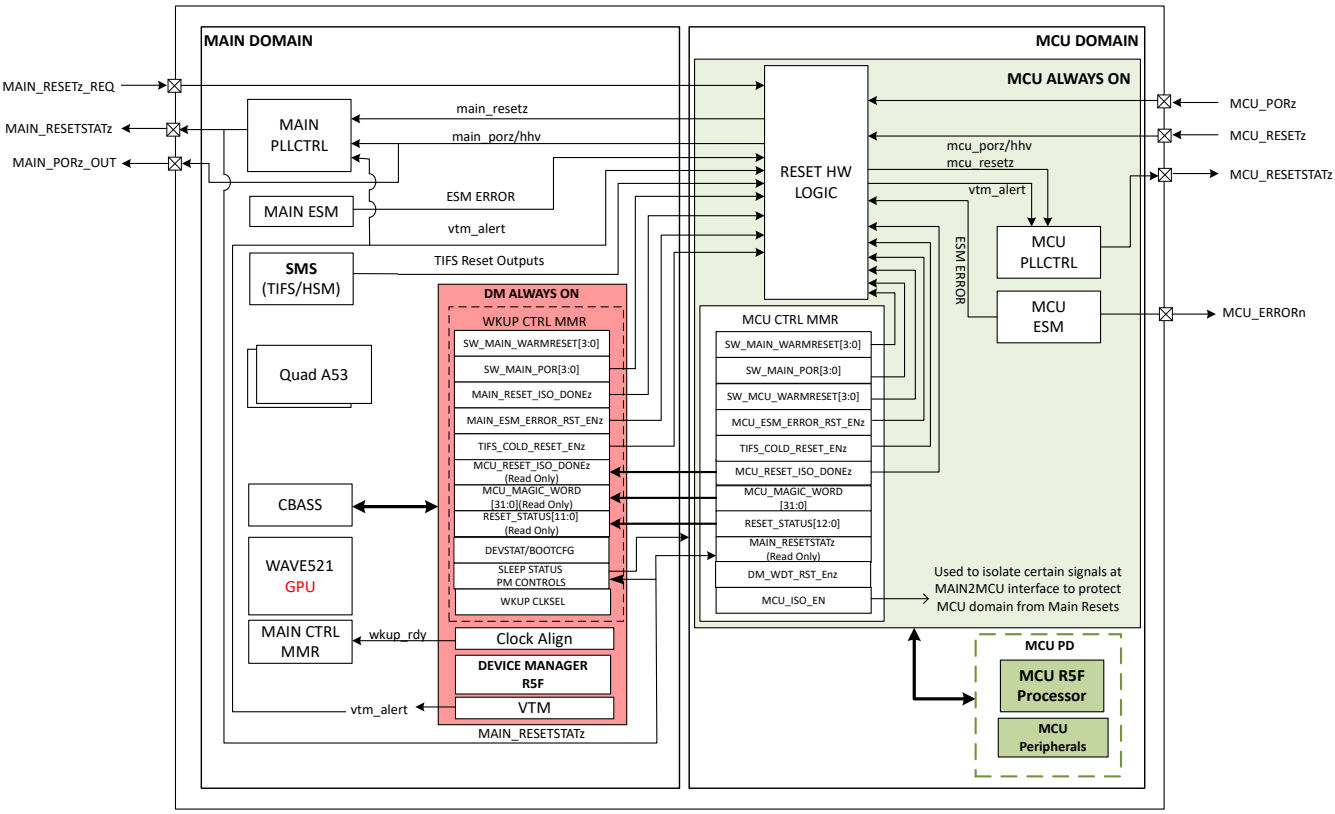


Figure 6-15. SoC Reset Hardware Logic Diagram

Copyright © 2026 Texas Instruments Incorporated

6.3.1.5.3 MCU Domain Reset Hardware Logic Diagram

MCU Domain Reset Hardware Logic Diagram shows the external and internal signals associated with the MCU domain resets.

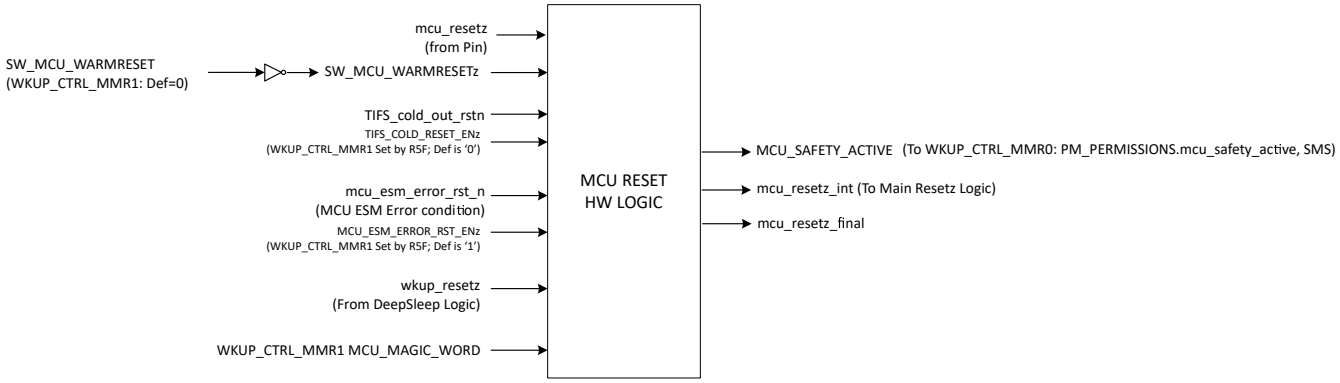


Figure 6-16. MCU Domain Reset Hardware Logic Diagram

Copyright © 2026 Texas Instruments Incorporated

6.3.1.5.4 MAIN Domain Reset and PORz Hardware Logic Diagrams

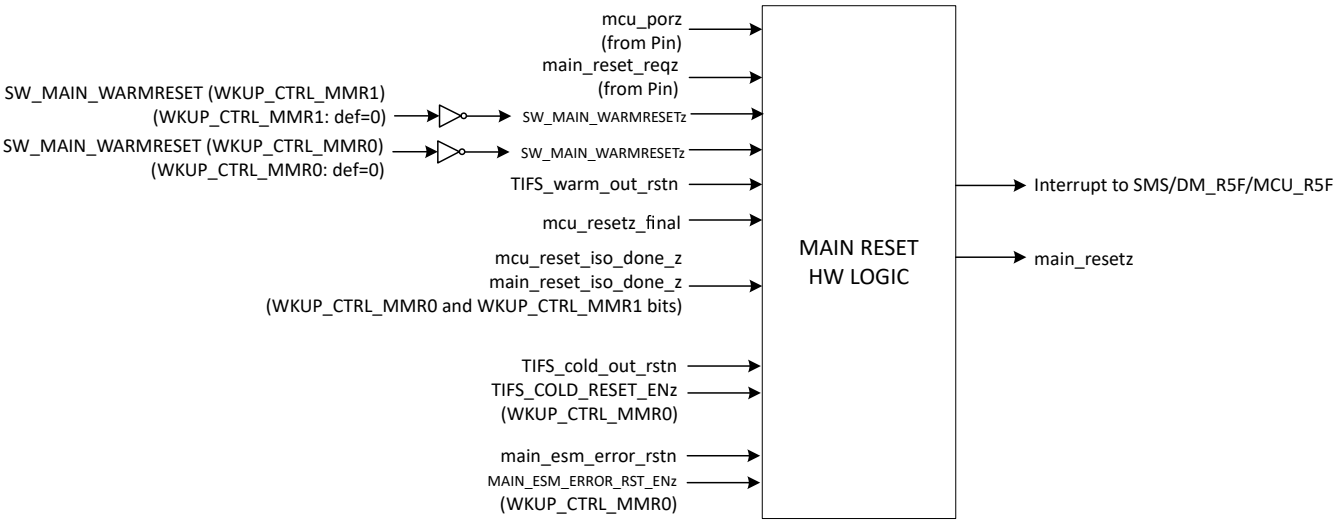


Figure 6-17. MAIN Domain Resetz Hardware Logic Diagram

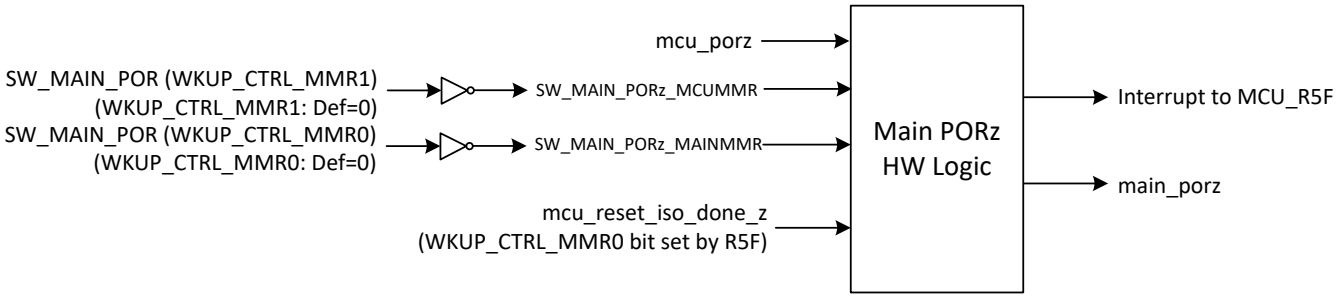


Figure 6-18. MAIN Domain PORz Hardware Logic Diagram

Copyright © 2026 Texas Instruments Incorporated

6.3.2 Reset Sources

Device Reset Sources summarizes the available reset sources that are supported by the device.

Table 6-33. Device Reset Sources

Reset Name	Device Domain	Reset Type	Sync/ Async	Pin/ Register/ Internal	Details
MCU_PORz	MCU MAIN	Hardware POR	Async	MCU_PORz HW Pin	MCU_PORz Details
SW_MAIN_PORz	MAIN	Software POR	Sync	MMR	SW_MAIN_PORz Details
MCU_RESETz	MCU MAIN	Hardware warm reset	Async	MCU_RESETz HW Pin	MCU_RESETz Details
SW_MCU_WARMRSTz	MCU MAIN	Software warm reset	Sync	MMR	SW_MCU_WARMRSTz Details
MAIN_RESETz_REQ	MAIN	Hardware warm reset	Sync	RESETz_REQ HW Pin	MAIN_RESETz_REQ Details
SW_MAIN_WARMRSTz	MAIN	Software warm reset	Sync	MMR	SW_MAIN_WARMRSTz Details
VTM Thermal Reset	MCU MAIN	Thermal error warm reset	Async	Internal Signal	VTM Thermal Alert Reset Details
ESM_ERRORz	MCU MAIN	ESM error warm reset	Async	Internal Signal	ESM_ERRORz Reset Details
SMS_COLD_OUT_RST_n	MCU MAIN	SMS0 cold reset	Async	Internal Signal	SMS Reset Details
SMS_WARM_OUT_RST_n	MAIN	SMS0 warm reset	Async	Internal Signal	SMS Reset Details
DM_WDT_RST_n	MCU MAIN	DM Watchdog Timeout Reset during Deepsleep	Async	Internal Signal	MCU_RESETz Details

Copyright © 2026 Texas Instruments Incorporated

6.3.3 Reset Status

Reset Status shows the Reset Status functionality. An output pin or software bit is used to represent the status of a specific individual reset.

Table 6-34. Reset Status

Reset Status Name	Reset Status Source	Reset Status Info	Signal Active Level	Reset Signal Details
MCU_CTRL_MMR_CFG0_RST_STAT Status Register	Register	4 bits in CTRLMMR on MCU warm reset status and assertion.	Status bits read active HIGH (1) when a particular reset is asserted.	
WKUP_CTRL_MMR_CFG0_RST_STAT Status Register	Register	4 bits in CTRLMMR on WKUP warm reset status and assertion.	Status bits read active HIGH (1) when a particular reset is asserted.	
PORz_OUTz	Output Pin	On/Off pin status of MAIN PORz reset	Active LOW (0)	MAIN_PORz_OUT Status Pin
RESETSTATz	Output Pin	On/Off pin status of MAIN warm reset	Active LOW (0)	MAIN_RESETSTATzStatus Pin
MCU_RESETSTATz	Output Pin	On/Off pin status of MCU PORz reset	Active LOW (0)	MCU_RESETSTATz Status Pin
MCU_ERRORn	Output Pin	On/Off pin status of MCU ESM_ERROR reset	Active LOW (0)	MCU_SAFETY_ERRORn Status Pin

Copyright © 2026 Texas Instruments Incorporated

6.3.3.1 Reset Source Status Registers

All warm reset sources which trigger a device/domain warm reset are captured in the MCU domain CTRLMMR reset status register **MCU_CTRL_MMR_CFG0_RST_SRC**. This register is only reset on MCU_PORz.

The MCU domain CTRLMMR **MCU_CTRL_MMR_CFG0_RST_SRC** register is shadowed in WKUP domain CTRLMMR **WKUP_CTRL_MMR_CFG0_RST_SRC** so that MAIN domain processors can read it without directly accessing the MCU domain CTRLMMR.

After recovery from warm reset, software can read the CTRLMMR reset source register **WKUP_CTRL_MMR_CFG0_RST_SRC** to identify the source of previous reset. After reading this reset source register, software must clear the register by writing to the MCU domain CTRLMMR, **MCU_CTRL_MMR_CFG0_RST_SRC**.

Reset status bits read active HIGH (1) when a particular reset is triggered.

The following reset sources, which cause MCU and MAIN domain resets, are captured in the MCU domain CTRLMMR reset source register:

- MCU_RESETz (Reset by Hardware Pin)
- SW_MCU_WARMRSTz (Reset by MCU domain CTRLMMR [1])
- SW_MAIN_PORz_MCUMMR (Reset by MCU domain CTRLMMR [1])
- SW_MAIN_PORz_MAINMMR (Reset by MAIN domain CTRLMMR [1])
- MAIN_RESET_REQz (Reset by Hardware Pin)
- SW_MAIN_WARMRSTz_MCUMMR (Reset by MCU domain CTRLMMR [1])
- SW_MAIN_WARMRSTz_MAINMMR (Reset by MAIN domain CTRLMMR [1])
- SMS_COLD_OUT_RST_n (Reset by SMS)
- SMS_WARM_OUT_RST_n (Reset by SMS)
- THERM_MAXTEMP_OUTRANGE_ALERT (Reset by VTM Max Temperature Alert)
- SYSR_ASSERTRESET_n (Reset by DEBUGSS)
- ESM_ERROR_RST_n (Reset by ESM Error)
- DM_WDT RST_n (Reset by WKUP WDT Timeout during Deepsleep mode)
- DM_MAIN_PORz (Reset by DS_MAIN_POR_PDOFF a WKUP domain CTRLMMR [1] during deepsleep)
  - MCU ESM Error will reset entire device
  - MAIN ESM Error will only reset MAIN domain when MCU domain is isolated. When MCU domain is not isolated, it will warm reset both MCU and MAIN domains.

Note

1. MAIN domain software WARMRSTz and PORz control registers are defined in both WKUP domain CTRLMMR and MCU domain CTRLMMR.

When the MCU domain is isolated from MAIN domain, MCU R5FSS processor can reset the MAINdomain, by programming the reset control registers defined in the MCU domain CTRLMMR. This way MCU R5FSS does not need to access MAIN domain CTRLMMR for MAIN domain reset action.

Whena MAIN domain processor (A53, SMS, R5F) needs to issue resets to MAIN domain on error detection they will use the WARMRESETz and PORz MMR bits defined in the WKUP domain CTRLMMR.

6.3.3.2 MCU_RESETSTATz Status Pin

The MCU_RESETSTATz pin indicates the internal reset status of the device (active LOW). When LOW, it indicates that the MCU and MAIN domains are in warm reset state. When HIGH, it indicates that the MCU is out of warm reset state. Refer to Section 6.3.3.4 for MAIN domain reset status.

Copyright © 2026 Texas Instruments Incorporated

6.3.3.3 MCU_ERRORn Status Pin

The MCU_ERRORn pin indicates the MCU domain ESM status. This Output is used to signal an external agent that it needs to (or may need to) intervene because of an error. The Error Pin Output is active Low or PWM, based on the ESM Error Pin Control register pwm_en field (Refer ESM IP) . This pwm_en field should only be modified when the ESM is disabled, based on the Global Enable register. Refer ESM chapter for details.

While in Power-On-Reset, the Error Pin is active (asserted low). SoC drives a Low via a weak internal pull-down. After Power-On-Reset condition is removed, ESM will drive the Error Pin. During a Warm Reset the state of the Error Pin is unchanged (i.e. the Error Pin logic is only reset by asserting MCU_PORz low).

When MCU_ERRORn output pin is configured to operate in level mode, a LOW indicates the MCU ESM module has registered an error. A HIGH indicates that there are no MCU ESM errors.

When MCU_ERRORn output pin is configured to operate in PWM mode, no error is indicated by continuous toggle according to programmable MMR widths for high and low periods. When an error occurs, the error pin stops toggling and remains constant until the error is cleared. An external device (such as a PMIC) that is detecting the PWM toggles can identify the error if the pin stops toggling. The periods should be programmed such that they fit within the expectation of the external device.

Note

This signal can be used by external safety devices to determine the device status and assert MCU_PORz if necessary.

6.3.3.4 RESETSTATz Status Pin

This pin indicates warm reset status (active LOW) of the MAIN domain.

When LOW, it indicates that the MAIN domain is in a warm reset state.

When HIGH, it indicates that the MAIN domain is out of reset.

The status of this signal will be captured in WKUP_CTRL_MMR1 registers.

6.3.3.5 PORz_OUT Status Pin

This pin indicates the POR status of the MAIN domain (active LOW).

When LOW, it indicates the MAIN domain is in the POR state.

When HIGH, it indicates the MAIN domain is out of the POR state.

Note

PORz_OUT may be used to enable and/or disable output buffers of attached devices that share pins associated with BOOTMODE inputs, such that appropriate boot configuration values are allowed to propagate into the device without contention.

Copyright © 2026 Texas Instruments Incorporated

6.3.4 Reset Controls

The reset control registers enable, disable, and adjust specific reset operations. This section explains how to control certain device resets.

6.3.4.1 Reset Control Registers

MCU Domain Reset Control Registers shows the CTRLMMR registers that are able to enable and disable reset abilities and options.

Table 6-35. MCU Domain Reset Control Registers

Control Name	Control Action	Control Register
MCU_ESM_ERROR_RST_ENz	Enable MCU ESM_ERRORz reset	**MCU_CTRL_MMR_CFG0_RST_CTRL**
SMS_COLD_RESET_ENz	Enable SMS cold reset of MCU domain	**MCU_CTRL_MMR_CFG0_RST_CTRL**

Table 6-36. MAIN Domain Reset Control Registers

Control Name	Control Action	Control Register
MAIN_ESM_ERROR_RST_ENz	Enable MAIN ESM_ERRORz Reset	**WKUP_CTRL_MMR_CFG0_RST_CTRL**
SMS_COLD_RESET_ENz	Enable SMS Cold Reset of MAIN Domain	**WKUP_CTRL_MMR_CFG0_RST_CTRL**

Note

For more information, see *Control Module (CTRLMMR)*.

Copyright © 2026 Texas Instruments Incorporated

6.3.4.2 Reset Isolation

Reset Isolation Introduction

It is possible to isolate certain SoC modules and subsystems and prevent them from undergoing an arbitrarily generated warm reset. This is known as Reset Isolation. This section describes the device reset isolation capabilities.

Simple Reset Isolation Sequence Diagram shows a simple description of the device reset isolation sequence.

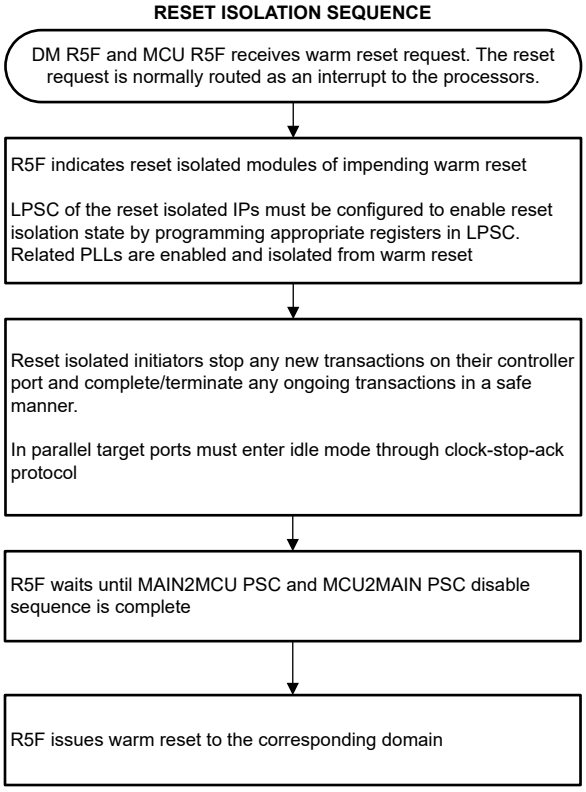


Figure 6-19. Simple Reset Isolation Sequence Diagram

MCU Reset Isolation Details

Detailed Reset Isolation Sequence Diagram shows a detailed description of the MCU reset isolation sequences.

Copyright © 2026 Texas Instruments Incorporated

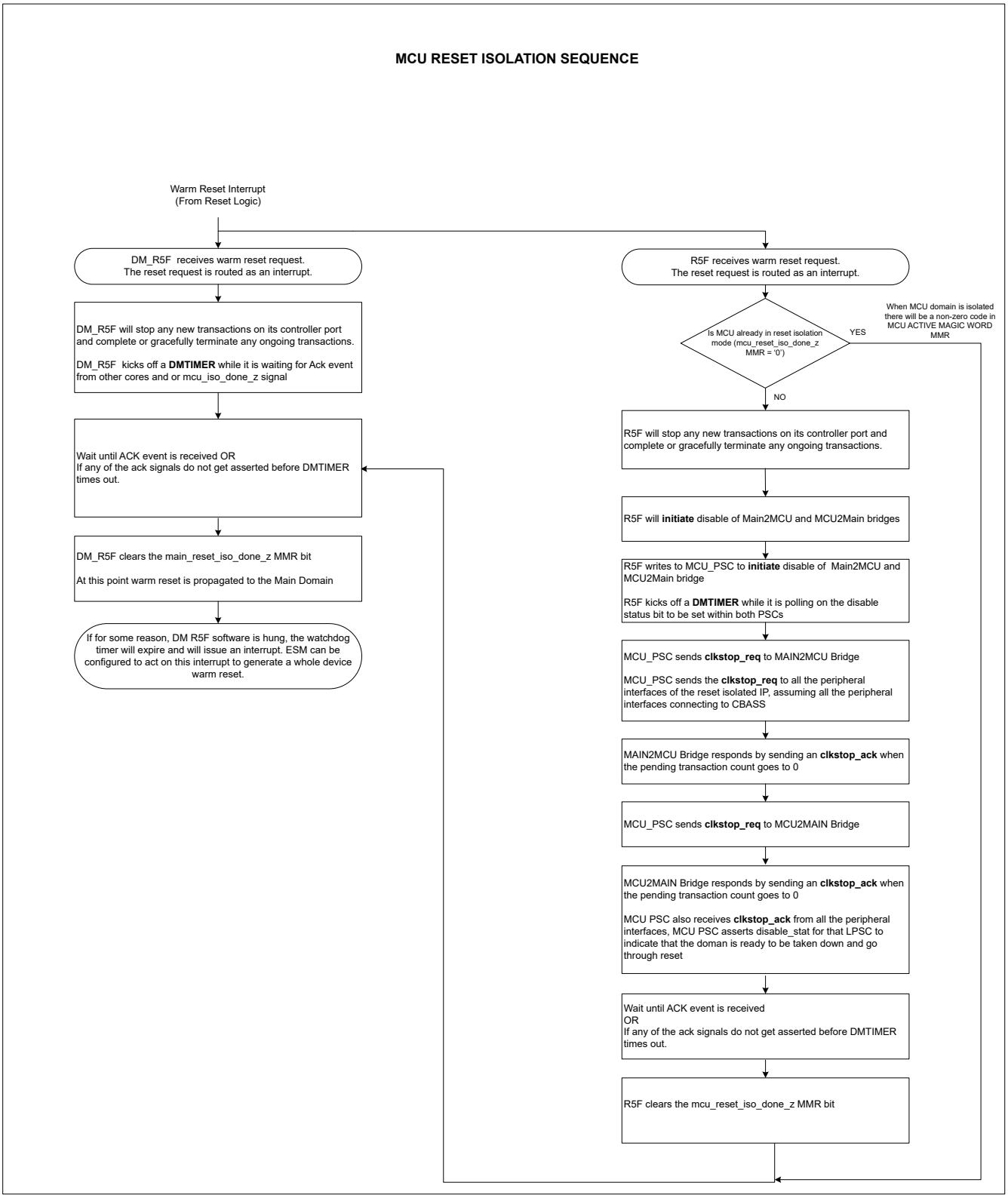


Figure 6-20. Detailed Reset Isolation Sequence Diagram

Copyright © 2026 Texas Instruments Incorporated

6.3.5 Reset Details

Reset Details Introduction

The Reset Details section will break down the available device resets and explain operating details such as sequencing and reset logic diagrams. It also includes any additional information related to the device resets.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.1 POR Resets

6.3.5.1.1 SW_MAIN_PORz Reset

Reset Overview

This reset is a software controlled MAIN domain POR reset defined in **WKUP_CTRL_MMR_CFG0_RST_CTRL** and **MCU_CTRL_MMR_CFG0_RST_CTRL**.

This software reset will generate a MAIN domain PORz.

R5FSS processor in MCU domain can trigger PORz to the MAIN domain by writing to the **MCU_CTRL_MMR_CFG0_RST_CTRL** register.

MAIN domain processors can reset the MAIN domain by writing to the **WKUP_CTRL_MMR_CFG0_RST_CTRL** register.

When the MCU domain is configured as a safety domain it must be isolated from SW_MAIN_PORz.

A reset isolation sequence for the MCU domain must be complete prior to reset propagation.

WKUP_CTRL_MMR_CFG0_RST_CTRL and **MCU_CTRL_MMR_CFG0_RST_CTRL** define a 4-bit field, SW_MAIN_POR[7:4], for generating a software-controlled POR for the MAIN domain (SW_MAIN_PORz).

When SW_MAIN_POR [7:4] field is set to “0110”, the MAIN domain is in POR reset state (SW_MAIN_PORz = LOW).

When SW_MAIN_POR [7:4] is set to any other value, the MAIN domain is out of POR reset state (SW_MAIN_PORz = HIGH).

This bit field is reset to “1111” (Inactive State) by default.

MCU Domain Effect

When MCU domain is configured as an independent domain, then the entire MCU domain is reset isolated. When MCU domain is not configured as independent domain then this SW_MAIN_PORz must not be used. Instead MCU_PORz pin functions should be used.

MCU IOs are not affected.

MAIN Domain Effect

All modules in MAIN domain are reset (including reset isolated modules).

IOs will enter safe state (as defined in the BALL STATE DURING RESET column of the Datasheet Pin Attributes table while this reset is active.

MAIN domain CTRLMMR boot configuration register is reset.

Device BOOTMODE pins will be re-latched after this reset is de-asserted.

Device will re-boot. During boot-up, R5FSS (secondary boot loader) will poll the CTRLMMR reset status and MCU ACTIVE MAGIC WORD registers and configure MCU domain/R5FSS processor accordingly.

Additional Details

PORz_OUT:

This pin indicates POR status output of the MAIN domain (active LOW).

When LOW, it indicates the MAIN domain is in POR state.

When HIGH, it indicates that the MAIN domain is out of POR state.

Copyright © 2026 Texas Instruments Incorporated

Note

PORz_OUT may be used to enable and/or disable output buffers of attached devices that share IOs associated with BOOTMODE inputs, such that appropriate boot configuration values are allowed to propagate into the device without contention

Note

The boot mode pins are latched internally on the rising edge of PORz_OUT and are reset isolated from all other device resets.

6.3.5.1.2 MCU_PORz Reset

Reset Overview

This is the POR reset signal (active LOW) for the entire device, controlled by the MCU_PORz HW Pin.

When LOW, it performs a POR reset on the entire device (MCU and MAIN domains) and puts IOs in a safe state (as defined in the BALL STATE DURING RESET column of the datasheet Pin Attributes table).

MCU Domain Effect

All modules in MCU domain are reset.

MCU IOs are in HHV state (Reset State).

When MCU_PORz is de-asserted, MCU IOs will enter the default state defined in the device Datasheet. MCU domain will be reconfigured by the device boot processor (secondary boot loader).

MAIN Domain Effect

All modules in the MAIN domain are reset.

MAIN domain IOs are in HHV state (as defined in the BALL STATE DURING RESET column of the datasheet Pin Attributes table).

When MCU_PORz is de-asserted, the MAIN domain IOs will enter default state as defined in the BALL STATE AFTER RESET column of the datasheet Pin Attributes table.

The device boot processor (secondary boot loader) will setup R5FSS as Safety or General-Purpose Processor.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.2 Warm Resets

6.3.5.2.1 MAIN Domain Warm Reset Sequence Flow

MAIN Domain Warm Reset Sequence Flowchart explains the process for a typical MAIN domain warm reset.

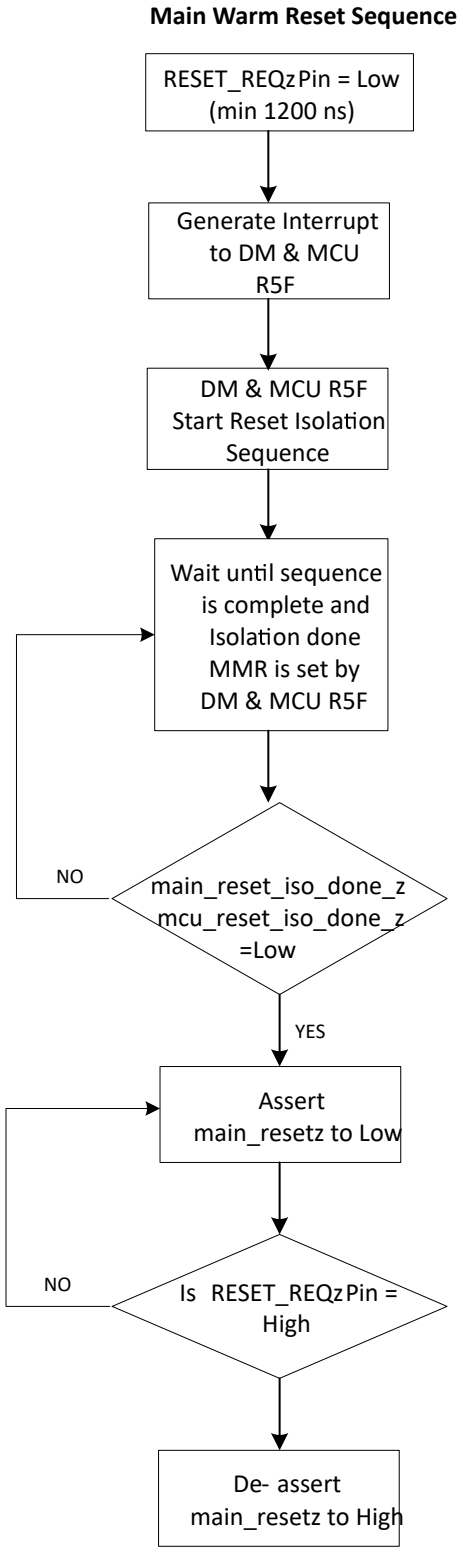


Figure 6-21. MAIN Domain Warm Reset Sequence Flowchart

Copyright © 2026 Texas Instruments Incorporated

6.3.5.2.2 MAIN_RESETz_REQ Reset

Reset Overview

This reset signal is the MAIN domain warm reset request (active LOW) controlled by the RESET_REQz HW pin.

When LOW is detected, reset hardware generates an interrupt to processors (A53SS/R5FSS) so they can complete the reset isolation sequence before reset is propagated.

MCU Domain Effect

When MCU domain is configured to operate independently, MCU domain reset isolation sequence is completed before propagating the RESETz to MAIN domain.

MCU IOs are not affected.

When MCU domain is not configured as independent, then this reset will also warm reset MCU domain.

MAIN Domain Effect

This is a MAIN domain warm reset request. First, the reset isolation sequence is applied and then the reset is propagated.

All modules in MAIN domain are reset except for warm reset isolated modules and MAIN domain CTRLMMR register bits which are reset only on MAIN_PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

Reason for this reset is captured in the CTRLMMR reset source register **WKUP_CTRL_MMR_CFG0_RST_SRC**. After reset is de-asserted, device will boot-up. During device boot-up, DM R5F secondary boot loader will read the reset status and MCU ACTIVE MAGIC WORD registers and reconfigure the MCU R5F processor accordingly.

Reset Sequence

The RESET_REQz reset sequence is described below.

RESET_REQz Sequence shows the timing between the RESET_REQz reset signal and the internal RESETz (MAIN domain warm reset) signal. Refer to the datasheet for timing and pulse width requirements.

Copyright © 2026 Texas Instruments Incorporated

MAIN WARM RESET (RESET_REQz Pin)

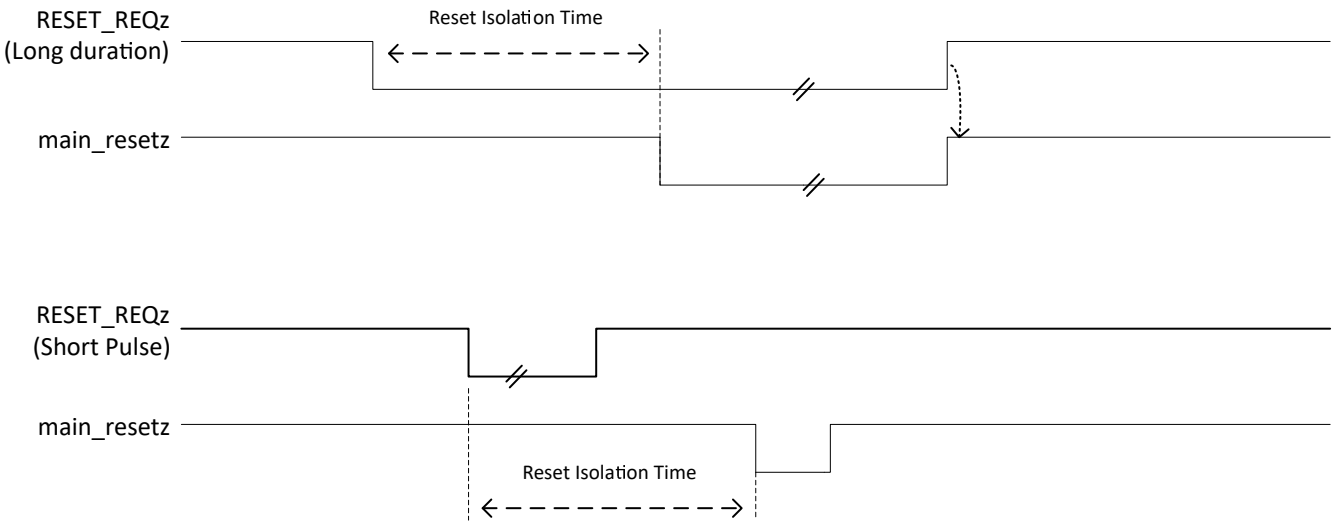


Figure 6-22. RESET_REQz Sequence

Additional Reset Details

RESETSTATz:

The RESETSTATz pin indicates the MAIN domain internal reset status (active LOW). This is a reflection of the reset status after the RESET_REQz reset signal (RESET_REQz HW Pin) is asserted.

When LOW, it indicates that the MAIN domain is in internal reset state.

When HIGH, it indicates that the MAIN domain is out of internal reset state.

RESETSTATz Timing shows the timing between the RESET_REQz reset signal and the RESETSTATz output. Refer to the datasheet for timing and pulse width requirements.

Copyright © 2026 Texas Instruments Incorporated

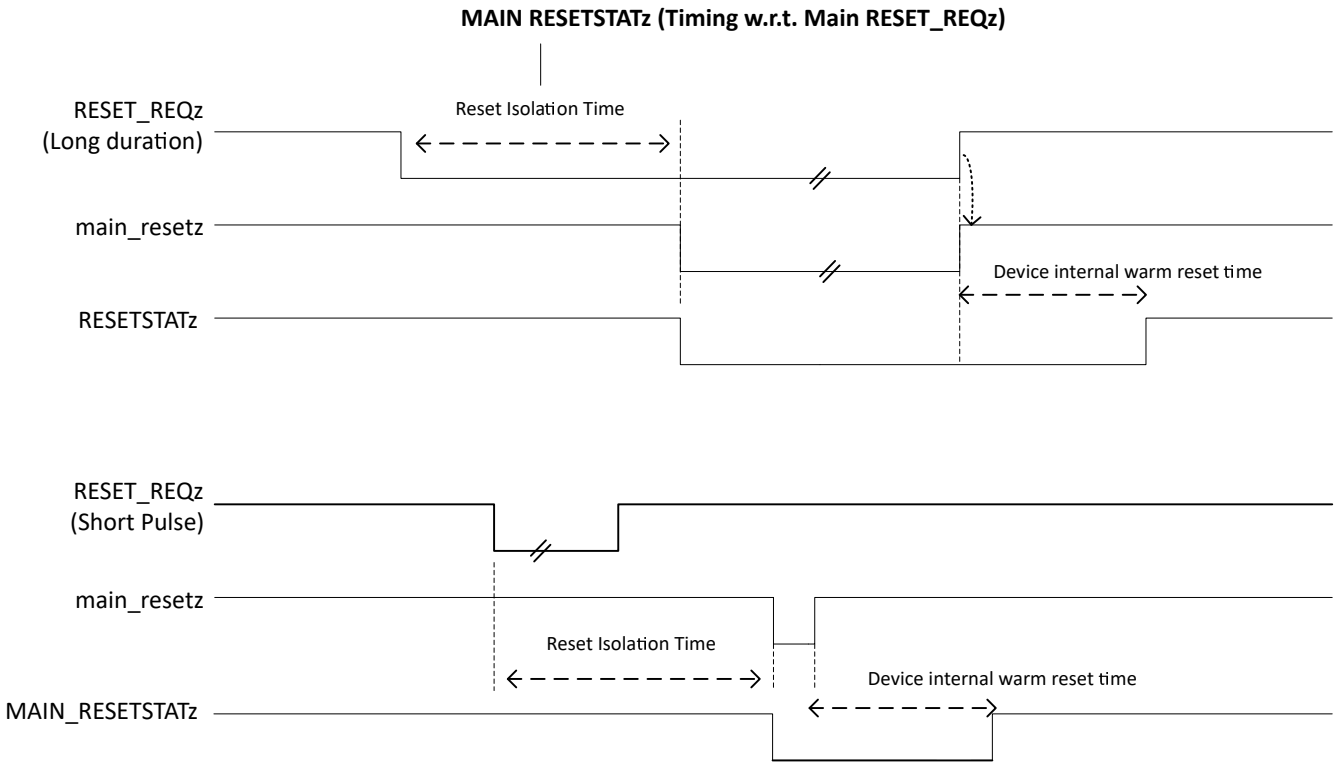


Figure 6-23. RESETSTATz Timing

Note

For more details see MAIN_RESETSTATz Status Pin.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.2.3 SW_MAIN_WARMRSTz Reset

Reset Overview

This reset is software controlled MAIN domain warm reset defined in **WKUP_CTRL_MMR_CFG0_RST_CTRL** and **MCU_CTRL_MMR_CFG0_RST_CTRL**.

The R5FSS can issue a warm reset to the MAIN domain, by writing to the MCU domain CTRLMMR register.

The MAIN domain processors can issue a warm reset the WKUP domain, by writing to the MAIN domain CTRLMMR register.

Note

When R5FSS is configured as safety processor it must be reset isolated from this MAIN domain warm reset.

A reset isolation sequence for MCU domain has to be complete prior to this reset propagation.

WKUP domain CTRLMMR and MCU domain CTRLMMR registers define a 4-bit field, SW_MAIN_WARMRST[3:0] for generating a software controlled warm reset for the MAIN domain (SW_MAIN_WARMRSTz).

When SW_MAIN_WARMRST[3:0] field is set to “0110”, a MAIN domain warm reset is active (SW_MAIN_WARMRSTz = LOW).

When SW_MAIN_WARMRST[3:0] is set to any other value, the MAIN domain warm reset is inactive (SW_MAIN_WARMRSTz = HIGH).

This bit field is reset to “1111” (Inactive State) by default.

Note

This software reset is equivalent to RESET_REQz reset signal (RESET_REQz HW Pin) functionality.

MCU Domain Effect

When MCU domain is configured to operate independently, MCU domain reset isolation sequence is completed before propagating the RESETz to main domain.

MCU IOs are not affected.

When MCU domain is not configured as independent then, this reset will also warm reset MCU domain.

MAIN Domain Effect

This is a MAIN domain reset request. First, the reset isolation sequence is applied and then the reset is propagated.

All modules in MAIN domain are reset except for reset isolated modules and MAIN domain CTRLMMR register bits which are reset only on MAIN_PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

Reason for this reset is captured in MAIN domain CTRLMMR reset status register **WKUP_CTRL_MMR_CFG0_RST_STAT**. After reset is de-asserted, device will boot-up. During device boot-up, R5FSS (secondary boot loader) will read the reset status and MCU ACTIVE MAGIC WORD registers and reconfigure the MCU domain/R5FSS processor accordingly.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.2.4 MCU_RESETz Reset

Reset Overview

The MCU_RESETz signal is the warm reset input (active low) to the entire device, controlled by the MCU_RESETz HW Pin.

When active, it resets all modules in the MCU and MAIN domains.

MCU_RESETz will not reset selected **WKUP_CTRL_MMR_CFG0_RST_CTRL** and **MCU_CTRL_MMR_CFG0_RST_CTRL** register bits. These bits are only reset by PORz.

MCU Domain Effect

All modules in MCU domain are reset except for **MCU_CTRL_MMR_CFG0_RST_CTRL** bits which are reset only on MCU_PORz.

IOs are not affected.

R5FSS is reset.

When MCU_RESETz is de-asserted, the MCU domain needs to be reconfigured by R5FSS (secondary boot loader) in the MAIN domain.

MAIN Domain Effect

All modules in the MAIN domain are reset except for CTRLMMR bits which are reset only by PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

When MCU_RESETz is de-asserted, the device goes through full boot. The reason for this reset is captured in the CTRLMMR reset source register **MCU_CTRL_MMR_CFG0_RST_SRC**.

During device boot-up, the R5FSS (secondary boot loader) will read the CTRLMMR reset status and MCU ACTIVE MAGIC WORD registers and reconfigure the MCU domain/R5FSS processor accordingly.

Additional Details

MCU_RESETSTATz:

This pin indicates internal MCU_RESETz reset status (active LOW).

When LOW, it indicates that the device, both MCU and MAIN domains, are in internal reset state.

When HIGH, it indicates that the device MAIN domains, are is out of internal reset state.

Note

In order for the MAIN domain to come out of reset, RESETz_REQ (RESET_REQz HW Pin) must also be de-asserted.

Note

For more details see MCU_RESETSTATz Status Pin.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.2.5 SW_MCU_WARMRSTz Reset

Reset Overview

This is a software MCU domain warm reset defined in MCU domain CTRLMMR.

MCU CTRLMMR defines a 4-bit field, SW_MCU_WARMRST[11:8] for generating a software warm reset in the MCU domain (SW_MCU_WARMRSTz).

When SW_MCU_WARMRST[11:8] field is set to “0110”, MCU warm reset is active (SW_MCU_WARMRSTz = LOW).

When SW_MCU_WARMRST[11:8] is set to any other value, MCU warm reset is inactive (SW_MCU_WARMRSTz = HIGH).

This bit field is reset to “1111” (Inactive State) by default.

Note

This software reset is equivalent to MCU_RESETz warm reset signal (MCU_RESETz HW Pin) functionality.

MCU Domain Effect

All modules in the MCU domain are reset except for MCU domain CTRLMMR bits which are reset only on MCU_PORz.

IOs are not affected.

R5FSS processor is reset.

When MCU_RESETz is de-asserted, the MCU domain needs to be reconfigured by the R5FSS (secondary boot loader) in the MAIN domain.

MAIN Domain Effect

All modules in the MAIN domain are reset except for CTRLMMR bits which are reset only on PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

When MCU_RESETz is de-asserted, the device goes through full boot. The reason for this reset is captured in CTRLMMR reset source register**WKUP_CTRL_MMR_CFG0_RST_SRC**.

During device boot-up, the R5FSS (secondary boot loader) will read the CTRLMMR reset status and MCU ACTIVE MAGIC WORD and reconfigure the MCU domain/R5FSS processor accordingly.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.3 SMS Resets

6.3.5.3.1 SMS_WARM_OUT_RST_n Reset (MAIN Domain)

Reset Overview

The SMS will issue this reset upon a security error.

When this reset is enabled in MAIN domain, it causes a MAIN domain warm reset.

This is a synchronous reset type (needs to complete reset isolation sequence).

Note

This reset behavior is same as the RESETz_REQ reset signal (RESET_REQz HW Pin).

MCU Domain Effect

When MCU domain is configured to operate independently, MCU domain reset isolation sequence is completed before propagating the RESETz to main domain.

MCU IOs are not affected.

When MCU domain is not configured as independent then, this reset will also warm reset MCU domain.

MAIN Domain Effect

This is a MAIN domain warm reset request. First, the reset isolation sequence is applied and then the reset is propagated.

All modules in MAIN domain are reset except for CTRLMMR register bits which are reset only on PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

Reason for this reset is captured in CTRLMMR reset source status register **WKUP_CTRL_MMR_CFG0_RST_SRC**. After reset is de-asserted, device will boot-up. During device boot-up, R5FSS (secondary boot loader) will read the reset status and MCU ACTIVE MAGIC WORD registers and reconfigure the MCU domain/R5FSS processor accordingly.

6.3.5.3.2 SMS_COLD_OUT_RST_n Reset (MAIN Domain)

Reset Overview

This reset is a MAIN domain warm reset that is executed from the SMS. This is an asynchronous reset type (takes effect immediately). This SMS reset of the Main domain is enabled only when the SMS_COLD_RESET_EN_z bit in WKUP domain CTRLMMR is '0'.

Note

This reset behavior is same as the RESETz_REQ reset signal (RESET_REQz HW Pin) with the exception that it takes effect immediately without need for any reset isolation sequence.

MCU Domain Effect

When MCU domain is configured to operate independently, MCU domain reset isolation sequence is completed before propagating the RESETz to main domain.

MCU IOs are not affected.

When MCU domain is not configured as independent then, this reset will also warm reset MCU domain.

Copyright © 2026 Texas Instruments Incorporated

MAIN Domain Effect

This is a MAIN domain asynchronous warm reset. Propagates immediately wihout any reset isolation sequence.

All modules in MAIN domain are reset except CTRLMMR register bits which are reset only on PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

Reason for this reset is captured in CTRLMMR reset status register. After reset is de-asserted, device will boot-up. During device boot-up, R5FSS (secondary boot loader) will read the reset status and MCU ACTIVE MAGIC WORD registers and reconfigure the MCU domain/R5FSS processor accordingly.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.3.3 SMS_COLD_OUT_RST_n Reset (MCU Domain)

Reset Overview

The device SMS will issue this reset upon a security error.

When this reset is enabled in MCU domain, it causes a whole device warm reset.

This SMS reset of the MCU domain is enabled only when the SMS_COLD_RESET_ENz bit in MCU domain CTRLMMR is '0'.

When the R5FSS is configured as a safety processor, it can block this reset from resetting the MCU domain.

This is an asynchronous reset type (takes effect immediately)

Note

This reset behavior is same as MCU_RESETz reset signal (MCU_RESETz HW Pin).

MCU Domain Effect

All modules in MCU domain are reset except for CTRLMMR bits which are reset only on MCU_PORz.

IOs are not affected.

R5FSS is reset.

When MCU_RESETz is de-asserted, the MCU domain will be reconfigured by R5FSS (secondary boot loader) in the MAIN domain.

MAIN Domain Effect

All modules in the MAIN domain are reset except for CTRLMMR bits which are reset only on PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

When MCU_RESETz is de-asserted, the device goes through full boot. The reason for this reset is captured in the CTRLMMR reset source register **WKUP_CTRL_MMR_CFG0_RST_SRC**.

During device boot-up, the R5FSS (secondary boot loader) will read the CTRLMMR reset status and MCU ACTIVE MAGIC WORD and reconfigure the MCU domain/R5FSS processor accordingly.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.4 VTM Thermal Alert Reset

Reset Overview

The VTM has a hardware alert signal that indicates a fault when any of the sensors listed in *Voltage and Thermal Manager Locations* detect that the device is nearing the maximum temperature allowed.

This alert can be used to reset the device or part of the device to reduce the temperature. This alert is not a standard interrupt and so it does not include the usual set, clear and enable features. It does include an overall enable MMR bit, VTM_CFG2_MISC_CTRL, along with enables per sensor from VTM_CFG1 Registers.

The VTM module outputs an alert signal (THERM_MAXTEMP_OUTRANGE_ALERT), when the device temperature goes beyond a maximum threshold.

This reset signal creates a MCU domain warm reset.

VTM alert will be active as long as the error condition (Device Temperature > Maximum Temperature Threshold) is still TRUE.

When the error goes away, VTM alert will be de-asserted and device will come out of warm reset.

Note

This signal, when asserted HIGH, will cause a device warm reset similar to MCU_RESETz.

MCU Domain Effect

All modules in MCU domain are reset except for MCU domain CTRLMMR bits which are reset only on MCU_PORz.

IOs are not affected.

This reset is active as long as the VTM alert is active.

When THERM_MAXTEMP_OUTRANGE_ALERT is de-asserted, the MCU domain needs to be reconfigured by R5FSS (secondary boot loader) in the MAIN domain.

MAIN Domain Effect

All modules in the MAIN domain are reset except for CTRLMMR bits which are reset only on MAIN_PORz.

IOs are not affected.

All processor cores are reset (A53SS and R5FSS).

When THERM_MAXTEMP_OUTRANGE_ALERT is de-asserted, the device goes through full initialization procedure. The reason for this reset is captured in the CTRLMMR reset source register **WKUP_CTRL_MMR_CFG0_RST_SRC**.

Additional Reset Details

VTM alert use-case example:

VTM sends out a warning signal before generating the Max Temp Alert.

There are 2 different threshold settings for max temperature limits.

The first one has a lower threshold which will generate an interrupt to the processor for taking action to reduce frequency, turn-off power domains, etc. This should eventually lower the temperature and avoid the Max Temp Alert.

Copyright © 2026 Texas Instruments Incorporated

If the interrupt does not fix the rising temperature, then VTM will issue a Max Temp Alert. This will cause a device warm reset. The VTM Max Temp Alert will be active as long as die temperature is above the programmed threshold.

Device warm reset will default certain power domains to off mode (default at power-up). This should eventually lower the die temperature further.

Software can read the reset status register to poll on the cause of the warm reset. If it is from VTM, then it can wait for additional time before turning on the power domains.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.5 MAIN ESM_ERRORz Reset

Reset Overview

This reset is issued when a MAIN domain processor (A53SS, SMS, or R5FSS) detects a catastrophic software error or a MAIN domain WDT timeout event occurs.

Errors in the MAIN domain cause the MAIN ESM module to trigger ESM_ERRORz.

This is routed as internal MAIN domain warm reset when enabled by the WKUP domain CTRLMMR reset control bit (MAIN_ESM_ERROR_RST_ENz).

This is an asynchronous reset type (takes effect immediately).

Reset MCU Domain Effect

When MCU domain is configured to operate independently, MCU domain reset isolation sequence is completed before propagating the RESETz to main domain.

MCU IOs are not affected.

When MCU domain is not configured as independent then, this reset will also warm reset MCU domain.

Reset MAIN Domain Effect

All modules in MAIN domain are reset except for MAIN domain CTRLMMR bits which are reset only on PORz.

IOs are not affected.

The device will re-boot. During boot-up, the R5FSS (secondary boot loader) will poll the CTRLMMR reset registers and reconfigure the MCU domain/R5FSS processor accordingly.

Note

If the device boot fails, and the MCU_ERRORn level is still LOW, then an external safety device should issue the MCU_PORz Reset to allow the device to recover from this error.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.6 MCU ESM_ERRORz Reset

Reset Overview

This reset is issued when the R5FSS detects a catastrophic safety error or a MCU WDT timeout occurs.

Errors in the MCU domain cause the MCU ESM module to trigger error output.

This is routed as a device warm reset when enabled by the MCU domain WKUP_CTRL_MMR1 bit (MCU_ESM_ERROR_RST_ENz).

This is an asynchronous reset type (takes effect immediately).

MCU Domain Effect

All modules in MCU domain are reset except for MCU domain WKUP_CTRL_MMR1 bits which are reset only on PORz.

IOs are not affected.

Error is reported on MCU_ERRORn pin.

When SAFETY_ERROR_RST_n is de-asserted, the MCU domain needs to be reconfigured by the R5FSS (secondary boot loader) in the MAIN domain.

External hardware (PMIC or secondary MCU safety channel) must issue MCU_PORz to recover from this error.

MAIN Domain Effect

All modules in MAIN domain are reset except for reset isolated modules and MAIN domain WKUP_CTRL_MMR0 bits which are reset only on PORz.

IOs are not affected.

Device will re-boot. During boot-up, the R5FSS (secondary boot loader) will poll the CTRLMMR reset source register and configure the MCU domain/R5FSS processor accordingly.

If the device boot fails, and MCU_ERRORn level is still LOW, then an external safety device should issue the MCU_PORz reset to allow the device to recover from the error.

6.3.5.7 DM_WDT_RST_n Reset

Reset Overview

During Deeplseep resume sequence if the Device Manager is unable to respond due to a software bug, the WKUP domain WDT timer will issue this reset signal.

When this reset is enabled in MCU domain, it causes a whole device warm reset.

This WKUP WDT reset is enabled only when the DM_WDT_RST_EN_z bit in MCU domain CTRLMMR is '0'.

When the R5FSS is configured as a safety processor, it can block this reset from resetting the MCU domain. This is an asynchronous reset type (takes effect immediately)

Note

This reset behavior is same as MCU_RESETz reset signal (MCU_RESETz HW Pin).

MCU Domain Effect

All modules in MCU domain are reset except for MCU domain CTRLMMR bits which are reset only on MCU_PORz.

IOs are not affected. R5FSS is reset.

Copyright © 2026 Texas Instruments Incorporated

When this reset is de-asserted, the MCU domain will be reconfigured by R5FSS (secondary boot loader) in the MAIN domain.

MAIN Domain Effect

All modules in the MAIN domain are reset except for CTRLMMR bits which are reset only on PORz.

IOs are not affected.

All processor cores are reset (A53SS, SMS, and R5FSS).

When this reset is de-asserted, the device goes through full boot. The reason for this reset is captured inthe CTRLMMR reset source register.

During device boot-up, the R5FSS (secondary boot loader) will read the CTRLMMR reset status and MCU ACTIVE MAGIC WORD and reconfigure the MCU domain/R5FSS processor accordingly.

Copyright © 2026 Texas Instruments Incorporated

6.3.5.8 HHV

IOs support HHV mode during power up. HHV is active when PORz signal is driven LOW.

HHV allows IO cells to be tri-stated and is an IO feature. All IO cells have HHV which is asserted (driven) by HHV generated during PORz assertion. There are default pull values during this HHV/PORz assertion specified in the device Datasheet for each pin. All HHV logic controlling the buffer output enable and associated default internal pull for each IO will be applied.

HHV Timing Diagram demonstrates the timing sequence for the device HHV signals.

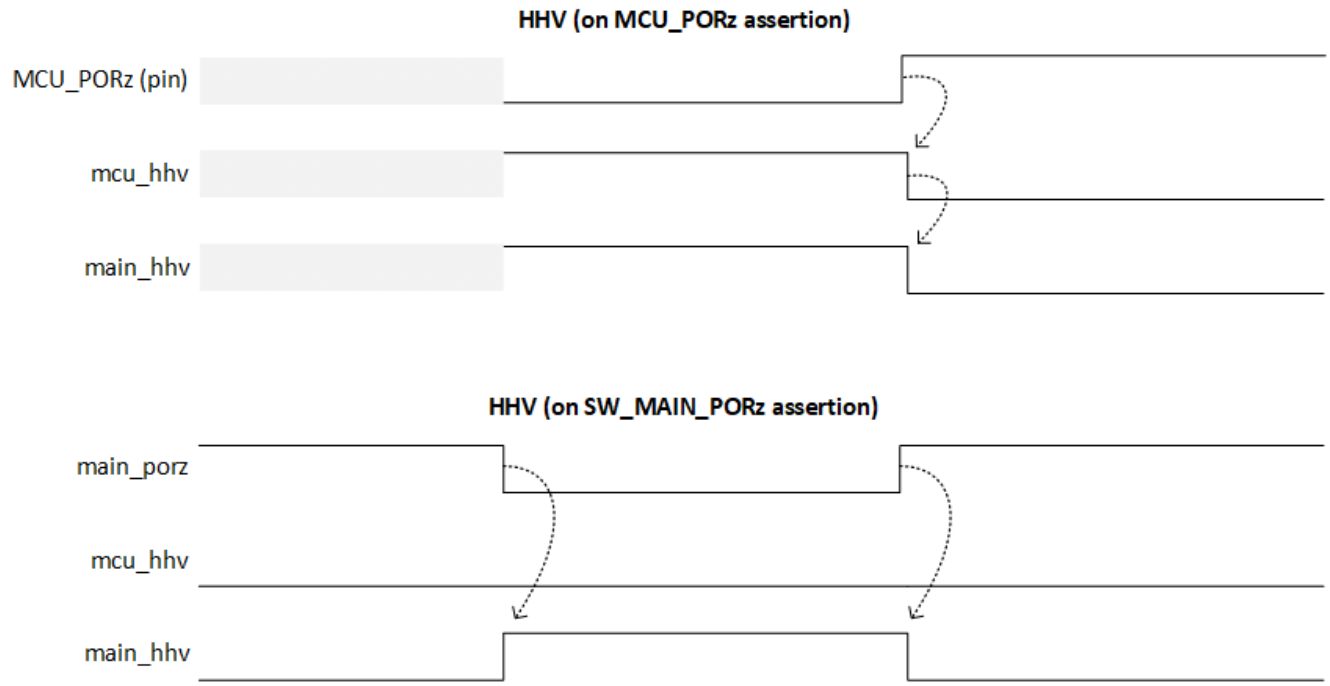


Figure 6-24. HHV Timing Diagram

6.3.5.9 BOOTMODE Pin Latching

This device requires several Boot Mode controls to enable device boot. The BOOTMODE inputs are not implemented on dedicated pins. They are multiplexed with other signal functions, where pins associated with the BOOTMODE inputs are defined in the Pin Attributes table. The logic state of the BOOTMODE inputs are latched on the rising edge of PORz_OUT. Valid logic states must be maintain as long as PORz_OUT is held low. The latched values are reflected in WKUP CTRL MMRs for read/write access.

Copyright © 2026 Texas Instruments Incorporated

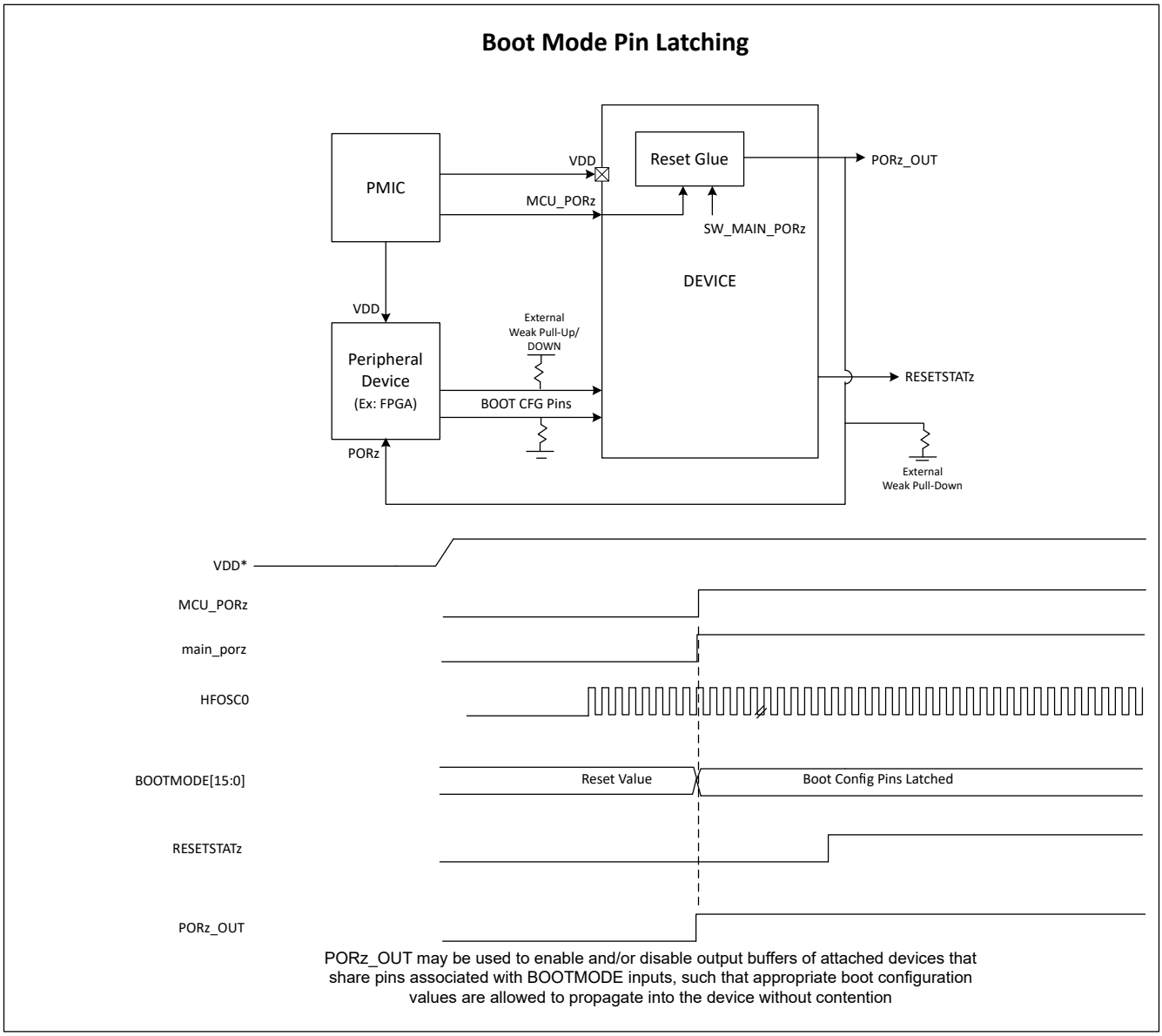


Figure 6-25. BOOTMODE Pin Latching

6.3.5.10 Power-on-Reset Sequence

Figure 6-26 and Figure 6-27 give a high-level overview of the device power-up sequence.

Copyright © 2026 Texas Instruments Incorporated

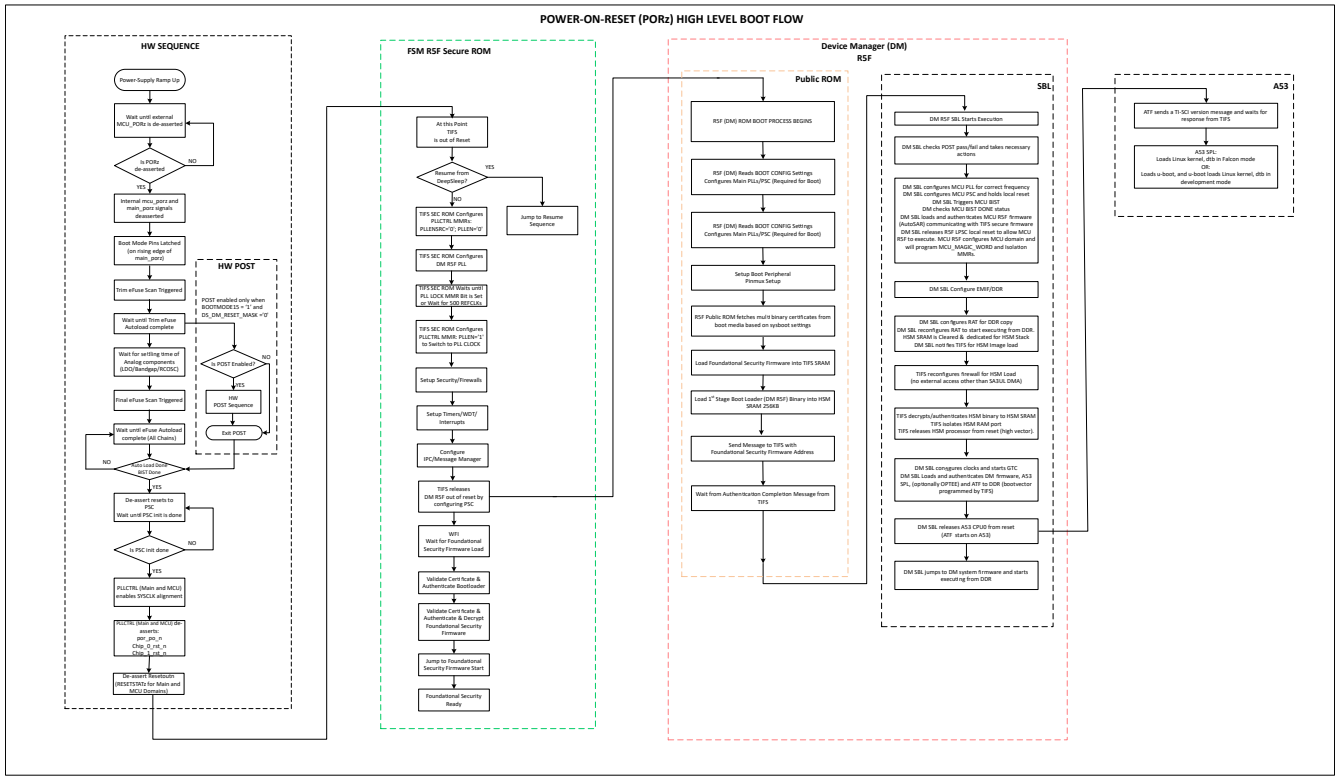


Figure 6-26. Power-up/Boot Flow Chart

Copyright © 2026 Texas Instruments Incorporated

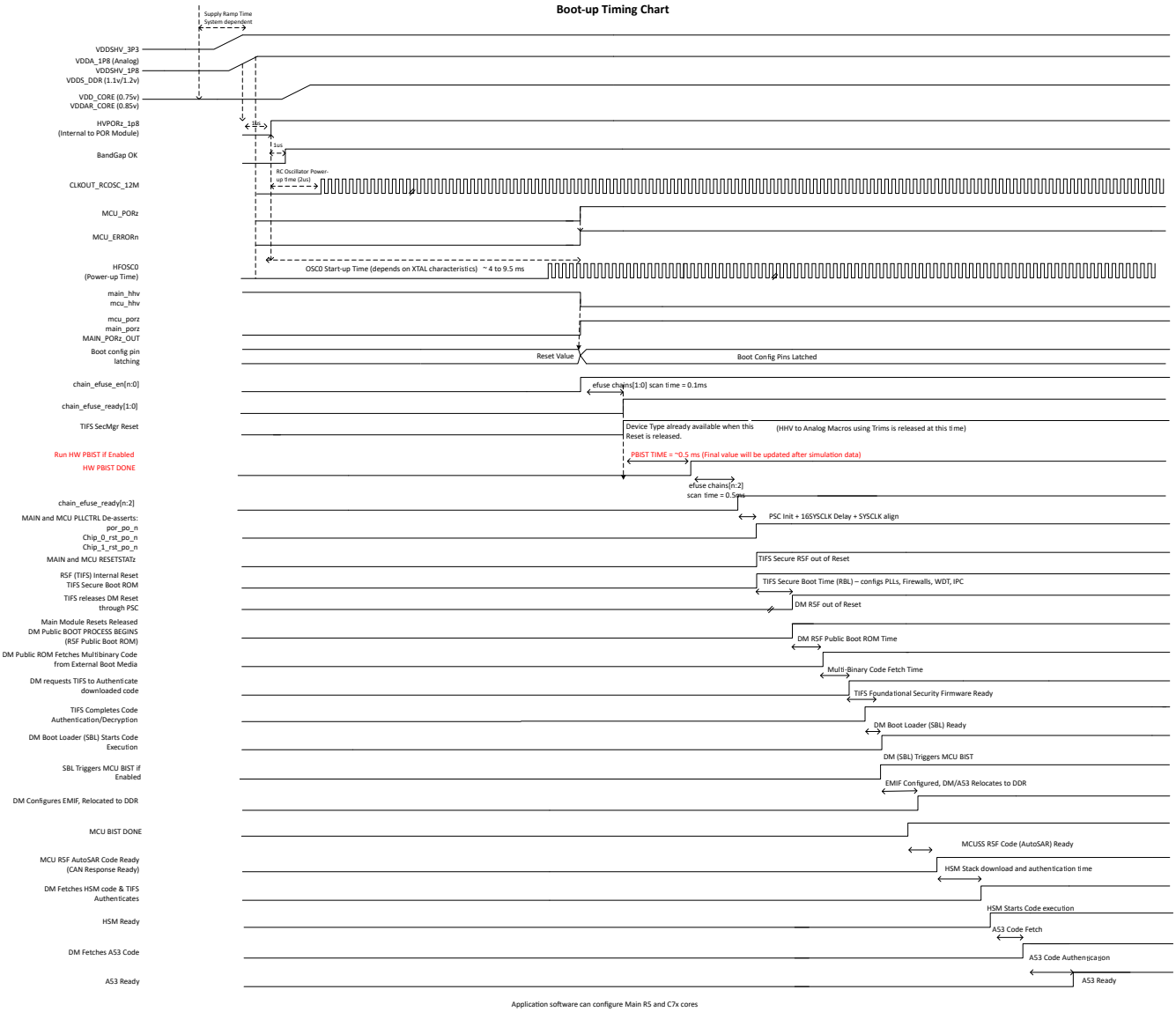


Figure 6-27. Power-up/Boot Timing Chart

Copyright © 2026 Texas Instruments Incorporated

6.4 Clocking

This chapter describes the clock architecture of the device.

6.4.1 Overview

To satisfy the various subsystems requirements, the device features multiple clock sources and clock generators:

- External Crystal Driver
- Internal Oscillator
- Phase-Locked Loop circuits (PLLs)
- Dividers

Figure 6-28 shows the device top-level clock diagram. The clocking is divided into two clocking domains - MCU and MAIN.

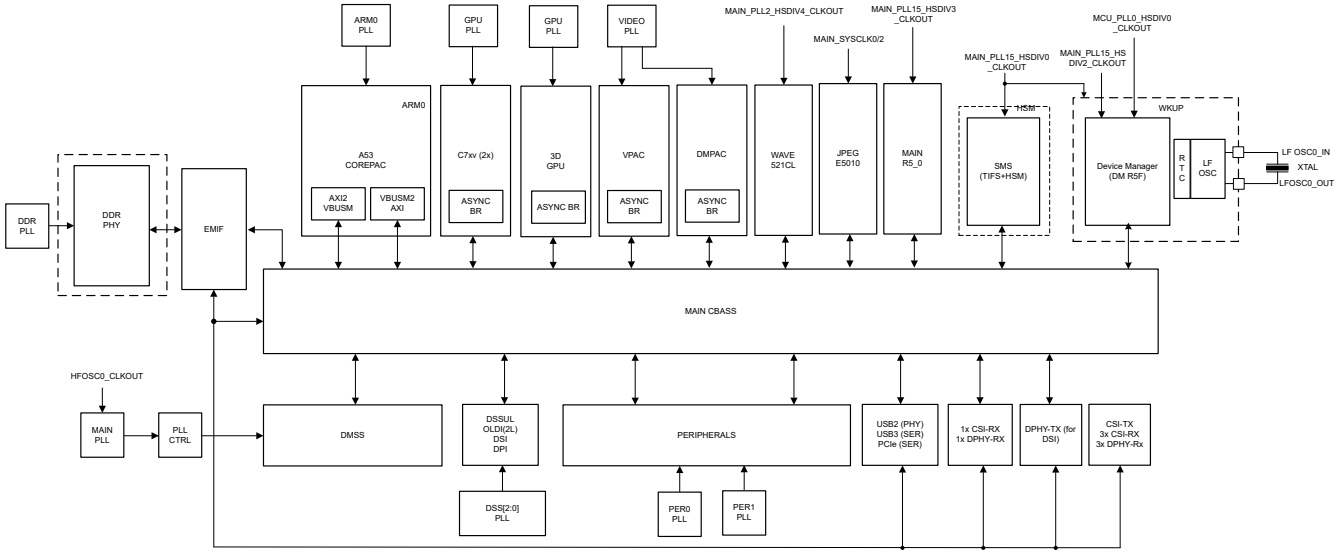


Figure 6-28. Top-Level Clock Diagram

Copyright © 2026 Texas Instruments Incorporated

6.4.2 Clock Inputs

6.4.2.1 Overview

Various external clock inputs are needed to drive the device. Summary of these input clock signals is as follows:

- High frequency oscillators
  - MCU_OSC0_XO/MCU_OSC0_XI — external main crystal interface pins of the internal oscillator which sources a reference clock. Provides reference clock to PLLs within MCU and MAIN domain.
- Low frequency oscillators
  - LSOSC0_IN/LFOSC0_OUT - external 32.768KHz crystal interface pins of the internal oscillator which sources a reference clock. Provides reference clock to RTC and Timers within MCU and MAIN domain.
- General purpose clock inputs
  - MCU_EXT_REFCLK0 — optional external system clock input (MCU domain).
  - EXT_REFCLK1— optional external system clock input (MAIN domain).
- External CPTS reference clock inputs
  - CP_GEMAC_CPTS0_RFT_CLK — CPTS reference clock input for CPSW and Timers in MAIN.
- External audio reference clock inputs/outputs
  - AUDIO_EXT_REFCLK[1:0] — optional McASP high-frequency input clocks when configured to operate as an input.

Note

Other clock inputs that are routed directly to the subsystems are described in the respective subsystem chapters.

Copyright © 2026 Texas Instruments Incorporated

6.4.3 Clock Outputs

The device provides several system clock outputs. Summary of these output clock signals is as follows:

- MCU_SYSCLKOUT0
- MCU_OBSCLK0
- SYSCLKOUT0
- OBSCLK0
- WKUP_CLKOUT0
- CLKOUT0
- AUDIO_EXT_REFCLK[1:0]
- AUDIO_EXT_REFCLK[2:0]

Other clock outputs, that are routed directly from subsystems to device pins, are described in the respective module chapter.

Observation clock pins - MCU_OBSCLK0 and OBSCLK0 serve the following purposes:

- During testing, PLLs on the device are configured to output all operating frequencies desired by each module to characterize PLL performance.
- System debug: during debug, clocks from various PLLs can be inspected for possible clues. Example clock glitches, lock loss etc.

Note

Maximum frequency supported on both MCU_OBSCLK0 and OBSCLK0 pins is 200 MHz. Hence the divider at the output of each OBSCLK mux must be programmed to meet this limitation.

Unlike the OBSCLK pins which can select several different clocks for output, system clock pins (MCU_SYSCLKOUT0 and SYSCLKOUT0) are hardwired to dedicated clock resources (see Section 6.4.3.2)

6.4.3.1 Observation Clock Pins

6.4.3.1.1 MCU_OBSCLK0 Pin

MCU_OBSCLK0 output is controlled by MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL register in the MCU_CTRL_MMR0 module; for more information about control registers, refer to *Control Module (CTRL_MMR)*. Two muxes are connected in series - MCU_OBSCLK0_MUX0 and MCU_OBSCLK0_MUX1, see Figure 6-29.

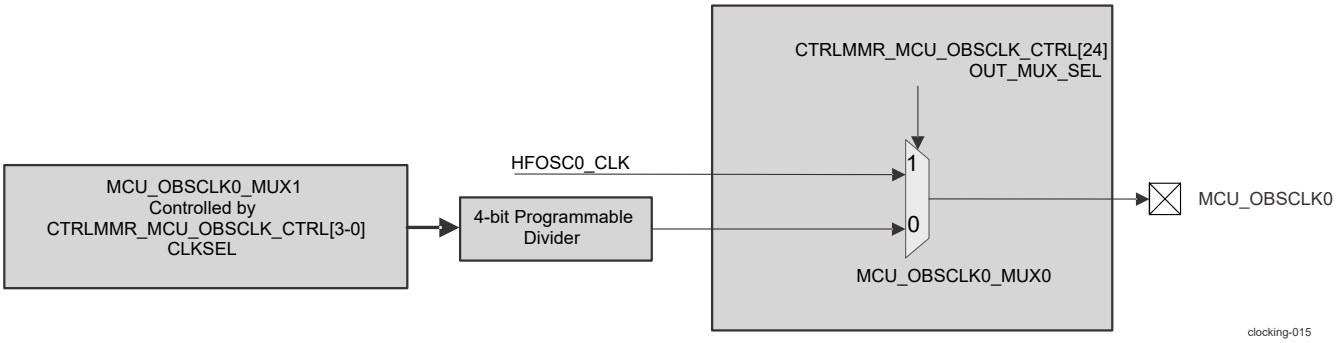


Figure 6-29. MCU_OBSCLK Muxes Diagram

How to select the output of MCU_OBSCLK0_MUX1 is described in Table 6-37.

Table 6-37. MCU_OBSCLK0_MUX1 Output Clock Selection

MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL[2] [3-0] CLK_SEL	MCU_OBSCLK0_MUX1 Output Clock Selection[1]
0x0	CLK_12M_RC
0x1	0 (GND)[3]
0x2	MCU_PLL0_HSDIV0_CLKOUT

Copyright © 2026 Texas Instruments Incorporated

Table 6-37. MCU_OBSCLK0_MUX1 Output Clock Selection (continued)

MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL(2)[3-0] CLK_SEL	MCU_OBSCLK0_MUX1 Output Clock Selection(1)
0x3	MCU_PLL0_HSDIV4_CLKOUT
0x4	MCU_PLLCTRL0_PLL_CTRL_OBSCLK_CLK
0x5	CLK_32K_RC
0x6	HFOSC0_CLKOUT
0x7	HFOSC0_CLKOUT_32K
0x8	MCU_SYSCLK0
0x9	DEVICE_CLKOUT_32K
0xA-0xF	RESERVED(3)

(1) Software-controlled 4-bit divider is used to obtain the divided versions of the clocks passed to MCU_OBSCLK1 mux
(2) For more information about control registers, refer to *Control Module (CTRL_MMR)*.
(3) GND - ground

The value of the software-controlled 4-bit divider is determined by the MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL[11-8] CLK _DIV field; for more information about control registers, see *Control Module (CTRL_MMR)*.

Note

MCU_OBSCLK1_MUX0 is provided as a low jitter output for HFOSC0_CLK. In this configuration, MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL[3-0] should be configured as 0x1 (1, selecting a logical low signal) and MCU_CTRL_MMR_CFG0_MCU_OBSCLK_CTRL[24] should be configured as 0x1.

6.4.3.1.2 OBSCLK0 Pin

The OBSCLK0 output pin is controlled by MAIN_CTRL_MMR_CFG0_OBSCLK0_CTRL register in the CTRL_MMR0 module; for more information about control registers, refer to *Control Module (CTRL_MMR)*. Figure 6-30 shows a block diagram of internal OBSCLK0 mux connections.

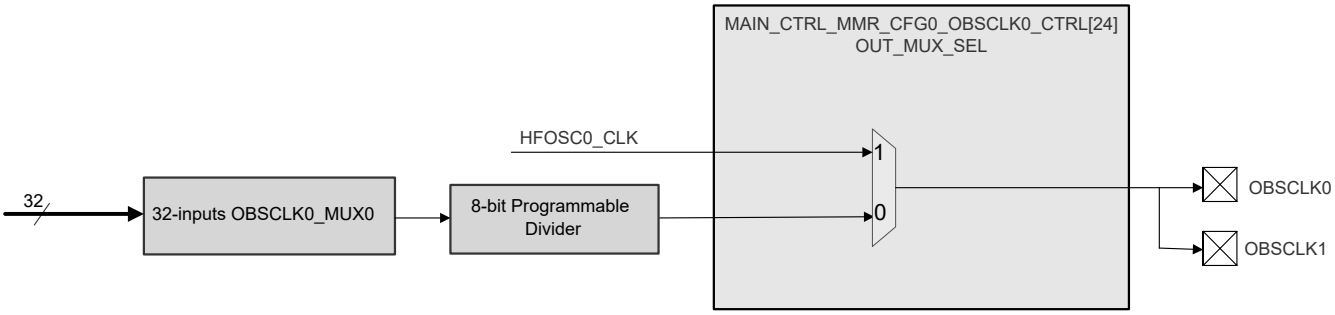


Figure 6-30. OBSCLK0 Muxes Diagram

Table 6-38. OBSCLK0 Clock Selection

MAIN_CTRL_MMR_CFG0_OBSCLK0_CTRL(2)[4-0] CLK_SEL	OBSCLK0 Selection (1)
0x0	MAIN_PLL0_HSDIV0_CLKOUT
0x1	MAIN_PLL1_HSDIV0_CLKOUT
0x2	MAIN_PLL2_HSDIV0_CLKOUT
0x3	MPUSS_CLK_DIV4
0x4	DDRSS_DDR_PLL_CLK_DIV4
0x5	CLK_12M_RC
0x6	HFOSC0_CLKOUT_32K

Copyright © 2026 Texas Instruments Incorporated

Table 6-38. OBSCLK0 Clock Selection (continued)

MAIN_CTRL_MMR_CFG0_OBSCLK0_CTRL(2)[4-0] CLK_SEL	OBSCLK0 Selection (1)
0x7	PLLCTRL0_PLL_CTRL_OBSCLK_CLK
0x8	HFOSC0_CLKOUT
0x9	CLK_32K_RC
0xA	CPSW0_CPTS_GENF0
0xB	CPSW0_CPTS_GENF1
0xC	MCU_PLL0_HSDIV0_CLKOUT
0xD	MAIN_PLL15_HSDIV0_CLKOUT
0xF	MAIN_PLL17_HSDIV0_CLKOUT
0x10	MAIN_SYSCLK0
0x11	DEVICE_CLKOUT_32K
0x16-0x1F	RESERVED(3)

(1) Software-controlled 8-bit divider is used to obtain the divided versions of the clocks passed to output pins
(2) For more information about control registers, refer to *Control Module (CTRL_MMR)*.
(3) GND - ground

The value of the software-controlled 8-bit divider is determined by register MAIN_CTRL_MMR_CFG0_OBSCLK0_CTRL[15-8] CLK_DIV; for more information about control registers, refer to *Control Module (CTRL_MMR)*.

6.4.3.2 System Clock Pins

6.4.3.2.1 MCU_SYSCLKOUT0

MCU_SYSCLK0 is divided by 4 and then send out of the device as a LVCMOS clock signal (MCU_SYSCLKOUT0). This signal can be used to test if the specific device clock is functioning or not.

Note

MCU_SYSCLKOUT0 cannot be used as a clock source for external devices on the board.

6.4.3.2.2 SYSCLKOUT0

SYSCLK0 is divided by 4 and then send out of the device as a LVCMOS clock signal (SYSCLKOUT0). This signal can be used to test if the specific device clock is functioning or not.

Note

SYSCLKOUT0 cannot be used as a clock source for external devices on the board.

6.4.3.2.3 WKUP_CLKOUT0

The WKUP_CLKOUT0 output pin is controlled by the WKUP_CTRL_MMR_CFG0_CLKOUT_CTRL register in the WKUP_CTRL_MMR0 module; for more information about control registers, refer Control Module (CTRL_MMR). Figure 6-31 shows a block diagram of internal WKUP_CLKOUT0 mux connections.

Copyright © 2026 Texas Instruments Incorporated

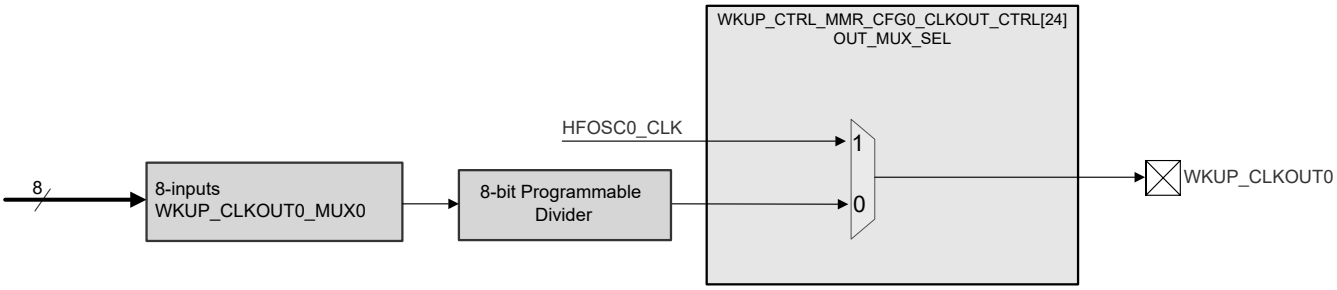


Figure 6-31. WKUP_CLKOUT0 Mux Diagram

Table 6-39. WKUP_CLKOUT0_MUX Output Clock Selection

WKUP_CTRL_MMR_CFG0_CLKOUT_CTRL(2)[2-0] WKUP_CLKOUT_SEL	WKUP_CLKOUT0_MUX Output Clock Selection(1)
0x0	0
0x1	LFOSC0_CLKOUT
0x2	MAIN_PLL0_HSDIV2_CLKOUT
0x3	MAIN_PLL1_HSDIV2_CLKOUT
0x4	MAIN_PLL2_HSDIV9_CLKOUT
0x5	DEVICE_CLKOUT_32K
0x6	CLK_12M_RC
0x7	HFOSC0_CLKOUT

(1) Software-controlled 8-bit divider is used to obtain the divided versions of the clocks passed to output pins.
(2) For more information about control registers, refer Control Module (CTRL_MMR).

6.4.4 Device Oscillators

The device has the possibility to source clocks of an external high-frequency oscillator - HFOSC0 and external 32.768KHz oscillator - LFOSC0. The device has also one internal RC oscillator - CLK_12M_RC.

6.4.4.1 Device Oscillators Integration

6.4.4.1.1 High-Frequency Oscillator with External Crystal

By default, HFOSC0 is enabled after PORz. It supports a bypass mode. In this mode, an external clock can be driven on the XI Pin.

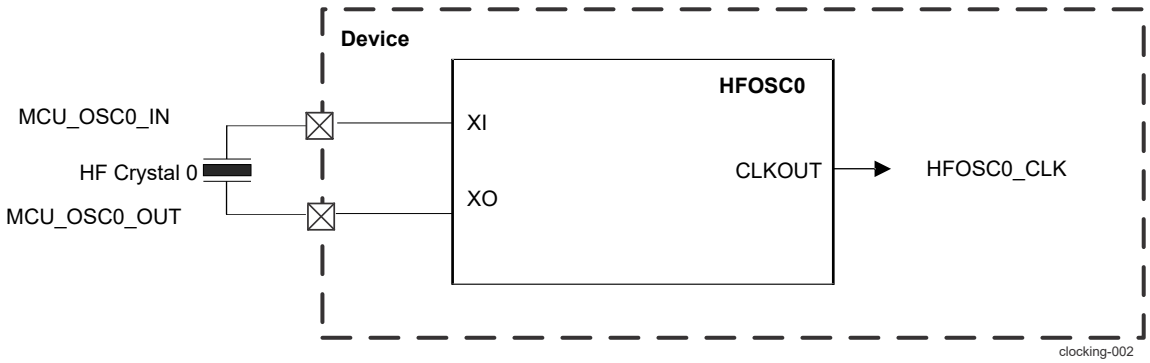


Figure 6-32. HFOSC0 Integration Diagram

6.4.4.1.2 Low-Frequency Oscillator with External Crystal

By default, LFOSC0 is disabled after PORz. It supports a bypass mode. In this mode, an external clock can be driven on the XI Pin.

Copyright © 2026 Texas Instruments Incorporated

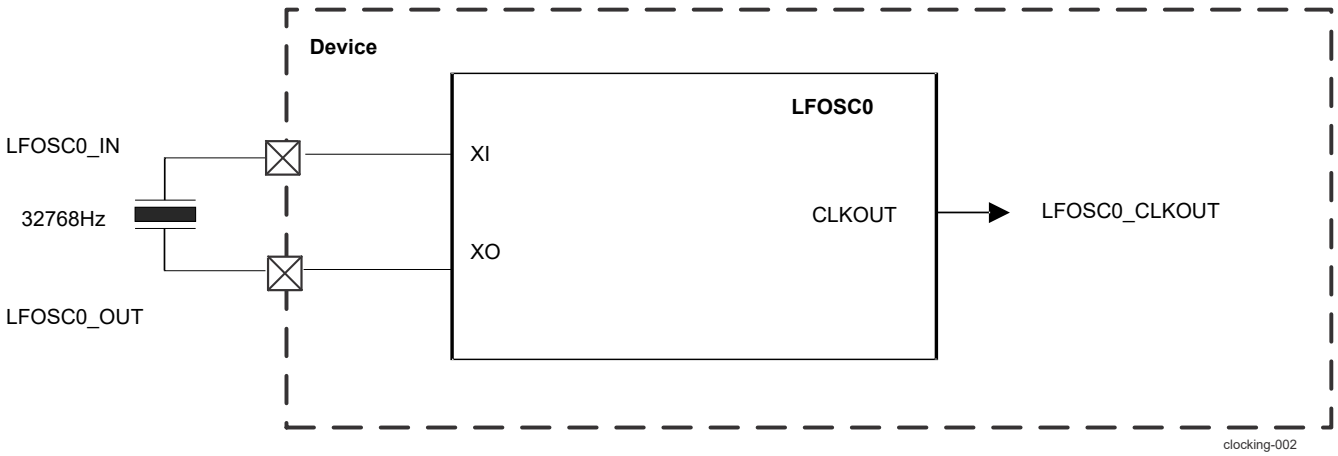


Figure 6-33. LFOSC0 Integration Diagram

6.4.4.1.3 Internal RC Oscillator

CLK_12M_RC in always oscillate mode.

The 12.5-MHz RC clock output (CLK_12M_RC) is used by the on-chip Reset Glue logic, Power Reset Generator (PRG) circuits, HFOSC0 Loss Circuits, Dual Clock Comparator (DCC) and Timer Modules.

Note

An output clock of CLK_12M_RC is further divided down to generate a 32-kHz clock (CLK_32K_RC). This clock is used by SMS and Timer Modules.

The actual value of the derived CLK_32K_RC will vary with PVT variations.

6.4.4.2 Oscillator Clock Loss Detection

The device supports HFOSC0 clock loss circuitry to detect when HFOSC0_CLK stops toggling. Dedicated hardware logic monitors HFOSC0 clock using CLK_12M_RC clock. When HFOSC0_CLK stops toggling for 9 CLK_12M_RC clock periods, a HFOSC0 clock stop loss condition is detected. If CTRLMMR_MCU_PLL_CLKSEL[8] CLKLOSS_SWTCH_EN is set , the reference clock is switched from HFOSC0_CLKOUT to CLK_12M_RC to allow the device to operate with a slower clock. The HFOSC0_CLK clock loss condition is reported as an error to MCU_ESM0 regardless of the value of MCU_CTRL_MMR_CFG0_MCU_PLL_CLKSEL[8] CLKLOSS_SWTCH_EN. Integration diagram of HFOSC0 clock loss detection is presented in Figure 6-34.

Copyright © 2026 Texas Instruments Incorporated

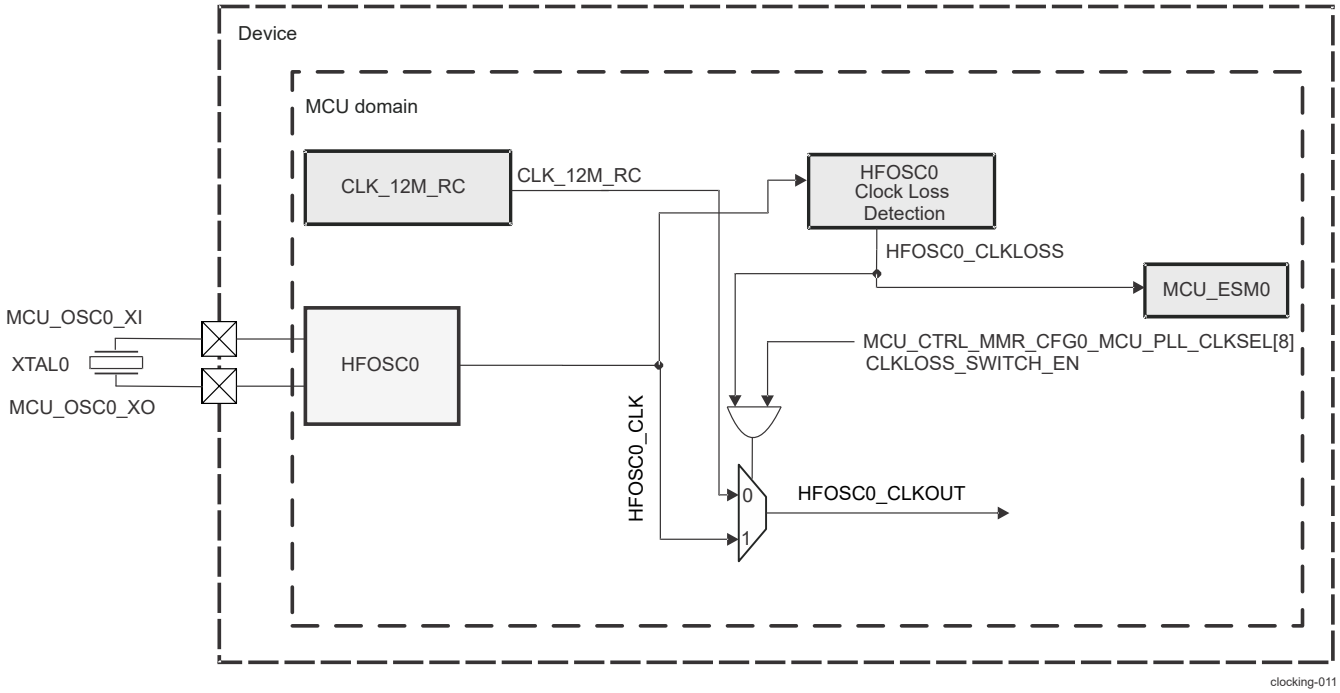


Figure 6-34. HFOSC0 Clock Loss Detection Integration Diagram

MCU_ESM0 can optionally generate an interrupt to MCU_ R5FSS and DM_R5FSS0 so they can save some critical contents such as error logging to scratch pad memory or external flash. ESM must also be configured to report this error on the MCU_ERRORn pin.

HFOSC0 clock loss is a catastrophic failure since this clock is the most critical system clock. During the clock loss condition an external system intervention is required.

The clock loss mux is not a glitch free mux. The mux control is synchronized to the CLK_12M_RC clock. Since HFOSC0_CLK has stopped switching the mux should transition to RC Clock without producing glitches.

During clock-loss condition, the device reports the error to the external device through MCU_ERRORn pin - the pin is driven Low. The recovery mechanism is up to the external system (such as a PMIC to take action). For example, it can try a full system power cycle to see if the system recovers. If the system does not recover then, it has to take some other action such as to check system clocks, external crystal or supply rails.

Device PLLs will lose lock when clock loss is detected. During this time 12.5-MHz RC clock is output through the bypass muxes. The PLL will relock to a new frequency based on 12.5 MHz RC clock.

In an event of clock glitch which may potentially hang the device, then the SMS watchdog timer will expire and will generate an internal reset for the whole device.

Copyright © 2026 Texas Instruments Incorporated

6.4.5 PLLs

Phase-Locked Loop circuits (PLLs) in the device are clock generator PLLs, which multiply the lower-frequency reference clock up to the operating frequency of the respective subsystem(s).

WARNING

PLL register descriptions are provided for debug purposes only. These registers should not be manipulated directly from software. PLL programming and configuration should only be performed using the appropriate APIs provided by TISCI services in TI's Processor Software Development Kits (SDKs). TISCI: Power Management (PM)

6.4.5.1 MCU Domain PLL Overview

MCU_PLL0 (MCU PLL) with MCU_PLLCTRL0 are present in the device in MCU domain. An overview is shown on Figure 6-35. For more specific information about PLLs see Section 6.4.5.5, *PLLs Device-Specific Information*.

Copyright © 2026 Texas Instruments Incorporated

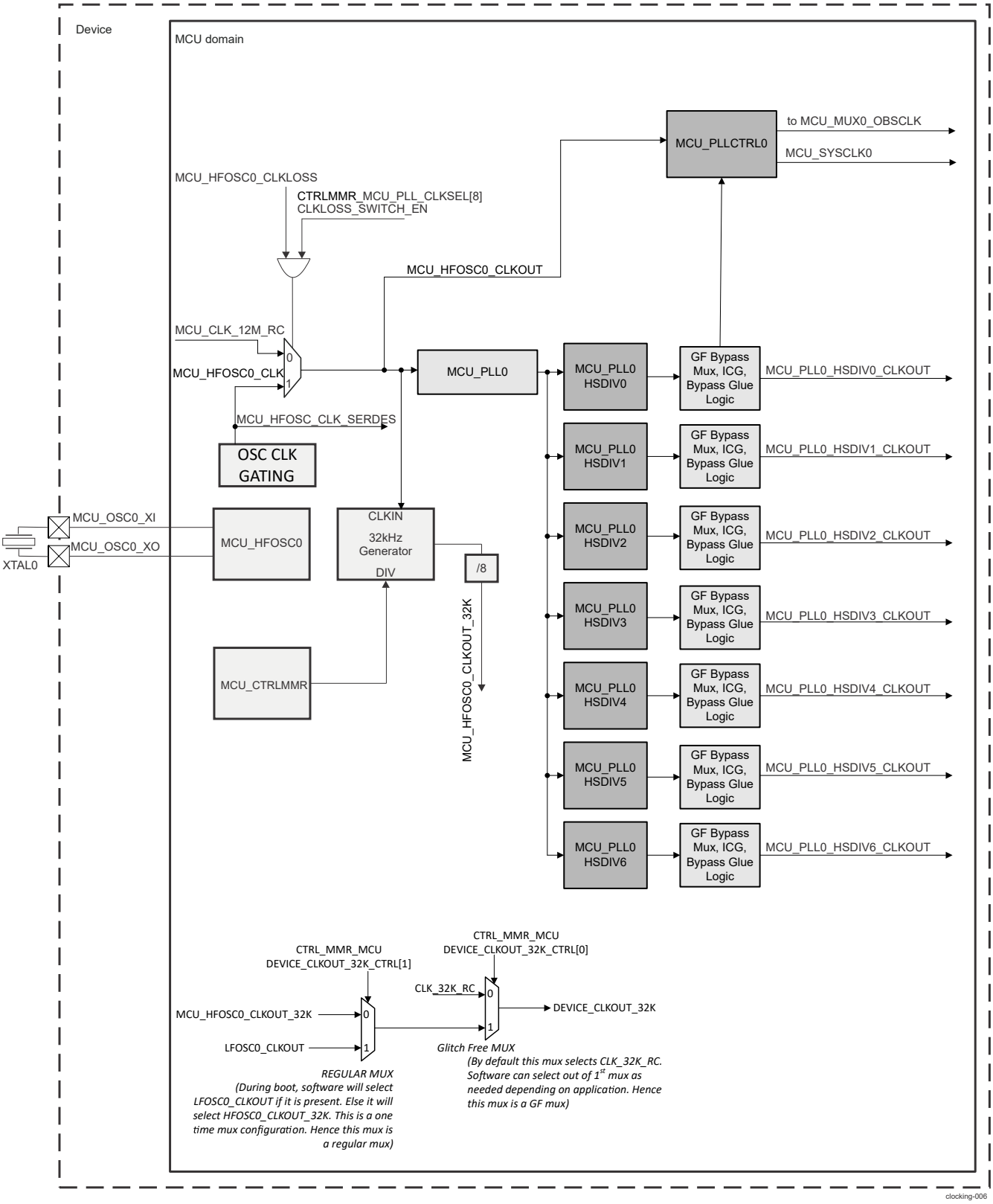


Figure 6-35. MCU Domain PLLs Integration

Copyright © 2026 Texas Instruments Incorporated

6.4.5.2 MAIN Domain PLLs Overview

The following are the PLLs in the device in MAIN domain:

- PLL0 (MAIN PLL) with PLLCTRL0
- PLL1 (PER0 PLL)
- PLL2 (PER1 PLL)
- PLL5 (VIDEO PLL)
- PLL6 (GPU PLL)
- PLL7 (C7x PLL)
- PLL8 (ARM0 PLL)
- PLL12 (DDR PLL)
- PLL15 (SMS PLL)
- PLL16 (DSS0 PLL)
- PLL17 (DSS1 PLL)
- PLL18 (DSS2 PLL)

Overview of the device PLLs with their reference clock options in MAIN domain is shown on Figure 6-36. For more specific information about PLLs see Section 6.4.5.5, *PLLs Device-Specific Information*.

Note

The external muxes of choosing the reference clocks are glitch-free muxes.

Copyright © 2026 Texas Instruments Incorporated

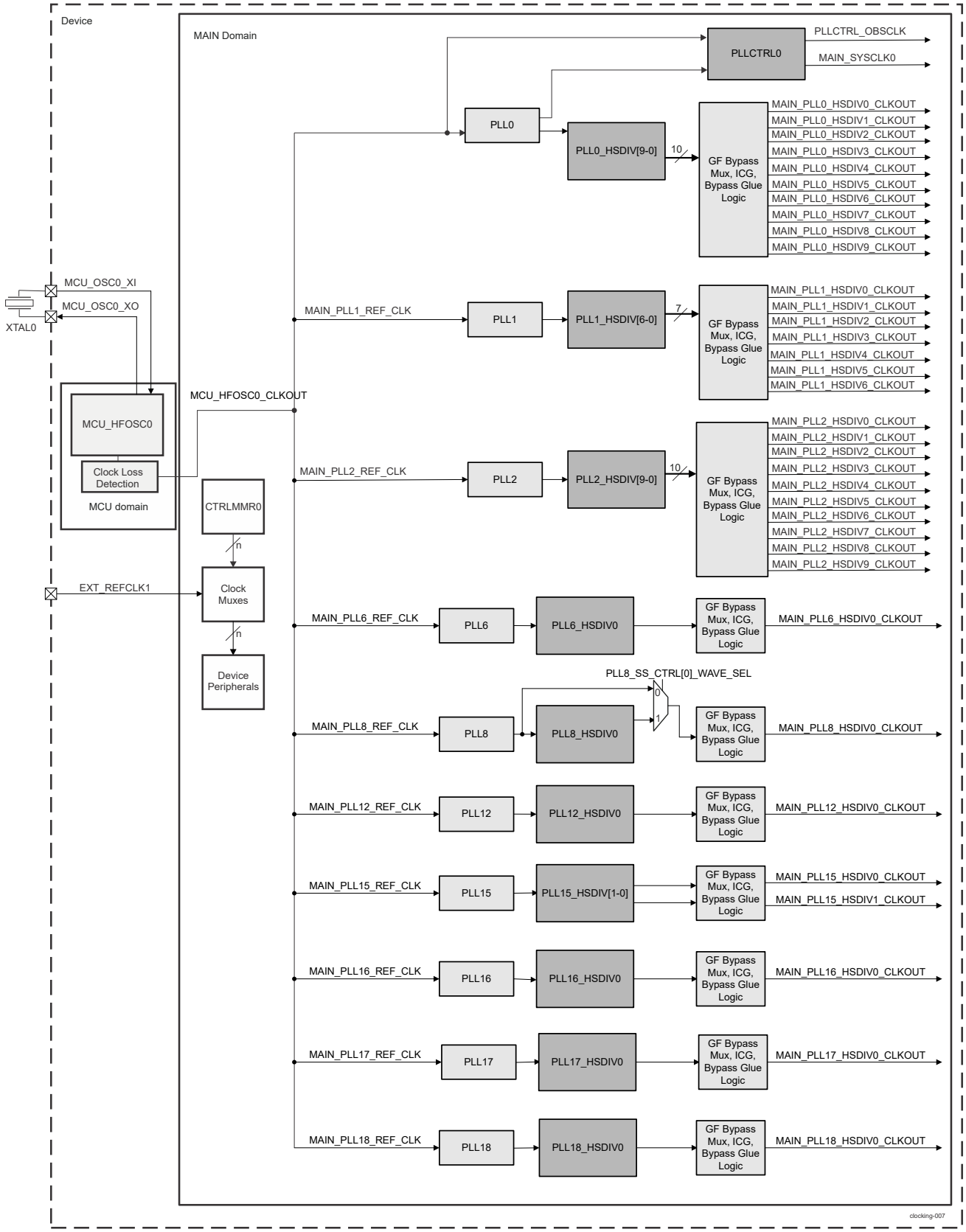


Figure 6-36. MAIN Domain PLLs Integration

Copyright © 2026 Texas Instruments Incorporated

6.4.5.3 PLL Reference Clocks

6.4.5.3.1 PLLs in MCU Domain

The reference clock MCU_PLL0_REF_CLK for the PLL in MCU domain is chosen between the internal high-frequency (HF) oscillator with external crystal (HFOSC0) and 12.5-MHz free-running RC oscillator. The selection is made through MCU_CTRL_MMR_CFG0_MCU_PLL_CLKSEL[8] CLKLOSS_SWTCH_EN, see Table 6-40.

Table 6-40. PLL MCU Domain Reference Clock Selection - HFOSC0_CLKOUT Selection

MCU_CTRL_MMR_CFG0_MCU_PLL_CLKSEL(1)[8] CLKLOSS_SWTCH_EN	MCU_PLL0_REF_CLK
0 (default)	HFOSC0_CLK
1	CLK_12M_RC if clock loss is detected or HFOSC0_CLK if clock loss is not detected

(1) For more information about control registers, see *Control Module (CTRL_MMR)*.

6.4.5.3.2 PLLs in MAIN Domain

Each PLL in MAIN domain has a dedicated register (**MAIN_CTRL_MMR_CFG0_MAIN_PLLn_CLKSEL**, n = 0 to 2, 8, 12,15, 16, 17) in CTRL_MMR0 that contains associated control bits.

6.4.5.4 Generic PLL Overview

To generate high-frequency clocks, the device supports multiple on-chip PLLs controlled directly by the Top-level Clocking. Their type is Fractional PLL with Calibration (PLLTS16FFCLAFRACF2).

Note

This chapter discusses only the PLLs that are directly controlled by the Top-level Clocking. The other PLLs embedded in and managed by other subsystems are described in their respective subsystems.

6.4.5.4.1 PLLs Output Clocks Parameters

Figure 6-37 shows the functional architecture of a generic PLL for PLLTS16FFCLAFRACF2 type.

Copyright © 2026 Texas Instruments Incorporated

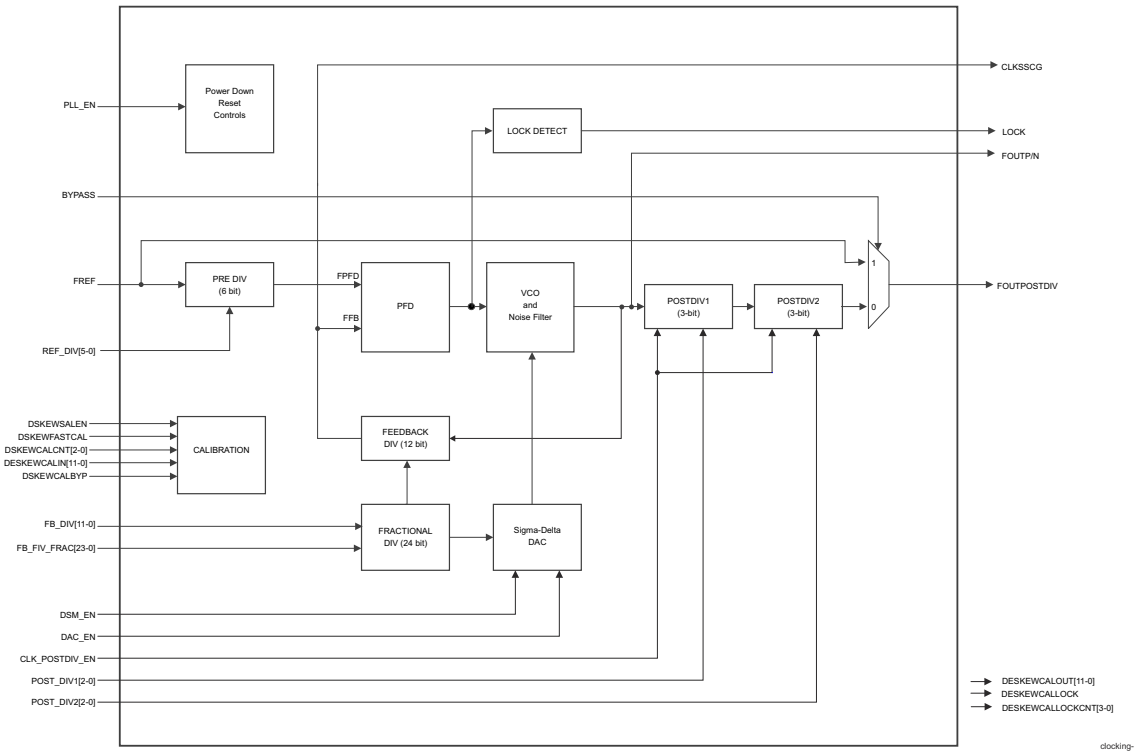


Figure 6-37. Generic PLL Functional Diagram for PLLTS16FFCLAFRACF2 Type

6.4.5.4.1.1 PLLs Input Clocks

As shown in Figure 6-37 the reference clock (FREF) input is used to generate the synthesized clock, but can also be used as the bypass clock for some outputs of the PLL whenever the PLL enters bypass mode. It is mandatory for the PLL clock synthesis.

6.4.5.4.1.2 PLL Output Clocks

6.4.5.4.1.2.1 PLLTS16FFCLAFRACF2 Type Output Clocks

Table 6-41 describes the output clocks of PLLTS16FFCLAFRACF2.

Table 6-41. PLLTS16FFCLAFRACF2 Output Clocks

Output	Description	Frequency
FOUTP	Positive phase VCO output (no post divider)	(FREF / REF_DIV) * (FB_DIV + FB_DIV_FRAC)
FOUTN	Negative phase VCO output (no post divider)	(FREF / REF_DIV) * (FB_DIV + FB_DIV_FRAC)
FOUTPOSTDIV	VCO-divided clock output.	FOUTP / (POST_DIV1*POST_DIV2)

Where:

- REF_DIV is the software-configured division ratio binary value.
- FB_DIV is the software-configured division ratio binary value.
- FB_DIV_FRAC is the software-configured fractional division.
- POST_DIV1 is the software-configured division ratio binary value.
- POST_DIV2 is the software-configured division ratio binary value.

Copyright © 2026 Texas Instruments Incorporated

Note

POST_DIV1 and POST_DIV2 valid values are from 1 to 7. To ensure correct operation, POST_DIV1 must always be programmed to a value equal to or greater than POST_DIV2.

Note

For device-specific information about clock output parameters and synthesized clocks, see Table 6-46 and Table 6-47.

6.4.5.4.1.2.2 PLL Lock

PLL module outputs a lock status signal to indicate that the PLL has achieved frequency lock. When the PLL detects no cycle slips between the feedback clock (FFB) and reference clock FPFD (FREF/ REF_DIV[5-0]) for 128 consecutive cycles, it asserts the lock signal high. When the PLL detects any cycle slip, lock signal will go low and stays low until it detects no cycle slip for 128 consecutive cycles. This lock signal is captured in <PLL_name>_STAT[0] LOCK bit. Software can read this bit to determine if the PLL has achieved frequency lock before selecting the PLL clock through external bypass mux.

When PLL losses lock, there is a hardware mechanism to automatically bypass the PLL clock to the reference clock (FREF) using the external glitch free mux. BYP_ON_LOCKLOSS bit in <PLL_name>_CTRL register enables this automatic bypass mode on PLL lock loss. When the PLL re-locks, the glitch free mux switches to PLL clock out.

The PLL Lock signal is also routed to ESM module for error reporting. ESM can be configured to generate interrupts to MCU_R5FSS and R5FSS0/1 and assert Low on MCU_ERRORn pin on PLL Lock loss for further action.

6.4.5.4.1.2.3 HSDIVIDER

A PLL may contain up to 10 HSDIVIDER modules to produce more clocks with divided ratio based on the PLL synthesized clock frequency. HSDIVIDER provides only one output. The HSDIVIDER output clock frequency is given by the equations in Table 6-42.

Table 6-42. HSDIV_CLKOUT Frequency With PLL State

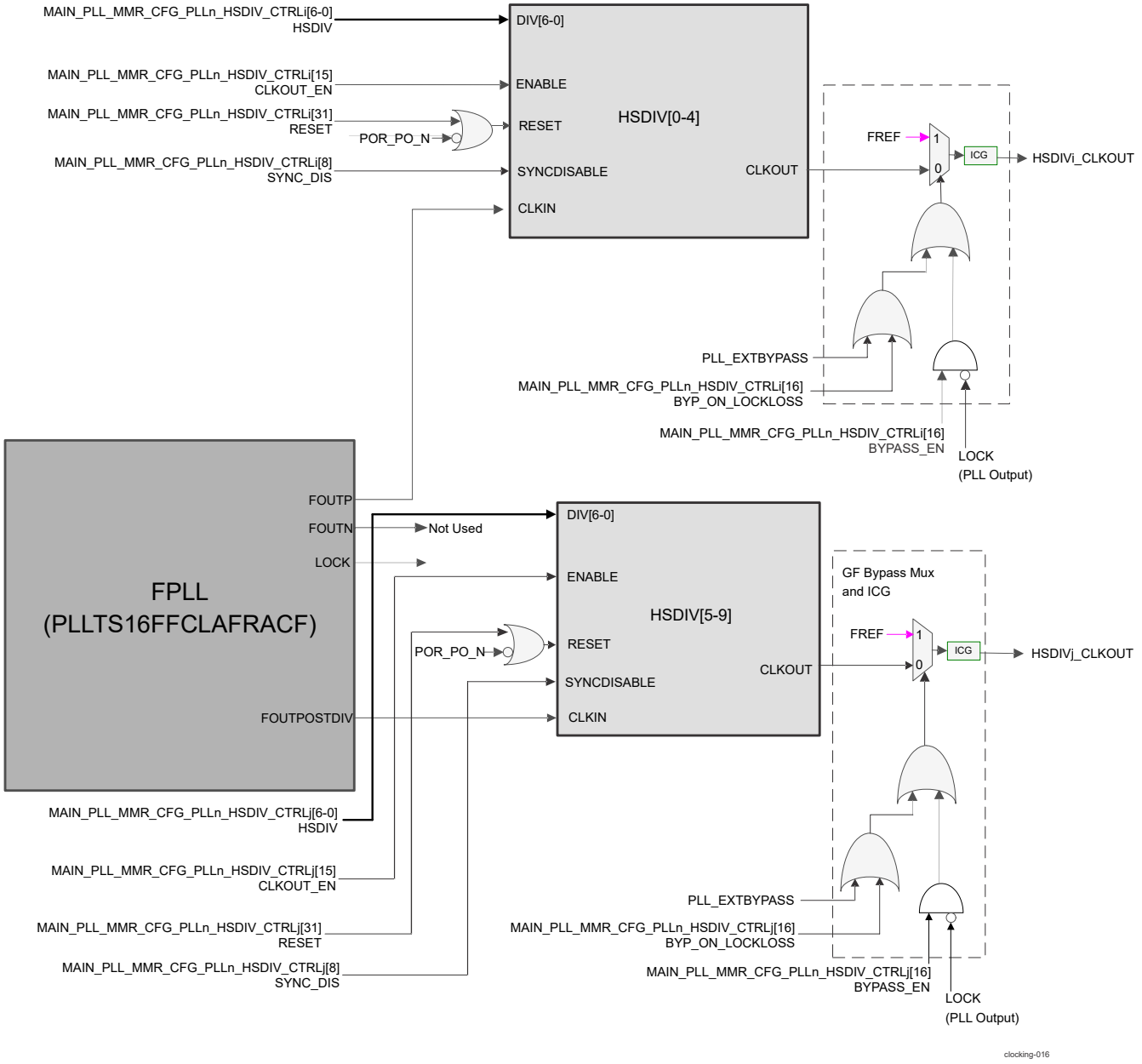
Equation	PLL Mode
HSDIV_CLKOUT = FOUTP / (HSDIV + 1)	Locked
HSDIV_CLKOUT = FREF	Before lock or during relock

Where:

- FOUTP is the PLL lock frequency.
- HSDIV is the software-configured division ratio binary value.

Figure 6-38 describes the connection between a HSDIV and a specific type of PLL. Signals related only to this connections are shown in both figures.

Copyright © 2026 Texas Instruments Incorporated



(1) i = 0 to 4; j = 5 to 9

Figure 6-38. PLLTS16FFCLAFRACF2 and HSDIV Connection

6.4.5.4.1.2.4 ICG Module

ICG module ensures that when a particular HSDIV is not enabled the HSDIV clock output is gated off. This is a hardware module with no software control.

6.4.5.4.1.2.5 PLL Power Down

PLLTS16FFCLAFRACF2 is disabled with MCU_PLL_MMR_CFG_PLL0_CTRL[15] PLL_EN or MAIN_PLL_MMR_CFG_PLLn_CTRL[15] PLL_EN.

6.4.5.4.1.2.6 PLL Calibration

PLLTS16FFCLAFRACF2 has calibration feature. The calibration settings are done via <PLL_Name>_CAL_CTRL register. The calibration status can be monitored by reading

Copyright © 2026 Texas Instruments Incorporated

<PLL_Name>_CAL_STAT register. For more information about device-specific information about calibration feature, see Table 6-48.

6.4.5.4.2 PLL Spread Spectrum Modulation Module

Spread Spectrum Modulation (SSMOD) modules on the device reduces Electro Magnetic Interference (EMI). It is a fully-digital circuit, used to modulate the frequency of the selected fractional PLLs. Programming options include selection of center spread or down spread, modulation depth, and modulation shape. The default modulation profile is triangular.

See *Spread Spectrum PLLs* for a list of PLLs that support Spread Spectrum.

SSMOD in the PLL is performed by changing the feedback divider (FRAC and FB_DIV) in a triangular pattern. This varies the frequency of the output clock in a triangular pattern. The frequency of this pattern is the modulation frequency (f_m). It is programmed as a ratio of the CLKIN/4.

Figure 6-39 shows a diagram of the spreading generation block.

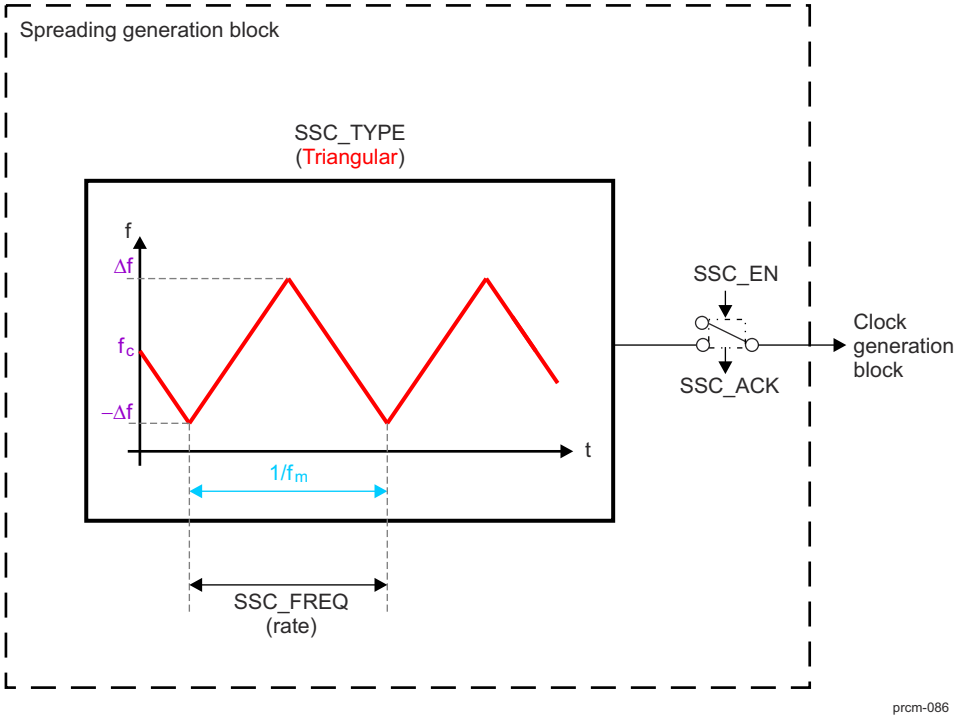


Figure 6-39. Spreading Generation Block Diagram

Note

Δf is the deviation from the center frequency. The total spreading deviation is equal to twice Δf. The peak (ΔM) or the amplitude of the triangular/square pattern as a percent of M would be equal to the percent of the frequency spread (Δf), $\Delta M/M = \Delta f/f_c$.

f_c is the original output clock frequency.

f_m is the spreading frequency.

This additional block generates the required waveform used to reduce EMI. This waveform is then modulated with the initial signal to add some controlled deviation to the clock signal frequency, which spreads the energy of the clock and its harmonics into a band of frequencies, and then reduces EMI. SSMOD_DOWNSPREAD can control the position of the generated signal. It is controlled by the <PLL_name>_CTRL[4] DOWNSPREAD_EN

Copyright © 2026 Texas Instruments Incorporated

bit of the corresponding registers. If the DOWNSPREAD_EN bit is set to 1, the frequency spread on the lower side is twice the programmed value. The frequency spread on the higher side is 0.

The value of SSMOD_FREQ controls the rate of the generated signal. It can be programmed as a ratio of the reference clock: f_{ref} / 4. The value that must be programmed is calculated as follows:
ModFreqDivider = f_{ref} / (4 × f_m), where $f_m < f_{ref} / 70$, $f_{ref} = f_{inp} / (1+N)$, and f_{inp} is the input clock for the PLL.

6.4.5.4.2.1 Definition of SSMOD

The aim of the SSMOD is to add a variation to the frequency of an original clock, which spreads the generated interference over a larger band of frequencies.
In theory, SSMOD means that the clock signal is varied around the desired frequency. For example, for a 1 GHz clock, the frequency might be 999.5 MHz at one time and 1.0005 GHz at another. Doing this constantly causes the power of the tone to be spread out more over a broader band of tight frequencies (centered at the desired tone). To realize this constant variation on the original signal, a modulation with an additional signal (called spreading waveform) is realized.
Creating an SSMOD by spreading the initial clock frequency is done by defining the following parameters:

- The spreading frequency (deviation), which is the ratio of the range of spreading frequency over the original clock frequency.

6.4.5.4.2.2 SSMOD Configuration

Table 6-43 describes the PLL SSMOD control bitfields. Figure 6-40 describes the connection between a SSMOD and a PLL.

Table 6-43. SSMOD related bitfields

Parameter	Register	Description
SSMOD enable control	<PLL_name>n_SS_CTRL[31](1) BYPASS_EN (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[31] BYPASS_EN)	Enable/disable the PLL SSMOD feature.
Wave table maximum address	<PLL_name>n_SS_CTRL[25-18](1) WV_TBL_MAXADDR (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[25-18] WV_TBL_MAXADDR)	Wave table maximum address. Indicates the maximum number of address bits used to access the externalwave table. These bits are not used if <PLL_name>n_SS_CTRL[0] WAVE_SEL = 0
Type of spread	<PLL_name>n_SS_CTRL[4](1) DOWNSPREAD_EN (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[4] DOWNSPREAD_EN)	Selects center spread or down spread clock variance
Wave table selection	<PLL_name>n_SS_CTRL[0] (1) WAVE_SEL (For example, MCU_PLL0 - MCU_PLL0_SS_CTRL[0] WAVE_SEL)	Wave pattern select External wave table should only be selected
Modulation dvider	<PLL_name>n_SS_SPREAD[19-16] (1) MOD_DIV (For example, MCU_PLL0 - MCU_PLL0_SS_SPREAD[19-16] MOD_DIV)	Input clock divider. This divider sets the modulation frequency.
Spread Depth	<PLL_name>n_SS_SPREAD[4-0] (1) SPREAD (For example, MCU_PLL0 - MCU_PLL0_SS_SPREAD[4-0] SPREAD)	Sets the spread modulation depth.

(1)

Copyright © 2026 Texas Instruments Incorporated

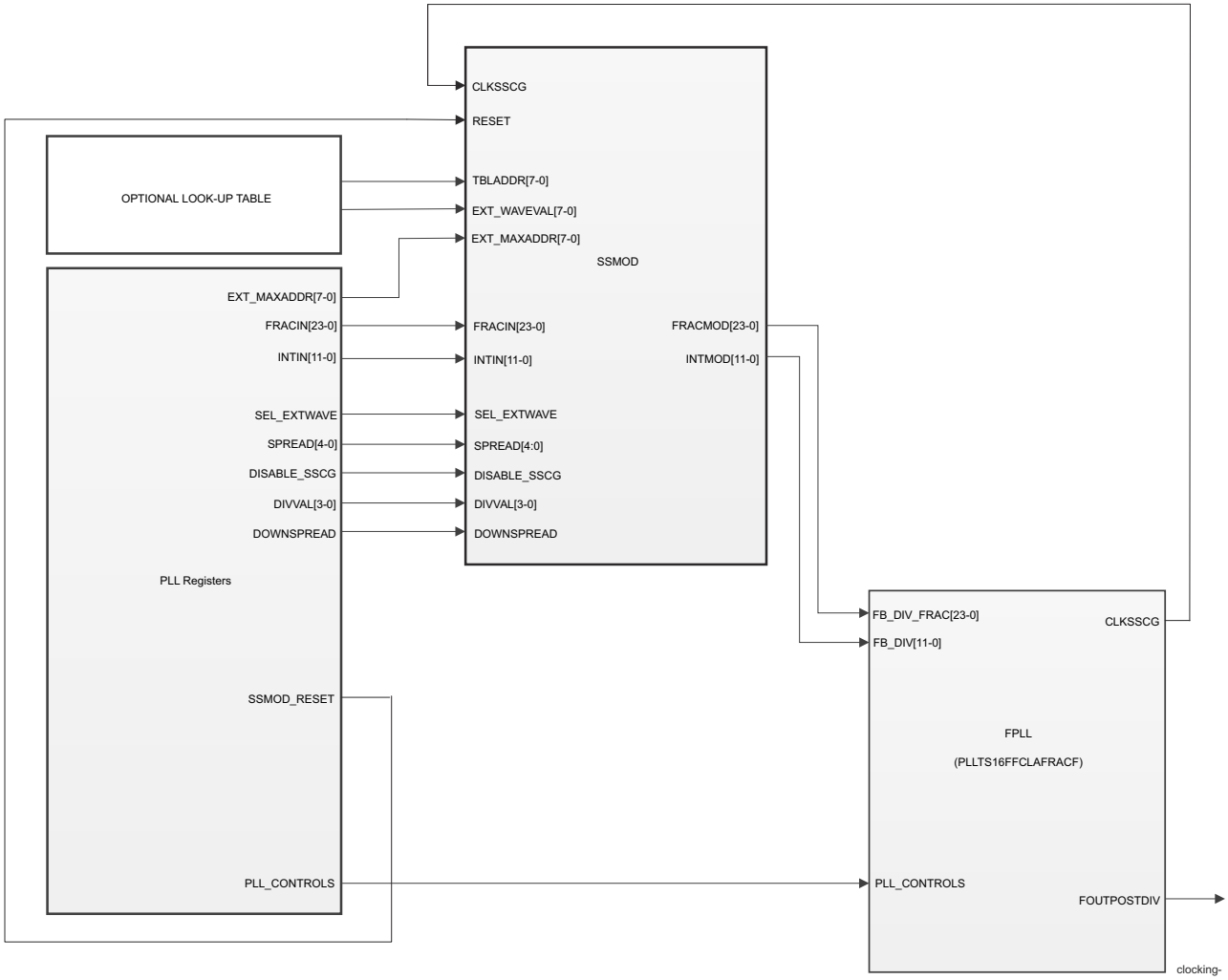


Figure 6-40. PLL and SSMOD Connection

Copyright © 2026 Texas Instruments Incorporated

6.4.5.5 PLLs Device-Specific Information

Table 6-44 lists the basic information of the MCU PLL.

Table 6-44. Basic Information of MCU_PLL0

PLL name	Type	HSDIV Available	Input clock	Comments
MCU_PLL0	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4	HFOSC0_CLKOUT	See (1), (2), and (3)

(1) See Figure 6-37, *Generic PLL Functional Diagram for PLLTS16FFCLAFRACF2 type* for overview of the PLL.
(2) See Section 6.4.5.5.2 for synthesized clock parameters.
(3) See Section 6.4.5.5.3 for clock output parameters.

Table 6-45 lists the basic information of each of the MAIN PLLs.

Table 6-45. Basic Information of MAIN Domain PLLs

PLL name	Type	HSDIV Available	Input clock	Comments
PLL0	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4, HSDIV5, HSDIV6, HSDIV7, HSDIV8, HSDIV9	MAIN_PLL0_REFCLK	See (1), (2), and (3)
PLL1	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4, HSDIV5, HSDIV6	MAIN_PLL1_REFCLK	See (1), (2), and (3)
PLL2	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1, HSDIV2, HSDIV3, HSDIV4, HSDIV5, HSDIV6, HSDIV7, HSDIV8, HSDIV9	MAIN_PLL2_REFCLK	See (1), (2), and (3)
PLL5	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1, HSDIV2	MAIN_PLL5_REFCLK	(1), (2), and (3)
PLL6	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL6_REFCLK	(1), (2), and (3)
PLL7	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1	MAIN_PLL7_REFCLK	(1), (2), and (3)
PLL8	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL8_REFCLK	See (1), (2), and (3)
PLL12	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL12_REFCLK	See (1), (2), and (3)
PLL15	PLLTS16FFCLAFRACF2	HSDIV0, HSDIV1	MAIN_PLL15_REFCLK	See (1), (2), and (3)
PLL16	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL16_REFCLK	See (1), (2), and (3)
PLL17	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL17_REFCLK	See (1), (2), and (3)
PLL18	PLLTS16FFCLAFRACF2	HSDIV0	MAIN_PLL18_REFCLK	See (1), (2), and (3)

(1) See Figure 6-37, *Generic PLL Functional Diagram for PLLTS16FFCLAFRACF2 type* for overview of the PLL.
(2) See Section 6.4.5.5.2 for synthesized clock parameters.
(3) See Section 6.4.5.5.3 for clock output parameters.

Copyright © 2026 Texas Instruments Incorporated

6.4.5.5.1 SSMOD Related Bitfields Table

Table 6-43 lists SSMOD related bitfields for all types of PLLs.

Copyright © 2026 Texas Instruments Incorporated

6.4.5.5.2 Clock Synthesis Inputs to the PLLs

Table 6-46 lists the clock synthesis parameters for PLLTS16FFCLAFRACF2 type.

Table 6-46. Clock synthesis Parameters for PLLTS16FFCLAFRACF2 Type

Parameter Name	Register
FB_DIV	<PLL_name>_FREQ_CTRL0[11-0] FB_DIV_INT (For example, MCU_PLL0 - MCU_PLL0_FREQ_CTRL0[11-0] FB_DIV_INT)
FB_DIV_FRAC	<PLL_name>_FREQ_CTRL1[23-0] FB_DIV_FRAC (For example, MCU_PLL0 - MCU_PLL0_FREQ_CTRL1[23-0] FB_DIV_FRAC)
POST_DIV2	<PLL_name>_DIV_CTRL[26-24] POST_DIV2 (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[26-24] POST_DIV2)
POST_DIV1	<PLL_name>_DIV_CTRL[18-16] POST_DIV1 (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[18-16] POST_DIV1)
REF_DIV	<PLL_name>_DIV_CTRL[5-0] REF_DIV (For example, MCU_PLL0 - MCU_PLL0_DIV_CTRL[5-0] REF_DIV)

Note

For PLLTS16FFCLAFRACF2 type - POST_DIV1 and POST_DIV2 values are from 1 to 7. To ensure correct operation, POST_DIV1 must always be programmed to a value equal to or greater that POST_DIV2.

6.4.5.5.3 Clock Output Parameter

Table 6-47 lists the control/status bit fields for output clocks.

Table 6-47. Clock Output Parameter for PLLTS16FFCLAFRACF2 Type

Clock Output/Divider	Parameter Name	Control/Status Bit Field
<PLL_name>_CLKOUT, <PLL_name>_HSDIVj_CLKOUT(1)	Control	<PLL_name>_CTRL[31] BYPASS_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[31] BYPASS_EN)
<PLL_name>_HSDIVj_CLKOUT(1)	Control	<PLL_name>_HSDIV_CTRLj[15] CLKOUT_EN (For example, MCU_PLL0 - MCU_PLL0_HSDIV_CTRL0[15] CLKOUT_EN and MCU_PLL0_HSDIV_CTRL1[15] CLKOUT_EN)
<PLL_name>_HSDIVj_CLKOUT(1)	Divider control	<PLL_name>_HSDIV_CTRLj[6-0] HSDIV (For example, MCU_PLL0 - MCU_PLL0_HSDIV_CTRL0[6-0] HSDIV and MCU_PLL0_HSDIV_CTRL1[6-0] HSDIV)
PLL bypass mux	Control	<PLL_name>_CTRL[16] BYP_ON_LOCKLOSS (For example, MCU_PLL0 - MCU_PLL0_CTRL[16] BYP_ON_LOCKLOSS)
Enable 4-phase clock generator	Control	<PLL_name>_CTRL[5] CLK_4PH_EN (ignored if <PLL_name>_CTRL[4] CLK_POSTDIV_EN = 0) (For example, MCU_PLL0 - MCU_PLL0_CTRL[5] CLK_4PH_EN)
Post divide CLK enable	Control	<PLL_name>_CTRL[4] CLK_POSTDIV_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[4] CLK_POSTDIV_EN)
Delta-Sigma modulator enable	Control	<PLL_name>_CTRL[1] DSM_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[1] DSM_EN)

Copyright © 2026 Texas Instruments Incorporated

Table 6-47. Clock Output Parameter for PLLTS16FFCLAFRACF2 Type (continued)

Clock Output/Divider	Parameter Name	Control/Status Bit Field
Enable fractional noise canceling DAC	Control	<PLL_name>_CTRL[0] DAC_EN (For example, MCU_PLL0 - MCU_PLL0_CTRL[0] DAC_EN)

(1) j is the number of HSDIV for the current PLL.

6.4.5.5.4 Calibration Related Bitfields

Table 6-48 lists the calibration related bitfields/bits.

Table 6-48. Recalibration Feature Parameters

Recalibration feture	Parameter Name	Control/Status Bit Field
Calibration enable to actively adjust for input skew	Control	<PLL_name>_CAL_CTRL[31] CAL_EN (For example, PLL12 - PLL12_CAL_CTRL[31] CAL_EN)
Fast calibration enabled	Control	<PLL_name>_CAL_CTRL[20] FAST_CAL (For example, PLL12 - PLL12_CAL_CTRL[20] FAST_CAL)
Calibration loop programmable counter	Control	<PLL_name>_CAL_CTRL[18-16] CAL_CNT (For example, PLL12 - PLL12_CAL_CTRL[18-16] CAL_CNT)
Calibration bypass	Control	<PLL_name>_CAL_CTRL[15] CAL_BYP (For example, PLL12 - PLL12_CAL_CTRL[15] CAL_BYP)
Calibration input	Control	<PLL_name>_CAL_CTRL[11-0] CAL_IN (For example, PLL12 - PLL12_CAL_CTRL[11-0] CAL_IN)
Output of the calibration block	Status	<PLL_name>_CAL_STAT[11-0] CAL_OUT (For example, PLL12 - PLL12_CAL_STAT[11-0] CAL_OUT)

6.4.5.6 PLL Controller (PLLCTRL)

The PLLCTRL manages clock alignment of CBASS clocks to the peripherals in Main and MCU domains. In addition PLLCTRL is also responsible for controlling reset propagation through the chip and providing DFT hooks for testability. Two PLL Controllers are implemented in the device - PLLCTRL0 and MCU_PLLCTRL0. They are respectively related to PLL0 and MCU_PLL0.

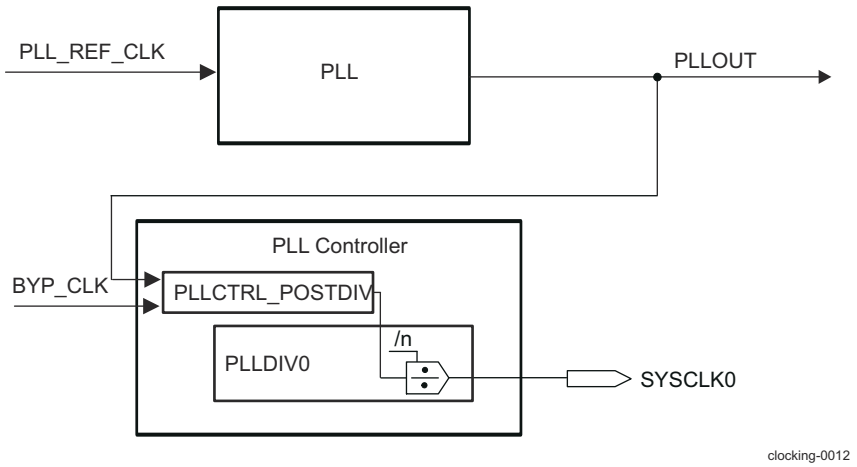


Figure 6-41. PLL Controller

Copyright © 2026 Texas Instruments Incorporated

Note

PLLCTRL_POSTDIV is not supported in this family of devices.

Note

The PLL Controllers registers can be accessed by any controller in the device.

A SYSCLK0 clock out of a PLLCTRL is the only synchronous clock in that domain. That means
- MCU_SYSCLK0 out of MCU_PLLCTRL is the synchronous CBASS clock in MCU domain and
SYSCLK0 out of Main PLLCTRL is the synchronous CBASS clock in Main domain.

6.4.5.7 PLL, PLLCTRL, and HSDIV Controllers Programming Guide

WARNING

PLL register descriptions are provided for debug purposes only. These registers should not be manipulated directly from software. PLL programming and configuration should only be performed using the appropriate APIs provided by TISCI services in TI's Processor Software Development Kits (SDKs). TISCI: Power Management (PM)

6.4.5.7.1 PLL Initialization

6.4.5.7.1.1 Kick Protection Mechanism

Note

PLL configuration registers are write-protected at power-up. Software must first un-lock the PLLn_LOCKKEY0 and PLLn_LOCKKEY1 registers prior to writing to any chip-level registers.

The un-lock process shall follow the steps:

1. Write value 0x68EF3490 to PLLn_LOCKKEY0 register.
2. Write value 0xD172BC5A to PLLn_LOCKKEY1 register.

After these two steps a write access to the PLL registers is allowed. Writing any other data value to either of these two registers locks the kicker mechanism and blocks any writes to the PLL registers.

Note

In order to ensure that all PLL registers are write protected, software must always re-lock the kicker mechanism after completing the register writes.

6.4.5.7.1.2 PLLCTRL Initialization

Note

For validated set of parameters that can be programmed to PLLs, please refer to the device-specific Datasheet.

Initially, the device powers up in PLL bypass mode. The bypass mux is a glitch free mux and is located outside the PLL module. BYPASS_EN bit controls the bypass mux section. During Power-up, the bypass mux defaults to select FREF clock (PLL clock bypass mode).

The following shows the PLL initialization sequence.

1. Wait until supplies are stable and PORz is de-asserted. By this time the reference clock source must be up and running.
2. Default controls settings:

Copyright © 2026 Texas Instruments Incorporated

- By default PLL_EN signal is Low provided by <PLL_name>_CTRL register default state. This signal behaves like a reset control to the PLL.
- By default INTL_BYP_EN signal is low provided by <PLL_name>_CTRL register default state.
- By default BYPASS_EN signal is high provided by PLL MMR CTRL default state. PLL clock will be bypassed externally by a glitch free mux. The output of the mux will be the reference clock FREF.
- LOCK output of PLL will be Low.

3. Program the PLL to a valid setting that runs the VCO within the specified range. The following bits/bitfields must be programmed appropriately by software: DSM_EN, DAC_EN, CLK_POSTDIV_EN, CLK_4PH_EN, REF_DIV[5-0], FB_DIV[11-0], FB_DIV_FRAC[23-0], POST_DIV1[2-0], POST_DIV2[2-0].
4. Wait for 1 µs to allow the PLL internal reset to complete (PLL powerdown switch pulls the loop filter voltage from rail-to-rail, which ensures the PLL will be completely powered down from any state).
5. Assert PLL_EN to a High (by writing a '1' into PLL_EN bit in <PLL_name>_CTRL register).
6. Wait for PLL Lock output to go high. Software can read the LOCK bit in <PLL_name>_STATS register to check if PLL has locked.
7. De-assert BYPASS_EN signal to a Low by writing 0 into BYPASS_EN bit in <PLL_name>_CTRL register.
8. Glitch free mux safely switches to the PLL clock output.

Note

In the case of brown out (supply dips below datasheet minimum) or loss of the reference clock, the state of the PLL will be undetermined. A full reset sequence should be initiated.

Changing PLL setting: During normal operation, before changing PLL settings, the PLL clock must be bypassed by setting BYPASS_EN control to a high. Then follow steps 2 to 8 from the above sequence.

6.4.5.7.1.3 PLL Programming Requirements

- A PLL VCO frequency must be set to > 1500 MHz
  - For best PLL performance, maximize VCO frequency where possible.
- A PLL must always operate in Fractional Mode (DAC_EN and DSM_EN set to ‘1’). This is true even when the FB_DIV is a integer.
- A PLL Reference clock frequency must be > 10 MHz.
- REF_DIV[5:0] must be set to “000001”.
- Calibration procedures for FRACF PLLs in integer mode or fractional mode:
  - **Programming FRACF in Integer Mode with calibration (Static or Auto Calibration)**
    - Set all the common PLL settings (PREDIV, FBDIV, …)
    - DACEN: 0
    - DSMEN = 0 (INT MODE)
    - FASTCAL=1
    - CALBYPASS=0
    - CALCNT[2:0]=2
    - CALIN[11:0]=0
    - CAL_EN=1
    - SET PLL_EN = 1
    - WAIT FOR PLL_LOCK AND CAL_LOCK GO HI
    - CAPTURE AND STORE CALCODE WHEN CAL_LOCK GOES HI IF IT WILL BE USED LATER
    - IF CAL_LOCK times out (> 2 ms) then continue
    - CAL_EN=0 (OPTIONAL, THIS WILL FREEZE CALIBRATION & INTERNAL CALCODE)
  - **Programming FRACF in Fractional Mode (with Static Calcode)**
    - Set all common PLL settings to nearest integer (PREDIV, FBDIV,...)
    - Run calibration as outlined in integer mode
    - CAPTURE AND STORE CALCODE WHEN CAL_LOCK GOES HI IF IT WILL BE USED LATER
    - IF CAL_LOCK times out (> 2 ms) then continue
    - CAL_EN=0 (THIS WILL FREEZE CALIBRATION & INTERNAL CALCODE)
    - IF CAL_LOCK timed out, keep CALBYPASS=0

Copyright © 2026 Texas Instruments Incorporated

- IF CAL_LOCK IS ASSERTED AND CALCODE IS CAPTURED APPLY CALCODE TO CALIN
- CALBYPASS=1 CALCODE IS SET BY CALIN
- ENABLE FRACTIONAL MODE

– **Programming FRACF in Fractional Mode (Auto-Calibration Not Allowed)**

- Set all the common PLL settings (PREDIV, FBDIV, …)
- DACEN=1
- DSMEN=1 (FRAC MODE)
- FASTCAL =X
- CALBYPASS=1 CALCODE IS SET BY CALIN
- CALCNT[2:0]=XX
- CALIN[11:0] = 0 (OR DESIRED CALCODE)
- CAL_EN = 0
- SET PLL_EN = 1
- WAIT FOR PLL_LOCK

PLL Integer mode with Calibration enabled is recommended for applications sensitive to phase noise or jitter: Examples include Refclk for SerDes, PHY, DDR etc. See individual product datasheets for specific SoC requirements. Calibration can be bypassed for digital clocks like ARM. If Calibration is bypassed, CALIN needs to be set to 0.

When Calibration is enabled, PLL Lock time will be longer since SOC needs to wait for CAL_LOCK. Applications sensitive to phase noise must wait for CAL_LOCK to go high.

6.4.5.7.2 HSDIV PLL Programming

<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN = 0 cleanly disables the HSDIV output clock

<PLL_Name>_HSDIV_CTRLk[6-0] HSDIV = <value> for divider

<PLL_Name>_HSDIV_CTRLk[8] SYNC_DIS = 0 insures that the divider changes occur synchronously to the internal divider of the HSDIV block.

when the clock is required,

<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN = 1 cleanly enables the HSDIV output clock

6.4.5.7.3 PLL Controllers Programming - Dividers PLLDIVn and GO Operation

The GO operation is required to change the divider ratios of the PLLDIVn registers. Section 6.4.5.7.3.1, *GO Operation*, discusses the GO operation. Section 6.4.5.7.3.2, *Software Steps to Modify PLLDIVn Ratios*, provides the software steps required to change the divider ratios.

6.4.5.7.3.1 GO Operation

The GO operation writes to the RATIO field in the PLLDIVn. Registers do not change the dividers' divide ratios immediately. The PLLDIV dividers change to the new RATIO rates only during a GO operation. This section discusses the GO operation and alignment of the SYSCLKs.

The PLL Controller clock align control register (ALNCTL) determines which SYSCLKs must be aligned. Before a GO operation, program ALNCTL so that the appropriate clocks are aligned during the GO operation.

A GO operation is initiated by setting the GOSET bit in PLLCMD to 1. During a GO operation:

- Any SYSCLK *n* with the corresponding ALN *n* bit in ALNCTL and SYS *n* bit in DCHANGE set to 1 is paused at the low edge. Then the PLL Controller restarts all these SYSCLKs simultaneously, aligned at the rising edge. When the SYSCLKs are restarted, SYSCLK *n* toggles at the rate programmed in the RATIO field in PLLDIVn.
- Any SYSCLK *n* with the corresponding ALN *n* bit in ALNCTL cleared and the SYS *n* bit in DCHANGE set immediately changes to the new rate programmed in the RATIO field.
- The GOSTAT bit in PLLSTAT is set throughout the duration of a GO operation.

Copyright © 2026 Texas Instruments Incorporated

CAUTION

To help prevent errors, all device operation must be stopped before the GO operation.

6.4.5.7.3.2 Software Steps to Modify PLLDIV Ratios

Perform the following steps to modify PLLDIV.

1. Check that the GOSTAT bit in PLLSTAT is cleared to show that no GO operation is currently in progress.
2. Program the RATIO field in PLLDIVn to the desired new divide-down rate. If the RATIO field changed, the PLL Controller will flag the change in the corresponding bit of DCHANGE.
3. Set the respective ALN *n* bits in ALNCTL to align any SYSCLKs after the GO operation.
4. Set the GOSET bit in PLLCMD to initiate the GO operation to change the divide values and align the SYSCLKs as programmed.
5. Read the GOSTAT bit in PLLSTAT to make sure the bit returns to 0 to indicate that the GO operation has completed.

6.4.5.7.4 Entire Sequence for Programming PLLCTRL, HSDIV, and PLL

Table 6-49 shows the entire sequence of programming PLLCTRL, HSDIV, and PLL from an unknown state to a defined non-spread-spectrum state.

Table 6-49. Programming Sequence of PLLCTRL, HSDIV, and PLL

Step	Description
Unlock PLL registers	<PLL_Name>_LOCKKEY0 (= 0x68EF3490)
	<PLL_Name>_LOCKKEY1 (= 0xD172BC5A)
If PLL0	Configure PLLCTRL block into bypass mode
	PLLCTL[0]PLLEN = 0
	PLLCTL[5] PLLENSRC = 0
Configure external bypass so that no transient clock propagates	<PLL_Name>_CTRL[31] BYPASS_EN = 1
Delay	
Disable HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN =0
Delay	
Disable PLL	<PLL_Name>_CTRL[15] PLL_EN =0
Delay	
Reset HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[31] RESET = 1
If PLL0	Check PLLSTAT[0] GOSTAT to make sure that divider change is not currently in-progress GOSTAT should equal 0.
	Clear GOSET bit PLLCMD[0] GOSET = 0
	Configure divider in PLLCTRL to /1 PLLDIV1 = 0x8000 (enable divider and divide by 1) PLLDIV2 = 0x00 (disable divider and divide by 1)
	Set alignment control bits ALNCTL = 3
	Set GOSET bit to initiate the divider change PLLCMD[0] GOSET = 1
	Check PLLSTAT[0] GOSTAT to make sure that divider change has completed GOSTAT should equal 0.
Configure HSDIV(s) divider value	<PLL_Name>_HSDIV_CTRLk[6-0] HSDIV (=val)
Clear HSDIV(s) SYNC_DIS	<PLL_Name>_HSDIV_CTRLk[8] SYNC_DIS = 0
Delay	

Copyright © 2026 Texas Instruments Incorporated

Table 6-49. Programming Sequence of PLLCTRL, HSDIV, and PLL (continued)

Step	Description
Clear Reset HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[31] RESET = 0
Configure PLL multiplier	Integer portion of divider <PLL_Name>_FREQ_CTRL0[11-0] (=val) Fractional portion of divider <PLL_Name>_FREQ_CTRL1[23-0] (=val)
Configure PLL dividers	Reference clock divider <PLL_Name>_DIV_CTRL[5-0] REF_DIV = 1
	Post divider 1 <PLL_Name>_DIV_CTRL[18-16] POST_DIV1 = 1
	Post divider 2 <PLL_Name>_DIV_CTRL[26-24] POST_DIV2 = 1
Configure “random” PLL controls	<PLL_Name>_CTRL[8] INTL_BYP_EN = 0
	<PLL_Name>_CTRL[5] CLK_4PH_EN = 0
	<PLL_Name>_DIV_CTRL[26-24] POST_DIV2 = 1
	<PLL_Name>_DIV_CTRL[18-16] POST_DIV1 = 1
	<PLL_Name>_DIV_CTRL[5-0] REF_DIV = 1
	<PLL_Name>_CTRL[1] DSM_EN = 1
	<PLL_Name>_CTRL[0] DAC_EN = 1
	<PLL_Name>_CTRL[4] CLK_POSTDIV_EN – if PLL has more than 5 HSDIV blocks, 1; else 0.
	<PLL_Name>_CTRL[16] BYP_ON_LOCKLOSS (= {0, 1})
Delay	
Enable PLL	<PLL_Name>_CTRL[15] PLL_EN = 1
Wait for Lock	Wait for MCU_PLL0_STAT[0] LOCK = 1
Enable HSDIV(s)	<PLL_Name>_HSDIV_CTRLk[15] CLKOUT_EN = 1
Delay	
Configure external bypass to pass PLL	<PLL_Name>_CTRL[31] BYPASS_EN = 0
Delay	
If PLL0	Configure PLLCTRL block into PLL mode PLLCTL[0] PLLEN = 0
Lock PLL registers	<PLL_Name>_LOCKKEY0 (= any value)
	<PLL_Name>_LOCKKEY1 (= any value)

6.4.5.7.5 Dual Clock Comparator (DCC)

The Dual Clock Comparator (DCC) is used to determine the accuracy of a clock signal during the time execution of an application. Specifically, the DCC is designed to detect drifts from the expected clock frequency. The desired accuracy can be programed based on calculation for each application. The DCC measures the frequency of a selectable clock source using another input clock as a reference. Internal 12.5 MHz RC oscillator clock is used to validate HFOSC0_CLKOUT. Then HFOSC0_CLKOUT (most accurate clock) is used to validate/track all other clocks in the device. An additional use of this module is to measure the frequency of a selectable clock source, using an external input clock source of known frequency (MCU_EXT_REFCLK0, EXT_REFCLK1 pins). There is one instance of DCC module in MCU domain and seven instances of DCC modules in Main domain to monitor critical clocks. Refer to DCC chapter for details.

Copyright © 2026 Texas Instruments Incorporated

Chapter 7

Processors and Accelerators

7.1 Arm® Cortex®-A53 Subsystem (A53SS) 528
7.3 Cortex R5F Subsystem (R5FSS) 553
7.4 Device Manager Cortex R5F Subsystem (WKUP_R5FSS) 585
7.5 Vectored Interrupt Manager (VIM) 607
7.6 Video Encoder/Decoder (VENC/VDEC) 625
7.7 Vision Pre-processing Accelerator (VPAC) 633
7.8 Depth and Motion Perception Accelerator (DMPAC) 786
7.9 Graphics Accelerator (GPU) 787

Copyright © 2026 Texas Instruments Incorporated

7.1 Arm® Cortex®-A53 Subsystem (A53SS)

This section describes the Arm® Cortex®-A53 Subsystem (A53SS) in the device.

Copyright © 2026 Texas Instruments Incorporated

7.1.1 A53SS Overview

7.1.1.1 A53SS Introduction

The SoC implements one cluster of quad-core Arm® Cortex® A53 MPCore™, with 32KB L1 instruction, 32KB L1 data, per core and 512KB L2 shared cache.

The Cortex®-A53 cores are general-purpose processors that can be used for running customer applications.

Note

Notes on references used in this document:
- A53SS is also referred to as Arm® CorePac.
- Cortex®-A53 is often shortened to A53.

The A53SS is built around the Cortex®-A53 MPCore™ (Arm® A53 Cluster), which is provided by Arm and configured by TI. It is based on the symmetric multiprocessor (SMP) architecture, and thus it delivers high performance and optimal power management, debug and emulation capabilities.

The A53 processor is an in-order, 8-stage, dual-issue pipeline with integrated L1 Instruction and Data Caches, compatible with Arm®v8-A architecture. It delivers significantly more performance than its predecessors at a higher level of power efficiency.

The Arm®v8-A architecture brings a number of new features. These include 64-bit data processing, extended virtual addressing and 64-bit general purpose registers. The A53 processor is Arm's first Arm®v8-A processor aimed at providing power-efficient 64-bit processing. It features an in-order, 8-stage, dual-issue pipeline, and improved integer, Arm® Neon™, Floating-Point Unit (FPU) and memory performance.

The A53 CPU supports two execution states: AArch32 and AArch64. The AArch64 state gives the A53 CPU its ability to execute 64-bit applications, while the AArch32 state allows the processor to execute existing Arm®v7-A applications.

7.1.1.2 A53SS Features

The A53SS module supports the following features:

- Quad Core A53 Cluster
  - Full Arm®v8-A Architecture Compliant
    - AArch32 and AArch64 Execution States
    - All exception levels EL0-3
    - A32 Instruction Set (Previously Arm instruction set)
    - T32 instruction set (Previously Thumb instruction set)
    - A64 Instruction Set
  - Data Coherency within Cluster (L1/L2 caches)
  - Advanced SIMD and Floating Point Extensions (Arm® Neon™)
  - Arm®v8 Cryptography Extensions
  - Arm GICv3 architecture
  - In-order pipeline with symmetric dual-issue of most instructions
  - Harvard L1 with system MMU
    - 32 KB Instruction Cache
    - 32 KB Data Cache
  - 512KB Shared L2 Cache
  - Generic Timer(s)
  - Debug
- 128-Bit VBUSM Initiator Interfaces (for axi_r and axi_r channels)
- 128-Bit VBUSM Target Interface (for Accelerator Coherency Port)
- 64-bit Grey-coded system input time
- 48-bit Grey-coded debug input time

Copyright © 2026 Texas Instruments Incorporated

- 48-bit Grey-coded debug input time
- Integrated PBIST controller with BISOR

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

7.1.2 A53SS Functional Description

7.1.2.1 A53SS Block Diagram

7.1.2.2 Arm Cortex-A53 Cluster

The Arm Cortex-A53 Cluster is provided by Arm and configured by TI. Table 7-1 summarizes the configuration of the Arm Cortex-A53 Cluster on this SoC.

Table 7-1. Arm A53 Cluster Configuration

Parameter	Value
Core Type	A53
Core Revision	r0p4
Number of Cores	4
Bus Width	256
L1 Instruction Cache Size	32K
L1 Data Cache Size	32K
L2 Cache Size	512K
SCU-L2 Cache Protection	Included
Advanced SIMD and Floating Point Extension	Included
Cryptography Extension	Included
CPU Cache Protection	Included
AMBA5 CHI or AMBA4 ACE Interface	AMBA4 ACE (configured for AXI using tie-offs)
Accelerator Coherency Port (ACP)	Included
V7 or v8 Debug Memory Map	v8

Note

For a brief list of features supported by the Arm Cortex-A53 Cluster, see A53SS Features.

For detailed description of the Arm® A53 Cluster, see the *Arm®Cortex®-A53 MPCore™ Processor Technical Reference Manual*.

7.1.2.3 A53SS Interfaces and Async Bridges

The A53SS has the following main interfaces:

- 64-bit graycoded system input time
  - Graycode value provided by Global Timebase Counter (GTC)
  - Dedicated decoder (64-bit input) for graycode-to-binary conversion
- 48-bit graycoded debug input time
  - Graycode value provided by GTC
  - Dedicated decoder (48-bit input) for graycode-to-binary conversion
- 32-bit VBUSP target interface for debug
  - Supported by VBUSP2APB Bridge, which performs VBUSP-to-APB conversion (for controlling the Arm A53 Cluster internal debug logic)
- 32-bit ATB output port for debug/trace
  - Supported by ATB Bridge, which performs clock and voltage level conversion on the combined ATB interface
  - Connected to the Debug Subsystem
- Cross Trigger Interface (CTI) for debug
  - Connected to the Debug Subsystem
- Interface(s) with Arm GIC-500 Interrupt Controller

Copyright © 2026 Texas Instruments Incorporated

  - Supported by GIC AXI Streaming Bridge, which performs clock and voltage level conversion on the AXI streaming protocol
  - Interrupts (PPIs) from Arm A53 Cluster to GIC-500
  - Interrupts (IRQ, FIQ, VIRQ, VFIQ) from GIC-500 to Arm A53 Cluster
- Power/clock interface(s)
  - Dedicated PLL for each Arm A53 Cluster
  - Dedicated LPSC for each Arm A53 Cluster, and also for each A53 core

7.1.2.4 A53SS Interrupts

7.1.2.4.1 A53SS Interrupt Inputs

The A53 CPU recieves interrupts at its inputs (IRQ, FIQ, VIRQ, VFIQ) via the dedicated Arm GIC-500 Interrupt Controller, which resides at the SoC level (MAIN Domain) and is integrated inside the GIC0 Subsystem. The GIC-500 supports all four A53 cores in the system.

The GIC-500 is compliant to the Arm GICv3 standard and supports four types of interrupts:

- *Software Generated Interrupts (SGI)*
  - There are 16 SGIs (ID0-ID15)
  - These are inter-processor interrupts
- *Private Peripheral Interrupts (PPI)*
  - There are 16 PPIs (ID16-ID31)
  - These are wired interrupts dedicated to a specific CPU
  - Many are reserved to specific functions via convention
- *Shared Peripheral Interrupts (SPIs)*
  - There are 256 SPIs (ID32-ID288)
  - These are wired interrupts that can be routed to any core or cluster, based on the programming of that interrupt in the GIC500
- *Locality-Specific Peripheral Interrupts (LPI)*
  - There are 57,344 LPIs
  - These interrupts are used for message-based interrupts from a peripheral

The mapping of PPIs and SPIs to the GIC-500 interrupt inputs can be found in *Interrupts*.

Note

For a brief list of features supported by the GIC-500 module, see the *Interrupts* chapter.

For detailed description of the GIC-500 module, see the *Arm®CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

7.1.2.4.2 A53SS Interrupt Outputs

A53SS Interrupt Outputs lists the interrupts generated by the A53SS.

Table 7-2. A53SS Interrupt Outputs

Interrupt Name (TI)[1]	Interrupt Name (Arm)	Interrupt Description
A53 Core Interrupts (PPIs)[2]		
A53_COREy_VCPUMNTIRQ	VCPUMNTIRQn	This interrupt indicates that a Virtual CPU Interface on the corresponding core needs serviced. This interrupt is serviced by software switching to the virtual CPU and finding what specific service needs done.
A53_COREy_CNTHPIRQ	CNTHPIRQn	This interrupt indicates a physical timer event at the EL2 (Hypervisor) exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL2 and service accordingly.

Copyright © 2026 Texas Instruments Incorporated

Table 7-2. A53SS Interrupt Outputs (continued)

Interrupt Name (TI)(1)	Interrupt Name (Arm)	Interrupt Description
A53_COREy_CNTPNSIRQ	CNTPNSIRQn	This interrupt indicates a physical timer event at the EL1 Non-Secure exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL1 Non-Secure and service accordingly.
A53_COREy_CNTPSIRQ	CNTPSIRQn	This interrupt indicates a physical timer event at the EL1 Secure exception level. Service of this interrupt is dependent on what the timer was set for. Software should take exception to EL1 Secure and service accordingly.
A53_COREy_CNTVIRQ	CNTVIRQn	This interrupt indicates a Virtual timer event. Service of this interrupt is dependent on what the timer was set for. Software should take exception and service accordingly.
A53_COREy_PMUIRQ	PMUIRQn	This interrupt is generated by the Performance Monitor Unit (PMU). The PMU can generate an interrupt based on several conditions, depending on programming. Software should take exception and query the PMU as to the cause of the interrupt, and act accordingly.
A53_COREy_DCCIRQ	COMMIRQn	Communications Channel receive or transmit interrupt
A53_COREy_CTIIRQ	CTIIRQn	Cross Trigger Interface interrupt

(1) In this column, *y* is the A53 Core index (0 or 1)
(2) These are 'per core' interrupts

Note

The mapping of these interrupts in the system is summarized in *A53SS Integration*, and can also be found in *Interrupts*.

For more detailed description of these interrupts and their handling, see the *Arm®Cortex®-A53 MPCore™ Processor Technical Reference Manual*.

7.1.2.5 A53SS Power Management and Clocking

7.1.2.5.1 A53SS Power Management

Each Arm A53 Cluster and each A53 CPU reside in a separate power domain, as follows:

There is a dedicated Local Power Sleep Controller (LPSC) for each Arm A53 Cluster, and for each A53 core, as well. The LPSC assignment is as follows:

For more details on these LPSCs, including power-up/down sequences, see *Power*.

7.1.2.5.2 A53SS Clocking

There is a dedicated PLL for each Arm A53 Cluster. The PLL assignment is as follows:

- PLL8 (ARM0 PLL): Dedicated for Arm A53 Cluster 0

For more details on these PLLs, see *Clocking*.

7.1.2.6 A53SS Debug

The A53SS supports the standard Arm debug architecture. Details on Arm debug can be found in the *On-chip Debug* chapter and the relevant Arm specifications.

7.1.2.7 A53SS Global and Debug Timestamps

The A53SS has two timebase input interfaces:

- 64-bit global timestamp: Used for synchronizing the Arm internal timers (via the ARM CNTVALUEB [63:0] bus)
- 48-bit debug timestamp: Can be embedded in Arm trace streams at periodic and strategic locations, allowing the temporal relationship of different trace sources to be determined

Copyright © 2026 Texas Instruments Incorporated

Both of them are fed by the Global Timebase Counter (GTC), which provides a 64-bit graycode value. The A53SS includes two graycode decoders (for global time and debug time, respectively), which take the asynchronous graycoded times, synchronize them to the appropriate clock, and convert them to binary time, as required by Arm.

Note

Both graycode decoders are 64-bit but the one dedicated to debug time has a 48-bit input (the upper 16 bits are tied to 0).

For more details on the GTC, see *Global Timebase Counter (GTC)*.

7.1.2.8 A53SS Watchdog

The A53SS does not have an integrated watchdog timer. Instead, this feature is provided externally by the Real Time Interrupt Module (RTI) module, which implements a windowed watchdog timer capable of issuing warm reset to the SoC, when necessary.

For more details on the RTI windowed watchdog feature, see *Real Time Interrupt Module (RTI/WWDT)*.

7.1.2.9 A53SS Functional Safety - ECC Error Injection Support

In the quad A53 CBA subsystem, there are five ECC aggregators – one for each core and one for the corepac level. Debug domain is not safety critical.

No CBASS/Bridge interconnect safety is supported. Only RAM safety is supported.

The following A53SS ECC Aggregator instances are instantiated:

- A53SS0_ECC_AGGR0: ECC Aggregator for Arm A53 Cluster 0, Core 0
- A53SS0_ECC_AGGR1: ECC Aggregator for Arm A53 Cluster 0, Core 1
- A53SS0_ECC_AGGR2: ECC Aggregator for Arm A53 Cluster 0, Core 2
- A53SS0_ECC_AGGR3: ECC Aggregator for Arm A53 Cluster 0, Core 3
- A53SS0_ECC_AGGR_COREPAC: ECC Aggregator for Arm A53 Cluster 0

7.1.2.9.1 A53 ECC Aggregators During Low Power States

- When a cpu core is in WFI/WFE, the corresponding core ECC Aggregator MMR regions is not accessible and will return error status.
- Similarly when L2 is in WFI, the corepac ECC Aggregator MMR region is not accessible and will return error status.

7.1.2.9.2 Auto-initialization of Memories

All A53 L1 and L2 memory initialization is handled by the Arm core. The TI ECC aggregator is used in inject-only mode and this cannot be used for initialization. The Arm CPU will initialize the cache memories after reset.

There are no other memories in the A53 subsystem beside the cache memories

7.1.2.9.3 A53 SRAM Safety

The Arm A53 Cluster natively supports ECC/parity protection on A53 SRAMs and error injection on few memories. The A53SS adds error injection capability on all internal SRAM. This is a valuable TI addition to the Arm native implementation.

Table 7-3. A53 SRAM Safety Support

RAM	A53 Error Injection Support	TI Error Injection Support
L1 I-Cache Data	No	Single error injection
L1 I-Cache Tag	No	Single error injection
L1 D-Cache Data	Double error injection	Single and double error injection
L1 D-Cache Tag	No	Single error injection
L1 Data Dirty	No	Single error injection

Copyright © 2026 Texas Instruments Incorporated

Table 7-3. A53 SRAM Safety Support (continued)

RAM	A53 Error Injection Support	TI Error Injection Support
TLB RAM	No	Single error injection
SCU Duplicate Tag	No	Single and double error injection
L2 Tag RAM	Double error injection	Single and double error injection
L2 Data RAM	Double error injection	Single and double error injection

The ECC Aggregator for the cores stimulates errors in the following RAMs:

- L1 I-Cache Data RAM
- L1 I-Cache Tag RAM
- L1 D-Cache Data RAM
- L1 D-Cache Tag RAM
- L1 D-Cache Dirty RAM
- TLB RAM
- L1 SCU L1-D Duplicate Tag RAM

The ECC Aggregator for the L2 cache stimulates errors in the following RAMs:

- L2 Data RAM
- L2 Tag RAM

7.1.2.9.4 A53 SRAM ECC Aggregator Configurations

There are several schemas of memory protection employed inside of the processors. The tables below describe the schema and arrangement. This describes what bits are protected, how they are protected, the behavior when an error is detected and what bits can be disturbed for ECC testing purposes. This also lists the RAMID associated with each memory to the corresponding ECC Aggregator.

The key for the protection schemes is as follows:

- Parity – Parity Bit(s) to protect data
- ECC – Error Correction Code to protect Data
- SED – Single Error Detection
- SECDED – Single Error Correction, Double Error Detection
- SEDSEC – Single Error Detection, Single Error Correction

Table 7-4. CPU Memories Safety Details

Memory	Protection	RAM Arrangement	CPU (Core ECC Aggr) Rams Ram ID	Notes
L1 I-Cache Data	Parity SED	41 – Parity for 40:21 40:21 – Instruction 20 – Parity for 19:0 19:0 – Instruction	0-3	Error detection results in both lines being invalidated then refetched from L2 or memory
L1 I-Cache Tag	Parity SED	31 – Parity for 30:0 30:0 – Data	4-5	Error detection results in both lines being invalidated, then line refetched from L2 or memory
L1 D-cache Data	ECC SECDED	38:32 – ECC 31:0 – Data	6-13	Error results in line being cleaned and invalidated from L1 with single bit errors corrected as part of eviction. Line refetched from L2 or memory
L1 D-Cache Tag	Parity SED	30 – Parity for 29:0 29:0 – Tag	14-17	Cache sizes makes LSB unnecessary (always 0) so they are removed from the actual RAM. Error results in line cleaned and invalidated from L1. SCU duplicate tags are used to get the correct address. Line refetched from L2 or memory

Copyright © 2026 Texas Instruments Incorporated

Table 7-4. CPU Memories Safety Details (continued)

Memory	Protection	RAM Arrangement	CPU (Core ECC Aggr) Rams Ram ID	Notes
L1 D-cache Dirty	Parity SEDSEC	11:10 – Way 1/3 Dirty copy 2 and 1. Parity for 4 9:8 – Way 0/2 Dirty copy 2 and 1. Parity for 0 7 – Way 1/3 Outer Allocation Hint 6 – Way 1/3 Age 5:4 – Way 1/3 Partial MOESI 3 – Way 0/2 Outer Allocation Hint 2 – Way 0/2 Age 1:0 – Way 0/2 Partial MOESI	18	Error results in line cleaned and invalidated from L1 with single bit errors corrected as part of the eviction. Only dirty bit is protected. Other bits are just performance hints
TLB	Parity SED	116 – parity for 113:62 115 – Parity for 61:31 114 – Parity for 30:0 113:62 – Page Attributes 61:31 – Entry Identifiers 30:0 – Address	19-22	Error detection results in entry invalidated, new pagewalk to refetch.
SCU L1 Duplicate Tag	ECC SECDED	37:31 - ECC for 30:0 30:0 - Duplicate Tag	23-26 (accessed by each Core's ecc_aggr)	Cache sizes makes LSB unnecessary (always 0) so they are removed from the actual RAM. Corretable Error – Tag rewritten with correct value, access retried Uncorrectable Error – Tag is invalidated

Table 7-5. L2 Memories Safety Details

Memory	Protection	RAM Arrangement	RAM_ID (L2 Cache RAMs)	Notes
L2 Tag	ECC SECDED	37:31 – ECC for 30:0 30:0 - Tag	0-15	Cache sizes makes LSB unnecessary (always 0) so they are removed from the actual RAM. Corretable Error – Tag rewritten with correct value, access retried. Uncorrectable Error – Tag is invalidated
L2 Victim	None	–	–	Performance Hint Only – Error has no functional impact
L2 Data	ECC SECDED	71:64 – ECC for 63:0 63:0 – Data	16-23	Error results in Data corrected inline, access may stall for 1-2 cycles. After correction, line might be evicted
Branch Predictor	None	–	–	Performance Hint Only – Error has no functional impact

7.1.2.10 A53SS Boot

For A53SS boot sequence and any other A53SS boot details, see *Control Module (CTRL_MMR)*.

7.1.2.11 A53SS Interprocessor Communication

The Arm A53 core(s) can communicate with other device cores by supporting interrupt generation to and from these cores. The interprocessor communication (IPC) interrupts are assigned in the corresponding BOOTCFG0 memory-mapped registers (MMRs) called IPC_SETx / IPC_CLRx. For more information, see *Control Module (CTRL_MMR)*.

Copyright © 2026 Texas Instruments Incorporated

7.2 Arm Cortex R5F Subsystem (R5FSS)

This chapter describes the Arm Cortex R5F real-time microcontroller unit subsystem (R5FSS) in the device.

7.2.1 R5FSS Overview 538
7.2.2 R5FSS Functional Description 540

Copyright © 2026 Texas Instruments Incorporated

7.2.1 R5FSS Overview

The R5FSS is a dual-core implementation of the Arm® Cortex®-R5F processor configured for split or single-core operation. It also includes accompanying memories (L1 caches and tightly-coupled memories), standard Arm CoreSight™ debug and trace architecture, integrated vectored interrupt manager (VIM), ECC aggregators, and various other modules for protocol conversion and address translation for easy integration into the SoC.

Note

The Cortex-R5F processor is a Cortex-R5 processor that includes the optional floating point unit (FPU) extension. In this TRM, all references to the Cortex-R5 processor apply to the Cortex-R5F processor by default.

7.2.1.1 R5FSS Features

Each R5FSS supports the following features:

- Dual-core Arm Cortex-R5F
  - Core revision: r1p3
  - Armv7-R profile
  - Boot-time configurable to be in single- or dual-core (split) mode
    - Dual-core mode: Two independently operating cores (asymmetric multi processing, no coherence)
    - Single-core mode: Only one operating core (CPU0)
      - CPU0 uses TCM resources of both cores
      - CPU1 caches and interrupts are unused in this mode
  - L1 memory system
    - 32KB instruction cache
      - 4x8KB ways
      - SECDED ECC protected per 64 bits
    - 32KB data cache
      - 4x8KB ways
      - SECDED ECC protected per 32 bits
    - 64KB tightly-coupled memory (TCM) per CPU
      - SECDED ECC protected per 32 bits
      - TCM hard error cache Implemented in CPU
      - Readable/writable from system
      - TCMs initialized (to 0's) at reset
      - Split into A and B banks (with B further splitting into B0 and B1 interleaved banks)
        - 32KB TCMA (ATCM)
        - 16KB TCMB0 (B0TCM)
        - 16KB TCMB1 (B1TCM)
      - In dual-core mode, TCM is 64KB in total (per core):
        - 32KB TCMA
        - 16KB TCMB0 + 16KB TCMB1
      - In single-core mode, TCM is 128KB in total (per core):
        - 64KB TCMA
        - 32KB TCMB0 + 32KB TCMB1
  - Low interrupt latency with restartable instructions
  - Non-maskable interrupt (NMI)
  - Full-precision floating point (VFPv3)
  - 16 region memory protection unit (MPU)
  - 8 breakpoints
  - 8 watchpoints
  - Dynamic branch prediction with global history buffer and 4-entry return stack

Copyright © 2026 Texas Instruments Incorporated

  - CoreSight debug access port (DAP)
  - CoreSight embedded trace macrocell (ETM-R5) interface
  - Performance monitoring unit (PMU)
- Interfaces
  - 64-bit VBUSM initiator pair (1 read, 1 write) for L3 memory accesses (per core)
  - 64-bit VBUSM target for TCM access (per core)
    - Also allows access to cache for debug purposes
  - 32-bit VBUSP initiator for peripheral access (per core)
  - 32-bit VBUSP target configuration port (per core)
  - 32-bit VBUSP target debug port
    - Allows access to all R5FSS internal debug logic
- Synchronous clock domain crossing on all interfaces
  - Both CPU and interface clocks run at the same frequency (1:1 ratio)
- 32-bit to 36-bit region-based address translation (RAT) on memory access initiators
  - 4 regions
    - Base address + size
    - Must be size aligned
- Integrated vectored interrupt manager (VIM)
  - 256 interrupts per core
    - Only interrupts connected to R5F core 0 are available in single-core mode
    - Each interrupt programmable as either IRQ or FIQ
    - Each interrupt has a programmable enable mask
    - Each interrupt has a programmable 4-bit priority
  - Priority interrupt supported
  - Vectored interrupt interface
    - Compatible with R5F VIC port
    - Programmable 32-bit vector address per interrupt
      - Address is SECDED error protected
      - Default vector addresses provided on DED
    - Split or single-core capable
    - Software interrupt generation
- Integrated ECC aggregators
  - Support for error injection to all supported ECC memory blocks to test ECC functionality (add-on function from TI)
  - One ECC aggregator per core to cover all RAMs and caches associated with that core
- Standard Arm CoreSight debug and trace architecture at the R5FSS level
  - Cross triggering: Supported by cross trigger interface (CTI) (per CPU) and cross trigger matrix (CTM) components
  - Processor trace: Supported by embedded trace macrocell (ETM) (per CPU) and advanced trace bus (ATB) funnel components
- Boot
  - From ROM or external memory
  - From TCM

See Section 7.2.2 for a functional block diagram and more details on the R5FSS.

7.2.1.2 R5FSS Not Supported Features

The R5FSS does *not* support the following native R5F features in this device:

- ACP port (no coherence)
- Bus parity / ECC
- Multiple power domains

Copyright © 2026 Texas Instruments Incorporated

7.2.2 R5FSS Functional Description

7.2.2.1 R5FSS Block Diagram

Figure 7-2 shows the R5FSS block diagram.

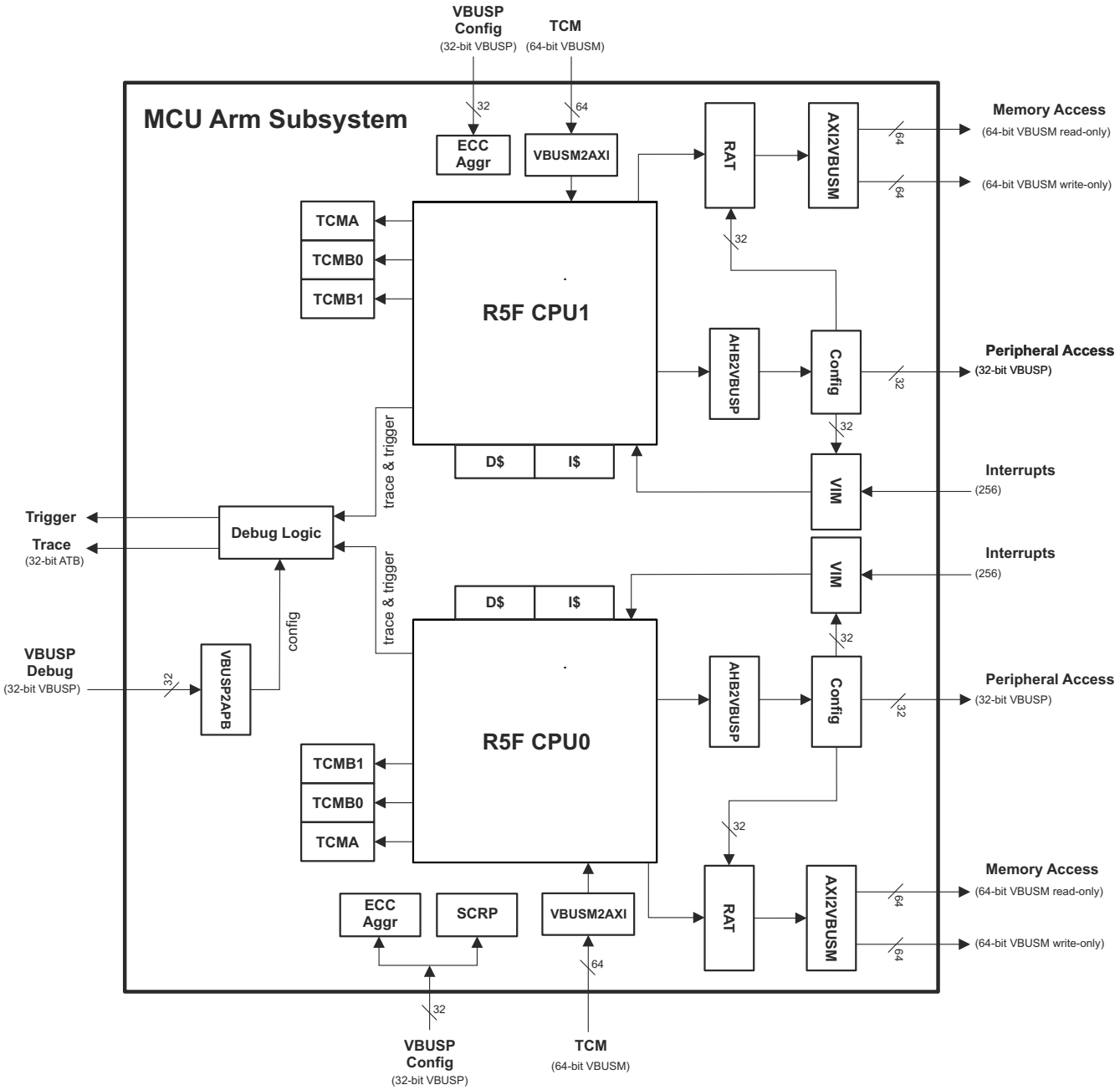


Figure 7-2. R5FSS Block Diagram

7.2.2.2 R5FSS Cortex-R5F Core

The Cortex-R5F is a processor from Arm, which is based on the Armv7-R profile. Each R5FSS implements two R5F cores, CPU0 and CPU1, each with their own RAMs and interfaces. While in reset, they can be bootstrapped to work in one of two modes: split or single-core.

In split mode, each R5F core works completely independent from the other (asymmetric multi-processing, or AMP). Each core uses its own RAMs and interfaces, with no coherence between the two cores. The only

Copyright © 2026 Texas Instruments Incorporated

restriction is that CPU0 must be in a higher power/reset state than CPU1. For instance, CPU1 cannot be out of reset if CPU0 is not.

In single-core mode:

- CPU0 is the only operating core
- CPU1 TCMs are stacked on CPU0 TCMs and are accessible only by CPU0 and CPU0 TCM interface
- The TCM size for CPU0 is essentially doubled in this mode (128KB)
- CPU1 caches and interrupts are not used

For a brief list of features supported by the R5F processor in this device, see Section 7.2.1.1. For more detailed description of this processor, see the *Arm Cortex-R5 Technical Reference Manual*.

7.2.2.2.1 L1 Caches

The R5F has a Harvard cache architrecture, which means it has an independent L1 instruction cache (16KB) and L1 data cache (16KB). The instruction cache is protected by SECDED ECC per 64 bits. The data cache is protected by SECDED ECC per 32 bits.

7.2.2.2.2 Tightly-Coupled Memories (TCMs)

The R5F has two tightly-coupled memories (TCMs), ATCM and BTCM. The BTCM is further broken down into two interleaved banks, B0TCM and B1TCM.

TCMs are low-latency, tightly integrated memories for the R5F to use. Either TCM can be used for any combination of instruction and/or data. TCM performance is equal to performance on instructions/data that are in cache. However, TCMs have some additional advantages over cache. TCMs can be loaded with instructions that do not cache well (such as ISRs) or preloaded with code by an external source, before that code is needed, to save cache miss time. TCMs are also a good place for blocks of data for intense processing. They can be loaded (or pre-loaded by an external source) before the data is needed, saving cache miss time. The data can then be directly accessed by an external source, instead of needing to do cache evicts.

As mentioned, TCMs can be accessed (either read or written) by an external source over the TCM VBUSM target interface. This allows instructions or data to be preloaded, or for data to be read out after the R5F has processed it. The VBUSM target has a lower priority to accessing TCMs than the R5F but care must be taken to keep an external source from reading or writing TCM data that the R5F is working on. This handshaking is external to any of the R5FSS hardware.

TCMs are protected by ECC per 32 bits. For this to work, ECC must be enabled before data is written in to the TCMs (either externally or from the R5F). ECC is enabled via the following R5F system control bits: ACTLR.ATCMPCEN, ACTLR.B0TCMPCEN, and ACTLR.B1TCMPCEN, respectively.

Whether or not the TCMs are enabled is controlled by the ENABLE bit in the corresponding ATCM/BTCM region register. The default (reset) value of this bit is determined by the CPUn_INITRAMA and CPUn_INITRAMB bootstraps, respectively. Both ATCM and BTCM are configured for a size of 32KB in this device. Note that the BTCM size is the total of both B0TCM and B1TCM (16KB each).

If a TCM is not enabled, then it does not appear in the R5F's memory view, but it can be accessed by an external source. If a TCM is enabled, then its place in the R5F memory map is determined by a combination of bootstrap signal and system register. If the CPUn_LOCZRAMA bootstrap signal is high, then the initial base address of ATCM is 0x0000_0000 and the initial address of BTCM is 20'h41010. If the CPUn_LOCZRAMA bootstrap signal is low, then the initial base address of BTCM is 0x0000_0000 and the initial base address of ATCM is 20'h41010.

Copyright © 2026 Texas Instruments Incorporated

Note

This base address of 0x41010 for ATCM/BTCM based on the CPUn_LOCZRAMA bootstrap only affects the R5F's memory view. The SoC will see the ATCM/BTCM based on the TCM target interface regions, as defined in Section 7.2.2.3.2. The base address of either TCM may be overwritten via the ATCM or BTCM region register. Care must be taken not to move the base address of a TCM when it may be being accessed.

It is possible to preload a TCM with instructions and boot from it. See Section 7.2.2.11 for details on TCM booting.

7.2.2.2.3 R5FSS Special Signals

Table 7-6 through Table 7-7 list some R5FSS features associated with special signals.

Table 7-6. R5FSS0 Special Features

Feature	Comment
Cluster affinity group ID	R5F Cluster 0 (ID = 0x0)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode
Dual- or single-core mode 0 = Dual mode 1 = Single mode	Controlled via MAIN_SEC_MMR register setting. Defaults to a value defined by eFuse
CPUn execution halt when coming out of reset (CPUn_HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPUn exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPUn VIM base address	0x2FFF_0000
CPUn RAT base address	0x2FFE_0000
CPUn RAT accesses ID	0x4 (CPU0); 0x5 (CPU1)
CPUn ATCM enable at reset (CPUn_INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn BTCM enable at reset (CPUn_INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPUn A/BTCM reset base address indicator (CPUn_LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPUn non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn VBUSM peripheral port enabled at reset	Enabled
CPUn VBUSP peripheral port enable at reset	Enabled
CPUn VBUSP peripheral port base address	Mapped to 0x0_2000_0000 for low latency MAIN peripherals
CPUn VBUSP peripheral port size	64MB for MAIN peripherals (0x0_2000_0000 to 0x0_2FFF_FFFF)
CPUn VBUSM normal peripheral port base address	Not used
CPUn VBUSM normal peripheral port size	Not used
CPUn VBUSM virtual peripheral port base address	Not used
CPUn VBUSM virtual peripheral port size	Not used
CPUn clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPUn WFI state	Status logged into MAIN_SEC_MMR register bit
CPUn WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

Copyright © 2026 Texas Instruments Incorporated

Table 7-7. R5FSS1 Special Features

Feature	Comment
Cluster affinity group ID	R5F Cluster 1 (ID = 0x1)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode
Dual- or single-core mode 0 = Dual mode 1 = Single mode	Controlled via MAIN_SEC_MMR register setting. Defaults to a value defined by eFuse
CPUn execution halt when coming out of reset (CPUn_HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPUn exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPUn VIM base address	0x2FFF_0000
CPUn RAT base address	0x2FFE_0000
CPUn RAT accesses ID	0x6 (CPU0); 0x7 (CPU1)
CPUn ATCM enable at reset (CPUn_INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn BTCM enable at reset (CPUn_INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPUn A/BTCM reset base address indicator (CPUn_LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPUn non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn VBUSM peripheral port enabled at reset	Enabled
CPUn VBUSP peripheral port enable at reset	Enabled
CPUn VBUSP peripheral port base address	Mapped to 0x0_2000_0000 for low latency MAIN peripherals
CPUn VBUSP peripheral port size	64MB for MAIN peripherals (0x0_2000_0000 to 0x0_2FFF_FFFF)
CPUn VBUSM normal peripheral port base address	Not used
CPUn VBUSM normal peripheral port size	Not used
CPUn VBUSM virtual peripheral port base address	Not used
CPUn VBUSM virtual peripheral port size	Not used
CPUn clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPUn WFI state	Status logged into MAIN_SEC_MMR register bit
CPUn WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

7.2.2.3 R5FSS Interfaces

7.2.2.3.1 Initiator Interfaces

The R5FSS has several initiator interfaces per core:

- 64-bit VBUSM initiator pair (1 read – “RMST”, 1 write – “WMST”) for L3 memory accesses; this is the main memory interface
  - Includes region-based address translation (RAT)
- 32-bit VBUSP initiator “PMST” for peripheral access
  - Includes logic that provides the R5F CPU with a private access to VIM and RAT
  - Enabled at reset

Copyright © 2026 Texas Instruments Incorporated

7.2.2.3.2 Target Interfaces

The R5FSS has several target interfaces that define its internal memory space:

- 32-bit VBUSP configuration target (per core)
  - Region [0]: ECC aggregator block
- 64-bit TCM target (per core)
  - Region [0]: ATCM
  - Region [1]: BTCM
  - Region [2]: Instruction cache RAMs
  - Region [3]: Data cache RAMs
- 32-bit VBUSP debug target
  - Provides access to all R5FSS internal debug logic

Regions [0] and [1] of the TCM target interface provide direct access to the TCM RAMs. Access to the RAMs is arbitrated with access from the R5F's L1 memory system. Excessive access while the R5F is also attempting access will degrade performance.

Regions [2] and [3] of the TCM target interface provide access to the cache RAMs for testing purposes. Access to the cache RAMs can only be done while the caches are disabled and should only be done for test purposes.

In addition to the target interfaces, there are peripherals (RAT and VIM) that are only accessible by the R5F. The R5F has an access to these modules via the VBUSP peripheral interface.

7.2.2.4 R5FSS Power, Clocking and Reset

7.2.2.4.1 R5FSS Power

The following R5FSS power considerations should be noted:

- R5FSS has a single power domain for all its internal logic
- When operating in split mode, CPU0 must be in a higher power/reset state than CPU1.

For more details on R5FSS power management, including power-up and power-down sequences, see *Power*.

7.2.2.4.2 R5FSS Clocking

The R5FSS has four clock inputs:

- CPU0_CLK: This is the clock for CPU0 logic
- CPU1_CLK: This is the clock for CPU1 logic
- CPU0_ICLK: This is the clock for CPU0 interfaces
- CPU1_ICLK: This is the clock for CPU1 interfaces

CPU0_CLK and CPU1_CLK are the clocks for all of the internal CPU logic, while CPU0_ICLK and CPU1_ICLK are the clocks for all of the interfaces for their associated CPU (for example: VBUSM and VBUSP bridges, exception generation, debug and trace logic). CPU1 clocks are automatically gated in single-core mode.

The interface clock is an integer ratio of the CPU clock. The exact ratio for this device is 1:1.

7.2.2.4.3 R5FSS Reset

The R5FSS has four reset inputs:

- CPU0_RST: This is the reset for the non-debug logic of CPU0
- CPU0_DBG_RST: This resets the CPU0 debug logic, excluding the APB interface
- CPU1_RST: This is the reset for the non-debug logic of CPU1
- CPU1_DBG_RST: This resets the CPU1 debug logic, excluding the APB interface

In addition to the reset signals, there are two halt signals:

- CPU0_HALT
- CPU1_HALT

Copyright © 2026 Texas Instruments Incorporated

These halt signals keep the CPUs from fetching instructions when they come out of reset. The main use is to have the CPUs halted until the TCMs are loaded (when booting from TCM), though halt could be used for any other purpose.

7.2.2.5 R5FSS Vectored Interrupt Manager (VIM)

7.2.2.5.1 VIM Overview

The VIM aggregates device interrupts and sends them to the R5F CPU(s). It can be used in either split or single-core configuration.

The VIM module supports the following features:

- 256 interrupt inputs per R5F core
- Each interrupt has its own 4-bit programmable priority
  - Defined via the R5FSS_VIM_PRI_INT_j register
  - The VIM provides support for priority interruption of interrupts
- Each interrupt has its own enable mask
  - Interrupt enable is done via the R5FSS_VIM_INTR_EN_SET_j register
  - Interrupt disable is done via the R5FSS_VIM_INTR_EN_CLR_j register
- Each interrupt can be programmed as either an IRQ or FIQ
  - Defined via the R5FSS_VIM_INTMAP_j register
- Each interrupt has its own programmable 32-bit vector address associated with it
  - Defined via the R5FSS_VIM_VEC_INT_j register
  - Protected with SECDED
- One IRQn and one FIQn output per core
- Vectored interrupt interface
  - Compatible with R5F VIC port
- Default vector provided when a double-bit error is detected
- Split or single-core capable
  - In single-core mode, only interrupts connected to VIM interrupt core 0 are available
- Software interrupt generation

7.2.2.5.2 VIM Interrupt Inputs

The VIM supports 256 interrupt inputs per core. Each interrupt can be either a level or a pulse (both active-high). The interrupt mapping for the two R5F cores can be found in *Interrupt Sources*.

7.2.2.5.3 VIM Interrupt Outputs

The VIM has two interrupt outputs per core:

- *CoreN_IRQn*: This is a normal interrupt for core *N* (active-low level). It can be serviced via the VIC interface or through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (via the R5FSS_VIM_INTMAP_j register) and is enabled (via the R5FSS_VIM_INTR_EN_SET_j register), then it will cause an IRQ to assert
- *CoreN_FIQn*: This is a fast (or non-maskable) interrupt for core *N* (active-low level). FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ and is enabled, then it will cause an FIQ to assert

7.2.2.5.4 VIM Interrupt Vector Table (VIM RAM)

For each VIM interrupt core, there is an associated interrupt vector table (VIM RAM) that is used to store the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. Note that both interrupt vector tables are identical in their memory organization.

The VIM RAM is basically comprised of a set of interrupt vector registers (R5FSS_VIM_VEC_INT_j). Hence, the interrupt vector table is organized in 256 words of 30 bits, with a base address corresponding to the physical address of the first register in the group.

Copyright © 2026 Texas Instruments Incorporated

Note

The lower two bits of the 32-bit interrupt vector are always 0s.

Figure 7-3 shows the VIM RAM interrupt vector map.

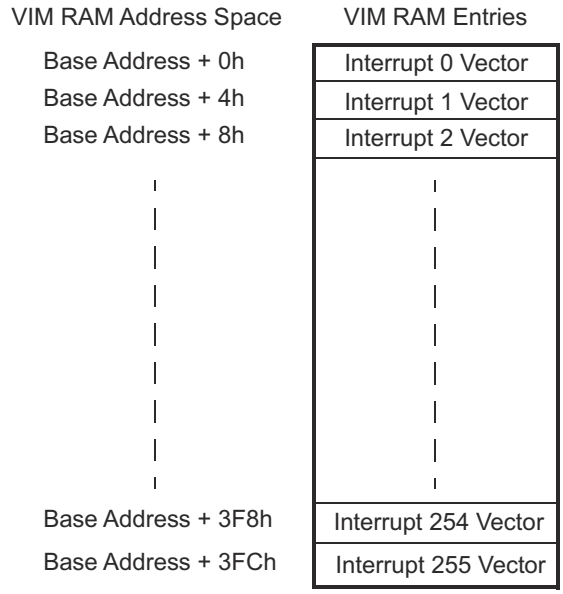


Figure 7-3. VIM RAM Interrupt Vector Map

The interrupt vector table has protection by ECC to indicate corruption due to soft errors. The ECC logic inside VIM supports SECDED. See Table 7-8 for the VIM RAM ID in the ECC aggregator map.

7.2.2.5.5 VIM Interrupt Prioritization

The VIM supports the interruption of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate but both use the same mechanism.

When an interrupt goes from pending to active (FIQ: reading the R5FSS_VIM_FIQVEC register; IRQ: reading the R5FSS_VIM_IRQVEC register, or the *coreN_IRQACK* going high), then the interrupt is loaded into the corresponding active register (R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ), and all interrupts of an equal or lesser priority are masked (discarded). If prior to this interrupt being cleared (by writing to the R5FSS_VIM_FIQVEC register, or R5FSS_VIM_IRQVEC register) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. If the CPU switches this interrupt to active (by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register), then the currently active interrupt will be pushed onto a stack. When an interrupt is cleared by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register, if there are any interrupts on the stack, the first entry is popped off and put back into the R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ register, so that software may continue where it left off.

7.2.2.5.6 VIM ECC Support

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the R5FSS_VIM_DEDVEC register is used to provide the default vector for the *coreN_IRQADDRV* signal, the R5FSS_VIM_IRQVEC register, and the R5FSS_VIM_FIQVEC register. The R5FSS_VIM_DEDVEC should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling.

Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
   a. Potentially switch to a completely software interrupt handler in the mean time

Copyright © 2026 Texas Instruments Incorporated

2. Restart the application from scratch
3. Reset the device
4. Sit in a loop (or WFI) while something external (for example, the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

Note

An interrupt that has an uncorrectable vector error (and thus uses the DED vector) will still have the priority of the original interrupt . This makes it possible for a higher priority interrupt to supercede the handling of the error.

Control and reporting are done by the R5FSS ECC aggregator.

7.2.2.5.7 VIM IDLE State

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface.

7.2.2.5.8 VIM Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM the software wants to take advantage of.

For IRQs, it is recommended to use the procedure in Section 7.2.2.5.8.1, but the procedures in Section 7.2.2.5.8.2 or Section 7.2.2.5.8.3 (if a user wants to implement a fully software prioritization scheme) may be used as alternatives.

For FIQs, it is recommended to use the procedure in Section 7.2.2.5.8.4, but the procedure in Section 7.2.2.5.8.5 may be used as an alternative.

Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

7.2.2.5.8.1 Servicing IRQ Through Vector Interface

If the associated CPU has the vector (VIC) interface enabled, then the following method is used for servicing IRQs:

1. Hardware handshake
   a. CPU asserts *coreN_IRQACK* high
   b. VIM asserts *coreN_IRQADDRV* to indicate that the *coreN_IRQADDR* bus is stable with the correct vector address
   c. CPU reads *coreN_IRQADDR*, jumps to that address, and de-asserts *coreN_IRQACK* low
   d. VIM de-asserts *coreN_IRQn* and *coreN_IRQADDRV*, VIM masks (discards) all IRQs with the same or lower priority
   e. VIM loads the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field (which corresponds to the vector address) into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, which causes the R5FSS_VIM_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTIRQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register

Copyright © 2026 Texas Instruments Incorporated

ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt

b. Level
   i. Clear the interrupt at the source
   ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt

4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.2.2.5.8.2 Servicing IRQ Through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the vector interface:

1. Read the R5FSS_VIM_IRQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_IRQn* output. If another interrupt of a higher priority becomes available, the *coreN_IRQn* will re-assert, allowing priority interruption of an interrupt
   b. Reading this register will cause the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, and the R5FSS_VIM_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
      ii. Clear the interrupt at the source

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register

4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.2.2.5.8.3 Servicing IRQ Through MMR Interface (Alternative)

If a user does not want to use the R5FSS_VIM_IRQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_IRQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_PRIIRQ register to determine which interrupt is the highest priority IRQ currently asserted, OR
   b. Optionally read the R5FSS_VIM_IRQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_IRQSTS_j register and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt

Copyright © 2026 Texas Instruments Incorporated

3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register

7.2.2.5.8.4 Servicing FIQ

When an FIQ interrupt is received, the CPU should follow these steps:

1. Read the R5FSS_VIM_FIQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_FIQn* output. If another interrupt of a higher priority becomes available, the *coreN_FIQn* will re-assert, allowing priority interruption of an interrupt.
   b. Reading this register will cause the value from the R5FSS_VIM_PRIFIQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_ACTFIQ[9:0] NUM bit field, and the R5FSS_VIM_ACTFIQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTFIQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
      ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
4. Write any value to the R5FSS_VIM_FIQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTFIQ[31] VALID bit

7.2.2.5.8.5 Servicing FIQ (Alternative)

If a user does not want to use the R5FSS_VIM_FIQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_FIQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_FIQVEC register to determine which interrupt is the highest priority FIQ currently asserted, OR

Copyright © 2026 Texas Instruments Incorporated

b. Optionally read the R5FSS_VIM_FIQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_FIQSTS_j register and use a software prioritization scheme to determine which FIQ to service

2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register.

7.2.2.6 R5FSS Region Address Translation (RAT)

The R5F is a 32-bit processor, which means it can only access 4GB of directly addressable memory. The R5FSS includes a region-based address translation (RAT) unit per core, which allows the R5F to access higher address ranges. The R5FSS RAT module translates a 32-bit input address into a 36-bit output address. It supports 4 regions. For more details on RAT functionality, refer to *Region-based Address Translation (RAT) Module*.

7.2.2.7 R5FSS ECC Support

The R5F provides native ECC and parity support on all related memories, generating and checking the redundancy automatically. The methods for checking and reporting errors are available in the *Arm Cortex-R5 Technical Reference Manual*.

The R5FSS adds the capability of testing this logic by allowing errors (single and double bit) to be injected into memories (for testing purposes) via an ECC aggregator (per core). Note that because the R5FSS ECC aggregator is only used in error-injection mode, it only supports a subset of the generic ECC aggregator functionality in the device.

For a detailed description of the generic ECC aggregator functionality, see *ECC Aggregator*. For register descriptions of R5FSS CPU0 and CPU1 ECC aggregators, see *R5FSS_CPU0_ECC_AGGR_CFG_REGS Registers* and *R5FSS_CPU1_ECC_AGGR_CFG_REGS Registers*, respectively.

Table 7-8 provides the RAM ID for each core. This is needed for bit field [10-0] ECC_VECTOR in the corresponding R5FSS_CPU0_VECTOR / R5FSS_CPU1_VECTOR register (part of the ECC aggregator register space).

Table 7-8. RAM ID Map for ECC Aggregator (Per Core)

RAM ID	Memory Name
0	CPU0/1 ITAG RAM0
1	CPU0/1 ITAG RAM1
2	CPU0/1 ITAG RAM2
3	CPU0/1 ITAG RAM3
4	CPU0/1 IDATA BANK0
5	CPU0/1 IDATA BANK1
6	CPU0/1 IDATA BANK2
7	CPU0/1 IDATA BANK3
8	CPU0/1 DTAG RAM0
9	CPU0/1 DTAG RAM1
10	CPU0/1 DTAG RAM2

Copyright © 2026 Texas Instruments Incorporated

Table 7-8. RAM ID Map for ECC Aggregator (Per Core) (continued)

RAM ID	Memory Name
11	CPU0/1 DTAG RAM3
12	CPU0/1 DDIRTY RAM
13	CPU0/1 DDATA RAM0
14	CPU0/1 DDATA RAM1
15	CPU0/1 DDATA RAM2
16	CPU0/1 DDATA RAM3
17	CPU0/1 DDATA RAM4
18	CPU0/1 DDATA RAM5
19	CPU0/1 DDATA RAM6
20	CPU0/1 DDATA RAM7
21	CPU0/1 ATCM BANK0
22	CPU0/1 ATCM BANK1
23	CPU0/1 B0TCM BANK0
24	CPU0/1 B0TCM BANK1
25	CPU0/1 B1TCM BANK0
26	CPU0/1 B1TCM BANK1
27	CPU0/1 VIM RAM

7.2.2.8 R5FSS Memory View

The memory view of each R5F (that is, the memory map as seen by each R5F) is a function of several things:

- Exception vector bootstrap: The R5F exception table (including boot vector) is always 32 bytes at address 0x00000000 as seen by the R5F. If not booting from a TCM, then boot is done over the main memory interface. The exception vector bootstrap is under software control, which allows these 32 bytes at address 0x00000000 to be remapped somewhere else in the SoC memory map.
- TCM locations: TCMs can be enabled or disabled and located at different places in the memory map, depending on bootstrap configuration. For more details, see Section 7.2.2.2.2.
- Peripheral interface locatons: The RF5 natively supports three interfaces for peripheral access. Each can be enabled/disabled and located based on bootstrap configuration. Note that the VBUSP peripheral interface must be enabled in order to use RAT and VIM.
- RAT base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- VIM base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- RAT programming: The RAT can take regions of memory accessible by the main memory interface and map them to different addresses.

The combination of the above determines what the R5F sees where in the memory map, and over what interface different transactions come out. Every transaction that does not directly address a TCM or a peripheral interface comes over the main memory interface. Transactions on the main memory interface can be further remapped with the RAT.

See *Memory Map*, for the complete R5F memory view for this device.

7.2.2.9 R5FSS Interrupts

All interrupts that are generated by the R5FSS are summarized in *R5FSS0/1 Hardware Requests*, along with their mapping. They can be divided into the following groups:

- R5F CPU internal interrupts: These are described in *Arm Cortex-R5 Technical Reference Manual*.
- ECC aggregator interrupts: These are described in the *ECC Aggregator* chapter.

Copyright © 2026 Texas Instruments Incorporated

- RAT exception interrupt.

Interrupt outputs: FPIXCm, FPUFCm, FPOFCm, FPDZCm, FPIDCm, and FPIOCm from R5F are not connected. Consequently, the only way to check the status of the cumulative exception status flags is by reading the FPSCR register.

7.2.2.10 R5FSS Debug and Trace

The R5FSS supports standard Arm CoreSight debug and trace architecture. For more details, see the *On-chip Debug* chapter.

7.2.2.11 R5FSS Boot Options

There are two methods of booting the R5F, or rather, two methods of placing the exception vectors (of which the boot vector is one).

The first method is to have the exception vectors external to the R5F. The user can place the exception vectors at the address indicated by the exception vector bootstrap and then program the boot vector there. When the processor exits reset, it will fetch the boot vector from this location.

The second method is to boot from a TCM. To do this, software should take the following steps:

1. Assert the correct bootstraps
   a. To boot from ATCM, set CPUn_INITRAMA (or CPUn_INITRAMB to boot from BTCM)
   b. Assert CPUn_LOCZRAMA properly for the desired TCM
2. Assert CPUn_HALT
3. Release the CPU from reset
4. Load the desired code into the TCM via the TCM target port
   a. Exception vectors should be located at address 0x00000000 of the TCM
5. De-assert CPUn_HALT

7.2.2.12 R5FSS Core Memory ECC Events

The R5F core generates several events as part of event bus that can be monitored by the PMU for debugging. The memory ECC related events from the event bus are exported to ESM for monitoring.

There are four ECC interrupts to the ESM that aggregate different categories of ECC events – CPU0 single error, CPU0 multi error, CPU1 single error, and CPU1 multi error events. Each ECC event has a 2-bit event bus counter associated with it. Everytime an event occurs, the counter is incremented by 1 till it reaches the max value of 3. The interrupt is asserted if the bus counter of any event associated with the interrupt is non-zero.

Each event bus counter has a MMR decrement control to decrement the counter by 1. So, for example, if a counter value is 2, the MMR to decrement the counter would need to be written 2 times to decrement the counter to 0. The reason decrement control has been added instead of clear control is if a new error occurs between the time the status register is read and the clear MMR is written, the new error would be lost. Write-to-decrement ensures that this does not happen.

When all event bus counters of an associated interrupt are zero, the interrupt is cleared. It takes three clock cycles for the event bus counter to be decremented once the write to the decrement control MMR presents itself at the R5FSS boundary.

Since each of the four ECC interrupts have single bit control to set the interrupt but multiple bits for clearing it, note that once an interrupt is set using the R5FSS_EVNT_BUS_ESM_SET register, it can be cleared by setting all the bits of the R5FSS_EVNT_BUS_ESM_CLR register that correspond to that particular interrupt. For example, if bit [0] of the R5FSS_EVNT_BUS_ESM_SET register is set, it can be cleared by setting bits [7-0] of the R5FSS_EVNT_BUS_ESM_CLR register to clear the interrupt.

The R5 core event bus only signals event when it is enabled. Non-invasive or invasive debug mode needs to be enabled to enable the PMU counters.

Copyright © 2026 Texas Instruments Incorporated

The export of the events to the event bus can be enabled by setting the X bit in the Performance Monitor Control Register of the R5 core. For more details, refer to Arm R5 TRM.

Table 7-9. R5 Event Bus Single-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
22	Instruction cache tag RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0
23	Instruction cache data RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1
24	Data cache tag or dirty RAM parity error or correctable ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2
25	Data cache data RAM parity error or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3
40	ATCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4
41	B0TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5
42	B1TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6
43	TCM correctable ECC error reported by load/ store unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7
44	TCM correctable ECC error reported by prefetch unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8

Table 7-10. R5 Event Bus Multi-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
26	TCM fatal ECC error reported from the prefetch unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0
27	TCM fatal ECC error reported from the load/ store unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1
33	Data cache data RAM fatal ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2
34	Data caches tag/dirty RAM fatal ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3
37	ATCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4
38	B0TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5
39	B1TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6

7.3 Cortex R5F Subsystem (R5FSS)

This chapter describes the Arm Cortex R5F real-time microcontroller unit subsystem (R5FSS) in the device.

7.3.1 R5FSS Overview

The R5FSS is a single-core implementation of the Arm® Cortex®-R5F processor that acts as the Device Manager responsible for boot, resource management, and power management functions. It also includes accompanying memories (L1 caches and tightly-coupled memories), standard Arm CoreSight™ debug and trace architecture, integrated vectored interrupt manager (VIM), ECC aggregators, and various other modules for protocol conversion and address translation for easy integration into the SoC.

Copyright © 2026 Texas Instruments Incorporated

Note

The Cortex-R5F processor is a Cortex-R5 processor that includes the optional floating point unit (FPU) extension. In this TRM, all references to the Cortex-R5 processor apply to the Cortex-R5F processor by default.

7.3.1.1 R5FSS Features

The R5FSS supports the following features:

- Single-core Arm Cortex-R5F
  - Core revision: r1p3
  - Armv7-R profile
  - L1 memory system
    - 32KB instruction cache
      - 4x8KB ways
      - SECDED ECC protected per 64 bits
    - 32KB data cache
      - 4x8KB ways
      - SECDED ECC protected per 32 bits
    - 64KB tightly-coupled memory (TCM)
      - SECDED ECC protected per 32 bits
      - TCM hard error cache Implemented in CPU
      - Readable/writable from system
      - TCMs initialized (to 0's) at reset
      - Split into A and B banks (with B further splitting into B0 and B1 interleaved banks)
        - 32KB TCMA (ATCM)
        - 16KB TCMB0 (B0TCM)
        - 16KB TCMB1 (B1TCM)
  - Low interrupt latency with restartable instructions
  - Non-maskable interrupt (NMI)
  - Full-precision floating point (VFPv3)
  - 16 region memory protection unit (MPU)
  - 8 breakpoints
  - 8 watchpoints
  - Dynamic branch prediction with global history buffer and 4-entry return stack
  - CoreSight debug access port (DAP)
  - CoreSight embedded trace macrocell (ETM-R5) interface
  - Performance monitoring unit (PMU)
- Interfaces
  - 64-bit VBUSM initiator pair (1 read, 1 write) for L3 memory accesses
  - 64-bit VBUSM target for TCM access
    - Also allows access to cache for debug purposes
  - 32-bit VBUSP initiator for peripheral access
  - 32-bit VBUSP target configuration port
  - 32-bit VBUSP target debug port
    - Allows access to all R5FSS internal debug logic
- Synchronous clock domain crossing on all interfaces
  - Interfaces can run at an integer multiple of the core frequency
- 32-bit to 36-bit region-based address translation (RAT) on memory access initiators
  - 4 regions
    - Base address + size
    - Must be size aligned

Copyright © 2026 Texas Instruments Incorporated

- Integrated vectored interrupt manager (VIM)
  - 256 interrupts
    - Each interrupt programmable as either IRQ or FIQ
    - Each interrupt has a programmable enable mask
    - Each interrupt has a programmable 4-bit priority
  - Priority interrupt supported
  - Vectored interrupt interface
    - Compatible with R5F VIC port
    - Programmable 32-bit vector address per interrupt
      - Address is SECDED error protected
      - Default vector addresses provided on DED
    - Software interrupt generation
- Integrated ECC aggregators
  - Support for error injection to all supported ECC memory blocks to test ECC functionality (add-on function from TI)
  - One ECC aggregator to cover all RAMs and caches
- Standard Arm CoreSight debug and trace architecture at the R5FSS level
  - Cross triggering: Supported by cross trigger interface (CTI) (per CPU) and cross trigger matrix (CTM) components
  - Processor trace: Supported by embedded trace macrocell (ETM) (per CPU) and advanced trace bus (ATB) funnel components
- Boot
  - From ROM only

See Section 7.3.2 for a functional block diagram and more details on the R5FSS.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

7.3.2 R5FSS Functional Description

7.3.2.1 R5FSS Block Diagram

Figure 7-4 shows the R5FSS block diagram.

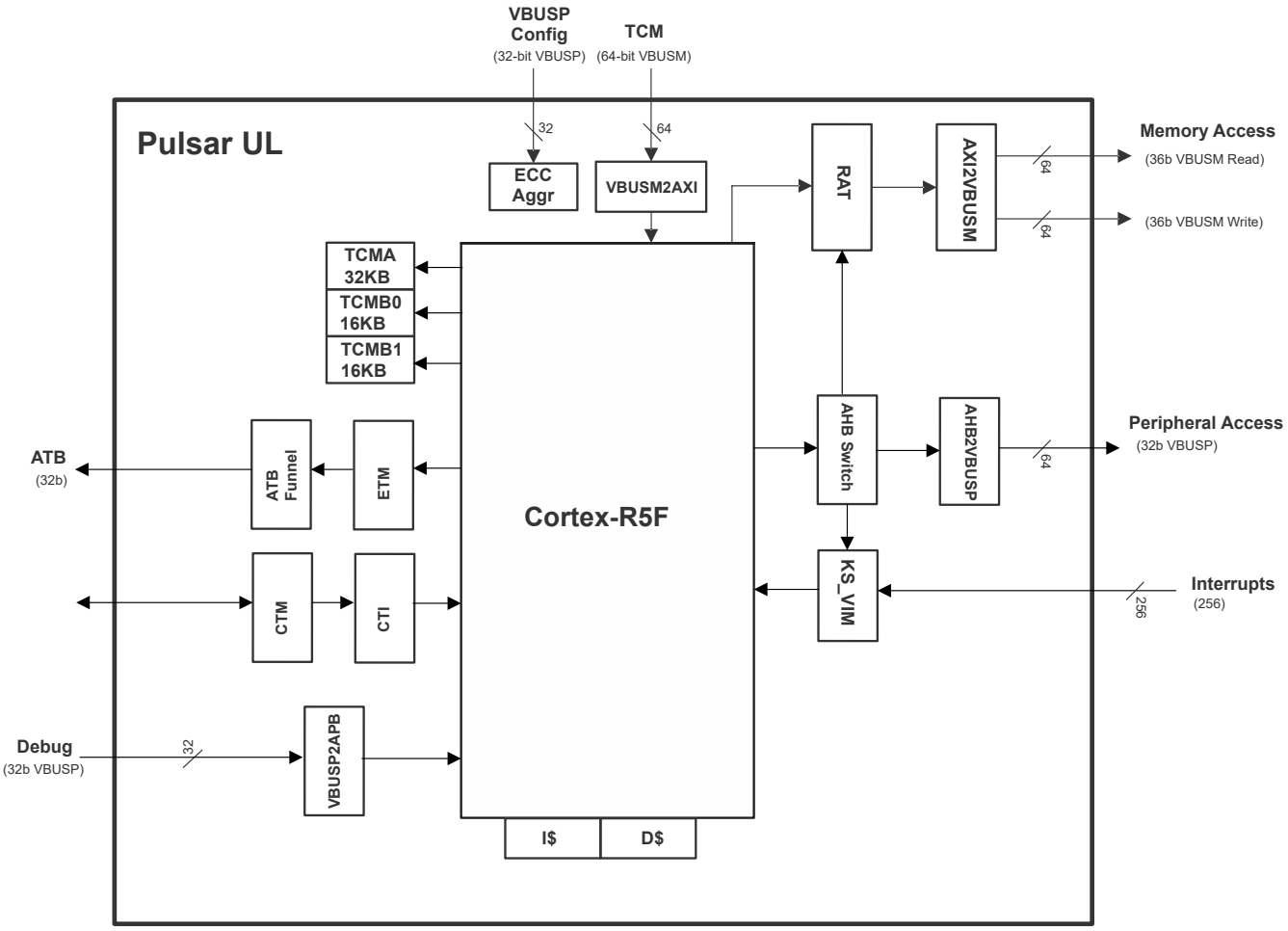


Figure 7-4. R5FSS Block Diagram

7.3.2.2 R5FSS Cortex-R5F Core

The Cortex-R5F is a processor from Arm, which is based on the Armv7-R profile.

For a brief list of features supported by the R5F processor in this device, see Section 7.3.1.1. For more detailed description of this processor, see the *Arm Cortex-R5 Technical Reference Manual*.

7.3.2.2.1 L1 Caches

The R5F has a Harvard cache architrecture, which means it has an independent L1 instruction cache (16KB) and L1 data cache (16KB). The instruction cache is protected by SECDED ECC per 64 bits. The data cache is protected by SECDED ECC per 32 bits.

7.3.2.2.2 Tightly-Coupled Memories (TCMs)

The R5F has two tightly-coupled memories (TCMs), ATCM and BTCM. The BTCM is further broken down into two interleaved banks, B0TCM and B1TCM.

TCMs are low-latency, tightly integrated memories for the R5F to use. Either TCM can be used for any combination of instruction and/or data. TCM performance is equal to performance on instructions/data that are in cache. However, TCMs have some additional advantages over cache. TCMs can be loaded with instructions

Copyright © 2026 Texas Instruments Incorporated

that do not cache well (such as ISRs) or preloaded with code by an external source, before that code is needed, to save cache miss time. TCMs are also a good place for blocks of data for intense processing. They can be loaded (or pre-loaded by an external source) before the data is needed, saving cache miss time. The data can then be directly accessed by an external source, instead of needing to do cache evicts.

As mentioned, TCMs can be accessed (either read or written) by an external source over the TCM VBUSM target interface. This allows instructions or data to be preloaded, or for data to be read out after the R5F has processed it. The VBUSM target has a lower priority to accessing TCMs than the R5F but care must be taken to keep an external source from reading or writing TCM data that the R5F is working on. This handshaking is external to any of the R5FSS hardware.

TCMs are protected by ECC per 32 bits. For this to work, ECC must be enabled before data is written in to the TCMs (either externally or from the R5F). ECC is enabled via the following R5F system control bits: ACTLR.ATCMPCEN, ACTLR.B0TCMPCEN, and ACTLR.B1TCMPCEN, respectively.

Whether or not the TCMs are enabled is controlled by the ENABLE bit in the corresponding ATCM/BTCM region register. The default (reset) value of this bit is determined by the CPUn_INITRAMA and CPUn_INITRAMB bootstraps, respectively. Both ATCM and BTCM are configured for a size of 32KB in this device. Note that the BTCM size is the total of both B0TCM and B1TCM (16KB each).

If a TCM is not enabled, then it does not appear in the R5F's memory view, but it can be accessed by an external source. If a TCM is enabled, then its place in the R5F memory map is determined by a combination of bootstrap signal and system register. If the CPUn_LOCZRAMA bootstrap signal is high, then the initial base address of ATCM is 0x0000_0000 and the initial address of BTCM is 20'h41010. If the CPUn_LOCZRAMA bootstrap signal is low, then the initial base address of BTCM is 0x0000_0000 and the initial base address of ATCM is 20'h41010.

Note

This base address of 0x41010 for ATCM/BTCM based on the CPUn_LOCZRAMA bootstrap only affects the R5F's memory view. The SoC will see the ATCM/BTCM based on the TCM target interface regions, as defined in Section 7.3.2.3.2. The base address of either TCM may be overwritten via the ATCM or BTCM region register. Care must be taken not to move the base address of a TCM when it may be being accessed.

It is possible to preload a TCM with instructions and boot from it. See Section 7.3.2.11 for details on TCM booting.

7.3.2.2.3 R5FSS Special Signals

Table 7-11 lists some R5FSS features associated with special signals.

Table 7-11. R5FSS Special Features

Feature	Comment
Cluster affinity group ID	R5F Cluster 0 (ID = 0x0)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode
CPUn execution halt when coming out of reset (CPUn_HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPUn exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPUn VIM base address	0x2FFF_0000
CPUn RAT base address	0x2FFE_0000
CPUn RAT accesses ID	0x4 (CPU0)
CPUn ATCM enable at reset (CPUn_INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state

Copyright © 2026 Texas Instruments Incorporated

Table 7-11. R5FSS Special Features (continued)

Feature	Comment
CPUn BTCM enable at reset (CPUn_INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPUn A/BTCM reset base address indicator (CPUn_LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPUn non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn VBUSM peripheral port enabled at reset	Enabled
CPUn VBUSP peripheral port enable at reset	Enabled
CPUn VBUSP peripheral port base address	Mapped to 0x0_2000_0000 for low latency MAIN peripherals
CPUn VBUSP peripheral port size	64MB for MAIN peripherals (0x0_2000_0000 to 0x0_2FFF_FFFF)
CPUn VBUSM normal peripheral port base address	Not used
CPUn VBUSM normal peripheral port size	Not used
CPUn VBUSM virtual peripheral port base address	Not used
CPUn VBUSM virtual peripheral port size	Not used
CPUn clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPUn WFI state	Status logged into MAIN_SEC_MMR register bit
CPUn WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

7.3.2.3 R5FSS Interfaces

7.3.2.3.1 Initiator Interfaces

The R5FSS has several initiator interfaces:

- 64-bit VBUSM initiator pair (1 read - "RMST", 1 write - "WMST") for L3 memory accesses; this is the main memory interface
  - Includes region-based address translation (RAT)
- 32-bit VBUSP initiator "PMST" for peripheral access
  - Includes logic that provides the R5F CPU with a private access to VIM and RAT
  - Enabled at reset

7.3.2.3.2 Target Interfaces

The R5FSS has several target interfaces that define its internal memory space:

- 32-bit VBUSP configuration target
  - Region [0]: ECC aggregator block
- 64-bit TCM target
  - Region [0]: ATCM
  - Region [1]: BTCM
  - Region [2]: Instruction cache RAMs
  - Region [3]: Data cache RAMs
- 32-bit VBUSP debug target
  - Provides access to all R5FSS internal debug logic

Regions [0] and [1] of the TCM target interface provide direct access to the TCM RAMs. Access to the RAMs is arbitrated with access from the R5F's L1 memory system. Excessive access while the R5F is also attempting access will degrade performance.

Regions [2] and [3] of the TCM target interface provide access to the cache RAMs for testing purposes. Access to the cache RAMs can only be done while the caches are disabled and should only be done for test purposes.

Copyright © 2026 Texas Instruments Incorporated

In addition to the target interfaces, there are peripherals (RAT and VIM) that are only accessible by the R5F. The R5F has an access to these modules via the VBUSP peripheral interface.

7.3.2.4 R5FSS Power, Clocking and Reset

7.3.2.4.1 R5FSS Power

The following R5FSS power considerations should be noted:

- R5FSS has a single power domain for all its internal logic

For more details on R5FSS power management, including power-up and power-down sequences, see *Power*.

7.3.2.4.2 R5FSS Clocking

The R5FSS has four clock inputs:

- CPU0_CLK: This is the clock for CPU0 logic
- CPU0_ICLK: This is the clock for CPU0 interfaces

CPU0_CLK is the clock for all of the internal CPU logic, while CPU0_ICLK is the clock for all of the interfaces for their associated CPU (for example: VBUSM and VBUSP bridges, exception generation, debug and trace logic).

The interface clock is an integer ratio of the CPU clock. The exact ratio for this device is 1:1.

7.3.2.4.3 R5FSS Reset

The R5FSS has two reset inputs:

- CPU0_RST: This is the reset for the non-debug logic of CPU0
- CPU0_DBG_RST: This resets the CPU0 debug logic, excluding the APB interface

In addition to the reset signals, there is one halt signal:

- CPU0_HALT

These halt signals keep the CPUs from fetching instructions when they come out of reset. The main use is to have the CPUs halted until the TCMs are loaded (when booting from TCM), though halt could be used for any other purpose.

7.3.2.5 R5FSS Vectored Interrupt Manager (VIM)

7.3.2.5.1 VIM Overview

The VIM aggregates device interrupts and sends them to the R5F CPU. It can be used in either split or single-core configuration.

The VIM module supports the following features:

- 256 interrupt inputs for the R5F core
- Each interrupt has its own 4-bit programmable priority
  - Defined via the R5FSS_VIM_PRI_INT_j register
  - The VIM provides support for priority interruption of interrupts
- Each interrupt has its own enable mask
  - Interrupt enable is done via the R5FSS_VIM_INTR_EN_SET_j register
  - Interrupt disable is done via the R5FSS_VIM_INTR_EN_CLR_j register
- Each interrupt can be programmed as either an IRQ or FIQ
  - Defined via the R5FSS_VIM_INTMAP_j register
- Each interrupt has its own programmable 32-bit vector address associated with it
  - Defined via the R5FSS_VIM_VEC_INT_j register
  - Protected with SECDED
- One IRQn and one FIQn output per core
- Vectored interrupt interface
  - Compatible with R5F VIC port
- Default vector provided when a double-bit error is detected

Copyright © 2026 Texas Instruments Incorporated

- Software interrupt generation

7.3.2.5.2 VIM Interrupt Inputs

The VIM supports 256 interrupt inputs per core. Each interrupt can be either a level or a pulse (both active-high). The interrupt mapping for the R5F core can be found in *Interrupt Sources*.

7.3.2.5.3 VIM Interrupt Outputs

The VIM has two interrupt outputs per core:

- *CoreN_IRQn*: This is a normal interrupt for core *N* (active-low level). It can be serviced via the VIC interface or through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (via the R5FSS_VIM_INTMAP_j register) and is enabled (via the R5FSS_VIM_INTR_EN_SET_j register), then it will cause an IRQ to assert
- *CoreN_FIQn*: This is a fast (or non-maskable) interrupt for core *N* (active-low level). FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ and is enabled, then it will cause an FIQ to assert

7.3.2.5.4 VIM Interrupt Vector Table (VIM RAM)

For each VIM interrupt core, there is an associated interrupt vector table (VIM RAM) that is used to store the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. Note that both interrupt vector tables are identical in their memory organization.

The VIM RAM is basically comprised of a set of interrupt vector registers (R5FSS_VIM_VEC_INT_j). Hence, the interrupt vector table is organized in 256 words of 30 bits, with a base address corresponding to the physical address of the first register in the group.

Note

The lower two bits of the 32-bit interrupt vector are always 0s.

Figure 7-5 shows the VIM RAM interrupt vector map.

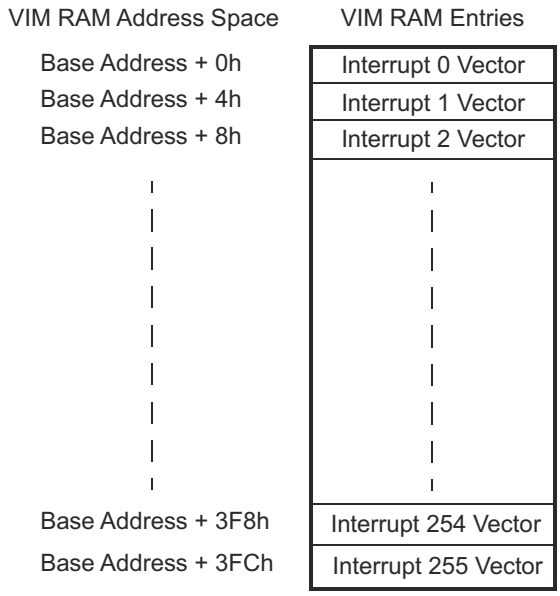


Figure 7-5. VIM RAM Interrupt Vector Map

The interrupt vector table has protection by ECC to indicate corruption due to soft errors. The ECC logic inside VIM supports SECDED. See Table 7-14 for the VIM RAM ID in the ECC aggregator map.

Copyright © 2026 Texas Instruments Incorporated

7.3.2.5.5 VIM Interrupt Prioritization

The VIM supports the interruption of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate but both use the same mechanism.

When an interrupt goes from pending to active (FIQ: reading the R5FSS_VIM_FIQVEC register; IRQ: reading the R5FSS_VIM_IRQVEC register, or the *coreN_IRQACK* going high), then the interrupt is loaded into the corresponding active register (R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ), and all interrupts of an equal or lesser priority are masked (discarded). If prior to this interrupt being cleared (by writing to the R5FSS_VIM_FIQVEC register, or R5FSS_VIM_IRQVEC register) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. If the CPU switches this interrupt to active (by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register), then the currently active interrupt will be pushed onto a stack. When an interrupt is cleared by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register, if there are any interrupts on the stack, the first entry is popped off and put back into the R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ register, so that software may continue where it left off.

7.3.2.5.6 VIM ECC Support

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the R5FSS_VIM_DEDVEC register is used to provide the default vector for the *coreN_IRQADDRV* signal, the R5FSS_VIM_IRQVEC register, and the R5FSS_VIM_FIQVEC register. The R5FSS_VIM_DEDVEC should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling.

Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
   a. Potentially switch to a completely software interrupt handler in the mean time
2. Restart the application from scratch
3. Reset the device
4. Sit in a loop (or WFI) while something external (for example, the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

Note

An interrupt that has an uncorrectable vector error (and thus uses the DED vector) will still have the priority of the original interrupt . This makes it possible for a higher priority interrupt to supercede the handling of the error.

Control and reporting are done by the R5FSS ECC aggregator.

7.3.2.5.7 VIM IDLE State

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface.

7.3.2.5.8 VIM Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM the software wants to take advantage of.

For IRQs, it is recommended to use the procedure in Section 7.3.2.5.8.1, but the procedures in Section 7.3.2.5.8.2 or Section 7.3.2.5.8.3 (if a user wants to implement a fully software prioritization scheme) may be used as alternatives.

For FIQs, it is recommended to use the procedure in Section 7.3.2.5.8.4, but the procedure in Section 7.3.2.5.8.5 may be used as an alternative.

Copyright © 2026 Texas Instruments Incorporated

Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

7.3.2.5.8.1 Servicing IRQ Through Vector Interface

If the associated CPU has the vector (VIC) interface enabled, then the following method is used for servicing IRQs:

1. Hardware handshake
   a. CPU asserts *coreN_IRQACK* high
   b. VIM asserts *coreN_IRQADDRV* to indicate that the *coreN_IRQADDR* bus is stable with the correct vector address
   c. CPU reads *coreN_IRQADDR*, jumps to that address, and de-asserts *coreN_IRQACK* low
   d. VIM de-asserts *coreN_IRQn* and *coreN_IRQADDRV*, VIM masks (discards) all IRQs with the same or lower priority
   e. VIM loads the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field (which corresponds to the vector address) into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, which causes the R5FSS_VIM_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTIRQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register
      ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.3.2.5.8.2 Servicing IRQ Through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the vector interface:

1. Read the R5FSS_VIM_IRQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_IRQn* output. If another interrupt of a higher priority becomes available, the *coreN_IRQn* will re-assert, allowing priority interruption of an interrupt
   b. Reading this register will cause the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, and the R5FSS_VIM_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse

Copyright © 2026 Texas Instruments Incorporated

i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
ii. Clear the interrupt at the source

b. Level
i. Clear the interrupt at the source
ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register

4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.3.2.5.8.3 Servicing IRQ Through MMR Interface (Alternative)

If a user does not want to use the R5FSS_VIM_IRQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_IRQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_PRIIRQ register to determine which interrupt is the highest priority IRQ currently asserted, OR
   b. Optionally read the R5FSS_VIM_IRQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_IRQSTS_j register and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register

7.3.2.5.8.4 Servicing FIQ

When an FIQ interrupt is received, the CPU should follow these steps:

1. Read the R5FSS_VIM_FIQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_FIQn* output. If another interrupt of a higher priority becomes available, the *coreN_FIQn* will re-assert, allowing priority interruption of an interrupt.
   b. Reading this register will cause the value from the R5FSS_VIM_PRIFIQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_PRIFIQ[9:0] NUM bit field, and the R5FSS_VIM_ACTFIQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTFIQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse

Copyright © 2026 Texas Instruments Incorporated

i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt

b. Level
i. Clear the interrupt at the source
ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt

4. Write any value to the R5FSS_VIM_FIQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTFIQ[31] VALID bit

7.3.2.5.8.5 Servicing FIQ (Alternative)

If a user does not want to use the R5FSS_VIM_FIQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_FIQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_FIQVEC register to determine which interrupt is the highest priority FIQ currently asserted, OR
   b. Optionally read the R5FSS_VIM_FIQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_FIQSTS_j register and use a software prioritization scheme to determine which FIQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register.

7.3.2.6 R5FSS Region Address Translation (RAT)

The R5F is a 32-bit processor, which means it can only access 4GB of directly addressable memory. The R5FSS includes a region-based address translation (RAT) unit per core, which allows the R5F to access higher address ranges. The R5FSS RAT module translates a 32-bit input address into a 36-bit output address. It supports 4 regions. For more details on RAT functionality, refer to *Region-based Address Translation (RAT) Module*.

7.3.2.6.1 WKUP R5FSS Usage

R5 is a micro controller with 32b address, which is only able to access up to 4GB address space. However, Sitara™ SoC supports 36b address and majority of the memory region beyond lower 4GB are implemented. Region based address translation block, called RAT, is introduced to allow R5 core to access full 36b SoC memory map.

Each R5 core has its own RAT block, and only R5 core itself is able to program the RAT.

Copyright © 2026 Texas Instruments Incorporated

Figure 7-6 shows the main R5's default 32b memory map view when main R5 is out of reset.

The full RAT functionality is described in Section 7.3.2.6.2, *RAT Function*.

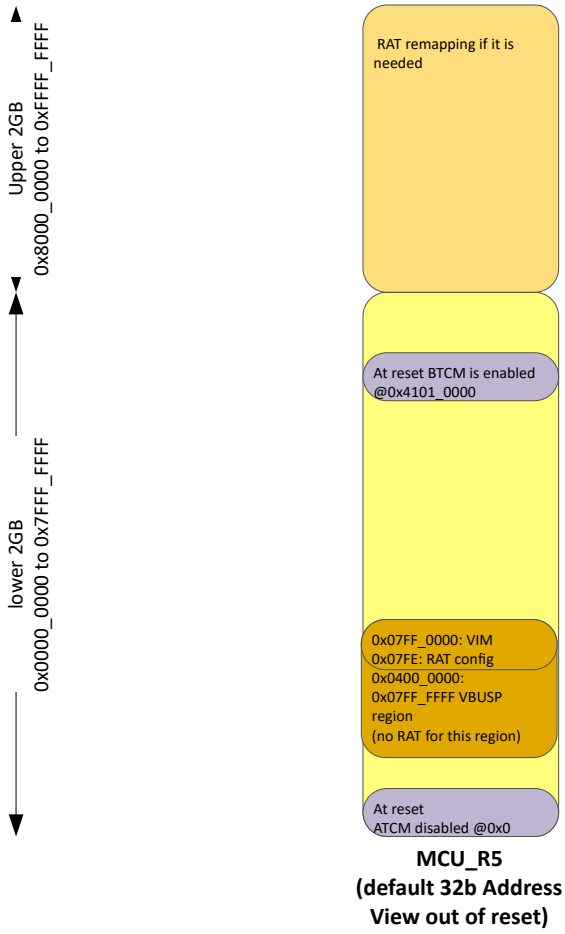


Figure 7-6. R5 Core Default Memory Map View

Copyright © 2026 Texas Instruments Incorporated

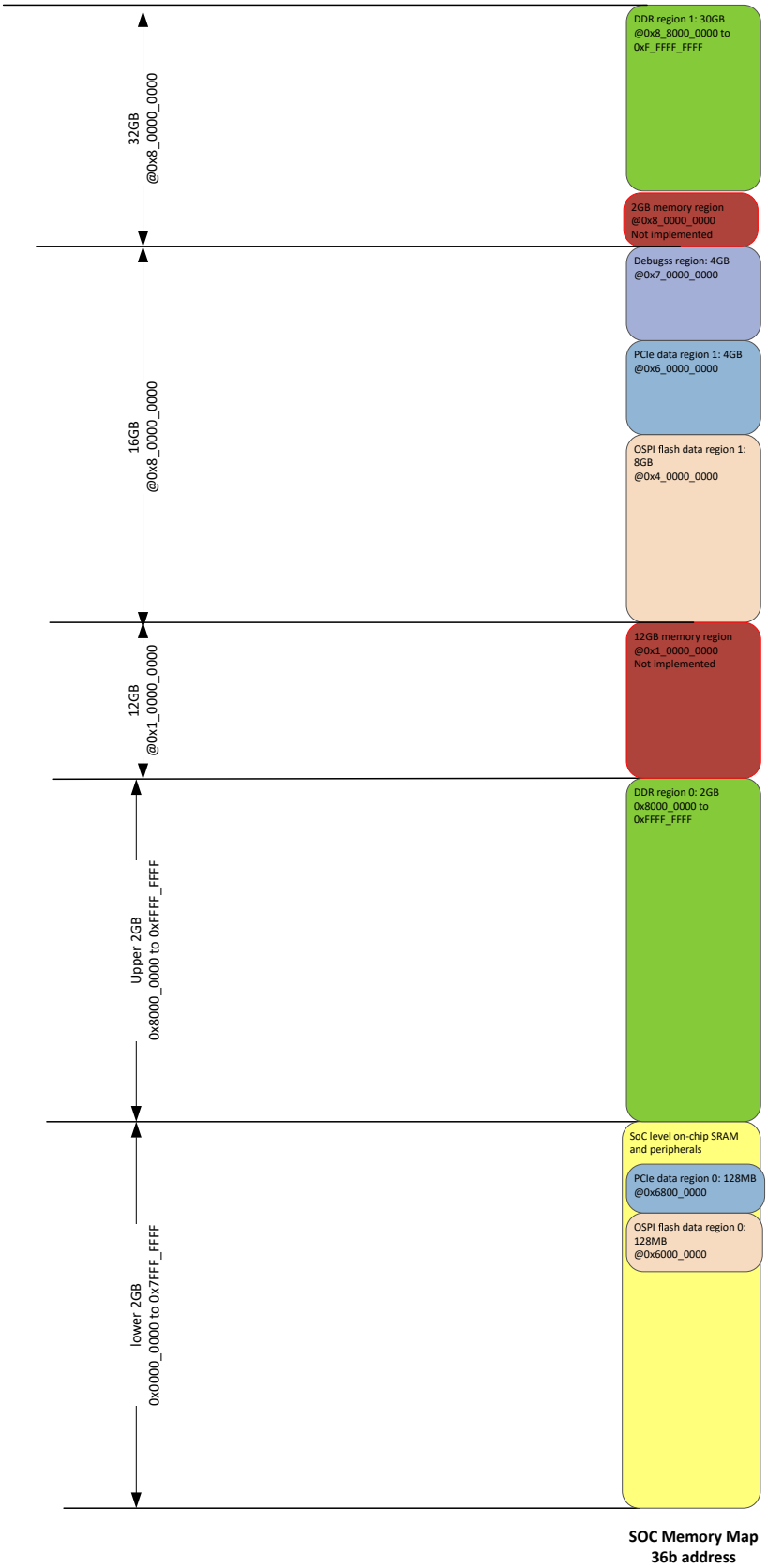


Figure 7-7. Typical Sitara SoC Memory Map View

Copyright © 2026 Texas Instruments Incorporated

RAT block enables R5 core to have full access on the SoC memory map including the memory region at and above 0x1_0000_0000.

Figure 7-8 shows how R5 access various end points, including its own ATCM, BTCM, VIM and SoC memory and peripherals. R5's ATCM is by default at address 0x0, and its BTCM is at address 0x4101_0000. R5 uses address 0x07FF_0000 to access its VIM and 0x07FE_0000 to access the RAT configuration region and all the other transactions between address range 0x0400_0000 to 0x07FF_FFFF are sent to the 32b peripheral interface to access the peripherals in SoC. The transactions with the rest of address are sent to RAT block, which could go through address remapping function from original R5's 32b address to 36b SoC address.

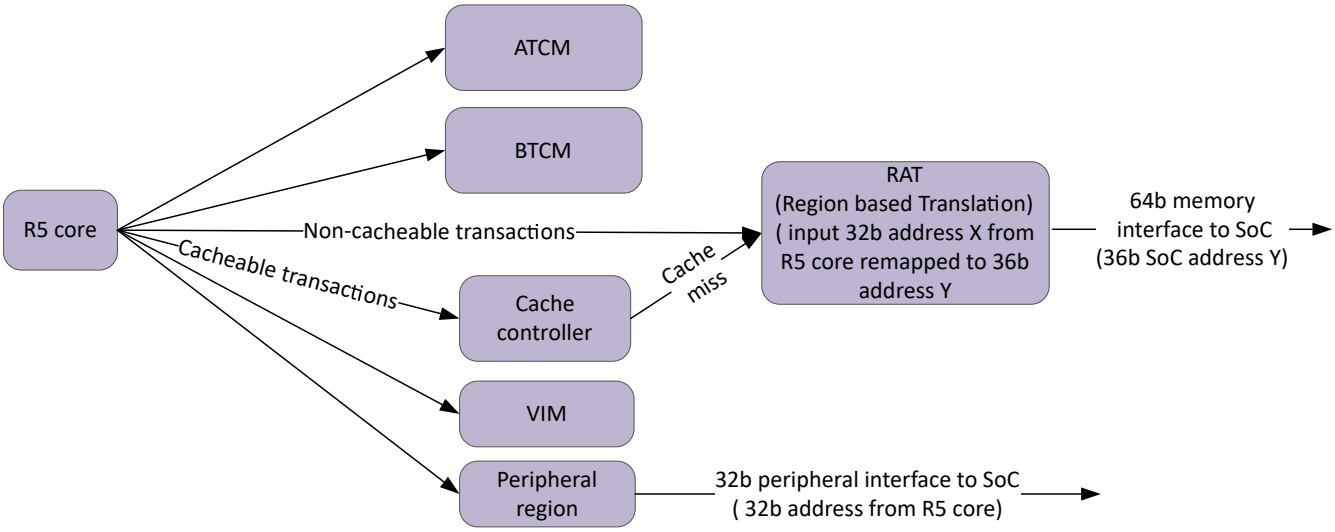


Figure 7-8. R5 Internal Address Decoding

7.3.2.6.2 RAT Function

RAT block is responsible to do a region based address translation. For each region, user can define the starting address and size of the region and remapping the transactions to use a new address range.

The starting address and region size need to be aligned. For example, if the region size is 16KB, the starting address for that region needs to be 16KB aligned. And region size should be minimum 4KB or larger to avoid transactions cross the region boundaries.

Each RAT block supports multiple programmable regions. And it is important that those regions are not overlapping with each other.

RAT block is by default disabled, which means RAT block does not do address remapping. The 32b address coming from R5 core is directly sent out to the SoC level. The user can program each RAT remapping region individually. For the transactions does not hit any remapping region, RAT block just simply forwards the transactions to the SoC level with its original address.

7.3.2.6.3 How to use RAT Block in R5

Table 7-12 shows the default R5 memory map view. Without enabling RAT function, all the regions beyond 4GB address space will not be accessible by R5.

Sitara™ SoC memory map is constructed the way to allow R5 and high level application processor such as A53 have similar memory map as possible. This methodology enables that R5 and ARM A53 core have a common memory map for all the SoC level memories and SoC level peripherals except a few exceptions:

- DDR's data space beyond 32b address, basically upper DDR data space beyond 2GB
- PCIe data space beyond 32b address
- OSPI data space beyond 32b address
- Debug configuration region

Copyright © 2026 Texas Instruments Incorporated

Table 7-12. R5 Memory map by Default

	R5 Memory Map(32b)	SoC Memory Map (36b)
SoC Level peripheral and on-chip SRAM	0x0 to 0x7FFF_FFFF Same address as SoC memory map	0x0 to 0x7FFF_FFFF
2GB DDR region	Same as SoC memory map: 0x8000_0000 to 0xFFFF_FFFF	0x80000_0000 to 0xFFFF_FFFF
Additional DDR region	Not accessible	0x8_8000_0000 (30GB)
Additional 8GB OSP Space	Not accessible	0x4_0000_0000 (4GB) 0x5_0000_0000 (4GB)
Additional PCIe data space	Not accessible	0x6_000_0000
Debug configuration region	Not accessible	0x7_0000_0000

Even though all the transactions other than going to ATCM/BTCM and transactions between address range 0x0400_0000 to 0x07FF_FFFF could use RAT block to remap to different address, it is recommended that only use RAT block to remap the R5 transactions with address between 0x8000_0000 to 0xFFFF_FFFF.

- Any transactions from R5 with address between 0x0 to 0x7FFF_FFFF are sent out to SoC directly (unless it is targeted to its internal end points such as A/B TCM, its VIM and RAT configuration)
- Only enable RAT to remap the transactions with address between 0x8000_0000 to 0xFFFF_FFFF if R5 needs to access the following address space:
  - PCIe data space at address 0x6_0000_0000
  - OSPI data space address 0x4_0000_0000 and 0x5_0000_0000
  - Debug configuration region at address 0x7_0000_0000
  - DDR data space at address 0x8_8000_0000

Copyright © 2026 Texas Instruments Incorporated

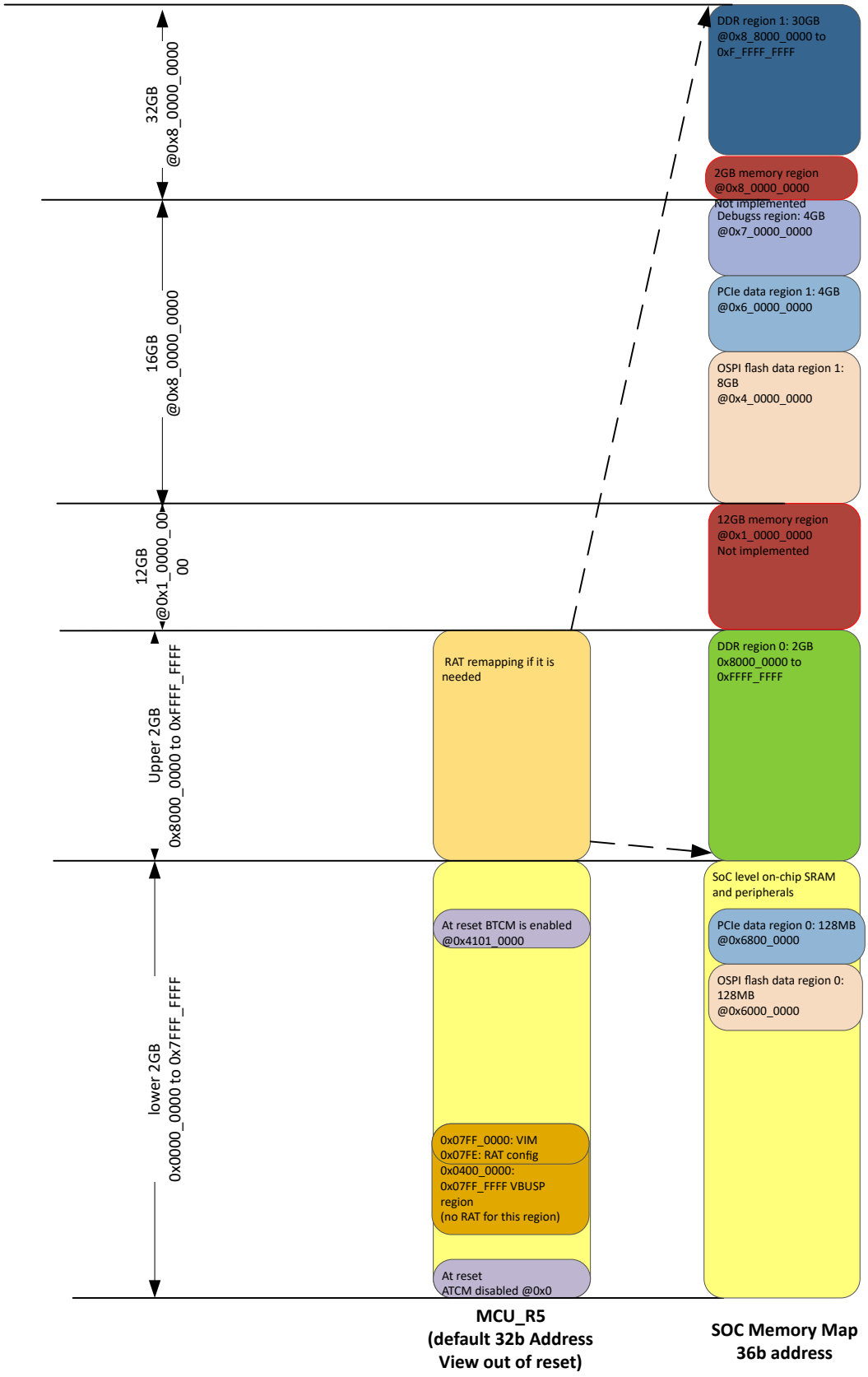


Figure 7-9. Example of Using RAT to Access Full 36b SoC Memory Map

Copyright © 2026 Texas Instruments Incorporated

7.3.2.6.4 Example of Using RAT to Access Full 36b SoC Memory Map

This section shows an example on how to utilize RAT block to allow R5 to access 36 SoC Memory Map. The RAT block inside R5 has up to 8 address remapping regions.

Assuming R5 needs to access the following region:

- All the SoC level on-chip SRAM
- All the SoC level peripherals
- Access 512MB OSPI region at address 0x4_0000_0000
- Access 1GB DDR region at address 0x9_0000_0000
- Access 512MB DDR region at address 0x8000_0000

First of all, we need to construct 32b R5 memory map as following and how those memory spaces are mapped to SoC level, shown in Table 7-13. Users can define where the OSPI and DDR data space placed inside R5's memory map and how they should be mapped to SoC level address map. Table 7-13 just shows one of the many possible ways to construct the memory map.

Table 7-13. Example of R5 Address Mapping

	R5 Memory Map(32b)	SoC Memory Map (36b)	RAT Remapping
SoC Level peripheral and on-chip SRAM (This is guaranteed by SoC Hardware design. No user configuration is needed)	0x0 to 0x7FFF_FFFF Same address as SoC memory map	0x0 to 0x7FFF_FFFF	Not RAT Remapping
512MB OSPI data region (user defined)	0x8000_0000 to 0x9FFF_FFFF	0x4_0000_0000 to 0x4_1FFF_FFFF	Using RAT remapping region 0
512MB DDR space (user defined)	0xA000_0000 to 0xBFFF_FFFF	0x8000_0000 to 0x9FFF_FFFF	Using RAT remapping region 1
Additional 1GB DDR Space (user defined)	0xC000_0000 to 0xFFFF_FFFF	0x9_0000_0000 to 0x9_3FFF_FFFF	Using RAT remapping region 2

In this example, only three RAT regions are needed:

- Region 0 is used to remap R5's address between 0x8000_0000 to 0x9FFF_FFFF to SoC level address 0x4_0000_0000 to 0x4_1FFF_FFFF
- Region 1 is used to remap R5's address between 0xA000_0000 to 0xBFFF_FFFF to 0x8000_0000 to 0x9FFF_FFFF
- Region 2 is used to remap R5's address 0xc000_0000 to 0xFFFF_FFFF to 0x9_0000_0000 to 0x9_3FFF_FFFF.

In order achieving those mappings, those are the setting of the RAT configuration registers:

	Region 0	Region 1	Region 2	Other regions
Region control register	Enabled, size set to 512MB	Enabled, size set to 512MB	Enabled, size set to 1GB	disabled
Region base(32b)	0x8000_0000	0xA000_0000	0xc000_0000	Don't care
Region translated lower address(32b)	0x0000_0000	0x8000_0000	0x0000_0000	Don't care
Region translated upper address	0x4	0x0	0x9	Don't care

Figure 7-10 shows how RAT is remapping the address between R5's 32b address to 36b SoC address.

Copyright © 2026 Texas Instruments Incorporated

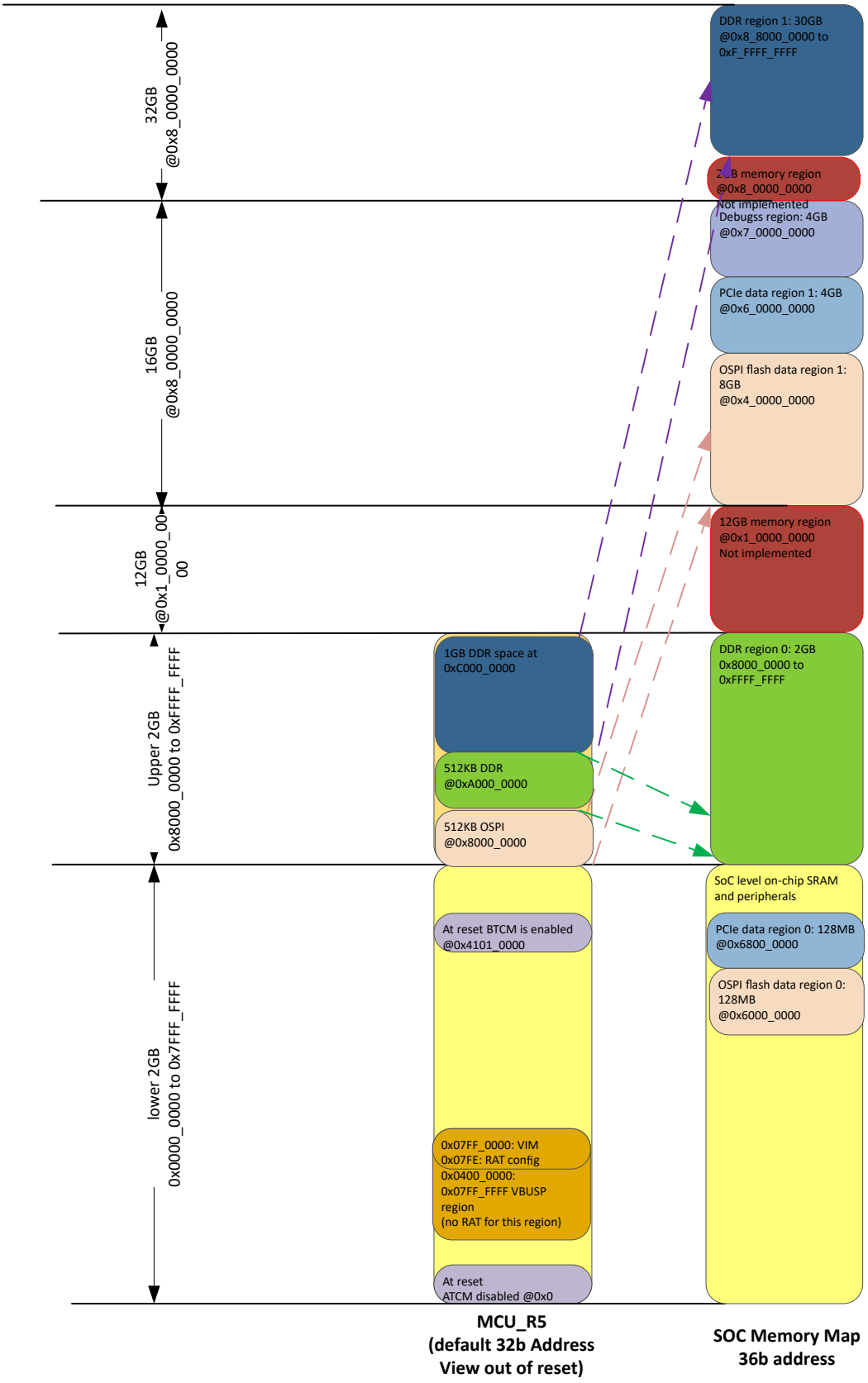


Figure 7-10. RAT Configuration Example

Copyright © 2026 Texas Instruments Incorporated

7.3.2.7 R5FSS ECC Support

The R5F provides native ECC and parity support on all related memories, generating and checking the redundancy automatically. The methods for checking and reporting errors are available in the *Arm Cortex-R5 Technical Reference Manual*.

The R5FSS adds the capability of testing this logic by allowing errors (single and double bit) to be injected into memories (for testing purposes) via an ECC aggregator (per core). Note that because the R5FSS ECC aggregator is only used in error-injection mode, it only supports a subset of the generic ECC aggregator functionality in the device.

For a detailed description of the generic ECC aggregator functionality, see *ECC Aggregator*. For register descriptions of the R5FSS aggregators, see R5FSS_CPU0_ECC_AGGR_CFG_REGS Registers and R5FSS_CPU1_ECC_AGGR_CFG_REGS Registers, respectively.

Table 7-14 provides the RAM ID. This is needed for bit field [10-0] ECC_VECTOR in the corresponding R5FSS_CPU0_VECTOR register (part of the ECC aggregator register space).

Table 7-14. RAM ID Map for ECC Aggregator (Per Core)

RAM ID	Memory Name
0	CPU0 ITAG RAM0
1	CPU0 ITAG RAM1
2	CPU0 ITAG RAM2
3	CPU0 ITAG RAM3
4	CPU0 IDATA BANK0
5	CPU0 IDATA BANK1
6	CPU0 IDATA BANK2
7	CPU0 IDATA BANK3
8	CPU0 DTAG RAM0
9	CPU0 DTAG RAM1
10	CPU0 DTAG RAM2
11	CPU0 DTAG RAM3
12	CPU0 DDIRTY RAM
13	CPU0 DDATA RAM0
14	CPU0 DDATA RAM1
15	CPU0 DDATA RAM2
16	CPU0 DDATA RAM3
17	CPU0 DDATA RAM4
18	CPU0 DDATA RAM5
19	CPU0 DDATA RAM6
20	CPU0 DDATA RAM7
21	CPU0 ATCM BANK0
22	CPU0 ATCM BANK1
23	CPU0 B0TCM BANK0
24	CPU0 B0TCM BANK1
25	CPU0 B1TCM BANK0
26	CPU0 B1TCM BANK1
27	CPU0 VIM RAM

Copyright © 2026 Texas Instruments Incorporated

7.3.2.8 R5FSS Memory View

The memory view of the R5F (that is, the memory map as seen by each R5F) is a function of several things:

- Exception vector bootstrap: The R5F exception table (including boot vector) is always 32 bytes at address 0x00000000 as seen by the R5F. If not booting from a TCM, then boot is done over the main memory interface. The exception vector bootstrap is under software control, which allows these 32 bytes at address 0x00000000 to be remapped somewhere else in the SoC memory map.
- TCM locations: TCMs can be enabled or disabled and located at different places in the memory map, depending on bootstrap configuration. For more details, see Section 7.3.2.2.2.
- Peripheral interface locations: The RF5 natively supports three interfaces for peripheral access. Each can be enabled/disabled and located based on bootstrap configuration. Note that the VBUSP peripheral interface must be enabled in order to use RAT and VIM.
- RAT base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- VIM base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- RAT programming: The RAT can take regions of memory accessible by the main memory interface and map them to different addresses.

The combination of the above determines what the R5F sees where in the memory map, and over what interface different transactions come out. Every transaction that does not directly address a TCM or a peripheral interface comes over the main memory interface. Transactions on the main memory interface can be further remapped with the RAT.

See *Memory Map*, for the complete R5F memory view for this device.

7.3.2.9 R5FSS Interrupts

All interrupts that are generated by the R5FSS are summarized in *Module Integration*, along with their mapping. They can be divided into the following groups:

- R5F CPU internal interrupts: These are described in *Arm Cortex-R5 Technical Reference Manual*.
- ECC aggregator interrupts: These are described in the *ECC Aggregator* chapter.
- RAT exception interrupt: This is described in Section 7.3.2.6.

7.3.2.10 R5FSS Debug and Trace

The R5FSS supports standard Arm CoreSight debug and trace architecture. For more details, see the *On-chip Debug* chapter.

7.3.2.11 R5FSS Boot Options

There are two methods of booting the R5F, or rather, two methods of placing the exception vectors (of which the boot vector is one).

The first method is to have the exception vectors external to the R5F. The user can place the exception vectors at the address indicated by the exception vector bootstrap and then program the boot vector there. When the processor exits reset, it will fetch the boot vector from this location.

The second method is to boot from a TCM. To do this, software should take the following steps:

1. Assert the correct bootstraps
   a. To boot from ATCM, set CPUn_INITRAMA (or CPUn_INITRAMB to boot from BTCM)
   b. Assert CPUn_LOCZRAMA properly for the desired TCM
2. Assert CPUn_HALT
3. Release the CPU from reset
4. Load the desired code into the TCM via the TCM target port
   a. Exception vectors should be located at address 0x00000000 of the TCM
5. De-assert CPUn_HALT

Copyright © 2026 Texas Instruments Incorporated

7.3.2.12 R5FSS Core Memory ECC Events

The R5F core generates several events as part of event bus that can be monitored by the PMU for debugging. The memory ECC related events from the event bus are exported to ESM for monitoring.

There are two ECC interrupts to the ESM that aggregate different categories of ECC events – CPU single error, CPU multi error. Each ECC event has a 2-bit event bus counter associated with it. Everytime an event occurs, the counter is incremented by 1 till it reaches the max value of 3. The interrupt is asserted if the bus counter of any event associated with the interrupt is non-zero.

Each event bus counter has a MMR decrement control to decrement the counter by 1. So, for example, if a counter value is 2, the MMR to decrement the counter would need to be written 2 times to decrement the counter to 0. The reason decrement control has been added instead of clear control is if a new error occurs between the time the status register is read and the clear MMR is written, the new error would be lost. Write-to-decrement ensures that this does not happen.

When all event bus counters of an associated interrupt are zero, the interrupt is cleared. It takes three clock cycles for the event bus counter to be decremented once the write to the decrement control MMR presents itself at the R5FSS boundary.

Since each of the four ECC interrupts have single bit control to set the interrupt but multiple bits for clearing it, note that once an interrupt is set using the R5FSS_EVNT_BUS_ESM_SET register, it can be cleared by setting all the bits of the R5FSS_EVNT_BUS_ESM_CLR register that correspond to that particular interrupt. For example, if bit [0] of the R5FSS_EVNT_BUS_ESM_SET register is set, it can be cleared by setting bits [7-0] of the R5FSS_EVNT_BUS_ESM_CLR register to clear the interrupt.

The R5F core event bus only signals event when it is enabled. Non-invasive or invasive debug mode needs to be enabled to enable the PMU counters.

The export of the events to the event bus can be enabled by setting the X bit in the Performance Monitor Control Register of the R5F core. For more details, refer to Arm R5 TRM.

Table 7-15. R5F Event Bus Single-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
22	Instruction cache tag RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0
23	Instruction cache data RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1
24	Data cache tag or dirty RAM parity error or correctable ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2
25	Data cache data RAM parity error or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3
40	ATCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4
41	B0TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5
42	B1TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6
43	TCM correctable ECC error reported by load/ store unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7
44	TCM correctable ECC error reported by prefetch unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8

Table 7-16. R5F Event Bus Multi-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
26	TCM fatal ECC error reported from the prefetch unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0

Copyright © 2026 Texas Instruments Incorporated

Table 7-16. R5F Event Bus Multi-Bit Error Events (continued)

Event Bus Bit #	Description	Associated Status Register
27	TCM fatal ECC error reported from the load/ store unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1
33	Data cache data RAM fatal ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2
34	Data caches tag/dirty RAM fatal ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3
37	ATCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4
38	B0TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5
39	B1TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6

7.3.3 Vectored Interrupt Manager (VIM)

This section describes the VIM module in the device.

7.3.3.1 VIM Overview

The Vectored Interrupt Manager (VIM) aggregates interrupts to a CPU. It is intended for use with a Cortex R5 from ARM. The VIM has up to 1024 interrupt inputs, which may be either level or pulse. Each interrupt has a programmable priority (0-highestthrough 15-lowest). Each interrupt may also be mapped as an IRQ or FIQ (FIQ is also often denoted as Non-Maskable Interrupt, or NMI).

7.3.3.1.1 VIM Features

- 32-1024 Interrupt inputs
  - Configurable by groups of 32
- Each interrupt has its own 4-bit programmable priority
  - Supports Priority interruption of interrupts
- Each interrupt has its own enable mask
- Each interrupt can be programmed as either an IRQ or FIQ
- Each interrupt has its own programmable 32-bit Vector address associated with it
  - Protected with SECDED
- One IRQn and FIQn output
- Vectored Interrupt Interface
  - Compatible with ARM Cortex R5 VIC Port
- Default Vector provided when a Double-Bit error is detected
- Provides clock phase inputs so VIM can run at CPU clock while interface runs at a different, synchronous clock

7.3.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

7.3.3.2 VIM Functional Description

7.3.3.2.1 Block Diagram

Figure 7-11 shows the Vectored Interrupt Manager block diagram.

Copyright © 2026 Texas Instruments Incorporated

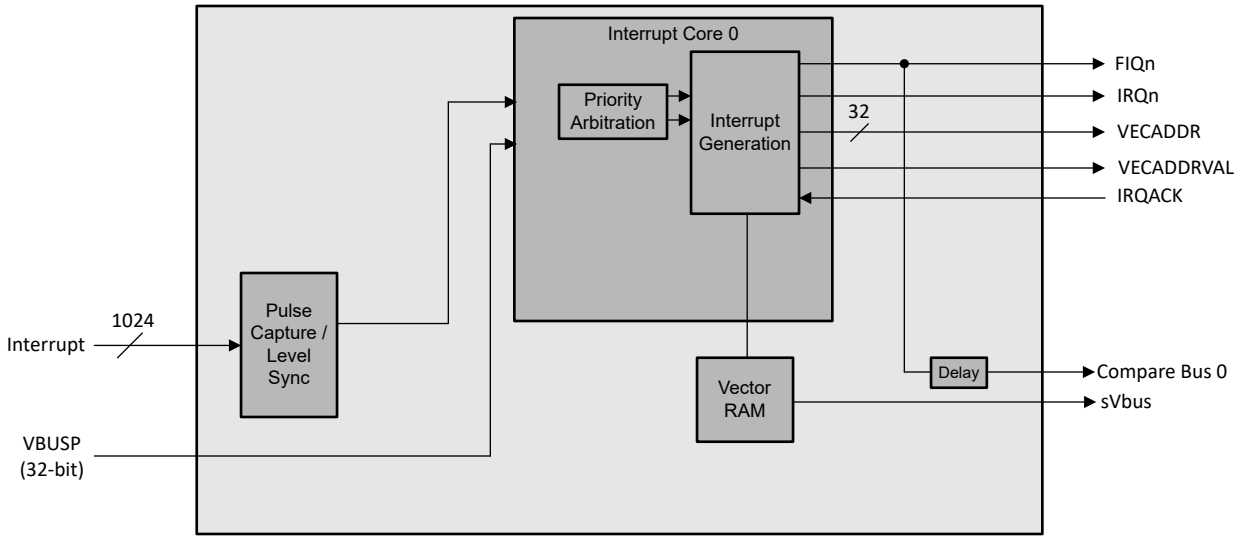


Figure 7-11. VIM Block Diagram

7.3.3.2.2 Interrupt Inputs

The VIM can have up to 1024 interrupt inputs, build-time configurable by multiples of 32 (*num_interrupts* parameter). Each interrupt can be either a level or a pulse (both active high).

Level interrupts are synchronized to the VIM clock. This synchronized value is captured in to a flop.

Pulse interrupts use rising edge detection. Each input has its own edge detection circuit. It is recommended that if pulses are pipelined between the source and the VIM, that the pipe stage replicates the pipe flop and ORs the output in order to protect against transient errors in the pipeline flop. Once an edge has been detected, the raw status is set.

There is no minimum pulse width, as true edge detection is used. The input signal should, however, be glitch free.

7.3.3.2.3 Interrupt Outputs

The VIM has two interrupt outputs, core*N*_IRQn and core*N*_FIQn (active low levels). Each interrupt input for core *N* can be programmed to influence either the core*N*_IRQn or core*N*_FIQn output via the Group *M* Interrupt Map Register (Base Address + 0x200 + *N**0x20 + 0x18).

7.3.3.2.4 Priority Interrupt / Nested Interrupts

Each interrupt has a priority number assigned to it (set using the 4.1.20 Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q**0x4) register). Legal values are 0 to 15 where 0 is the highest priority and 15 is the lowest priority. The highest priority interrupt is the pending interrupt with the smallest priority number. If two pending interrupts have the same priority, the interrupt lowest numerically (0 through maximum number of interrupts) is prioritized. IRQs and FIQs are prioritized separately.

The VIM supports the interrupt of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate, but both use the same mechanism. When an interrupt goes from pending to active (FIQ: reading the FIQ Vector Address (Base Address + 0x1C), IRQ: reading the IRQ Vector Address (Base Address + 0x18) or the core*N*_IRQACK going high), then the interrupt is loaded into the corresponding Active Register, and all interrupts of an equal or lesser priority are masked off (see note below). If, before this interrupt is cleared (writing the FIQ Vector Address (Base Address + 0x1C) or IRQ Vector Address (Base Address + 0x18)) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. The CPU may or may not service the higher priority interrupt. If the CPU switches this interrupt to active, by reading the corresponding Vector Address Register (or core*N*_IRQACK going high for an IRQ) , then the currently active interrupt will be pushed on to a stack. When an interrupt is cleared by writing the Vector Address Register, if there are any interrupts on the stack, the first entry is popped off and put back into the Active

Copyright © 2026 Texas Instruments Incorporated

Register, so that software may continue where it left off. Note that the IRQVEC/FIQVEC address registers are *not* repopulated with the old vector as it's assumed that the ISR is picking back up where it left off. Only the interrupt number and priority are restored to the ACTIRQ/ACTFIQ registers. If software needs the vector again, it will have to read it by using the interrupt number.

Note

"Masked off" means that they are masked off from priority arbitration to interrupt the currently active interrupt, it does NOT mean that the status bits in the regsiters are masked off. That is, this priority masking has NO EFFECT on whether the status bits are visible in the masked registers such as the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04).

7.3.3.2.5 VIC Port

The VIM is designed to work with the VIC interface of an ARM Cortex R5.

7.3.3.2.6 Latency

There are several factors that go in to the speed of processing an interrupt: the interrupt controller, the processor, and the system latency.

The VIM behaves deterministically (except for synchronizer delay). Table 7-17 shows the latency of the VIM.

Table 7-17. VIM Latency

Step	Cycles	Note
Edge Detection	2-3	Synchronizing to VIM Clock
Interrupt Capture	1	
Prioritization and Vector Read	1	IRQ Could be stalled here if an FIQ is reading the Vector RAM
		IRQ/FIQ output asserted

The next factor is the processor itself

Table 7-18. Processor Interrupt Latency

Step	Cycles(1)	Note
Take Exception		
Store State		Stack system registers etc
Read Vector	1	Through MMR or VIC interface (for IRQ. VIC interface will be faster)
Jump to Vector		

(1) These steps are dependent on the processor. See R5 spec for details.

The final factor is system latency to access the actually instructions for the ISR. This will be dependent on where the instructions are (cache, TCM, system memory etc) and dependent on the system latency to reach those memories (1 cycle for cache/TCM, System dependent for external memory)

7.3.3.2.7 Safety

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the DED Vector Address (Base Address + 0x30) is used instead for the core*N*_IRQADDRV, IRQ Vector Address (Base Address + 0x18), and FIQ Vector Address (Base Address + 0x1C). The DED Vector Address should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling. Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
   a. Potentially switch to a completely software interrupt handler in the mean time
2. Restart the application from scratch
3. Reset the device

Copyright © 2026 Texas Instruments Incorporated

4. Sit in a loop (or WFI) while something external (say the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

An interrupt that has an uncorrectable vector error and thus uses the DED Vector, will still have the priority of the original interrupt (that is for masking purposes). This makes it possible for a higher priority interrupt to supercede the handling of the error.

Control and reporting are done by an external ECC aggregator through sVbus ports.

7.3.3.2.8 IDLE

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface because it does not have a clock IPG. To clock stop the sub-system that includes the VIM, disable (mask) all interrupts and wait for the IDLE signal to assert.

7.3.3.3 Interrupt Conditions

7.3.3.3.1 CPU Interrupts

See Table 7-19 for the interrupts generated by the module.

Table 7-19. Interrupts

Interrupt	Description
core0_IRQn	IRQ Interrupt (Active Low)
core0_FIQn	FIQ Interrupt (Active Low)

7.3.3.3.2 Interrupt Description

7.3.3.3.2.1 coreN_IRQn

This is a normal interrupt, active low level, for Core N. It can be serviced via the VIC interface or through the MMR interface.

7.3.3.3.2.2 coreN_FIQn

This is a fast (or non-maskable) interrupt, active low level, for Core N. FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface.

7.3.3.3.3 Interrupt Condition Control

7.3.3.3.3.1 coreN_IRQn

Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (Group *N* Interrupt Map Register (Base Address + 0x200 + N*0x20 + 0x18)) and is enabled (Group *N* Interrupt Enabled Set Register (Base Address + 0x200 + N*0x20 + 0x08)), and its priority is not masked (4.1.11 IRQ Priority Mask Register (Base Address + 0x28), and it is not masked because of nested interrupt priority masking (2.2.5 Priority Interrupt / Nested Interrupts), then it will cause an IRQ to assert.

7.3.3.3.3.2 coreN_FIQn

Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ (Group *N* Interrupt Map Register (Base Address + 0x200 + N*0x20 + 0x18)) and is enabled (Group *N* Interrupt Enabled Set Register (Base Address + 0x200 + N*0x20 + 0x08)), and its priority is not masked (4.1.12 FIQ Priority Mask Register (Base Address + 0x2C)), and it is not masked because of nested interrupt priority masking (2.2.5 Priority Interrupt / Nested Interrupts), then it will cause an FIQ to assert.

7.3.3.3.4 Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM software wants to take advantage of. For IRQs, it is recommended to use 3.4.1 IRQ through the Vector

Copyright © 2026 Texas Instruments Incorporated

Interface, but 3.4.2 or 3.4.3 (if a user wants to implement a fully software prioritization scheme) may be used as alternatives. For FIQs, it is recommended to use 3.4.4 FIQ, but 3.4.5 may be used as an alternative.

Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

7.3.3.3.4.1 IRQ through the Vector Interface

If the attached CPU has the Vector Interface enabled, then the following method is used for servicing IRQs

1. Hardware handshake
   a. CPU asserts coreN_IRQACK signal high
   b. VIM asserts coreN_IRQADDRV to indicate that the coreN_IRQADDR bus is stable with the correct Vector Address
   c. CPU reads the coreN_IRQADDR, jumps to that address, and de-asserts coreN_IRQACK signal low
   d. VIM de-asserts coreN_IRQn_intr and coreN_IRQADDRV, VIM masks all IRQs with the same or lower priority
   e. VIM loads the value from the Prioritized IRQ (Base Address + 0x08) (which corresponds to the vector address) to be loaded into the Active IRQ (Base Address + 0x20) and the valid bit to be set
2. Service an interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active IRQ (Base Address + 0x20) to determine number and reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group M Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group M Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group M Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group M Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. 4. Write any value to the IRQ Vector Address (Base Address + 0x18)
   a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
   b. This will clear the valid bit of the Active IRQ (Base Address + 0x20)

7.3.3.3.4.2 IRQ through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the Vector Interface.

1. Read the IRQ Vector Address (Base Address + 0x18) and jump to that address to service the ISR

Copyright © 2026 Texas Instruments Incorporated

a. Reading this register will mask all interrupts of an equal or lower priority and de-assert the IRQn output. If another interrupt of a higher priority becomes available, the IRQn will re-assert, allowing priority interruption of an interrupt.
b. Reading this register will cause the value from the Prioritized IRQ (Base Address + 0x08) (which corresponds to the vector address) to be loaded into the Active IRQ (Base Address + 0x20) and the valid bit to be set

2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active IRQ (Base Address + 0x20) to determine number and reading the Group M Type Map Register (Base Address + 0x200 + M*0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. Write any value to the IRQ Vector Address (Base Address + 0x18)
   a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
   b. This will clear the valid bit of the Active IRQ (Base Address + 0x20)

7.3.3.3.4.3 IRQ through MMR Interface (Alternative)

If a user does not want to use the Vector Address registers, the VIM may be used as a more traditional interrupt controller. Note that in this mode, priority masking will not work if route 1b is used (below) as the hardware prioritization may not match the software prioritization scheme. In this case, software would be responsible for doing all priority operations

1. Determine which interrupt to service
   a. Read the Prioritized IRQ (Base Address + 0x08) to determine which interrupt is the highest priority IRQ currently asserted OR
   b. Optionally read the IRQ Group Status (Base Address + 0x10) to determine which groups have IRQs pending, then read the Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10) and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt
3. Read the IRQ Vector Address (Base Address + 0x18)
   a. Note, this step can be done any time before step 4
   b. Value is ignored
4. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)

Copyright © 2026 Texas Instruments Incorporated

a. Pulse
   i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
   ii. Clear the interrupt at the source
      1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

b. Level
   i. Clear the interrupt at the source
   ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
      2. If the source maintains the level, then it means there is another interrupt

5. Write any value to the IRQ Vector Address (Base Address + 0x18)

7.3.3.3.4.4 FIQ

When an FIQ interrupt is received, the CPU should follow these steps.

1. Read the FIQ Vector Address (Base Address + 0x1C) and jump to that address to service the ISR
   a. Reading this register will mask all interrupts of an equal or lower priority and de-assert the FIQn output. If another interrupt of a higher priority becomes available, the FIQn will re-assert, allowing priority interruption of an interrupt.
   b. Reading this register will cause the value from the Prioritized FIQ (Base Address + 0x0C) (which corresponds to the vector address) to be loaded into the Active FIQ (Base Address + 0x24) and the valid bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active FIQ (Base Address + 0x24) to determine number and reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. Write any value to the FIQ Vector Address (Base Address + 0x1C)

Copyright © 2026 Texas Instruments Incorporated

a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
b. This will clear the valid bit of the Active FIQ (Base Address + 0x24)

7.3.3.3.4.5 FIQ (Alternative)

If a user does not want to use the Vector Address registers, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the FIQ Vector Address (Base Address + 0x1C) is never read). Software would be responsible for doing all priority operations

1. Determine which interrupt to service
   a. Read the Prioritized FIQ (Base Address + 0x0C) to determine which interrupt is the highest priority FIQ currently asserted OR
   b. Optionally read the FIQ Group Status (Base Address + 0x14) to determine which groups have IRQs pending, then read the Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14) and use a software prioritization scheme to determine which FIQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

7.3.3.4 Memory Map

Each Interrupt Core (*N* = 0) has its own MMR interface with its own independent memory map

Table 7-20. Core *N* Memory Map

Address Offset	Register
0x00	Revision Register
0x04	Info Register
0x08	Prioritized IRQ
0x0C	Prioritized FIQ
0x10	IRQ Group Status
0x14	FIQ Group Status
0x18	IRQ Vector Address
0x1C	FIQ Vector Address
0x20	Active IRQ
0x24	Active FIQ
0x28-0x2F	*Reserved*

Copyright © 2026 Texas Instruments Incorporated

Table 7-20. Core *N* Memory Map (continued)

Address Offset	Register
0x30	DED Vector Address
0x34-0x1FF	*Reserved*
0x400 + *M**0x20 + 0x00	Group *M* Interrupt Raw Status/Set Register
0x400 + *M**0x20 + 0x04	Group *M* Interrupt Enabled Status/Clear Register
0x400 + *M**0x20 + 0x08	Group *M* Interrupt Enabled Set Register
0x400 + *M**0x20 + 0x0C	Group *M* Interrupt Enabled Clear Register
0x400 + *M**0x20 + 0x10	Group *M* Interrupt IRQ Enabled Status/Clear Register
0x400 + *M**0x20 + 0x14	Group *M* Interrupt FIQ Enabled Status/Clear Register
0x400 + *M**0x20 + 0x18	Group *M* Interrupt Map Register
0x400 + *M**0x20 + 0x1C	Group *M* Type Map Register
0x1000 + *Q**0x4 – 00x1FFF	Interrupt *Q* Priority Register
0x2000 + *Q**0x4 – 0x2FFF	Interrupt *Q* Vector Register

There are *M* interrupt groups (0 through *num_groups*-1) with 32 interrupts per group.

There are *Q* interrupt inputs where *Q* = *M* * 32.

Accesses to the Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4) must be word-aligned, 32-bit accesses. Any write received with no bytes enabled will be ignored. Any writes receive with all bytes enabled will be executed. Any other write will not execute and will return an error status of Addressing Error.

7.3.3.5 Module I/O

7.3.3.5.1 Clocks, Reset, Emulation

Table 7-21. Clocks, Reset, Emulation

Pin Name	Type	Function
core0_clk	In	Clock for Core 0
core0_phase	In	Phase Indicator for Core 0
core0_srst_n	In	Sync Module Reset for Core 0
core0_arst_n	In	Async Module Reset for Core 0
core0_clkstop_idle	Out	Clock Stop Idle

7.3.3.5.2 VBUSP Target Interface

(N=0). Core0 interface is used for interrupt core 0.

Table 7-22. VBUSP Target Interface

Pin Name	Type	Function
core*N*_cfg_req	In	Request
core*N*_cfg_address[13:0]	In	Address (byte address)
core*N*_cfg_dir	In	Direction (read or write)
core*N*_cfg_xcnt[2:0]	In	Transaction Count
core*N*_cfg_byten[3:0]	In	Byte enables
core*N*_cfg_wdata[31:0]	In	Write data
core*N*_cfg_rdatap[31:0]	Out	Read data
core*N*_cfg_rready	Out	Read ready
core*N*_cfg_wready	Out	Write ready
core*N*_cfg_rstatus[2:0]	Out	Read Status

Copyright © 2026 Texas Instruments Incorporated

Table 7-22. VBUSP Target Interface (continued)

Pin Name	Type	Function
core*N*_cfg_emudbg	In	Emulation Debug

7.3.3.5.3 Interrupt Inputs

Core0 interface is used for interrupt core 0.

Table 7-23. Interrupt Inputs

Pin Name	Type	Function
core0_in_intr[*Q*-1:0]	In	Core 0 Interrupt Inputs

7.3.3.5.4 Interrupt Outputs

Table 7-24. Interrupt Outputs

Pin Name	Type	Function
core0_IRQn_intr	Out	IRQ Interrupt for core 0– Active Low Level
core0_FIQn_intr	Out	FIQ Interrupt for core 0 – Active Low Level

7.3.3.5.5 VIC Interfaces

(*N*=0). Core0 interface is used for interrupt core 0.

Table 7-25. VIC Interfaces

Pin Name	Type	Function
core*N*_vic_IRQADDRV	Out	IRQ Address Valid
core*N*_vic_IRQADDR[31:2]	Out	IRQ Address
core*N*_vic_IRQACK	In	Interrupt Acknowledge

7.3.3.5.6 Compare Outputs

These pins will never be on an isolation boundary and therefore their isolation values are don't-care. *Y* is dependent on the configuration

Table 7-26. Compare Outputs

Pin Name	Type	Function
core0_compare_bus[*Y*:0]	Out	Comparison Bus for Core0 to CCMR5

7.3.3.5.7 ECC Control and Status Bus

(*N*=0). Core0 interface is used for interrupts for core 0.

Table 7-27. ECC Control and Status Bus

Signal Name	Dir	Default Val	Description
core*N*_ramecc_strb	In	1'b0	Strobe to sample the incoming serial data
core*N*_ramecc_txd	In	1'b0	Output serial data
core*N*_ramecc_rxd	Out	1'b0	Input serial data
core*N*_ramecc_pend[1:0]	Out	2'd0	Level Interrupt source

7.3.3.5.8 DFT

Table 7-28. DFT

Signal Name	Dir	Default Val	Description
dft_clk_force_en	In	1'b0	DFT Clock Force Enable
dft_partition_enn	In	1'b0	DFT Clock Partition Enable
dft_scan_en	In	1'b0	DFT Scan Enable

Copyright © 2026 Texas Instruments Incorporated

Table 7-28. DFT (continued)

Signal Name	Dir	Default Val	Description
dft_async_rst_sel	In	1'b0	DFT Async Reset Select
dft_test_async_rst_n	In	1'b0	DFT Async Test Reset
dft_tft_mcp_dis	In	1'b0	DFT MCP Disable
dft_local_clk_en	In	1'b0	DFT Local Clock Enable

7.3.3.5.9 RAM GPIO

Table 7-29. RAM GPIO

Signal Name	Dir	Default Val	Description
core*N*_ramdft_gpi[255:0]	In	256'd0	RAM GPI
core*N*_ramdft_gpo[255:0]	Out	256'd0	RAM GPO

7.3.3.6 Programmer's Guide

7.3.3.6.1 Initialization Sequence

At initialization, the following tasks should be performed:

1. Program the DED Vector Address (Base Address + 0x30)
2. Program Group *M* Interrupt Map Register (Base Address + 0x200 + *M**0x20 + 0x18)
3. Program Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C)
4. Program Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q**0x4)
5. Program Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4)
6. Enable interrupts via the Group *M* Interrupt Enabled Set Register (Base Address + 0x400 + *M**0x20 + 0x08)

Note that the Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4) for each interrupt that will be enabled must be written before the interrupt is enabled, even if software is not going to use the vector. Whenever an interrupt is prioritized, the RAM location for the interrupt is read, and if that location is un-initialized, an ECC error will be reported. If the vectors aren't being used, then the RAM can be initialized by writing 0x0 to every location.

7.3.3.6.2 DED Behavior

Whenever there is 2-bit error detect on a Vector Address, the DED Vector Address (Base Address + 0x30) overrides all other vectors. Software should provide an ISR to handle this scenario at this address.

7.3.3.6.3 Power Up/Down Sequence

Before a clean power down, software should check that there are no pending interrupts, disable all interrupts, and wait for the clkstop_idle output.

7.4 Device Manager Cortex R5F Subsystem (WKUP_R5FSS)

This chapter describes the Arm Cortex R5F real-time microcontroller unit subsystem (WKUP_R5FSS) in the device.

7.4.1 WKUP_R5FSS Overview

The WKUP_R5FSS is a single-core implementation of the Arm® Cortex®-R5F processor that acts as the Device Manager responsible for boot, resource management, and power management functions. It also includes accompanying memories (L1 caches and tightly-coupled memories), standard Arm CoreSight™ debug and trace architecture, integrated vectored interrupt manager (VIM), ECC aggregators, and various other modules for protocol conversion and address translation for easy integration into the SoC.

Copyright © 2026 Texas Instruments Incorporated

Note

The Cortex-R5F processor is a Cortex-R5 processor that includes the optional floating point unit (FPU) extension. In this TRM, all references to the Cortex-R5 processor apply to the Cortex-R5F processor by default.

There is one WKUP_R5FSS subsystem in the device. Table 7-30 shows WKUP_R5FSS allocation across device domains.

Table 7-30. R5FSS Allocation Across Device Domains

Module Instance	Domain		
	MCU	MAIN	WKUP
WKUP_R5FSS	–	–	✓

7.4.1.1 WKUP_R5FSS Features

The WKUP_R5FSS supports the following features:

- Single-core Arm Cortex-R5F
  - Core revision: r1p3
  - Armv7-R profile
  - L1 memory system
    - 32KB instruction cache
      - 4x8KB ways
      - SECDED ECC protected per 64 bits
    - 32KB data cache
      - 4x8KB ways
      - SECDED ECC protected per 32 bits
    - 64KB tightly-coupled memory (TCM)
      - SECDED ECC protected per 32 bits
      - TCM hard error cache Implemented in CPU
      - Readable/writable from system
      - TCMs initialized (to 0's) at reset
      - Split into A and B banks (with B further splitting into B0 and B1 interleaved banks)
        - 32KB TCMA (ATCM)
        - 16KB TCMB0 (B0TCM)
        - 16KB TCMB1 (B1TCM)
  - Low interrupt latency with restartable instructions
  - Non-maskable interrupt (NMI)
  - Full-precision floating point (VFPv3)
  - 16 region memory protection unit (MPU)
  - 8 breakpoints
  - 8 watchpoints
  - Dynamic branch prediction with global history buffer and 4-entry return stack
  - CoreSight debug access port (DAP)
  - CoreSight embedded trace macrocell (ETM-R5) interface
  - Performance monitoring unit (PMU)
- Interfaces
  - 64-bit VBUSM initiator pair (1 read, 1 write) for L3 memory accesses
  - 64-bit VBUSM target for TCM access
    - Also allows access to cache for debug purposes
  - 32-bit VBUSP initiator for peripheral access
  - 32-bit VBUSP target configuration port
  - 32-bit VBUSP target debug port

Copyright © 2026 Texas Instruments Incorporated

        - Allows access to all WKUP_R5FSS internal debug logic
    - ECC/Parity support on all internal SoC bus interfaces
    - ECC/Parity support on internal SoC bus bridges
- Synchronous clock domain crossing on all interfaces
    - Interfaces can run at an integer multiple of the core frequency
- 32-bit to 36-bit region-based address translation (RAT) on memory access initiators
    - 4 regions
        - Base address + size
        - Must be size aligned
- Integrated vectored interrupt manager (VIM)
    - 256 interrupts
        - Each interrupt programmable as either IRQ or FIQ
        - Each interrupt has a programmable enable mask
        - Each interrupt has a programmable 4-bit priority
    - Priority interrupt supported
    - Vectored interrupt interface
        - Compatible with R5F VIC port
        - Programmable 32-bit vector address per interrupt
            - Address is SECDED error protected
            - Default vector addresses provided on DED
        - Software interrupt generation
- Integrated ECC aggregators
    - Support for error injection to all supported ECC memory blocks to test ECC functionality (add-on function from TI)
    - One ECC aggregator to cover all RAMs and caches
- Standard Arm CoreSight debug and trace architecture at the R5FSS level
    - Cross triggering: Supported by cross trigger interface (CTI) (per CPU) and cross trigger matrix (CTM) components
    - Processor trace: Supported by embedded trace macrocell (ETM) (per CPU) and advanced trace bus (ATB) funnel components
- Boot
    - From ROM only

See Section 7.4.2 for a functional block diagram and more details on the WKUP_R5FSS.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

7.4.2 WKUP_R5FSS Functional Description

7.4.2.1 WKUP_R5FSS Block Diagram

Figure 7-12 shows the WKUP_R5FSS block diagram.

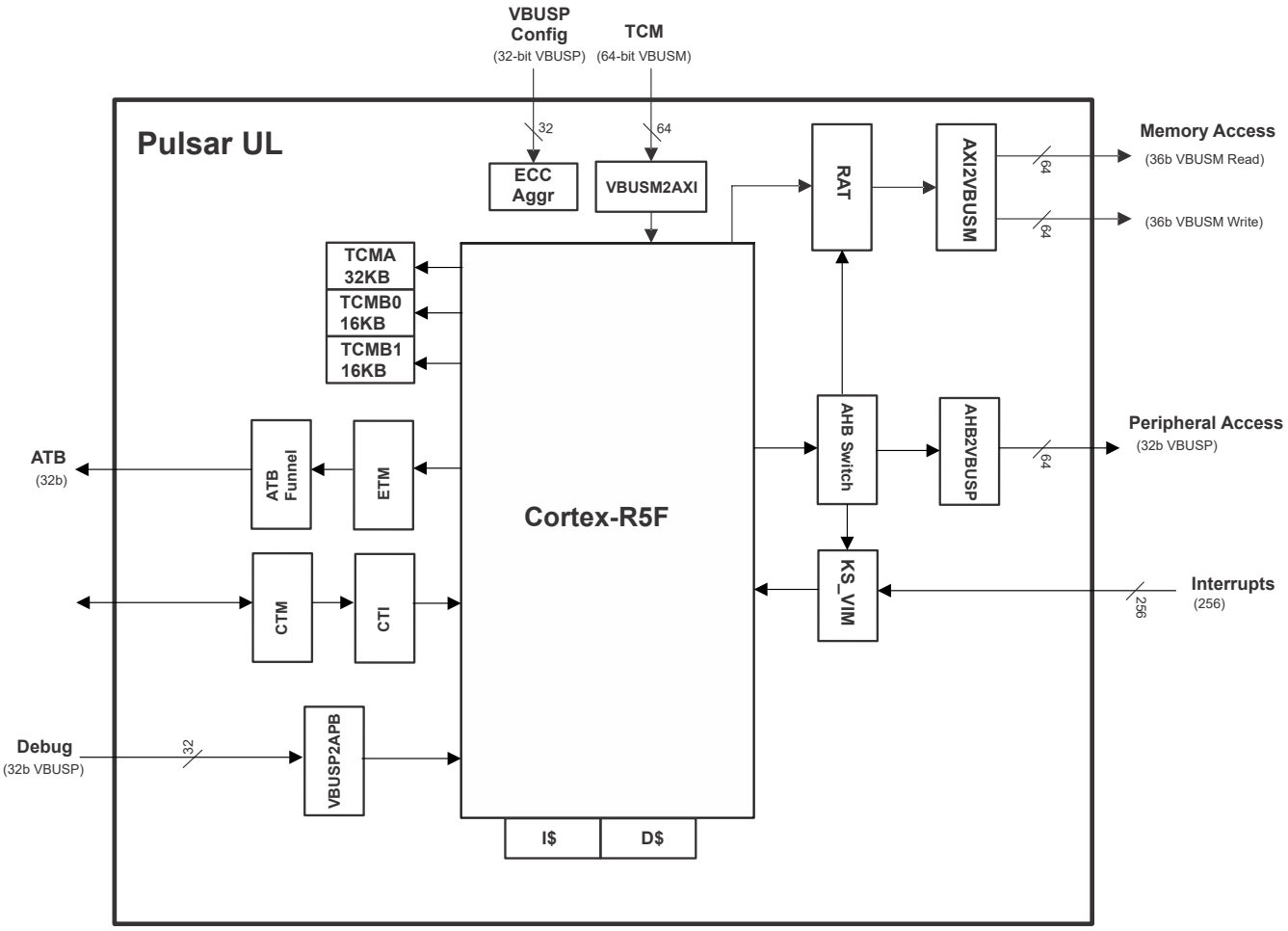


Figure 7-12. WKUP_R5FSS Block Diagram

7.4.2.2 WKUP_R5FSS Cortex-R5F Core

The Cortex-R5F is a processor from Arm, which is based on the Armv7-R profile.

For a brief list of features supported by the R5F processor in this device, see Section 7.4.1.1. For more detailed description of this processor, see the *Arm Cortex-R5 Technical Reference Manual*.

7.4.2.2.1 L1 Caches

The R5F has a Harvard cache architrecture, which means it has an independent L1 instruction cache (16KB) and L1 data cache (16KB). The instruction cache is protected by SECDED ECC per 64 bits. The data cache is protected by SECDED ECC per 32 bits.

7.4.2.2.2 Tightly-Coupled Memories (TCMs)

The R5F has two tightly-coupled memories (TCMs), ATCM and BTCM. The BTCM is further broken down into two interleaved banks, B0TCM and B1TCM.

TCMs are low-latency, tightly integrated memories for the R5F to use. Either TCM can be used for any combination of instruction and/or data. TCM performance is equal to performance on instructions/data that are in cache. However, TCMs have some additional advantages over cache. TCMs can be loaded with instructions

Copyright © 2026 Texas Instruments Incorporated

that do not cache well (such as ISRs) or preloaded with code by an external source, before that code is needed, to save cache miss time. TCMs are also a good place for blocks of data for intense processing. They can be loaded (or pre-loaded by an external source) before the data is needed, saving cache miss time. The data can then be directly accessed by an external source, instead of needing to do cache evicts.

As mentioned, TCMs can be accessed (either read or written) by an external source over the TCM VBUSM target interface. This allows instructions or data to be preloaded, or for data to be read out after the R5F has processed it. The VBUSM target has a lower priority to accessing TCMs than the R5F but care must be taken to keep an external source from reading or writing TCM data that the R5F is working on. This handshaking is external to any of the R5FSS hardware.

TCMs are protected by ECC per 32 bits. For this to work, ECC must be enabled before data is written in to the TCMs (either externally or from the R5F). ECC is enabled via the following R5F system control bits: ACTLR.ATCMPCEN, ACTLR.B0TCMPCEN, and ACTLR.B1TCMPCEN, respectively.

Whether or not the TCMs are enabled is controlled by the ENABLE bit in the corresponding ATCM/BTCM region register. The default (reset) value of this bit is determined by the CPUn_INITRAMA and CPUn_INITRAMB bootstraps, respectively. Both ATCM and BTCM are configured for a size of 32KB in this device. Note that the BTCM size is the total of both B0TCM and B1TCM (16KB each).

If a TCM is not enabled, then it does not appear in the R5F's memory view, but it can be accessed by an external source. If a TCM is enabled, then its place in the R5F memory map is determined by a combination of bootstrap signal and system register. If the CPUn_LOCZRAMA bootstrap signal is high, then the initial base address of ATCM is 0x0000_0000 and the initial address of BTCM is 20'h41010. If the CPUn_LOCZRAMA bootstrap signal is low, then the initial base address of BTCM is 0x0000_0000 and the initial base address of ATCM is 20'h41010.

Note

This base address of 0x41010 for ATCM/BTCM based on the CPUn_LOCZRAMA bootstrap only affects the R5F's memory view. The SoC will see the ATCM/BTCM based on the TCM target interface regions, as defined in Section 7.4.2.3.2. The base address of either TCM may be overwritten via the ATCM or BTCM region register. Care must be taken not to move the base address of a TCM when it may be being accessed.

It is possible to preload a TCM with instructions and boot from it. See Section 7.4.2.11 for details on TCM booting.

7.4.2.2.3 WKUP_R5FSS Special Signals

Table 7-31 lists some WKUP_R5FSS features associated with special signals.

Table 7-31. WKUP_R5FSS Special Features

Feature	Comment
Cluster affinity group ID	R5F Cluster 0 (ID = 0x0)
Exception handling state at reset 0 = Arm 1 = Thumb	Controlled via MAIN_SEC_MMR register setting. Defaults to Arm mode
CPUn execution halt when coming out of reset (CPUn_HALT)	Controlled via MAIN_SEC_MMR register setting. Defaults to halted state
CPUn exception vectors base address	Controlled via MAIN_SEC_MMR register setting. Defaults to Bootvector RAM address 0x0000_0000_0200
CPUn VIM base address	0x2FFF_0000
CPUn RAT base address	0x2FFE_0000
CPUn RAT accesses ID	0x4 (CPU0)
CPUn ATCM enable at reset (CPUn_INITRAMA)	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state

Copyright © 2026 Texas Instruments Incorporated

Table 7-31. WKUP_R5FSS Special Features (continued)

Feature	Comment
CPUn BTCM enable at reset (CPUn_INITRAMB)	Controlled via MAIN_SEC_MMR register setting. Defaults to enabled state
CPUn A/BTCM reset base address indicator (CPUn_LOCZRAMA) 0 = B at 0x0 1 = A at 0x0	Controlled via MAIN_SEC_MMR register setting. Defaults to 1
CPUn non-maskable fast interrupts enable	Controlled via MAIN_SEC_MMR register setting. Defaults to disabled state
CPUn VBUSM peripheral port enabled at reset	Enabled
CPUn VBUSP peripheral port enable at reset	Enabled
CPUn VBUSP peripheral port base address	Mapped to 0x0_2000_0000 for low latency MAIN peripherals
CPUn VBUSP peripheral port size	64MB for MAIN peripherals (0x0_2000_0000 to 0x0_2FFF_FFFF)
CPUn VBUSM normal peripheral port base address	Not used
CPUn VBUSM normal peripheral port size	Not used
CPUn VBUSM virtual peripheral port base address	Not used
CPUn VBUSM virtual peripheral port size	Not used
CPUn clock stopped indication	Status logged into MAIN_SEC_MMR register bit
CPUn WFI state	Status logged into MAIN_SEC_MMR register bit
CPUn WFE state	Status logged into MAIN_SEC_MMR register bit
CPU clockstop behavior 0: CPU clocks stopped in standby 1: CPU clocks not stopped in standby	Controlled via MAIN_SEC_MMR register setting. Defaults to 0

7.4.2.3 WKUP_R5FSS Interfaces

7.4.2.3.1 Initiator Interfaces

The WKUP_R5FSS has several initiator interfaces:

- 64-bit VBUSM initiator pair (1 read - "RMST", 1 write - "WMST") for L3 memory accesses; this is the main memory interface
  - Includes region-based address translation (RAT)
- 32-bit VBUSP initiator "PMST" for peripheral access
  - Includes logic that provides the R5F CPU with a private access to VIM and RAT
  - Enabled at reset

7.4.2.3.2 Target Interfaces

The WKUP_R5FSS has several target interfaces that define its internal memory space:

- 32-bit VBUSP configuration target
  - Region [0]: ECC aggregator block
- 64-bit TCM target
  - Region [0]: ATCM
  - Region [1]: BTCM
  - Region [2]: Instruction cache RAMs
  - Region [3]: Data cache RAMs
- 32-bit VBUSP debug target
  - Provides access to all R5FSS internal debug logic

Regions [0] and [1] of the TCM target interface provide direct access to the TCM RAMs. Access to the RAMs is arbitrated with access from the R5F's L1 memory system. Excessive access while the R5F is also attempting access will degrade performance.

Regions [2] and [3] of the TCM target interface provide access to the cache RAMs for testing purposes. Access to the cache RAMs can only be done while the caches are disabled and should only be done for test purposes.

Copyright © 2026 Texas Instruments Incorporated

In addition to the target interfaces, there are peripherals (RAT and VIM) that are only accessible by the R5F. The R5F has an access to these modules via the VBUSP peripheral interface.

7.4.2.4 WKUP_R5FSS Power, Clocking and Reset

7.4.2.4.1 WKUP_R5FSS Power

The following WKUP_R5FSS power considerations should be noted:

- WKUP_R5FSS has a single power domain for all its internal logic

For more details on WKUP_R5FSS power management, including power-up and power-down sequences, see *Power*.

7.4.2.4.2 WKUP_R5FSS Clocking

The WKUP_R5FSS has four clock inputs:

- CPU0_CLK: This is the clock for CPU0 logic
- CPU0_ICLK: This is the clock for CPU0 interfaces

CPU0_CLK is the clock for all of the internal CPU logic, while CPU0_ICLK is the clock for all of the interfaces for their associated CPU (for example: VBUSM and VBUSP bridges, exception generation, debug and trace logic).

The interface clock is an integer ratio of the CPU clock. The exact ratio for this device is 1:1.

7.4.2.4.3 WKUP_R5FSS Reset

The R5FSS has two reset inputs:

- CPU0_RST: This is the reset for the non-debug logic of CPU0
- CPU0_DBG_RST: This resets the CPU0 debug logic, excluding the APB interface

In addition to the reset signals, there is one halt signal:

- CPU0_HALT

These halt signals keep the CPUs from fetching instructions when they come out of reset. The main use is to have the CPUs halted until the TCMs are loaded (when booting from TCM), though halt could be used for any other purpose.

7.4.2.5 WKUP_R5FSS Vectored Interrupt Manager (VIM)

7.4.2.5.1 VIM Overview

The VIM aggregates device interrupts and sends them to the R5F CPU. It can be used in either split or single-core configuration.

The VIM module supports the following features:

- 256 interrupt inputs for the R5F core
- Each interrupt has its own 4-bit programmable priority
  - Defined via the R5FSS_VIM_PRI_INT_j register
  - The VIM provides support for priority interruption of interrupts
- Each interrupt has its own enable mask
  - Interrupt enable is done via the R5FSS_VIM_INTR_EN_SET_j register
  - Interrupt disable is done via the R5FSS_VIM_INTR_EN_CLR_j register
- Each interrupt can be programmed as either an IRQ or FIQ
  - Defined via the R5FSS_VIM_INTMAP_j register
- Each interrupt has its own programmable 32-bit vector address associated with it
  - Defined via the R5FSS_VIM_VEC_INT_j register
  - Protected with SECDED
- One IRQn and one FIQn output per core
- Vectored interrupt interface
  - Compatible with R5F VIC port

Copyright © 2026 Texas Instruments Incorporated

- Default vector provided when a double-bit error is detected
- Software interrupt generation

7.4.2.5.2 VIM Interrupt Inputs

The VIM supports 256 interrupt inputs per core. Each interrupt can be either a level or a pulse (both active-high). The interrupt mapping for the R5F core can be found in *Interrupt Sources*.

7.4.2.5.3 VIM Interrupt Outputs

The VIM has two interrupt outputs per core:

- *CoreN_IRQn*: This is a normal interrupt for core *N* (active-low level). It can be serviced via the VIC interface or through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (via the R5FSS_VIM_INTMAP_j register) and is enabled (via the R5FSS_VIM_INTR_EN_SET_j register), then it will cause an IRQ to assert
- *CoreN_FIQn*: This is a fast (or non-maskable) interrupt for core *N* (active-low level). FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface. Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ and is enabled, then it will cause an FIQ to assert

7.4.2.5.4 VIM Interrupt Vector Table (VIM RAM)

For each VIM interrupt core, there is an associated interrupt vector table (VIM RAM) that is used to store the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. Note that both interrupt vector tables are identical in their memory organization.

The VIM RAM is basically comprised of a set of interrupt vector registers (R5FSS_VIM_VEC_INT_j). Hence, the interrupt vector table is organized in 256 words of 30 bits, with a base address corresponding to the physical address of the first register in the group.

Note

The lower two bits of the 32-bit interrupt vector are always 0s.

Figure 7-13 shows the VIM RAM interrupt vector map.

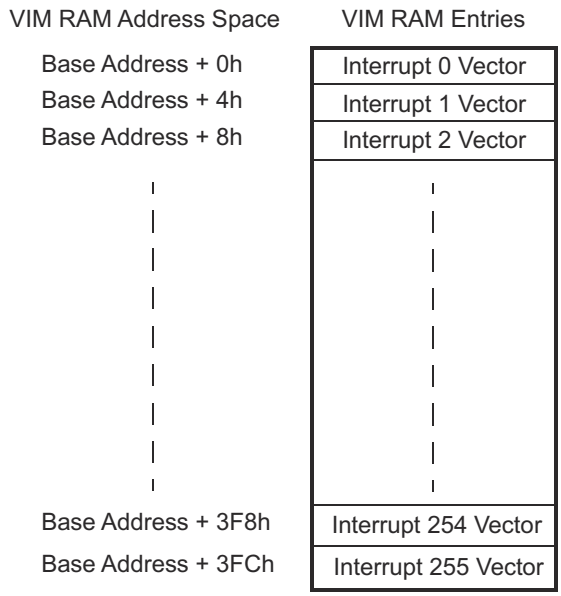


Figure 7-13. VIM RAM Interrupt Vector Map

Copyright © 2026 Texas Instruments Incorporated

The interrupt vector table has protection by ECC to indicate corruption due to soft errors. The ECC logic inside VIM supports SECDED. See Table 7-34 for the VIM RAM ID in the ECC aggregator map.

7.4.2.5.5 VIM Interrupt Prioritization

The VIM supports the interruption of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate but both use the same mechanism.

When an interrupt goes from pending to active (FIQ: reading the R5FSS_VIM_FIQVEC register; IRQ: reading the R5FSS_VIM_IRQVEC register, or the *coreN_IRQACK* going high), then the interrupt is loaded into the corresponding active register (R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ), and all interrupts of an equal or lesser priority are masked (discarded). If prior to this interrupt being cleared (by writing to the R5FSS_VIM_FIQVEC register, or R5FSS_VIM_IRQVEC register) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. If the CPU switches this interrupt to active (by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register), then the currently active interrupt will be pushed onto a stack. When an interrupt is cleared by reading the R5FSS_VIM_FIQVEC / R5FSS_VIM_IRQVEC register, if there are any interrupts on the stack, the first entry is popped off and put back into the R5FSS_VIM_ACTFIQ / R5FSS_VIM_ACTIRQ register, so that software may continue where it left off.

7.4.2.5.6 VIM ECC Support

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the R5FSS_VIM_DEDVEC register is used to provide the default vector for the *coreN_IRQADDRV* signal, the R5FSS_VIM_IRQVEC register, and the R5FSS_VIM_FIQVEC register. The R5FSS_VIM_DEDVEC should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling.

Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
   a. Potentially switch to a completely software interrupt handler in the mean time
2. Restart the application from scratch
3. Reset the device
4. Sit in a loop (or WFI) while something external (for example, the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

Note

An interrupt that has an uncorrectable vector error (and thus uses the DED vector) will still have the priority of the original interrupt . This makes it possible for a higher priority interrupt to supercede the handling of the error.

Control and reporting are done by the R5FSS ECC aggregator.

7.4.2.5.7 VIM IDLE State

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface.

7.4.2.5.8 VIM Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM the software wants to take advantage of.

For IRQs, it is recommended to use the procedure in Section 7.4.2.5.8.1, but the procedures in Section 7.4.2.5.8.2 or Section 7.4.2.5.8.3 (if a user wants to implement a fully software prioritization scheme) may be used as alternatives.

Copyright © 2026 Texas Instruments Incorporated

For FIQs, it is recommended to use the procedure in Section 7.4.2.5.8.4, but the procedure in Section 7.4.2.5.8.5 may be used as an alternative.

Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

7.4.2.5.8.1 Servicing IRQ Through Vector Interface

If the associated CPU has the vector (VIC) interface enabled, then the following method is used for servicing IRQs:

1. Hardware handshake
   a. CPU asserts *coreN_IRQACK* high
   b. VIM asserts *coreN_IRQADDRV* to indicate that the *coreN_IRQADDR* bus is stable with the correct vector address
   c. CPU reads *coreN_IRQADDR*, jumps to that address, and de-asserts *coreN_IRQACK* low
   d. VIM de-asserts *coreN_IRQn* and *coreN_IRQADDRV*, VIM masks (discards) all IRQs with the same or lower priority
   e. VIM loads the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field (which corresponds to the vector address) into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, which causes the R5FSS_VIM_ACTIRQ[31] VALID bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTIRQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register
      ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_IRQSTS_j register, or R5FSS_VIM_STS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.4.2.5.8.2 Servicing IRQ Through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the vector interface:

1. Read the R5FSS_VIM_IRQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_IRQn* output. If another interrupt of a higher priority becomes available, the *coreN_IRQn* will re-assert, allowing priority interruption of an interrupt
   b. Reading this register will cause the value from the R5FSS_VIM_PRIIRQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_ACTIRQ[9:0] NUM bit field, and the R5FSS_VIM_ACTIRQ[31] VALID bit to be set

Copyright © 2026 Texas Instruments Incorporated

2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
      ii. Clear the interrupt at the source
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
4. Write any value to the R5FSS_VIM_IRQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTIRQ[31] VALID bit

7.4.2.5.8.3 Servicing IRQ Through MMR Interface (Alternative)

If a user does not want to use the R5FSS_VIM_IRQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_IRQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_PRIIRQ register to determine which interrupt is the highest priority IRQ currently asserted, OR
   b. Optionally read the R5FSS_VIM_IRQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_IRQSTS_j register and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_IRQSTS_j register

7.4.2.5.8.4 Servicing FIQ

When an FIQ interrupt is received, the CPU should follow these steps:

1. Read the R5FSS_VIM_FIQVEC register and jump to that address to service the ISR
   a. Reading this register will mask (discard) all interrupts of an equal or lower priority and de-assert the *coreN_FIQn* output. If another interrupt of a higher priority becomes available, the *coreN_FIQn* will re-assert, allowing priority interruption of an interrupt.
   b. Reading this register will cause the value from the R5FSS_VIM_PRIFIQ[9:0] NUM bit field to be loaded into the R5FSS_VIM_ACTFIQ[9:0] NUM bit field, and the R5FSS_VIM_ACTFIQ[31] VALID bit to be set
2. Service the interrupt

Copyright © 2026 Texas Instruments Incorporated

3. Depending on whether the original source of the interrupt was a pulse or a level (determined by reading the R5FSS_VIM_ACTFIQ[9:0] NUM bit field to determine number, and reading the appropriate bit in the R5FSS_VIM_INTTYPE_j register to determine type)
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
      ii. Clear the interrupt at the source. This way, the source can generate another pulse, if it needs to, and the VIM will process this as a new interrupt
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt. If the source maintains the level, then it means there is another interrupt
4. Write any value to the R5FSS_VIM_FIQVEC register
   a. This will clear the priority mask and will cause all interrupts to be re-evaluated for the new highest priority interrupt
   b. This will also clear the R5FSS_VIM_ACTFIQ[31] VALID bit

7.4.2.5.8.5 Servicing FIQ (Alternative)

If a user does not want to use the R5FSS_VIM_FIQVEC register, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the R5FSS_VIM_FIQVEC register is never read). Software would be responsible for doing all priority operations.

1. Determine which interrupt to service
   a. Read the R5FSS_VIM_FIQVEC register to determine which interrupt is the highest priority FIQ currently asserted, OR
   b. Optionally read the R5FSS_VIM_FIQGSTS register to determine which groups have IRQs pending, then read the R5FSS_VIM_FIQSTS_j register and use a software prioritization scheme to determine which FIQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level
   a. Pulse
      i. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register
      ii. Clear the interrupt at the source.
   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a '1' to the appropriate bit in the R5FSS_VIM_STS_j register, or R5FSS_VIM_FIQSTS_j register.

7.4.2.6 WKUP_R5FSS Region Address Translation (RAT)

7.4.2.6.1 WKUP R5FSS Usage

R5 is a micro controller with 32b address, which is only able to access up to 4GB address space. However, Sitara™ SoC supports 36b address and majority of the memory region beyond lower 4GB are implemented. Region based address translation block, called RAT, is introduced to allow R5 core to access full 36b SoC memory map.

Each R5 core has its own RAT block, and only R5 core itself is able to program the RAT.

Copyright © 2026 Texas Instruments Incorporated

Figure 7-14 shows the main R5's default 32b memory map view when main R5 is out of reset.

The full RAT functionality is described in Section 7.3.2.6.2, *RAT Function*.

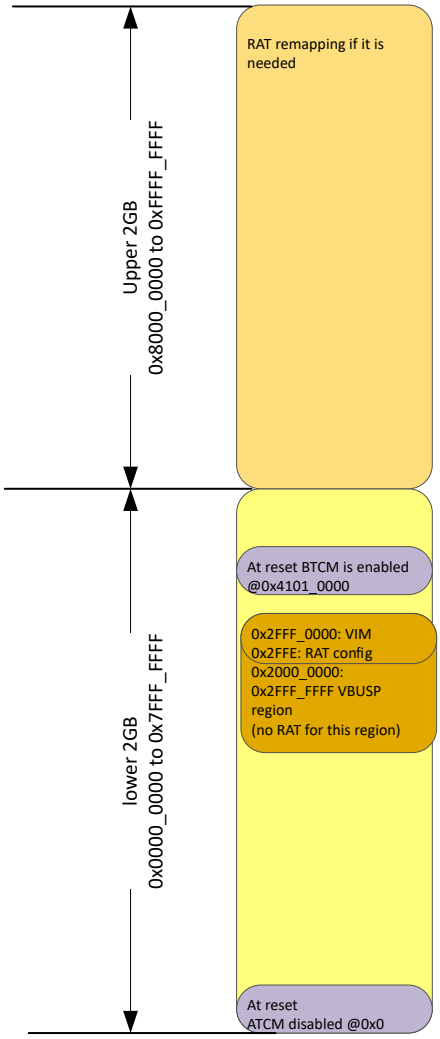


Figure 7-14. R5 Core Default Memory Map View

Copyright © 2026 Texas Instruments Incorporated

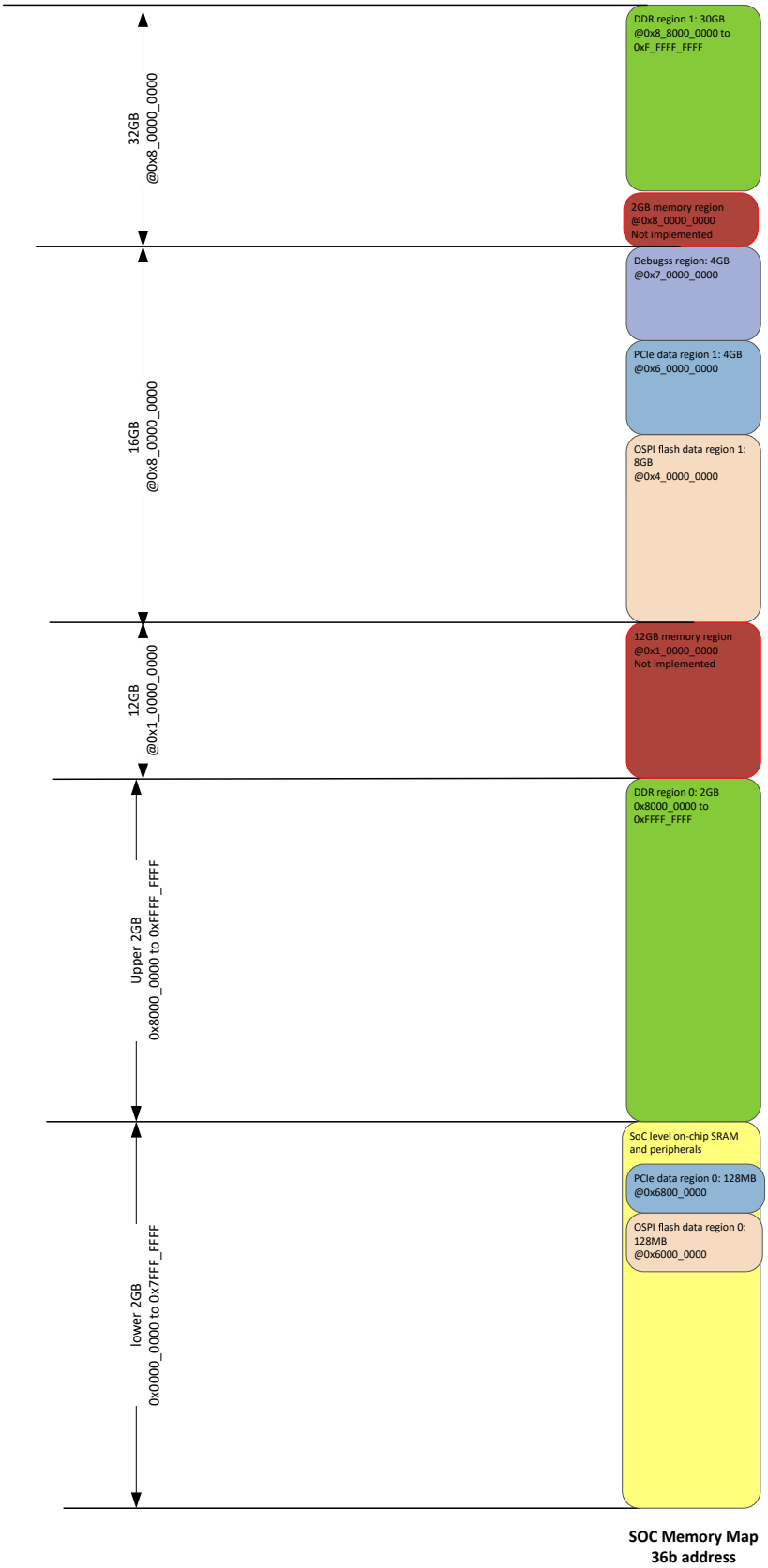


Figure 7-15. Typical Sitara SoC Memory Map View

Copyright © 2026 Texas Instruments Incorporated

RAT block enables R5 core to have full access on the SoC memory map including the memory region at and above 0x1_0000_0000.

Figure 7-16 shows how R5 access various end points, including its own ATCM, BTCM, VIM and SoC memory and peripherals. R5's ATCM is by default at address 0x0, and its BTCM is at address 0x4101_0000. R5 uses address 0x2FFF_0000 to access its VIM and 0x2FFE_0000 to access the RAT configuration region and all the other transactions between address range 0x2000_0000 to 0x2FFF_FFFF are sent to the 32b peripheral interface to access the peripherals in SoC. The transactions with the rest of address are sent to RAT block, which could go through address remapping function from original R5's 32b address to 36b SoC address.

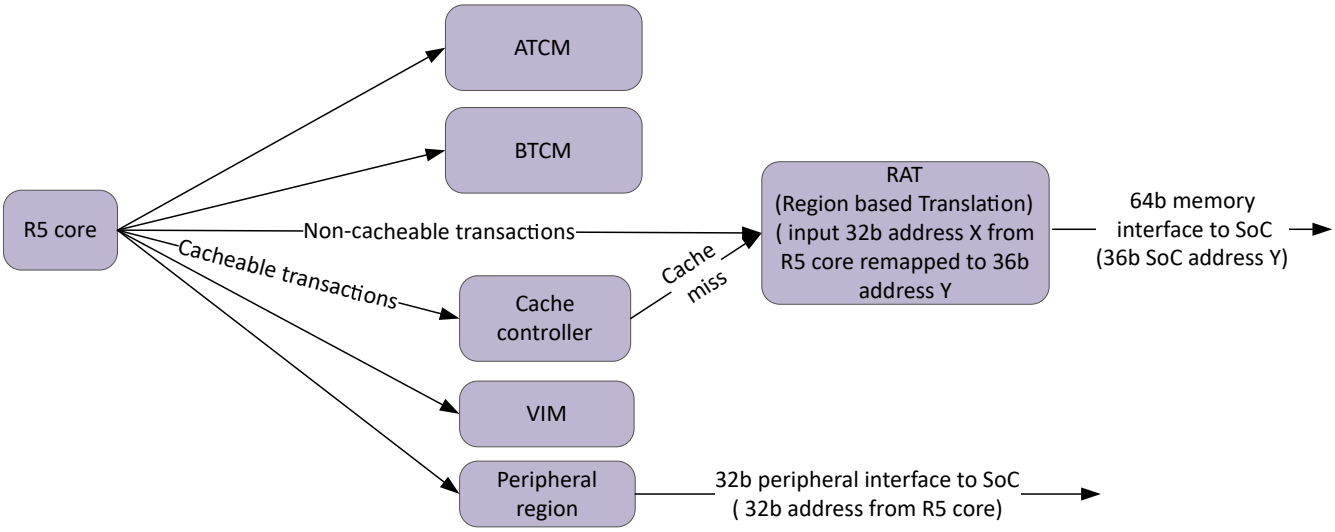


Figure 7-16. R5 Internal Address Decoding

7.4.2.6.2 RAT Function

RAT block is responsible to do a region based address translation. For each region, user can define the starting address and size of the region and remapping the transactions to use a new address range.

The starting address and region size need to be aligned. For example, if the region size is 16KB, the starting address for that region needs to be 16KB aligned. And region size should be minimum 4KB or larger to avoid transactions cross the region boundaries.

Each RAT block supports multiple programmable regions. And it is important that those regions are not overlapping with each other.

RAT block is by default disabled, which means RAT block does not do address remapping. The 32b address coming from R5 core is directly sent out to the SoC level. The user can program each RAT remapping region individually. For the transactions does not hit any remapping region, RAT block just simply forwards the transactions to the SoC level with its original address.

7.4.2.6.3 How to use RAT Block in R5

Table 7-32 shows the default R5 memory map view. Without enabling RAT function, all the regions beyond 4GB address space will not be accessible by R5.

Sitara™ SoC memory map is constructed the way to allow R5 and high level application processor such as A53 have similar memory map as possible. This methodology enables that R5 and ARM A53 core have a common memory map for all the SoC level memories and SoC level peripherals except a few exceptions:

- DDR's data space beyond 32b address, basically upper DDR data space beyond 2GB
- PCIe data space beyond 32b address
- OSPI data space beyond 32b address
- Debug configuration region

Copyright © 2026 Texas Instruments Incorporated

Table 7-32. R5 Memory map by Default

	R5 Memory Map(32b)	SoC Memory Map (36b)
SoC Level peripheral and on-chip SRAM	0x0 to 0x7FFF_FFFF Same address as SoC memory map	0x0 to 0x7FFF_FFFF
2GB DDR region	Same as SoC memory map: 0x8000_0000 to 0xFFFF_FFFF	0x80000_0000 to 0xFFFF_FFFF
Additional DDR region	Not accessible	0x8_8000_0000 (30GB)
Additional 8GB OSP Space	Not accessible	0x4_0000_0000 (4GB) 0x5_0000_0000 (4GB)
Additional PCIe data space	Not accessible	0x6_000_0000
Debug configuration region	Not accessible	0x7_0000_0000

Even though all the transactions other than going to ATCM/BTCM and transactions between address range 0x2000_0000 to 0x2FFF_FFFF could use RAT block to remap to different address, it is recommended that only use RAT block to remap the R5 transactions with address between 0x8000_0000 to 0xFFFF_FFFF.

- Any transactions from R5 with address between 0x0 to 0x7FFF_FFFF are sent out to SoC directly (unless it is targeted to its internal end points such as A/B TCM, its VIM and RAT configuration)
- Only enable RAT to remap the transactions with address between 0x8000_0000 to 0xFFFF_FFFF if R5 needs to access the following address space:
  - PCIe data space at address 0x6_0000_0000
  - OSPI data space address 0x4_0000_0000 and 0x5_0000_0000
  - Debug configuration region at address 0x7_0000_0000
  - DDR data space at address 0x8_8000_0000

Copyright © 2026 Texas Instruments Incorporated

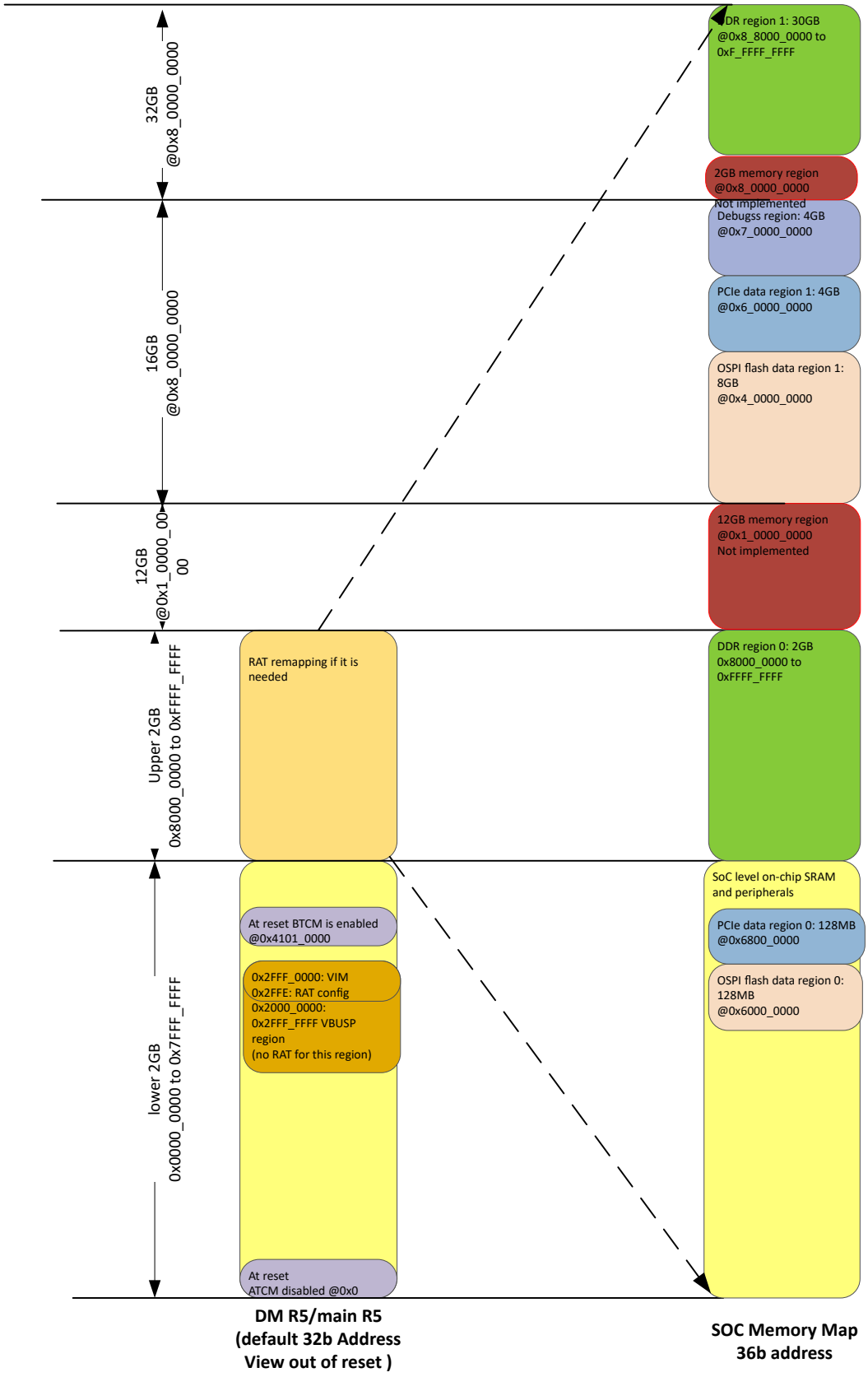


Figure 7-17. Example of Using RAT to Access Full 36b SoC Memory Map

Copyright © 2026 Texas Instruments Incorporated

7.4.2.6.4 Example of Using RAT to Access Full 36b SoC Memory Map

This section shows an example on how to utilize RAT block to allow R5 to access 36 SoC Memory Map. The RAT block inside R5 has up to 8 address remapping regions.

Assuming R5 needs to access the following region:

- All the SoC level on-chip SRAM
- All the SoC level peripherals
- Access 512MB OSPI region at address 0x4_0000_0000
- Access 1GB DDR region at address 0x9_0000_0000
- Access 512MB DDR region at address 0x8000_0000

First of all, we need to construct 32b R5 memory map as following and how those memory spaces are mapped to SoC level, shown in Table 7-33. Users can define where the OSPI and DDR data space placed inside R5's memory map and how they should be mapped to SoC level address map. Table 7-33 just shows one of the many possible ways to construct the memory map.

Table 7-33. Example of R5 Address Mapping

	R5 Memory Map(32b)	SoC Memory Map (36b)	RAT Remapping
SoC Level peripheral and on-chip SRAM (This is guaranteed by SoC Hardware design. No user configuration is needed)	0x0 to 0x7FFF_FFFF Same address as SoC memory map	0x0 to 0x7FFF_FFFF	Not RAT Remapping
512MB OSPI data region (user defined)	0x8000_0000 to 0x9FFF_FFFF	0x4_0000_0000 to 0x4_1FFF_FFFF	Using RAT remapping region 0
512MB DDR space (user defined)	0xA000_0000 to 0xBFFF_FFFF	0x8000_0000 to 0x9FFF_FFFF	Using RAT remapping region 1
Additional 1GB DDR Space (user defined)	0xC000_0000 to 0xFFFF_FFFF	0x9_0000_0000 to 0x9_3FFF_FFFF	Using RAT remapping region 2

In this example, only three RAT regions are needed:

- Region 0 is used to remap R5's address between 0x8000_0000 to 0x9FFF_FFFF to SoC level address 0x4_0000_0000 to 0x4_1FFF_FFFF
- Region 1 is used to remap R5's address between 0xA000_0000 to 0xBFFF_FFFF to 0x8000_0000 to 0x9FFF_FFFF
- Region 2 is used to remap R5's address 0xc000_0000 t0 0xFFFF_FFFF to 0x9_0000_0000 to 0x9_3FFF_FFFF.

In order achieving those mappings, those are the setting of the RAT configuration registers:

	Region 0	Region 1	Region 2	Other regions
Region control register	Enabled, size set to 512MB	Enabled, size set to 512MB	Enabled, size set to 1GB	disabled
Region base(32b)	0x8000_0000	0xA000_0000	0xc000_0000	Don't care
Region translated lower address(32b)	0x0000_0000	0x8000_0000	0x0000_0000	Don't care
Region translated upper address	0x4	0x0	0x9	Don't care

Figure 7-18 shows how RAT is remapping the address between R5's 32b address to 36b SoC address.

Copyright © 2026 Texas Instruments Incorporated

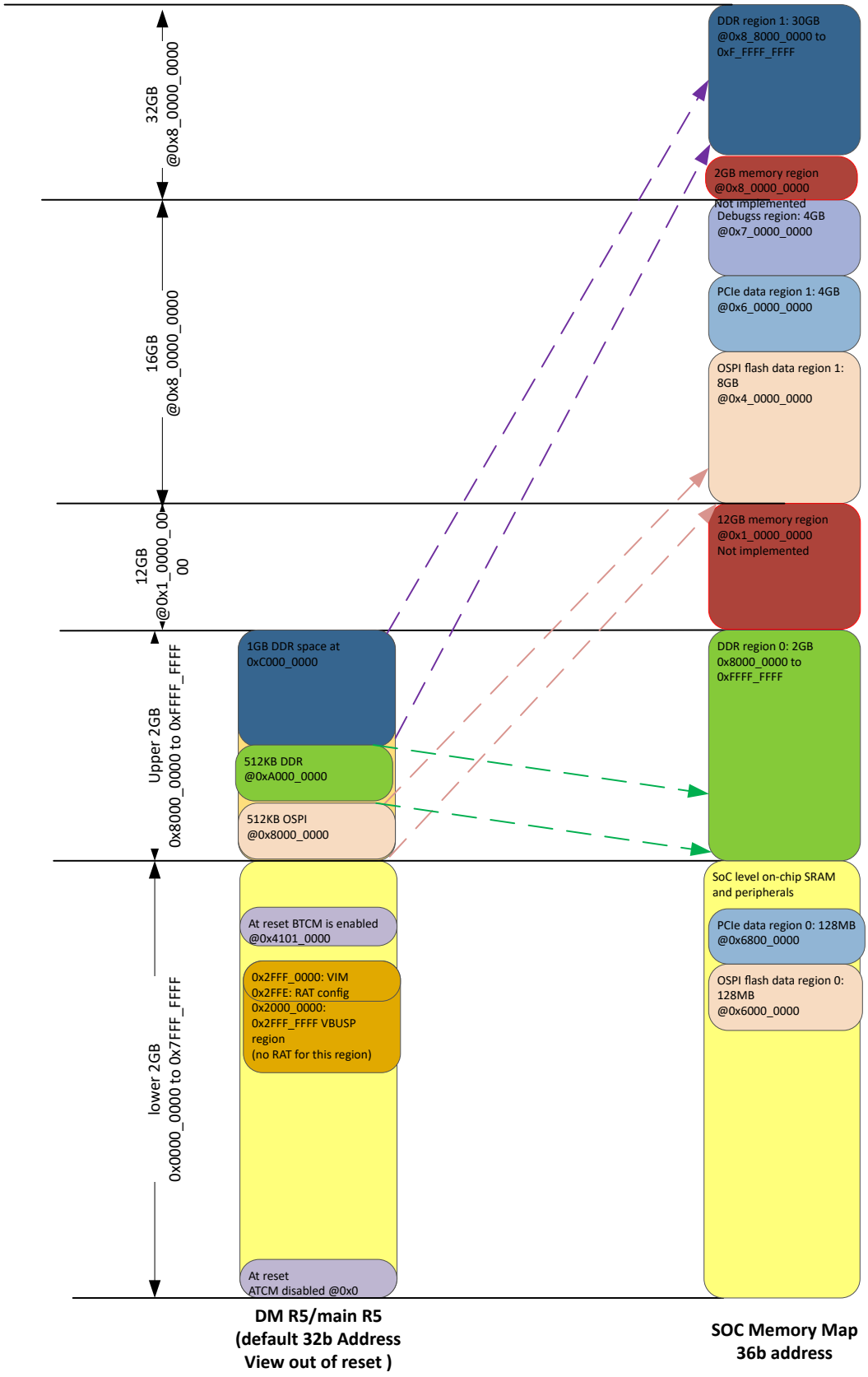


Figure 7-18. RAT Configuration Example

Copyright © 2026 Texas Instruments Incorporated

7.4.2.7 WKUP_R5FSS ECC Support

The R5F provides native ECC and parity support on all related memories, generating and checking the redundancy automatically. The R5F can also monitor any ECC errors on it's SoC buses and bridges through the ECC aggregator. The methods for checking and reporting errors are available in the *Arm Cortex-R5 Technical Reference Manual*.

The R5FSS adds the capability of testing this logic by allowing errors (single and double bit) to be injected into memories (for testing purposes) via an ECC aggregator (per core). Note that because the R5FSS ECC aggregator is only used in error-injection mode, it only supports a subset of the generic ECC aggregator functionality in the device.

For a detailed description of the generic ECC aggregator functionality, see *ECC Aggregator*. For register descriptions of the WKUP_R5FSS aggregators, see R5FSS_CPU0_ECC_AGGR_CFG_REGS Registers and R5FSS_CPU1_ECC_AGGR_CFG_REGS Registers, respectively.

Table 7-34 provides the RAM ID. This is needed for bit field [10-0] ECC_VECTOR in the corresponding R5FSS_CPU0_VECTOR**R5FSS_CPU0_VECTOR** register (part of the ECC aggregator register space).

Table 7-34. RAM ID Map for ECC Aggregator (Per Core)

RAM ID	Memory Name
0	CPU0 ITAG RAM0
1	CPU0 ITAG RAM1
2	CPU0 ITAG RAM2
3	CPU0 ITAG RAM3
4	CPU0 IDATA BANK0
5	CPU0 IDATA BANK1
6	CPU0 IDATA BANK2
7	CPU0 IDATA BANK3
8	CPU0 DTAG RAM0
9	CPU0 DTAG RAM1
10	CPU0 DTAG RAM2
11	CPU0 DTAG RAM3
12	CPU0 DDIRTY RAM
13	CPU0 DDATA RAM0
14	CPU0 DDATA RAM1
15	CPU0 DDATA RAM2
16	CPU0 DDATA RAM3
17	CPU0 DDATA RAM4
18	CPU0 DDATA RAM5
19	CPU0 DDATA RAM6
20	CPU0 DDATA RAM7
21	CPU0 ATCM BANK0
22	CPU0 ATCM BANK1
23	CPU0 B0TCM BANK0
24	CPU0 B0TCM BANK1
25	CPU0 B1TCM BANK0
26	CPU0 B1TCM BANK1
27	CPU0 VIM RAM

Copyright © 2026 Texas Instruments Incorporated

7.4.2.8 WKUP_R5FSS Memory View

The memory view of the R5F (that is, the memory map as seen by each R5F) is a function of several things:

- Exception vector bootstrap: The R5F exception table (including boot vector) is always 32 bytes at address 0x00000000 as seen by the R5F. If not booting from a TCM, then boot is done over the main memory interface. The exception vector bootstrap is under software control, which allows these 32 bytes at address 0x00000000 to be remapped somewhere else in the SoC memory map.
- TCM locations: TCMs can be enabled or disabled and located at different places in the memory map, depending on bootstrap configuration. For more details, see Section 7.4.2.2.2.
- Peripheral interface locations: The RF5 natively supports three interfaces for peripheral access. Each can be enabled/disabled and located based on bootstrap configuration. Note that the VBUSP peripheral interface must be enabled in order to use RAT and VIM.
- RAT base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- VIM base address: This is determined by a bootstrap. This address is located within the VBUSP peripheral interface address space.
- RAT programming: The RAT can take regions of memory accessible by the main memory interface and map them to different addresses.

The combination of the above determines what the R5F sees where in the memory map, and over what interface different transactions come out. Every transaction that does not directly address a TCM or a peripheral interface comes over the main memory interface. Transactions on the main memory interface can be further remapped with the RAT.

See *Memory Map* for the complete R5F memory view for this device.

7.4.2.9 WKUP_R5FSS Interrupts

All interrupts that are generated by the WKUP_R5FSS are summarized in *Module Integration*, along with their mapping. They can be divided into the following groups:

- R5F CPU internal interrupts: These are described in *Arm Cortex-R5 Technical Reference Manual*.
- ECC aggregator interrupts: These are described in the *ECC Aggregator* chapter.
- RAT exception interrupt: This is described in Section 7.4.2.6.

7.4.2.10 WKUP_R5FSS Debug and Trace

The WKUP_R5FSS supports standard Arm CoreSight debug and trace architecture. For more details, see the *On-chip Debug* chapter.

7.4.2.11 WKUP_R5FSS Boot Options

There are two methods of booting the R5F, or rather, two methods of placing the exception vectors (of which the boot vector is one).

The first method is to have the exception vectors external to the R5F. The user can place the exception vectors at the address indicated by the exception vector bootstrap and then program the boot vector there. When the processor exits reset, it will fetch the boot vector from this location.

The second method is to boot from a TCM. To do this, software should take the following steps:

1. Assert the correct bootstraps
   a. To boot from ATCM, set CPUn_INITRAMA (or CPUn_INITRAMB to boot from BTCM)
   b. Assert CPUn_LOCZRAMA properly for the desired TCM
2. Assert CPUn_HALT
3. Release the CPU from reset
4. Load the desired code into the TCM via the TCM target port
   a. Exception vectors should be located at address 0x00000000 of the TCM
5. De-assert CPUn_HALT

Copyright © 2026 Texas Instruments Incorporated

7.4.2.12 WKUP_R5FSS Core Memory ECC Events

The R5F core generates several events as part of event bus that can be monitored by the PMU for debugging. The memory ECC related events from the event bus are exported to ESM for monitoring.

There are two ECC interrupts to the ESM that aggregate different categories of ECC events – CPU single error, CPU multi error. Each ECC event has a 2-bit event bus counter associated with it. Everytime an event occurs, the counter is incremented by 1 till it reaches the max value of 3. The interrupt is asserted if the bus counter of any event associated with the interrupt is non-zero.

Each event bus counter has a MMR decrement control to decrement the counter by 1. So, for example, if a counter value is 2, the MMR to decrement the counter would need to be written 2 times to decrement the counter to 0. The reason decrement control has been added instead of clear control is if a new error occurs between the time the status register is read and the clear MMR is written, the new error would be lost. Write-to-decrement ensures that this does not happen.

When all event bus counters of an associated interrupt are zero, the interrupt is cleared. It takes three clock cycles for the event bus counter to be decremented once the write to the decrement control MMR presents itself at the WKUP_R5FSS boundary.

Since each of the four ECC interrupts have single bit control to set the interrupt but multiple bits for clearing it, note that once an interrupt is set using the R5FSS_EVNT_BUS_ESM_SET register, it can be cleared by setting all the bits of the R5FSS_EVNT_BUS_ESM_CLR register that correspond to that particular interrupt. For example, if bit [0] of the R5FSS_EVNT_BUS_ESM_SET register is set, it can be cleared by setting bits [7-0] of the R5FSS_EVNT_BUS_ESM_CLR register to clear the interrupt.

The R5F core event bus only signals event when it is enabled. Non-invasive or invasive debug mode needs to be enabled to enable the PMU counters.

The export of the events to the event bus can be enabled by setting the X bit in the Performance Monitor Control Register of the R5F core. For more details, refer to Arm R5 TRM.

Table 7-35. R5F Event Bus Single-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
22	Instruction cache tag RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS0
23	Instruction cache data RAM parity or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS1
24	Data cache tag or dirty RAM parity error or correctable ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS2
25	Data cache data RAM parity error or correctable ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS3
40	ATCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS4
41	B0TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS5
42	B1TCM single-bit ECC error.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS6
43	TCM correctable ECC error reported by load/ store unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS7
44	TCM correctable ECC error reported by prefetch unit.	R5FSS_CPU0_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8 R5FSS_CPU1_EVNT_BUS_SB_ERR_CNT_STATUS[0] EVNT_BUS8

Table 7-36. R5F Event Bus Multi-Bit Error Events

Event Bus Bit #	Description	Associated Status Register
26	TCM fatal ECC error reported from the prefetch unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS0

Copyright © 2026 Texas Instruments Incorporated

Table 7-36. R5F Event Bus Multi-Bit Error Events (continued)

Event Bus Bit #	Description	Associated Status Register
27	TCM fatal ECC error reported from the load/ store unit.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS1
33	Data cache data RAM fatal ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS2
34	Data caches tag/dirty RAM fatal ECC error, from data-side or ACP.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS3
37	ATCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS4
38	B0TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS5
39	B1TCM multi-bit ECC error.	R5FSS_CPU0_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6 R5FSS_CPU1_EVNT_BUS_MB_ERR_CNT_STATUS[0] EVNT_BUS6

7.5 Vectored Interrupt Manager (VIM)

This section describes the VIM module in the device.

7.5.1 VIM Overview

The Vectored Interrupt Manager (VIM) aggregates interrupts to a CPU. It is intended for use with a Cortex R5 from ARM. The VIM has up to 1024 interrupt inputs, which may be either level or pulse. Each interrupt has a programmable priority (0-highestthrough 15-lowest). Each interrupt may also be mapped as an IRQ or FIQ (FIQ is also often denoted as Non-Maskable Interrupt, or NMI).

7.5.1.1 VIM Features

- 32-1024 Interrupt inputs
  - Configurable by groups of 32
- Each interrupt has its own 4-bit programmable priority
  - Supports Priority interruption of interrupts
- Each interrupt has its own enable mask
- Each interrupt can be programmed as either an IRQ or FIQ
- Each interrupt has its own programmable 32-bit Vector address associated with it
  - Protected with SECDED
- One IRQn and FIQn output
- Vectored Interrupt Interface
  - Compatible with ARM Cortex R5 VIC Port
- Default Vector provided when a Double-Bit error is detected
- Provides clock phase inputs so VIM can run at CPU clock while interface runs at a different, synchronous clock

7.5.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

7.5.2 VIM Functional Description

7.5.2.1 Block Diagram

Figure 7-19 shows the Vectored Interrupt Manager block diagram.

Copyright © 2026 Texas Instruments Incorporated

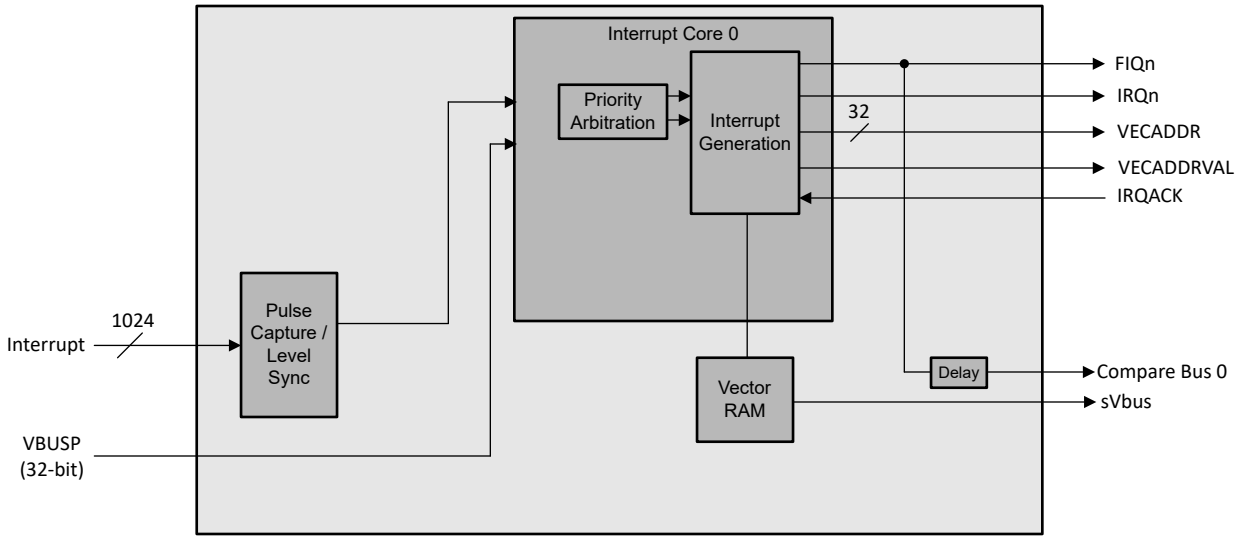


Figure 7-19. VIM Block Diagram

7.5.2.2 Interrupt Inputs

The VIM can have up to 1024 interrupt inputs, build-time configurable by multiples of 32 (*num_interrupts* parameter). Each interrupt can be either a level or a pulse (both active high).

Level interrupts are synchronized to the VIM clock. This synchronized value is captured in to a flop.

Pulse interrupts use rising edge detection. Each input has its own edge detection circuit. It is recommended that if pulses are pipelined between the source and the VIM, that the pipe stage replicates the pipe flop and ORs the output in order to protect against transient errors in the pipeline flop. Once an edge has been detected, the raw status is set.

There is no minimum pulse width, as true edge detection is used. The input signal should, however, be glitch free.

7.5.2.3 Interrupt Outputs

The VIM has two interrupt outputs, core*N*_IRQn and core*N*_FIQn (active low levels). Each interrupt input for core *N* can be programmed to influence either the core*N*_IRQn or core*N*_FIQn output via the Group *M* Interrupt Map Register (Base Address + 0x200 + *N**0x20 + 0x18).

7.5.2.4 Priority Interrupt / Nested Interrupts

Each interrupt has a priority number assigned to it (set using the 4.1.20 Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q**0x4) register). Legal values are 0 to 15 where 0 is the highest priority and 15 is the lowest priority. The highest priority interrupt is the pending interrupt with the smallest priority number. If two pending interrupts have the same priority, the interrupt lowest numerically (0 through maximum number of interrupts) is prioritized. IRQs and FIQs are prioritized separately.

The VIM supports the interrupt of the currently active interrupt by one with a higher priority. FIQs and IRQs are completely separate, but both use the same mechanism. When an interrupt goes from pending to active (FIQ: reading the FIQ Vector Address (Base Address + 0x1C), IRQ: reading the IRQ Vector Address (Base Address + 0x18) or the core*N*_IRQACK going high), then the interrupt is loaded into the corresponding Active Register, and all interrupts of an equal or lesser priority are masked off (see note below). If, before this interrupt is cleared (writing the FIQ Vector Address (Base Address + 0x1C) or IRQ Vector Address (Base Address + 0x18)) another interrupt of higher priority arrives, then the FIQn/IRQn will be asserted and that interrupt made pending as normal. The CPU may or may not service the higher priority interrupt. If the CPU switches this interrupt to active, by reading the corresponding Vector Address Register (or core*N*_IRQACK going high for an IRQ) , then the currently active interrupt will be pushed on to a stack. When an interrupt is cleared by writing the Vector Address Register, if there are any interrupts on the stack, the first entry is popped off and put back into the Active

Copyright © 2026 Texas Instruments Incorporated

Register, so that software may continue where it left off. Note that the IRQVEC/FIQVEC address registers are *not* repopulated with the old vector as it's assumed that the ISR is picking back up where it left off. Only the interrupt number and priority are restored to the ACTIRQ/ACTFIQ registers. If software needs the vector again, it will have to read it by using the interrupt number.

Note

"Masked off" means that they are masked off from priority arbitration to interrupt the currently active interrupt, it does NOT mean that the status bits in the regsiters are masked off. That is, this priority masking has NO EFFECT on whether the status bits are visible in the masked registers such as the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04).

7.5.2.5 VIC Port

The VIM is designed to work with the VIC interface of an ARM Cortex R5.

7.5.2.6 Latency

There are several factors that go in to the speed of processing an interrupt: the interrupt controller, the processor, and the system latency.

The VIM behaves deterministically (except for synchronizer delay). Table 7-37 shows the latency of the VIM.

Table 7-37. VIM Latency

Step	Cycles	Note
Edge Detection	2-3	Synchronizing to VIM Clock
Interrupt Capture	1	
Prioritization and Vector Read	1	IRQ Could be stalled here if an FIQ is reading the Vector RAM
		IRQ/FIQ output asserted

The next factor is the processor itself

Table 7-38. Processor Interrupt Latency

Step	Cycles(1)	Note
Take Exception		
Store State		Stack system registers etc
Read Vector	1	Through MMR or VIC interface (for IRQ. VIC interface will be faster)
Jump to Vector		

(1) These steps are dependent on the processor. See R5 spec for details.

The final factor is system latency to access the actually instructions for the ISR. This will be dependent on where the instructions are (cache, TCM, system memory etc) and dependent on the system latency to reach those memories (1 cycle for cache/TCM, System dependent for external memory)

7.5.2.7 Safety

The memory that holds the interrupt vector for each interrupt is protected by SECDED ECC. Single-bit errors are corrected and written back. Double-bit errors are not corrected. If a double-bit error occurs while trying to load a vector, then the DED Vector Address (Base Address + 0x30) is used instead for the core*N*_IRQADDRV, IRQ Vector Address (Base Address + 0x18), and FIQ Vector Address (Base Address + 0x1C). The DED Vector Address should point to an ISR that handles the fact that there was an uncorrectable error in the interrupt handling. Some possible remediating actions would be to:

1. Reconstruct the vector table and re-start the application
   a. Potentially switch to a completely software interrupt handler in the mean time
2. Restart the application from scratch
3. Reset the device

Copyright © 2026 Texas Instruments Incorporated

4. Sit in a loop (or WFI) while something external (say the ESM) responds to the DED interrupt that will be generated

It is up to the user and the application to determine the appropriate action.

An interrupt that has an uncorrectable vector error and thus uses the DED Vector, will still have the priority of the original interrupt (that is for masking purposes). This makes it possible for a higher priority interrupt to supercede the handling of the error.

Control and reporting are done by an external ECC aggregator through sVbus ports.

7.5.2.8 IDLE

The VIM will indicate IDLE when there are no pending unmasked interrupts or MMR accesses. The VIM does not have a clock stop interface because it does not have a clock IPG. To clock stop the sub-system that includes the VIM, disable (mask) all interrupts and wait for the IDLE signal to assert.

7.5.3 Interrupt Conditions

7.5.3.1 CPU Interrupts

See Table 7-39 for the interrupts generated by the module.

Table 7-39. Interrupts

Interrupt	Description
core0_IRQn	IRQ Interrupt (Active Low)
core0_FIQn	FIQ Interrupt (Active Low)

7.5.3.2 Interrupt Description

7.5.3.2.1 coreN_IRQn

This is a normal interrupt, active low level, for Core N. It can be serviced via the VIC interface or through the MMR interface.

7.5.3.2.2 coreN_FIQn

This is a fast (or non-maskable) interrupt, active low level, for Core N. FIQs always have priority over IRQs. An FIQ can be serviced through the MMR interface.

7.5.3.3 Interrupt Condition Control

7.5.3.3.1 coreN_IRQn

Whenever an interrupt input goes high, if that interrupt is mapped as an IRQ (Group *N* Interrupt Map Register (Base Address + 0x200 + N*0x20 + 0x18)) and is enabled (Group *N* Interrupt Enabled Set Register (Base Address + 0x200 + N*0x20 + 0x08)), and its priority is not masked (4.1.11 IRQ Priority Mask Register (Base Address + 0x28), and it is not masked because of nested interrupt priority masking (2.2.5 Priority Interrupt / Nested Interrupts), then it will cause an IRQ to assert.

7.5.3.3.2 coreN_FIQn

Whenever an interrupt input goes high, if that interrupt is mapped as an FIQ (Group *N* Interrupt Map Register (Base Address + 0x200 + N*0x20 + 0x18)) and is enabled (Group *N* Interrupt Enabled Set Register (Base Address + 0x200 + N*0x20 + 0x08)), and its priority is not masked (4.1.12 FIQ Priority Mask Register (Base Address + 0x2C)), and it is not masked because of nested interrupt priority masking (2.2.5 Priority Interrupt / Nested Interrupts), then it will cause an FIQ to assert.

7.5.3.4 Interrupt Handling

There are multiple ways to service an interrupt depending on how much of the hardware assistance offered by the VIM software wants to take advantage of. For IRQs, it is recommended to use 3.4.1 IRQ through the Vector

Copyright © 2026 Texas Instruments Incorporated

Interface, but 3.4.2 or 3.4.3 (if a user wants to implement a fully software prioritization scheme) may be used as alternatives. For FIQs, it is recommended to use 3.4.4 FIQ, but 3.4.5 may be used as an alternative.

Note

These descriptions do not include steps such as stack pushes and state retention that software must take in order to return from the ISR. It is assumed that the programmer is aware of these steps.

7.5.3.4.1 IRQ through the Vector Interface

If the attached CPU has the Vector Interface enabled, then the following method is used for servicing IRQs

1. Hardware handshake
   a. CPU asserts coreN_IRQACK signal high
   b. VIM asserts coreN_IRQADDRV to indicate that the coreN_IRQADDR bus is stable with the correct Vector Address
   c. CPU reads the coreN_IRQADDR, jumps to that address, and de-asserts coreN_IRQACK signal low
   d. VIM de-asserts coreN_IRQn_intr and coreN_IRQADDRV, VIM masks all IRQs with the same or lower priority
   e. VIM loads the value from the Prioritized IRQ (Base Address + 0x08) (which corresponds to the vector address) to be loaded into the Active IRQ (Base Address + 0x20) and the valid bit to be set
2. Service an interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active IRQ (Base Address + 0x20) to determine number and reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group M Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group M Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group M Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group M Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. 4. Write any value to the IRQ Vector Address (Base Address + 0x18)
   a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
   b. This will clear the valid bit of the Active IRQ (Base Address + 0x20)

7.5.3.4.2 IRQ through MMR Interface

When an IRQ interrupt is received, the CPU should follow these steps if not using the Vector Interface.

1. Read the IRQ Vector Address (Base Address + 0x18) and jump to that address to service the ISR

Copyright © 2026 Texas Instruments Incorporated

a. Reading this register will mask all interrupts of an equal or lower priority and de-assert the IRQn output. If another interrupt of a higher priority becomes available, the IRQn will re-assert, allowing priority interruption of an interrupt.
b. Reading this register will cause the value from the Prioritized IRQ (Base Address + 0x08) (which corresponds to the vector address) to be loaded into the Active IRQ (Base Address + 0x20) and the valid bit to be set

2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active IRQ (Base Address + 0x20) to determine number and reading the Group M Type Map Register (Base Address + 0x200 + M*0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. Write any value to the IRQ Vector Address (Base Address + 0x18)
   a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
   b. This will clear the valid bit of the Active IRQ (Base Address + 0x20)

7.5.3.4.3 IRQ through MMR Interface (Alternative)

If a user does not want to use the Vector Address registers, the VIM may be used as a more traditional interrupt controller. Note that in this mode, priority masking will not work if route 1b is used (below) as the hardware prioritization may not match the software prioritization scheme. In this case, software would be responsible for doing all priority operations

1. Determine which interrupt to service
   a. Read the Prioritized IRQ (Base Address + 0x08) to determine which interrupt is the highest priority IRQ currently asserted OR
   b. Optionally read the IRQ Group Status (Base Address + 0x10) to determine which groups have IRQs pending, then read the Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10) and use a software prioritization scheme to determine which IRQ to service
2. Service the interrupt
3. Read the IRQ Vector Address (Base Address + 0x18)
   a. Note, this step can be done any time before step 4
   b. Value is ignored
4. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)

Copyright © 2026 Texas Instruments Incorporated

a. Pulse
   i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
   ii. Clear the interrupt at the source
      1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

b. Level
   i. Clear the interrupt at the source
   ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)
      1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
      2. If the source maintains the level, then it means there is another interrupt

5. Write any value to the IRQ Vector Address (Base Address + 0x18)

7.5.3.4.4 FIQ

When an FIQ interrupt is received, the CPU should follow these steps.

1. Read the FIQ Vector Address (Base Address + 0x1C) and jump to that address to service the ISR
   a. Reading this register will mask all interrupts of an equal or lower priority and de-assert the FIQn output. If another interrupt of a higher priority becomes available, the FIQn will re-assert, allowing priority interruption of an interrupt.
   b. Reading this register will cause the value from the Prioritized FIQ (Base Address + 0x0C) (which corresponds to the vector address) to be loaded into the Active FIQ (Base Address + 0x24) and the valid bit to be set
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Active FIQ (Base Address + 0x24) to determine number and reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

4. Write any value to the FIQ Vector Address (Base Address + 0x1C)

Copyright © 2026 Texas Instruments Incorporated

a. This will clear the priority mask and all interrupts to be re-evaluated for the new highest priority interrupt.
b. This will clear the valid bit of the Active FIQ (Base Address + 0x24)

7.5.3.4.5 FIQ (Alternative)

If a user does not want to use the Vector Address registers, the VIM may be used as a more traditional interrupt controller. Note that in this mode, there is no hardware priority masking (because the FIQ Vector Address (Base Address + 0x1C) is never read). Software would be responsible for doing all priority operations

1. Determine which interrupt to service
   a. Read the Prioritized FIQ (Base Address + 0x0C) to determine which interrupt is the highest priority FIQ currently asserted OR
   b. Optionally read the FIQ Group Status (Base Address + 0x14) to determine which groups have IRQs pending, then read the Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14) and use a software prioritization scheme to determine which FIQ to service
2. Service the interrupt
3. Depending on whether the original source of the interrupt was a pulse or a level (Determined by reading the Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C) to determine type)
   a. Pulse
      i. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
      ii. Clear the interrupt at the source
         1. This way, the source can generate another pulse if it needs to and the VIM will process this as a new interrupt

   b. Level
      i. Clear the interrupt at the source
      ii. Clear the status by writing a 1 to the appropriate bit in the Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04) or Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)
         1. This way, the level should be gone at the input to the VIM, it will avoid falsely re-calling the interrupt
         2. If the source maintains the level, then it means there is another interrupt

7.5.4 Memory Map

Each Interrupt Core (*N* = 0) has its own MMR interface with its own independent memory map

Table 7-40. Core *N* Memory Map

Address Offset	Register
0x00	Revision Register
0x04	Info Register
0x08	Prioritized IRQ
0x0C	Prioritized FIQ
0x10	IRQ Group Status
0x14	FIQ Group Status
0x18	IRQ Vector Address
0x1C	FIQ Vector Address
0x20	Active IRQ
0x24	Active FIQ

Copyright © 2026 Texas Instruments Incorporated

Table 7-40. Core *N* Memory Map (continued)

Address Offset	Register
0x28-0x2F	*Reserved*
0x30	DED Vector Address
0x34-0x1FF	*Reserved*
0x400 + *M**0x20 + 0x00	Group *M* Interrupt Raw Status/Set Register
0x400 + *M**0x20 + 0x04	Group *M* Interrupt Enabled Status/Clear Register
0x400 + *M**0x20 + 0x08	Group *M* Interrupt Enabled Set Register
0x400 + *M**0x20 + 0x0C	Group *M* Interrupt Enabled Clear Register
0x400 + *M**0x20 + 0x10	Group *M* Interrupt IRQ Enabled Status/Clear Register
0x400 + *M**0x20 + 0x14	Group *M* Interrupt FIQ Enabled Status/Clear Register
0x400 + *M**0x20 + 0x18	Group *M* Interrupt Map Register
0x400 + *M**0x20 + 0x1C	Group *M* Type Map Register
0x1000 + *Q**0x4 – 00x1FFF	Interrupt *Q* Priority Register
0x2000 + *Q**0x4 – 0x2FFF	Interrupt *Q* Vector Register

There are *M* interrupt groups (0 through *num_groups*-1) with 32 interrupts per group.

There are *Q* interrupt inputs where *Q* = *M* * 32.

Accesses to the Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4) must be word-aligned, 32-bit accesses. Any write received with no bytes enabled will be ignored. Any writes receive with all bytes enabled will be executed. Any other write will not execute and will return an error status of Addressing Error.

7.5.5 VIM Registers

7.5.5.1 Revision Register (Base Address + 0x00)

Table 7-41. Revision Register (Base Address + 0x00)

Bits	Field	Type	Reset	Description
31:30	scheme	r	01	Always read as 01. Writes have no effect.
29:28	bu	r	10	Always read as 10.
27:16	func	r	0x090	Always read as the assigned func id.
15:11	rtl	r	0x0	RTL version of the module. The “R” value in the X.Y.R.Z as defined by K3.
10:8	major	r	0x0	Major revision of module. The “X” value in the X.Y.R.Z defined in K3.
7:6	custom	r	0x0	Special version.
5:0	minor	r	0x1	Minor revision of module. The “Y” value in the X.Y.R.Z defined in K3

7.5.5.2 Info Register (Base Address + 0x04)

Table 7-42. Info Register (Base Address + 0x04)

Bits	Field	Type	Reset	Description
31:11	*reserved*	r/o	0	Always read as 0. Writes have no effect.
10:0	Interrupts	r/o	*num_groups* 32*	Indicates the number of interrupts supported by this VIM

Copyright © 2026 Texas Instruments Incorporated

7.5.5.3 Prioritized IRQ (Base Address + 0x08)

Table 7-43. Prioritized IRQ (Base Address + 0x08)

Bits	Field	Type	Reset	Description
31	valid	r/o	0	Indicates that the num field of this register is valid 1 – num field is valid (pending IRQ interrupt) 0 – num field is invalid (no pending IRQ interrupts)
30:20	*reserved*	r/o	0	Always read as 0. Writes have no effect.
19:16	pri	r/o	0	This field indicates the priority of the pending IRQ interrupt. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable
15:10	*reserved*	r/o	0	Always read as 0. Writes have no effect.
9:0	num	r/o	0	This field indicates the interrupt number of the pending IRQ interrupt with the highest priority. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable 0 – Interrupt 0 1 – Interrupt 1 1023 – Interrupt 1023 Note: The highest value is determined by the *num_groups* parameter

7.5.5.4 Prioritized FIQ (Base Address + 0x0C)

Table 7-44. Prioritized FIQ (Base Address + 0x0C)

Bits	Field	Type	Reset	Description
31	valid	r/o	0	Indicates that the num field of this register is valid 1 – num field is valid (pending FIQ interrupt) 0 – num field is invalid (no pending FIQ interrupts)
30:20	*reserved*	r/o	0	Always read as 0. Writes have no effect.
19:16	pri	r/o	0	This field indicates the priority of the pending FIQ interrupt. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable
15:10	*reserved*	r/o	0	Always read as 0. Writes have no effect.
9:0	num	r/o	0	This field indicates the interrupt number of the pending FIQ interrupt with the highest priority. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable 0 – Interrupt 0 1 – Interrupt 1 1023 – Interrupt 1023 Note: The highest value is determined by the *num_groups* parameter

7.5.5.5 IRQ Group Status (Base Address + 0x10)

Table 7-45. IRQ Group Status (Base Address + 0x10)

Bits	Field	Type	Reset	Description
31:0	*sts*	r/o	0	This field indicates that one or more interrupts in Group *M* are mapped to IRQ, unmasked, and pending. Bit 0 corresponds to Group 0; Bit 1 corresponds to Group 1 etc. The interrupts associated with each group are [(*M**32)+31:*M**32]

Copyright © 2026 Texas Instruments Incorporated

7.5.5.6 FIQ Group Status (Base Address + 0x14)

Table 7-46. FIQ Group Status (Base Address + 0x14)

Bits	Field	Type	Reset	Description
31:0	*sts*	r/o	0	This field indicates that one or more interrupts in Group *M* are mapped to FIQ, unmasked, and pending. Bit 0 corresponds to Group 0; Bit 1 corresponds to Group 1 etc. The interrupts associated with each group are [(*M**32)+31:*M**32]

7.5.5.7 IRQ Vector Address (Base Address + 0x18)

Table 7-47. IRQ Vector Address (Base Address + 0x18)

Bits	Field	Type	Reset	Description
31:2	addr	r/w	0x0	This field contains the upper 30 bits of the32-bit interrupt vector address (addresses must be 32-bit aligned) of the currently pending highest priority IRQ (as indicated by the num field of the Prioritized IRQ (Base Address + 0x08). This field is only valid if the valid flag in the Prioritized IRQ (Base Address + 0x08) register is set. • Reading this register returns the Interrupt Vector Address of the pending IRQ with the highest priority. It also has the following effects – Mask all IRQ interrupts of an equivalent or lower priority – De-asserts the IRQn signal – De-asserts the core*N*_IRQADDRV signal (if set) – Loads Active IRQ (Base Address + 0x20) • Writing any value to this register will not alter its contents, but will have the following effect – Remove the mask on all priorities
1:0	*reserved*	r/o	0x0	Reserved. Read as 0. The lower 2 bits of the 32-bit vector address are always 0. Vector addresses must be 32-bit aligned.

7.5.5.8 FIQ Vector Address (Base Address + 0x1C)

Table 7-48. FIQ Vector Address (Base Address + 0x1C)

Bits	Field	Type	Reset	Description
31:0	addr	r/w	0x0	This field contains the upper 30 bits of the 32-bit interrupt vector address (addresses must be 32-bit aligned) of the currently pending highest priority FIQ (as indicated by the num field of the Prioritized FIQ (Base Address + 0x0C). This field is only valid if the valid flag in the Prioritized FIQ (Base Address + 0x0C) register is set. • Reading this register returns the Interrupt Vector Address of the pending FIQ with the highest priority. It also has the following effects – Mask all FIQ interrupts of an equivalent or lower priority – De-asserts the FIQ n signal – Loads Active FIQ (Base Address + 0x24) • Writing any value to this register will not alter its contents, but will have the following effect – Remove the mask on all FIQ priorities
1:0	*reserved*	r/o	0x0	Reserved. Read as 0. The lower 2 bits of the 32-bit vector address are always 0. Vector addresses must be 32-bit aligned.

Copyright © 2026 Texas Instruments Incorporated

7.5.5.9 Active IRQ (Base Address + 0x20)

Table 7-49. Active IRQ (Base Address + 0x20)

Bits	Field	Type	Reset	Description
31	valid	r/o	0	Indicates that the num field of this register is valid. This field is set whenever the IRQ Vector Address (Base Address + 0x18) is read. It is cleared whenever the IRQ Vector Address (Base Address + 0x18) is written. 1 – num field is valid (active IRQ interrupt) 0 – num field is invalid (no active IRQ interrupts)
30:20	*reserved*	r/o	0	Always read as 0. Writes have no effect.
19:16	pri	r/o	0	This field indicates the priority of the active IRQ interrupt. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable
15:10	*reserved*	r/o	0	Always read as 0. Writes have no effect.
9:0	num	r/o	0	This field indicates the interrupt number of the active IRQ interrupt. This field is loaded with the value from the Prioritized IRQ (Base Address + 0x08) whenever the IRQ Vector Address (Base Address + 0x18) is read. This field is only valid if the valid flag (bit 8) is set. Otherwise the value is unpredictable 0 – Interrupt 0 1 – Interrupt 1 1023 – Interrupt 1023 Note: The highest value is determined by the *num_groups* parameter

7.5.5.10 Active FIQ (Base Address + 0x24)

Table 7-50. Active FIQ (Base Address + 0x24)

Bits	Field	Type	Reset	Description
31	valid	r/o	0	Indicates that the num field of this register is valid. This field is set whenever the FIQ Vector Address (Base Address + 0x1C) is read. It is cleared whenever the FIQ Vector Address (Base Address + 0x1C) is written. 1 – num field is valid (active FIQ interrupt) 0 – num field is invalid (no active FIQ interrupts)
30:20	*reserved*	r/o	0	Always read as 0. Writes have no effect.
19:16	pri	r/o	0	This field indicates the priority of the ACTIVE FIQ interrupt. This field is only valid if the valid flag (bit 31) is set. Otherwise the value is unpredictable
15:10	*reserved*	r/o	0	Always read as 0. Writes have no effect.
9:0	num	r/o	0	This field indicates the interrupt number of the active FIQ interrupt. This field is loaded with the value from the Prioritized FIQ (Base Address + 0x0C) whenever the FIQ Vector Address (Base Address + 0x1C) is read. This field is only valid if the valid flag (bit 8) is set. Otherwise the value is unpredictable 0 – Interrupt 0 1 – Interrupt 1 1023 – Interrupt 1023 Note: The highest value is determined by the *num_groups* parameter

7.5.5.11 IRQ Priority Mask Register (Base Address + 0x28)

Table 7-51. IRQ Priority Mask Register (Base Address + 0x28)

Bits	Field	Type	Reset	Description
31:16	*reserved*	r/o	0	Always read as 0. Writes have no effect.

Copyright © 2026 Texas Instruments Incorporated

Table 7-51. IRQ Priority Mask Register (Base Address + 0x28) (continued)

Bits	Field	Type	Reset	Description
15:0	*msk*	r/w	0xFFFF	This field is a bit vector to enable all IRQ interrupts of a given priority. Bit 0 corresponds to priority = 0 (highest) Bit 1 corresponds to priority = 1 etc.… 0 – IRQ of this priority are disabled 1 – IRQ of this priority are enabled

7.5.5.12 FIQ Priority Mask Register (Base Address + 0x2C)

Table 7-52. FIQ Priority Mask Register (Base Address + 0x2C)

Bits	Field	Type	Reset	Description
31:16	*reserved*	r/o	0	Always read as 0. Writes have no effect.
15:0	*msk*	r/w	0xFFFF	This field is a bit vector to enable all FIQ interrupts of a given priority. Bit 0 corresponds to priority = 0 (highest) Bit 1 corresponds to priority = 1 etc.… 0 – FIQ of this priority are disabled 1 – FIQ of this priority are enabled

7.5.5.13 DED Vector Address (Base Address + 0x30)

Table 7-53. DED Vector Address (Base Address + 0x30)

Bits	Field	Type	Reset	Description
31:2	addr	r/w	0x0	This field contains the upper 30 bits of the 32-bit interrupt vector address (the address must be 32-bit aligned) of an interrupt to be used if an uncorrectable double-bit error (DED) is detected in any of the interrupt vector addresses. If there is a DED, both the IRQ Vector Address (Base Address + 0x18) and FIQ Vector Address (Base Address + 0x1C) registers (along with the VECADDR output) will be populated with the value in this field instead of their normal vector. See section Section 7.3.3.2.4 , *Priority Interrupt / Nested Interrupts*. Safety on how to handle interrupts when there has been a DED • ECC Aggregator will indicate where there was a DED. Software may go correct that one location • Software may keep an ISR at the DED Vector which corrects the vectors or otherwise deals with the scenario • This mechanism is provided because there is no way to indicate on the coreN_IRQADDR to the CPU that the vector is invalid, and it is undesirable to have the CPU jump to a random address
1:0	*reserved*	r/o	0x0	Reserved. Read as 0. The lower 2 bits of the 32-bit vector address are always 0. Vector addresses must be 32-bit aligned.

Copyright © 2026 Texas Instruments Incorporated

7.5.5.14 Group *M* Interrupt Raw Status/Set Register (Base Address + 0x400 + *M**0x20 + 0x00)

Table 7-54. Group *M* Interrupt Raw Status/Set Register (Base Address + 0x400 + *M0x20 + 0x00)**

Bits	Field	Type	Reset	Description
31:0	sts	r/w1ts	0x0	This is the raw status of the events in Group *M*. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Inactive 1 – Active/Pending Write: 0 – No effect 1 – Set to Interrupt Raw Status

7.5.5.15 Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x04)

Table 7-55. Group *M* Interrupt Enabled Status/Clear Register (Base Address + 0x400 + *M0x20 + 0x04)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w1tc	0x0	This is the masked status of the events in Group M. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Inactive or Disabled 1 – Active/Pending and Enabled Write: 0 – No effect 1 – Clear Interrupt Raw Status

7.5.5.16 Group *M* Interrupt Enabled Set Register (Base Address + 0x400 + *M**0x20 + 0x08)

Table 7-56. Group *M* Interrupt Enabled Set Register (Base Address + 0x400 + *M0x20 + 0x08)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w1ts	0x0	This field is used to enable the mask of events in Group *M*. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Disabled 1 – Enabled Write: 0 – No effect 1 – Set Enable

Copyright © 2026 Texas Instruments Incorporated

7.5.5.17 Group *M* Interrupt Enabled Clear Register (Base Address + 0x400 + *M**0x20 + 0x0C)

Table 7-57. Group *M* Interrupt Enabled Clear Register (Base Address + 0x400 + *M0x20 + 0x0C)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w1tc	0x0	This field is used to disable the mask of events in Group *M*. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Disabled 1 – Enabled Write: 0 – No effect 1 – Clear Enable

7.5.5.18 Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x10)

Table 7-58. Group *M* Interrupt IRQ Enabled Status/Clear Register (Base Address + 0x400 + *M0x20 + 0x10)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w1tc	0x0	This is the masked status of the events in group *M* that are mapped to IRQ. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Inactive, Disabled, or not an IRQ 1 – Active/Pending, Enabled, and IRQ Write: 0 – No effect 1 – Clear Interrupt Raw Status (if IRQ)

7.5.5.19 Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M**0x20 + 0x14)

Table 7-59. Group *M* Interrupt FIQ Enabled Status/Clear Register (Base Address + 0x400 + *M0x20 + 0x14)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w1tc	0x0	This is the masked status of the events in group *M* that are mapped to FIQ. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) Read: 0 – Inactive, Disabled, or not an FIQ 1 – Active/Pending, Enabled, and FIQ Write: 0 – No effect 1 – Clear Interrupt Raw Status (if FIQ)

Copyright © 2026 Texas Instruments Incorporated

7.5.5.20 Group *M* Interrupt Map Register (Base Address + 0x400 + *M**0x20 + 0x18)

Table 7-60. Group *M* Interrupt Map Register (Base Address + 0x400 + *M0x20 + 0x18)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w	0x0	This field is used to indicate which interrupt the corresponding event influences (if enabled) for event group *M*. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) 0 – IRQ Interrupt (default) 1 – FIQ Interrupt

7.5.5.21 Group *M* Type Map Register (Base Address + 0x400 + *M**0x20 + 0x1C)

Table 7-61. Group *M* Type Map Register (Base Address + 0x400 + *M0x20 + 0x1C)**

Bits	Field	Type	Reset	Description
31:0	msk	r/w	0x0	This field is used to indicate whether the source of an interrupt is a level (default) or a pulse for event group *M*. This is informational so that an ISR may query this register and know whether it has to clear a pulse event or a level event *(see Section 7.3.3.3.4 Interrupt Handling)*. The value has no effect on how the VIM hardware functions. The input interrupts are agnostic as to whether they are pulse or level. Each bit corresponds to event *Q* where *Q* = *M**32+Bit (Example: bit 0 is event *M**32+0, bit 1 is *M**32 + 1 etc…) 0 – Level (default) 1 – Pulse

7.5.5.22 Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q**0x4)

Table 7-62. Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q0x4)**

Bits	Field	Type	Reset	Description
31:4	*reserved*	r/o	0	Always read as 0. Writes have no effect.
3:0	pri	r/w	0xF	This is the priority for interrupt *Q*. If two interrupts have the same priority, then whichever interrupt has the lower number *Q* wins arbitration 0 – Highest Priority 1 …. 14 15 – Lowest Priority (Default)

Copyright © 2026 Texas Instruments Incorporated

7.5.5.23 Interrupt Q Vector Register (Base Address + 0x2000 + Q*0x4)

Table 7-63. Interrupt Q Vector Register (Base Address + 0x2000 + Q*0x4)

Bits	Field	Type	Reset	Description
31:2	addr	r/w		This is the 32-bit Vector Address associated with interrupt *Q*. It is the address that will be reflected in the IRQ Vector Address (Base Address + 0x18) or FIQ Vector Address (Base Address + 0x1C) and the VECADDR pin when interrupt *Q* is the active interrupt. Internally, these values are kept in a RAM. The FIQ and IRQ state machines have priority access to this RAM. Writes to this register will be piped internally, but further writes to the MMR interface may be stalled until this write has a chance to complete in the RAM. The new Vector Address will not take effect until this write completes to the RAM. In order to tell if this write has completed, software may read this register back. That read will not be able to complete unless the write has landed. Reads to this register will stall the MMR interface until the read is able to be completed at the RAM.
1:0	*reserved*	r/o	0x0	Reserved. Read as 0. The lower 2 bits of the 32-bit vector address are always 0. Vector addresses must be 32-bit aligned.

7.5.6 Module I/O

7.5.6.1 Clocks, Reset, Emulation

Table 7-64. Clocks, Reset, Emulation

Pin Name	Type	Function
core0_clk	In	Clock for Core 0
core0_phase	In	Phase Indicator for Core 0
core0_srst_n	In	Sync Module Reset for Core 0
core0_arst_n	In	Async Module Reset for Core 0
core0_clkstop_idle	Out	Clock Stop Idle

7.5.6.2 VBUSP Target Interface

(N=0). Core0 interface is used for interrupt core 0.

Table 7-65. VBUSP Target Interface

Pin Name	Type	Function
core*N*_cfg_req	In	Request
core*N*_cfg_address[13:0]	In	Address (byte address)
core*N*_cfg_dir	In	Direction (read or write)
core*N*_cfg_xcnt[2:0]	In	Transaction Count
core*N*_cfg_byten[3:0]	In	Byte enables
core*N*_cfg_wdata[31:0]	In	Write data
core*N*_cfg_rdatap[31:0]	Out	Read data
core*N*_cfg_rready	Out	Read ready
core*N*_cfg_wready	Out	Write ready
core*N*_cfg_rstatus[2:0]	Out	Read Status
core*N*_cfg_emudbg	In	Emulation Debug

Copyright © 2026 Texas Instruments Incorporated

7.5.6.3 Interrupt Inputs

Core0 interface is used for interrupt core 0.

Table 7-66. Interrupt Inputs

Pin Name	Type	Function
core0_in_intr[*Q*-1:0]	In	Core 0 Interrupt Inputs

7.5.6.4 Interrupt Outputs

Table 7-67. Interrupt Outputs

Pin Name	Type	Function
core0_IRQn_intr	Out	IRQ Interrupt for core 0– Active Low Level
core0_FIQn_intr	Out	FIQ Interrupt for core 0 – Active Low Level

7.5.6.5 VIC Interfaces

(*N*=0). Core0 interface is used for interrupt core 0.

Table 7-68. VIC Interfaces

Pin Name	Type	Function
core*N*_vic_IRQADDRV	Out	IRQ Address Valid
core*N*_vic_IRQADDR[31:2]	Out	IRQ Address
core*N*_vic_IRQACK	In	Interrupt Acknowledge

7.5.6.6 Compare Outputs

These pins will never be on an isolation boundary and therefore their isolation values are don't-care. *Y* is dependent on the configuration

Table 7-69. Compare Outputs

Pin Name	Type	Function
core0_compare_bus[*Y*:0]	Out	Comparison Bus for Core0 to CCMR5

7.5.6.7 ECC Control and Status Bus

(*N*=0). Core0 interface is used for interrupts for core 0.

Table 7-70. ECC Control and Status Bus

Signal Name	Dir	Default Val	Description
core*N*_ramecc_strb	In	1'b0	Strobe to sample the incoming serial data
core*N*_ramecc_txd	In	1'b0	Output serial data
core*N*_ramecc_rxd	Out	1'b0	Input serial data
core*N*_ramecc_pend[1:0]	Out	2'd0	Level Interrupt source

7.5.6.8 DFT

Table 7-71. DFT

Signal Name	Dir	Default Val	Description
dft_clk_force_en	In	1'b0	DFT Clock Force Enable
dft_partition_enn	In	1'b0	DFT Clock Partition Enable
dft_scan_en	In	1'b0	DFT Scan Enable
dft_async_rst_sel	In	1’b0	DFT Async Reset Select
dft_test_async_rst_n	In	1’b0	DFT Async Test Reset
dft_tft_mcp_dis	In	1’b0	DFT MCP Disable

Copyright © 2026 Texas Instruments Incorporated

Table 7-71. DFT (continued)

Signal Name	Dir	Default Val	Description
dft_local_clk_en	In	1'b0	DFT Local Clock Enable

7.5.6.9 RAM GPIO

Table 7-72. RAM GPIO

Signal Name	Dir	Default Val	Description
core*N*_ramdft_gpi[255:0]	In	256'd0	RAM GPI
core*N*_ramdft_gpo[255:0]	Out	256'd0	RAM GPO

7.5.7 Programmer's Guide

7.5.7.1 Initialization Sequence

At initialization, the following tasks should be performed:

1. Program the DED Vector Address (Base Address + 0x30)
2. Program Group *M* Interrupt Map Register (Base Address + 0x200 + *M**0x20 + 0x18)
3. Program Group *M* Type Map Register (Base Address + 0x200 + *M**0x20 + 0x1C)
4. Program Interrupt *Q* Priority Register (Base Address + 0x1000 + *Q**0x4)
5. Program Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4)
6. Enable interrupts via the Group *M* Interrupt Enabled Set Register (Base Address + 0x400 + *M**0x20 + 0x08)

Note that the Interrupt *Q* Vector Register (Base Address + 0x2000 + *Q**0x4) for each interrupt that will be enabled must be written before the interrupt is enabled, even if software is not going to use the vector. Whenever an interrupt is prioritized, the RAM location for the interrupt is read, and if that location is un-initialized, an ECC error will be reported. If the vectors aren't being used, then the RAM can be initialized by writing 0x0 to every location.

7.5.7.2 DED Behavior

Whenever there is 2-bit error detect on a Vector Address, the DED Vector Address (Base Address + 0x30) overrides all other vectors. Software should provide an ISR to handle this scenario at this address.

7.5.7.3 Power Up/Down Sequence

Before a clean power down, software should check that there are no pending interrupts, disable all interrupts, and wait for the clkstop_idle output.

7.6 Video Encoder/Decoder (VENC/VDEC)

The following sections describe the video accelerator details for the device.

7.6.1 Introduction

The Video Accelerator is a 4K codec that supports both HEVC and H.264/AVC video formats. It provides high performance encode and decode capability up to 8bit 4K@60fps with a single-core architecture.

The Video Accelerator can encode and/or decode any resolution up to 8192 x 4320. It guarantees real-time performance for encoding/decoding 4K 60fps based on its sophisticated, latency tolerant hardware architecture. The Video Accelerator is highly optimized for memory bandwidth loading and excellent power management.

The Video Accelerator contains a 32-bit processor called V-CPU, which is responsible for parsing bitstream syntax in decoder orencoding bitstream syntax in encoder from sequence to slice header unit, pre-scanning slice data, controlling the underlying video hardware blocks called V-CORE. The V-CPU also communicates with host CPU through host register interface. The V-CORE performs actual processing of coded slice data: entropy decoding, inverse scan, inverse transform/quantization, motion compensation, and loop filtering in decoder and motion estimation, intra prediction, RDO, and entropy coding in encoder. This software and hardware combined architecture can provide flexibility and high throughput at the same time.

Copyright © 2026 Texas Instruments Incorporated

7.6.2 Features

7.6.2.1 Performance

H.265/HEVC Encoder

- Capable of encoding HEVC Main and Main Still Picture Profile @ L5.1 High tier
  - Maximum resolution: 8192x8192
    - The maximum resolution means the largest image size that the hardware can process with. Performance is not considered in this term.
  - Minimum resolution: 256x128
  - Constraints
    - A picture width shall be multiple of 8.
    - A picture height shall be multiple of 8.

Table 7-73. HEVC Encoder Performance

PicWidth	PicHeight	MHz/60fps 160Mbps
3840	2160	500

H.264/AVC Encoder

- Capable of encoding Baseline/Constrained Baseline/Main/High Profiles Level @ L5.2
  - Maximum resolution: 8192x8192
    - The maximum resolution means the largest image size that the hardware can process with. Performance is not considered in this term.
  - Minimum resolution: 256x128
  - Constraints
    - A picture width shall be multiple of 8.
    - A picture height shall be multiple of 8.

Table 7-74. H.264/AVC Encoder Performance

PicWidth	PicHeight	MHz/60fps 72Mbps
3840	2160	500

H.265/HEVC Decoder

- Capable of decoding HEVC Main and Main Still Picture Profile @ L5.1 High tier
  - Maximum resolution: 8192x4320
    - The Maximum resolution means the largest image size that the hardware can process with. Performance is not considered in this term.
  - Minimum resolution: 8x8

Table 7-75. H.265/HEVC Decoder Performance

PicWidth	PicHeight	MHz/60fps 160Mbps
3840	2160	450.00

H.264/AVC Decoder

- Capable of decoding Baseline/Constrained Baseline/Main/High Profiles @ L5.2
  - Maximum resolution: 8192x4320
    - The Maximum resolution means the largest image size that the hardware can process with. Performance is not considered in this term.
  - Minimum resolution: 32x32

Copyright © 2026 Texas Instruments Incorporated

Table 7-76. H.264/AVC Decoder Performance

PicWidth	PicHeight	MHz/60fps 72Mbps
3840	2160	450.00

7.6.2.2 Codec Related Features

H.265/HEVC Encoder

- Fully compatible with ISO/IEC 23008-2 high efficiency video coding main profile
- I/P slices
- CTU64
  - Supportable prediction unit (PU) size: 32 x 32, 16 x 16, 8 x 8
  - Supportable transform unit (TU) size: 32 x 32 to 4 x 4
- Parallel tools
  - Wavefront parallel processing (WPP) encoding with a single slice
  - Multi slice: Independent slice segment and dependent slice segment
- High performance offline CABAC encoding
- Motion estimation
  - 1/4-pel precision motion vectors
  - Search range [+/-128H, +/-64V] with an adaptive search center
  - Two reference frames for P-slice
  - Long-term reference for P picture
- Custom tuning tools
  - Custom Lambda map and lambda table
  - Custom mode decision
  - Fully programmable user scaling list
- In-loop filter
  - Deblocking filter
  - Sample adaptive offset (SAO)
  - Loop filtering across slices
- Strong intra smoothing on/off
- Transform skip
- Lossless coding
- Picture/CTU/subCTU level of rate control
- Region of interest (ROI) encoding with custom QP map
- 3DNR
- Adaptive intra refresh (AIR) for error resilience

H.264/AVC Encoder

- Compatible with the ITU-T recommendation H.264 specification. All coding tools in the baseline, con- strained baseline, main, and high profiles are supported.
  - With a few exceptions:
    - Interlaced coding tools are not supported.
    - FMO/ASO tool of H.264 is not supported.
- 16 x 16, 8 x 8 and 4 x 4 block sizes are supported and configurable.
- Motion estimation
  - 1/4-pel accuracy motion estimation with programmable search range up to [+/-64, +/-48]
  - One reference frame for P-slice
- Intra prediction
  - Luma I4 x 4 Mode: 9 modes
  - Luma I8 x 8 Mode: 9 modes
  - Luma I16 x 16 Mode: 4 modes (vertical, horizon, DC, plane)
  - Chroma mode: 3 modes (vertical, horizon, DC)
- Custom tuning tools

Copyright © 2026 Texas Instruments Incorporated

  - User-defined mode (skip, intra) map
  - User-defined QP map
  - Lambda tuning for custom mode decision
  - Fully programmable user scaling list
- In-loop deblocking filter
- CABAC/CAVLC support
- Error resilience tools:
  - Cyclic intra refresh (CIR)
  - Multi-slice structure
- A frame level and MB level of rate control
- Region of interest (ROI) encoding with custom QP map

H.265/HEVC decoder

- Fully compatible with ISO/IEC 23008-2 high efficiency video coding main/MSP (main still picture) profile. All coding tools in the profile are supported.
- I/P/B slices
  - All intra-prediction modes
  - All inter-prediction modes
- Variable CTU size: 64 x 64 to 16 x 16
  - Variable prediction unit (PU) size: 64 x 64 to 4 x 4
  - Variable transform unit (TU) size: 32 x 32 to 4 x 4
- Advanced motion vector prediction (AMVP) and merge mode
- A quarter motion compensation with 8 tap filters
- Uniform reconstruction quantization (URQ)
- Parallel coding tools
  - Multi tile
  - Wavefront parallel processing (WPP)-encoded bitstream support
  - Multi slice: Independent slice segment and dependent slice segment
- High performance CABAC decoding
- In-loop deblocking filtering
- Sample adaptive offset (SAO)
- Loop filtering across slice/tile boundaries
- Data reporting to the external host
- Robust error concealment
- Sequence change detection

H.264/AVC Decoder

- Fully compatible with the ITU-T recommendation H.264 specification. All coding tools in the baseline/constrained baseline/main/high profile are supported.
  - With a few exceptions:
    - Interlaced coding tools are not supported.
    - FMO/ASO tool of H.264 is not supported.
- Variable block size (16 x 16, 16 x 8, 8 x 16, 8 x 8, 8 x 4, 4 x 8 and 4 x 4)
- CABAC/CAVLC support
- In-loop deblocking filter
- Error detection, concealment and error resilience tools

7.6.2.3 Non-Codec Related Features

Rotation and mirroring

The Video Accelerator supports 8 modes of rotation and mirroring. The PRP block can do rotations of the source picture in 90, 180, or 270 degrees and vertical or horizontal flip before starting to encode the operation.

Bit-depth and chroma format conversion

Copyright © 2026 Texas Instruments Incorporated

The PRP block is responsible for source format conversion in the encoder IP. 422 (planar/semi-planar) to 420 conversion can be made before encoding operation.

Frame buffer compression

To overcome bandwidth suffering and at the same time to ensure real-time encode/decode performance, TI has carefully designed lossless, great access patterned frame buffer compression technology and employed it into the WAVE IP series. WAVE521CL achieves significant reduction in bandwidth while sustaining fast processing capability.

Long Burst Write

The Video Accelerator includes burst write back (BWB) module that enables 128-byte burst write access to external memory for better bus utilization. It collects a group of short write requests from loop filter block in 128-byte unit and sends it out to the external memory more efficiently. BWB works for an external Display module or other post processing module that needs to read decoded frames in raster scan order.

The FBC/FBD block also writes or reads 128-byte burst length of compressed frame data from/to the externalmemory.

3DNR

The Video Accelerator employs temporal noise reduction technology, also called 3DNR (3D noise reduction), for better quality of image under low light. Video noise appears easily in a low light level. It is an important issue in surveillance camera because images are captured often under dark conditions or indoors, which can make identification difficult.

The Video Accelerator can detect and filter noise for each color component Y, Cb, and Cr of a frame. This achieves both good video quality and enhancement of coding efficiency. Noise reduction is done while encoding. Therefore, additional read/write bandwidth is not required.

Note

3DNR is supported in the H.265/HEVC encoder only.

Latency tolerance

The Video Accelerator can afford to reach real time performance under memory access latency if the number of cycles per CTU retains less than 500 cycles. The Video Accelerator is designed to be less sensitive to pipeline delay, especially at a peak bitrate, which might eventually incur performance drop. With use of decoupling techniques and inter-pipe queues, it can hide this sort of delay and deliver high performance at any situation.

Subframe synchronization

The Video Accelerator supports subframe synchronization for low latency coding. By receiving dedicated subframe-ready signals from the image process unit (IPU) or by setting host interface registers from host CPU, the Video Accelerator can start encoding process as early as the minimum set of raw video data is ready.

Programmability

The Video Accelerator embeds the 32-bit CPU and 16-bit DSP dedicated to bitstream processing and their video hardware control. Host interface registers and interrupts are provided for communication between a host processor and the Video Accelerator.

Low Power Consumption

The Video Accelerator can ensure ultra-low power operation by gating the clock sources for some internal hardware blocks in an idle state.

Trick mode

The Video Accelerator supports trick mode such as fast/slow forward playback and fast/slow reverse playback for video applications. By seeking an intra random access point (IRAP) picture as entry points of bitstream, the

Copyright © 2026 Texas Instruments Incorporated

Video Accelerator can skip non-IRAP pictures and start decoding from an IRAP picture or decode a thumbnail of a picture.

Frame-based processing

V-CPU processor operates video operation on a frame by frame basis. While frame operation is running, there is no need for communication between the host processor and the Video Accelerator. This is the key feature for promising the lowest burden to host processor for video operations.

By issuing a picture processing, the host application can perform its own operations until it receives an interrupt from the Video Accelerator informing of the completion of picture processing.

Handling multi-instances

The Video Accelerator supports multiple instances, which can be helpful for multi-channel encoder and/or decoder applications. To support this multi-instance operation, the Video Accelerator uses an internal context parameter set for each instance. While creating a new instance and starting picture processing, a set of these context parameters is created and updated automatically in the Video Accelerator. Because of this internal context management scheme, different encoder and/or decoder tasks running on the host processor can control Video Accelerator operations independently with their own instances.

When creating a new instance, an application task will get a new handle specifying an instance if a new handle is available on the Video Accelerator. All of the following operations for the given application task could be separately handled on the Video Accelerator by using this task-specific handle. Because the Video Accelerator can only perform one picture processing task at a time, each application task should check whether the Video Accelerator is ready or not before starting a new picture operation. By calling a function for closing a certain instance, the application can easily terminate a single task of video operation on the Video Accelerator.

7.6.3 Block Diagram

Figure 7-20 shows the hierarchical architecture of the Video Accelerator and external bus interfaces.

Copyright © 2026 Texas Instruments Incorporated

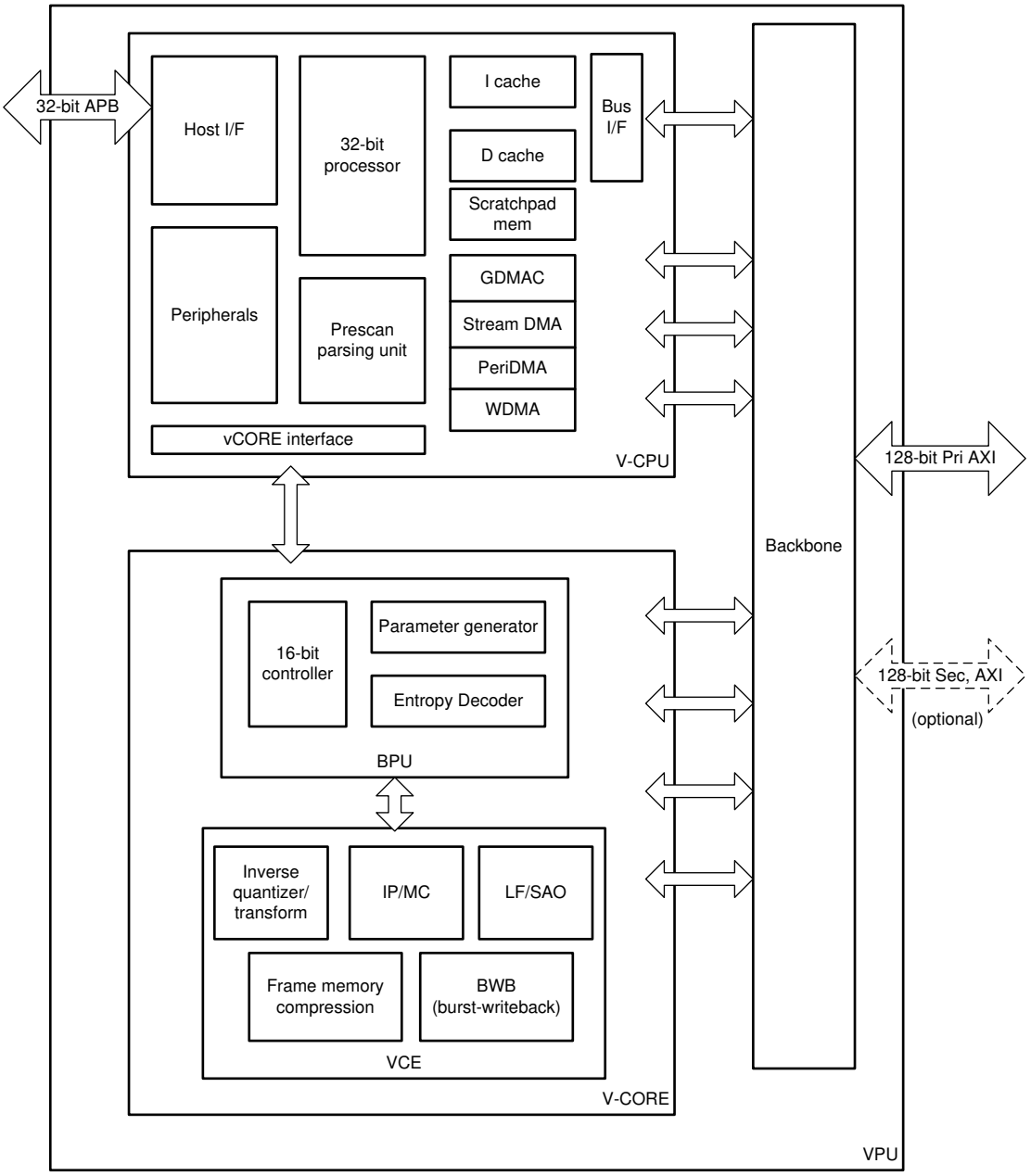


Figure 7-20. Video Accelerator Block Diagram

The Video Accelerator is built on a layer structure for efficient video processing. It is composed of two main parts, V-CPU and V-CORE. V-CPU is a 32-bit processor where the Video Accelerator L1 driver software is running. It mainly encodes and decodes a high-level syntax of bitstream - SPS/PPS/SH, and communicates with the host processor, such as receiving a picture level command from the host processor or sending the result of video task through the 32-bit AMBA3 APB bus. V-CPU can also parse necessary parameters and give necessary slice/tile-level information to the underlying V-CORE hardware.

Particularly in case of an HEVC decoder, V-CPU uses a virtual wavefront parallel processing (VWPP) scheme to balance processing load among V-COREs and to simplify its operation. During the VWPP, V-CPU parses bitstream and extracts some data from bitstream such as CABAC context, QP, and start address for each row and tile. After that, V-CPU reorders the data to meet the WPP processing sequence and dispatches them to the appropriate V-COREs.

Copyright © 2026 Texas Instruments Incorporated

V-CORE can encode and decode a slice unit of data. It consists of a 16-bit DSP called a bit processor unit (BPU) and a group of video codec hardware blocks called a video codec engine (VCE).

- BPU is responsible for packing and parsing of coded slice data, CTU/CU/PU/TU-level parameter derivation, and finally passing slice data over to the VCE. To speed up entropy encoding/decoding, some hardware accelerators are included in the BPU.
- On the other hand, VCE is an actual hardware core executing a series of decoding process - encoding and decoding process - motion estimation, intra-prediction, RDO, and entropy coding with their hardwired logics, motion compensation, inverse-transformation/quantization, and loop filter with their hardwired logics. HEVC and H.264/AVC decoders share most of the internal VCE memories and full logics in certain VCE blocks like intra-prediction for small gate count.

They are doing these tasks with CTU-based pipelines and FIFO queues to ensure real-time speed and performance. When it comes to system connection as shown in the Figure 7-20, there is one 32-bit AMBA3 APB interface connecting to the host processor, and there are two 128-bit AMBA3 AXI bus interfaces connecting to the external memory controller for reading and writing picture data and temporal data.

- Primary AXI: AXI interface to read or write work data and coded picture (bitstream), decoded picture data and so forth.
- Secondary AXI (optional): On-chip-SRAM connected AXI interface to read or write temporal buffer required for V-CORE operation. This is an optional interface that can be used to alleviate bandwidth usage.

Copyright © 2026 Texas Instruments Incorporated

7.7 Vision Pre-processing Accelerator (VPAC)

This chapter describes the Vision Pre-processing Accelerator (VPAC) in the device.

7.7.1 VPAC Overview

The Vision Pre-processing Accelerator (VPAC) subsystem is a set of common vision primitive functions, performing pixel data processing tasks, such as: color processing and enhancement, noise filtering, wide dynamic range (WDR) processing, lens distortion correction, pixel remap for de-warping, on-the-fly scale generation, on-the-fly pyramid generation. The VPAC offloads these common tasks from the main SoC processors (ARM, DSP, etc.), so these CPUs can be utilized for differentiated high-lelvel algorithms. The VPAC is designed to support multiple cameras by working in time-multiplexing mode. The VPAC also includes an imaging pipe, which can be integrated on-the-fly with external camera sensor, as well as does memory-to-memory (M2M) processing on pixel data.

Figure 7-21 provides an overview of the VPAC subsystem.

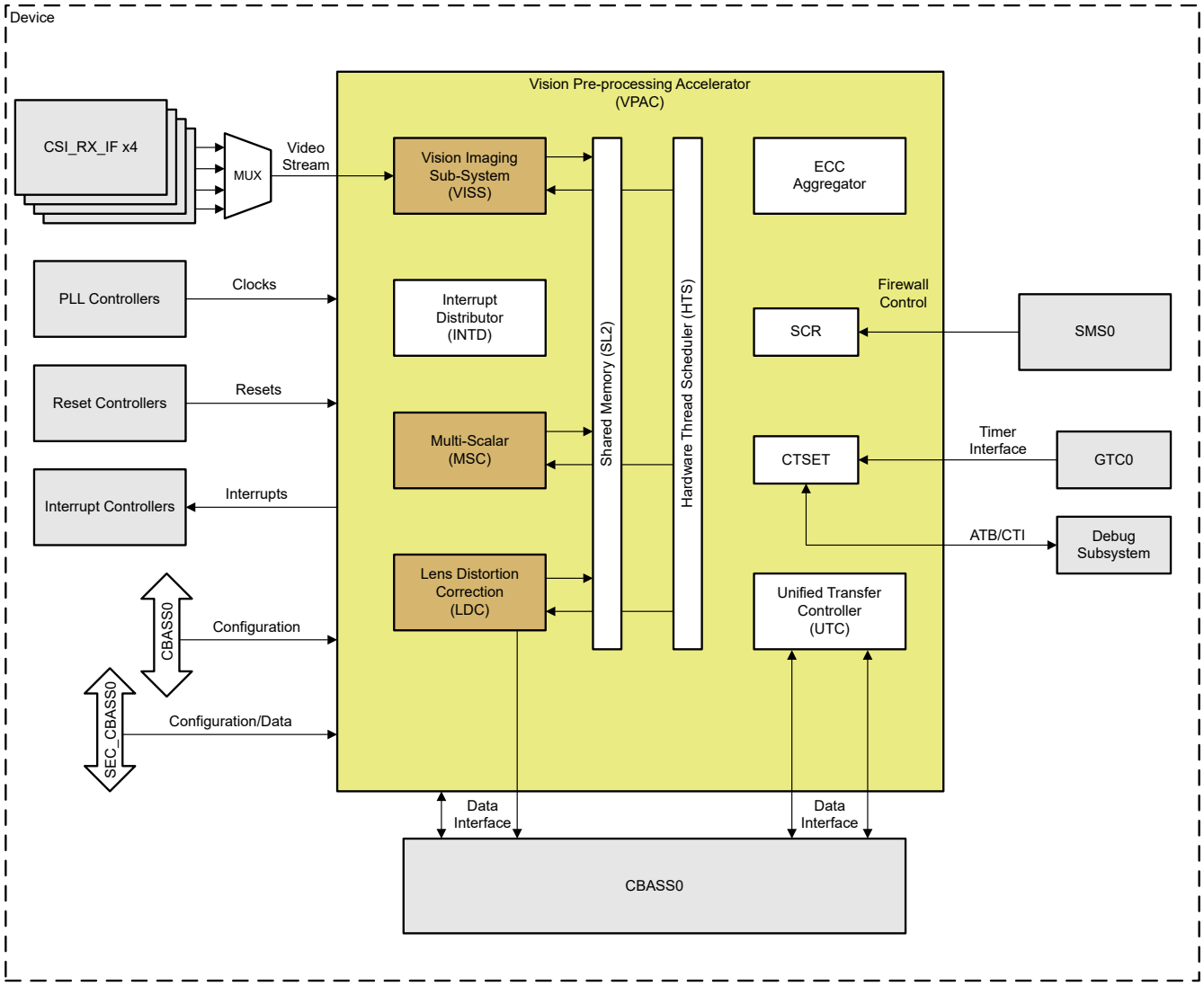


Figure 7-21. VPAC Overview

7.7.1.1 VPAC Features

The VPAC includes the following processing and infrastructure sub-modules:

Copyright © 2026 Texas Instruments Incorporated

- Vision Imaging Sub-System (VISS). The vision imaging pipe does image processing on raw data which includes Wide Dynamic Range (WDR) merge, Defect Pixel Correction (DPC), Lens Shading Correction (LSC), Global/Local Brightness and Contrast Enhancement (GLBCE), Spatial Noise Filtering, demosaicing, color conversion, and Edge Enhancement (EE). It can operate on sensor data either on-the-fly or in memory-to-memory mode. The VISS provides the following main features:
  - Performance is up to 1 pixel/cycle
  - Up to 16-bit input RAW formats (8b, 12b, 14b, 16b) (see Section 7.7.2.7)
    - Support for generic 2x2 RAW format (Bayer, RCCB, RCCC, etc.)
    - Support for 4x4 RGB-IR RAW format
  - Processing up to 4096 pixels/line (8MP frame)
  - Support for concurrent 12-bit and 8-bit YUV output
  - Support for other color spaces:
    - RGB output
    - Support for color saturation and gray scale for HSV/HSL, etc.
  - Support for multiple WDR/HDR formats (for 2x2 RAW formats):
    - Combined Companded format (up to 16-bit companded)
    - Separated Exposures – up to 3 frame Motion Adaptive HDR Merge
    - Staggered HDR
    - Digital Lateral Overlap (DLO)
  - Support of statistics for Auto-Exposure/Auto-White-Balance or Auto-Focus (configurable option to select either AE/AWB or AF)
  - Support for Flexible CFA for generating multi-plane output for any 2x2 RAW format sensor configuration.
    - Supports RCBC, RCCC, Bayer, RGBC as well as other formats
    - Can generate up to 4 independent color planes
    - Support for up two 3 directions per color plane
    - Adaptive threshold based on intensity measure
  - Supports pattern conversion (remosaicing RGB-IR(4x4) CFA format raw data into 2x2 Bayer pattern) and IR demosaicing
  - Support for multiple color format output generation
    - Supports traditional 12-bit YUV output
    - Support for 8-bit YUV output) (see Section 7.7.2.7)
    - Supports RGB output (8-bit only)
    - Supports saturation outputs (8 bits only)
    - Supports Grayscale/Luma/IR/Clear output
  - Global/Local Brightness and Contrast Enhancement module (16-bit)
  - Spatial Noise Filter (NSF4V)
    - Supports raw input image in generic 2x2 color pattern format including Bayer pattern
    - Supports 16-bit raw input image
    - Spatially adaptive noise level threshold
    - Configurable soft or hard thresholding
    - Supports elliptical lens shading gain adjustment of adaptive noise threshold with 2 sets of independent settings
    - White Balance gain
    - Signal level detection with flexible cross-color-channel or independent averaging
  - Option to bypass visual enhancement functions (NSF4V and GLBCE)
  - Edge Enhancement (EE)
    - Enhance the visual quality of the image (Luma only)
    - Edge enhancer filter and the edge sharpener filter can be combined by configuration
    - Option to bypass visual enhancement functions (CAC, NSF4V, and GLBCE)
- Lens Distortion Correction (LDC) block. The LDC engine is YUV domain processor designed to perform perspective and geometric transforms. This can be used for creating several effects, including but not limited to:
  - Lens Distortion Correction, including Fish Eye Lenses

Copyright © 2026 Texas Instruments Incorporated

  - Epipolar rectification for Stereo
  - Generic perspective transforms for rotation scaling or Augmented Reality like effects

  The output of the LDC block can be sent to external memory (DDR) or sent to other VPAC sub-module (MSC) for further pre-processing via local shared memory (SL2). The LDC block provides the following main features:

  - Performance up to 1 cycle/pixel (bilinear) and 2 cycles/pixel (bicubic)
  - Autonomous memory-to-memory operation
  - Tile based processing
  - Generic mesh based distortion model to correct multiple distortion types
  - Perspective warp transformation for perspective correction; affine transform is a subset of perspective warp and supports scaling and rotation
  - Supported formats (see Section 7.7.2.7):
    - YUV420 12-bit packet input and YUV420 12-bit packed output
    - YUV420 12-bit unpacked input and YUV420 12-bit unpacked output
    - YUV420 8-bit packet input and YUV420 8-bit packed output
    - YUV422 8-bit packed input and YUV420 8-bit packed output
    - YUV422 8-bit packed input and YUV422 8-bit packed output
  - Supports up to 8192 x 8192 image dimension
  - Pixel Interpolation
    - Bicubic interpolation for Y and bilinear interpolation for Cb/Cr
    - Bilinear interpolation mode to offer double throughput
  - Support of Luma Only or Chroma Only Mode
  - Supports variable block size frame processing (9 regions)
- Multi-Scalar (MSC) block. The MSC block reads data from memory (DDR or on-chip) to shared memory (SL2) and generates up to 10 scaled outputs from a given input with varios scaling ratios (between 1x and x0.25). The output of the MSC block can be sent to external memory (DDR) from the local shared memory (SL2). The MSC supports two independent threads (scalars), which combined can generate up to max 10 scales. The MSC block provides the following main features:
  - Multi-scaling capable: 10 simultaneous scaled outputs from 1 or 2 input planes; up to 2 simultaneous processing thread with total of 10 scaled output with 1 to N and 1 to M configurations (where N+M <=10)
  - Each scaling engine can be configured to perform Pyramid or inter-octave scale generation
  - Scaling ratio supported: 1x ~ x/4 (with 1x scaling, MSC can perform a 3~5-tap generic separable convolution filtering)
  - 5-tap separable filter kernel implementation
  - Programmable kernel sizes for vertical/horizontal filter (3, 4, or 5)
  - Poly phase implementation with support of 64 or 32 phases poly phase support
  - 4 sets of 5-tap x 32 phase coefficients (two can be combined to support 5-tap x 64 phase coefficients and one of set can hold additional 5-tap customer filter coefficients for non-resizing filter)
  - 2 additional sets of 5-tap Gaussian filter coefficients (single-phase) dedicated for Pyramid (Octave) generation
  - Coefficients in 10-bit (S10Q8 format - signed value of 10 bits with 8-bit fraction)
  - Separate initial horizontal and vertical phase programming option
  - Support for two planes of data processing
    - Data in a plane can be uniform (for example, Y plane) or interleaved (Cb/Cr interleaved)
  - 8-bit or 12-bit component data
  - Performance up to 1 input pixel/cycle
  - Output data rate is dependent on downscaling ratio
  - Programmable input and output ROI (or clipping) for each scaler
  - On-the-fly (OTF) or memory-to-memory (M2M) operation with line buffers in SL2 (frame resolution support is not constrained by MSC line buffer)
- Hardware Thread Scheduler (HTS). The HTS block used for inter-processor communication among various VPAC sub-modules (VISS, LDC, MSC) and with the local DMA Engine (UTC). The HTS block provides the following main features:

Copyright © 2026 Texas Instruments Incorporated

  - Scheduling the HWA threads and associated tasks
  - Enables rate-controlled task execution
  - Support running several independent threads asynchronously in the sub-system
  - Debug functions:
    - Halting HWA threads at logical task boundary
    - Handling pipeline suspend and subsequent abort by software intervention
- Shared Level 2 (SL2) memory. The SL2 is used as intermediate storage for Hardware Accelerator (HWA) to exchange data across VPAC sub-modules (that is, VISS, LDC, MSC) and from DDR/System Memory. VPAC supports Realtime and Non-Realtime processing threads simultaneously.

Copyright © 2026 Texas Instruments Incorporated

7.7.2 VPAC Subsystem Level

This section covers VPAC subsystem-level details.

7.7.2.1 VPAC Subsystem Block Diagram

The VPAC subsystem includes imaging pipe to do image processing on raw pixels (bayer, RCCC, etc.). The VPAC subsystem loads pixel data either from memory (for example, DDR or on-chip memory), or captured from camera sensor (via external module on SoC level), and applies many vision primitive functions such as image processing, scaling to generate image pyramids, noise filtering, correcting lens distortion, applying perspective transformation for stereo rectification, etc. Optionally, the VPAC subsystem can also do dynamic range management on received image data. The output image processed data generated by the VPAC subsystem is written into memory (for example, DDR or on-chip). The VPAC subsystem supports software managed and HTS controlled flexible load/store of data using K3 DMA infrastructure sub-modules.

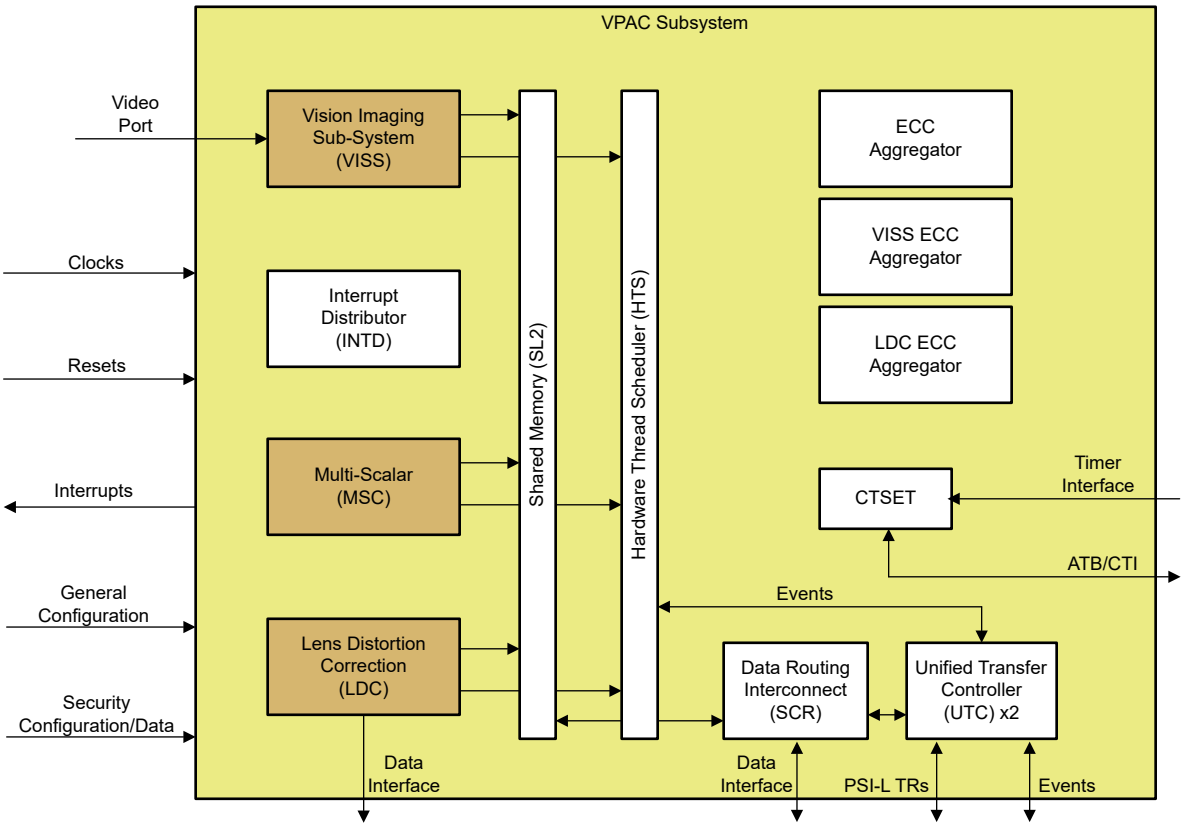


Figure 7-22. VPAC Subsystem Block Diagram

The VPAC subsystem includes the following image processing harware accelerators (HWAs) and infrastructure blocks:

- Vision Imaging Subsystem (VISS). The VISS does on-the-fly (OTF) processing on raw pixels captured from camera sensor. It also does memory to memory processing for data captured from other sources to its parallel port or video port. The VISS uses VPAC infrastructure modules (HTS, DRU/UTC, interrupt aggregator, etc.) for flow and data management. For more information on the VISS operation and the sub-modules integrated inside, see Section *VPAC Vision Imaging Subsystem (VISS)*. The VISS consists of following Hardware Accelerators (HWA):
  - RAW Front End (RFE) block: The RFE block does RAW pixel (that is, Bayer, RCCC, RGBW, 4x4 RGBIR, etc.) processing on captured image data from sensor. The processed data is later passed on to the NSF4V block, or otherwise it is passed onto the Flexible Color Proccesing (FCP) block for demosaicing and color conversion. In case of the 4x4 RGBIR image data passed on to Pattern Conversion and IR Demosaicing (PCID) before being processed by the downstream blocks. The RFE includes also the H3A

Copyright © 2026 Texas Instruments Incorporated

block, which supports the control loops for auto focus (AF), auto white balance (AWB) and auto exposure (AE) by computing image statistics.
  - Pattern Conversion and IR Demosaicing (PCID) module: The PCID module is used only to process the 4x4 RGB-IR CFA pattern RAW data to up-sample/demosaic IR channel to full resolution and re-mosaic the input pattern to create 2x2 Bayer CFA pattern data to be processed by existing VISS blocks. The PCID module can also be used to decontaminate the 2x2 Bayer CFA pattern data of IR contamination. The PCID module can further remap the demosaiced IR channel image to compress the dynamic range. Further the 2x2 Bayer CFA pattern generated by the PCID block can be looped back to the H3A block within RAWFE to collect AE/AWB statistics.
  - Global Local Brightness Contrast Enhancement (GLBCE) module: The GLBCE module is used for dynamic range control within image for visual quality. If a contrast enhancement on the input image is required for visual quality, the RFE output is processed by the NSF4v and GLBCE blocks and then passed onto the FCP block. Otherwise, if contrast enhancement is not required, the NSF4v and GLBCE blocks can be bypassed, and the RFE output is directly provided to the FCP block.
  - Flexible Color Processing (FCP) block: The FCP block receives data from GLBCE and does demosaicing and color conversion. The output of the color processing is sent to the internal EE (Edge Enhancer) block, if it is enabled, before being ouput to VPAC shared memory. Otherwise, the FCP output data is directly sent to VPAC shared memory to be written into external memory for further vision processing by programmable processors (that is, DSP or Arm), or other vision hardware accelerators at SoC level.
  - Load/Store Engine (LSE): The LSE is an infrastructure block, which provides the following functions:
    - Receives data, captured through MIPI CSI-2 image sensor (by external module at SoC level), for on-the-fly image processing to reduce DDR bandwidth and latency. LSE also provides horizontal and vertical blanking cycles to allow core data path to settle at proper boundary of line and frame.
    - Provides access to VPAC SL2 memory for loading/storing data. Loaded data from SL2 are passed on to unpacker function of LSE for RFE. Packed data after FCP/H3A processing is written into SL2.
    - Event control. HTS events are generated at line level. These events are routed to RFE to start the processing. For each consumed line HTS needs *task done* indication. For some initial lines consumed, there would not be any valid data output. Similarly, H3A generated data will be at paxel/window height number of lines.
- Lens Distortion Correction (LDC) module. The LDC module deals with lens geometric distortion issues in the camera system. The distortion can be common optical distortions, such as barrel distortion, pin-cushion distortion, or fisheye distortion. Correction is not limited to just these types of distortions. The LDC module consists of a Back Mapping block (which gives coordinates of the distorted image as a function of coordinates of the undistorted output image), a delta_x/delta_y offset table, frame buffer inetrface, buffer, an interpolation block, and SL2 interface. The frame buffer is external to the LDC module, and is usually in an off-chip SDRAM. The LDC module uses the common VPAC infratructure modules (HTS, DRU/UTC, interrupt aggregator, etc.) for flow and data management, except for loading input data for which it has its own DMA engine. For more details on the LDC operation, see Section *VPAC Lens Distortion Correction (LDC)*.
- Multi-Scaler (MSC) module. The MSC consists of 10 programmable resizers performing multi-thread/multi-scaling operations (2-input to N-outputs, and 2-input to M-outputs, where N+M is 10 or less). Each processing thread of MSC reads its input plane data from the SL2 circular line buffer, performs multi-scaling operations (ratios between X and 0.25X) on the same input, and writes out results to SL2 circular line buffers. In case of on-the-fly operation, the source data is generated from another VPAC HWA. In case of memory-to-memory operation, the source data is read from the external memory. Data transfers from/to SL2 to/from external memory (or another HWA) are handled by the VPAC DMA controller (DRU/UTC), with transfer request events coming from the VPAC HTS module. The MSC module uses the VPAC infrastrucutre modules (HTS, DRU/UTC, interrupt aggregator, etc.) for flow and data management. For more details on the the MSC operation, see Section *VPAC Subsystem Multi-Scaler (MSC)*.
- Hardware Thread Scheduler (HTS). The HTS module is a messaging layer for low-overhead synchronization of the parallel computing tasks and DMA transfers, and is independent from the host processor. It allows autonomous frame level processing for the VPAC subsystem. The HTS module defines various aspects of synchronization and data sharing between the VPAC HWAs. With regards to producer and consumer dependencies, the HTS module ensures that a task starts only when input data and adequate space to write out data is available. In addition to this, it also takes care of pipe-up, debug, abort and interrupt management

Copyright © 2026 Texas Instruments Incorporated

aspects related to the VPAC HWAs. For more details on the HTS operation, see Section *VPAC Hardware Thread Scheduler (HTS)*.
- Common shared level 2 (SL2) memory. The SL2 memory subsystem implements a full crossbar inteconnect, and serves as input/output scratch memory for the VPAC HWAs.
- Unified Transfer Controller (UTC): UTC is a VPAC DMA engine to handle real-time and nonreal-time data transfers.
- Data Routing Unit. The SCR acts like a 256-bit data routing interconnect within the VPAC subsystem, and is used to route UTC traffic onto SL2 memory or for external (system level) access.
- Counter, Timer and System Event Trace (CTSET) module. The CTSET module provides event tracing capabilities for debug purposes.
- ECC Aggregator. The ECC Aggregator modules provide a mechanism to control and monitor certain internal ECC RAMs via Single Error Correction (SEC) and Double Error Detection (DED) functions.

7.7.2.1.1 Notes on VISS RFE H3A Usage

The H3A module supports the control loops for auto focus (AF), auto white balance (AWB) and auto exposure (AE) by computing image statistics. The statistics are typically used by the host CPU to adjust the various parameters for processing the image data. The H3A module comprises two sub modules: the AF and AE/AWB. The H3A module, which is integrated inside RFE module of VISS, can be configured to produce either AE/AWB or AF statistics in a sigle run. Data generated is stored into a separate SL2 memory buffer by writes of 32-Byte data in a single burst. There is support of only one H3A output channel between LSE and HTS.

The H3A output data is a single dimensional data and the amount of data produced depends on number of windows/paxel in a frame, varions functional modes and corner cases of multiple of 8 windows/paxels in a row, finishing row level statistics at 32-Byte boundary, etc. Table 7-77 shows the size of data produced, depending on H3A configuration.

Table 7-77. VPAC Subsystem H3A Output Data Size

Register Settings		Number of Bytes	Special care
AF H3A_PCR[0] AF_EN = ‘1’	Vertical AF disabled H3A_PCR[20] AF_VF_EN = ‘0’	48 Bytes/paxel	If paxel_no_x odd, add 16B per row to make it 32B multiple per row.
	Vertical AF enabled H3A_PCR[20] AF_VF_EN = ‘1’	64 Bytes/paxel	None
AE and AWB H3A_PCR[16] AEW_EN = ‘1’	Sum of square H3A_AEW_CFG[9-8] AEFMT = ‘0’	34 Bytes/window	(+16B) After 8 windows (+16B) At end of row, but no end of windows, if H3A size at Row != 32B (+16B) Num of windows %8 !=0 (+16B) At end of windows, if H3A size at end of windows != 32B
	Min/Max H3A_AEW_CFG[9-8] AEFMT = ‘1’	34 Bytes/window	
	Sum only H3A_AEW_CFG[9-8] AEFMT = ‘2’	18 Bytes/window	

The parameters of H3A output buffer need to be configured via VISS LSE register settings. Some special considerations include:
- Configure the H3A line size to be more or equal to maximum H3A data produced in a row. That is, VISS_LSE_BUF_ATTR[15-6] H3A_LN_SIZE >= ceil((# of output bytes per H3A paxel/window row)/64)
- The H3A output data is 1-dimensional in the external storage (DDR or on-chip memory). For ease of synchronization, this 1-D data is converted into multiple of H3A line data. The TR (transfer request) written to transfer H3A data from SL2 to external storage must make sure to configure offset such that data get stored as if 1-D continuous data.
- Configure the minimum VISS LSE_BUF_ATTR[24-16] CBUF_SIZE = 3, to take care of corner cases.

7.7.2.2 VPAC Subsystem Clocks

Each block within the VPAC subsystem receives its own clocking signal:

Copyright © 2026 Texas Instruments Incorporated

- MAIN_CLK is the operation clock for all infrastructure modules, such as UTC/DRU, HTS, ECC Aggregator, INTD, etc.
- VISS0_CLK is the operation clock for the VISS and its sub-modules
- LDC0_CLK is the operation clock for the LDC module
- MSC_CLK is the operation clock for the MSC module

For more details on the source of each clock at SoC level, see *VPAC Integration*.

7.7.2.3 VPAC Subsystem Resets

Each block within the VPAC subsystem can be reset through its own HW reset signal:

- MAIN_RST is the reset for all infrastructure modules, such as UTC/DRU, HTS, ECC Aggregator, etc.
- VISS0_RST is the reset for the VISS and its sub-modules
- LDC0_RST is the reset for the LDC module
- MSC_RST is the reset for the MSC module

For more details on the source of each reset signal at SoC level, see *VPAC Integration*.

7.7.2.4 VPAC Subsystem Interrupts

The VPAC subsystem outputs six physical system interrupt lines: VPAC_LEVEL_0_INTR through VPAC_LEVEL_5_INTR. For more information on how the interrupt lines are connected at SoC level, see *VPAC Integration*. The interrupt aggregation within the VPAC subsystem is done using the internal INTD module. Each interrupt line can be mapped to the same list of source interrupt events generated by the modules within the VPAC subsystem.

Table 7-78 shows the INTD interrupt control registers for each VPAC interrupt line. Each interrupt line has a separate set of registers for level and pulse input event types. Each bit in a register corresponds to a particular interrupt event.

- The ENABLE registers enable the mapping of the internal VPAC interrupt events, generated by a module within the VPAC subsystem, to the VPAC output system interrupt lines. Writing a '1' in a register bit position allows the corresponding module interrupt event to trigger the respective system interrupt. Writing a '0' in the same bit position blocks the module interrupt event from triggering the system interrupt.
- The ENABLE_CLR registers disables the mapping of the internal VPAC interrupt events to the VPAC output system interrupt lines.
- The STATUS registers are used to capture which VPAC sub-module interrupt events, that have been mapped to a system interrupt, are active and pending. This status is used when multiple module interrupts are grouped into a single system interrupt, so the software can read the status register to determine exactly which module interrupt has caused the system interrupt.
  - For level interrupts, the status is based on the active level of the module interrupt input, and when the input is inactive then the status is inactive as well.
  - For pulsed interrupts, the status is set upon the pulse and reflects a pending status and software must clear the pending status since the module does not indicate the inactive status (unlike a level interrupt, which does).
- The STATUS_CLR registers allows software to clear the pending status of interrupt events. This register is only valid and functional for pulsed interrupts and has no effect for level interrupts.

Copyright © 2026 Texas Instruments Incorporated

Table 7-78. VPAC Subsystem Interrupt Control Registers

Output System Interrupt Line	Input Interrupt Event Type	Interrupt Enable Registers	Interrupt Clear Registers	Interrupt Status Registers	Interrupt Status Clear Registers
VPAC_LEVEL_0_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_0_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_0_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_0_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_0_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_0_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_0_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_0_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_0_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_0_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_0_7	VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_0_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_0_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_0_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_0_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_0_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_0_7
VPAC_LEVEL_1_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_1_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_1_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_1_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_1_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_1_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_1_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_1_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_1_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_1_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_1_7	VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_1_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_1_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_1_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_1_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_1_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_1_7
VPAC_LEVEL_2_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_2_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_2_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_2_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_2_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_2_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_2_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_2_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_2_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_2_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_2_7	VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_2_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_2_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_2_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_2_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_2_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_2_7
VPAC_LEVEL_3_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_3_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_3_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_3_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_3_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_3_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_3_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_3_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_3_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_3_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_3_7	PAC_INTD_ENABLE_CLR_REG_PULSE_VPAC_OUT_3_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_3_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_3_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_3_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_3_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_3_7

Copyright © 2026 Texas Instruments Incorporated

Table 7-78. VPAC Subsystem Interrupt Control Registers (continued)

Output System Interrupt Line	Input Interrupt Event Type	Interrupt Enable Registers	Interrupt Clear Registers	Interrupt Status Registers	Interrupt Status Clear Registers
VPAC_LEVEL_4_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_4_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_4_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_4_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_4_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_4_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_4_7
VPAC_LEVEL_5_INTR	Level	VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_CFG_ENABLE_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_CFG_STATUS_REG_LEVEL_VPAC_OUT_5_7	VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_5_0 through VPAC_INTD_CFG_STATUS_CLR_REG_LEVEL_VPAC_OUT_5_7
	Pulse	VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_CFG_ENABLE_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_CFG_ENABLE_CLR_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_CFG_STATUS_REG_PULSE_VPAC_OUT_5_7	VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_5_0 through VPAC_INTD_CFG_STATUS_CLR_REG_PULSE_VPAC_OUT_5_7

The VISS interrupt events are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_0 through VPAC_INTD_xxx_VPAC_OUT_5_0 registers. Table 7-79 provides more details on the VISS interrupt events. All VISS interrupts are single pulse event signals.

Table 7-79. VPAC Subsystem VISS Interrupt Events

Register Bit	Interrupt Event Name	Description
0	RAWFE_CFG_ERR_INTR	Config read or write memory acccess occurred during functional operation and likely corrupted functional opertion. VISS merges all config error sources from RawFE and see section *RAWFE Interrupts* for the entire error source list. h3a_line_cfg_err_intr and h3a_accm_cfg_err_intr sources can be generated errornously on access outside of active frame window to H3A Line and Accumalator memories respectvely if none of the H3A functions are enable in previous frame.
1	RAWFE_AEW_PULSE_INTR	H3A AEW interrupt.
2	RAWFE_AF_PULSE_INTR	H3A AF interrupt.
3	RAWFE_H3A_PULSE_INTR	H3A interrupt.
4	RAWFE_H3A_BUF_OVRFLOW_PULSE_INTR	H3A output buffer overflow.
5	NSF4V_LINEMEM_CFG_ERR_INTR	VBUSP diagnostic read access of RAM, while NSF data using RAM for functional purpose.
6	NSF4V_HBLANK_ERR_INTR	Horizontal Blanking too short between lines.
7	NSF4V_VBLANK_ERR_INTR	Vertical Blanking too short between frames.
8	GLBCE_CFG_ERR_INTR	Either non-shadowed registers written or statastic memories are accessed during active window.
9	GLBCE_FILT_START_INTR	GLBCE started filtering. This interrupt is issued at the rising edge of filtering signal.
10	GLBCE_FILT_DONE_INTR	GLBCE ended filtering. This interrupt is issued at the falling edge of filtering signal.
11	GLBCE_HSYNC_ERR_INTR	Generated when delayed HS/HE signals doesn't match with derived signals from GLBCE core.

Copyright © 2026 Texas Instruments Incorporated

Table 7-79. VPAC Subsystem VISS Interrupt Events (continued)

Register Bit	Interrupt Event Name	Description
12	GLBCE_VSYNC_ERR_INTR	Generated when delayed VS/VE signals doesn't match with derived signals from GLBCE core.
13	GLBCE_VP_ERR_INTR	This interrupt is issued, if there is a data input while filtering is high.
14	FCFA_CFG_ERR_INTR	Either non-shadowed registers written or line memories are accessed during active window.
15	FCC_CFG_ERR_INTR	Configuration access to registers/memories has corrupted functional operation.
		Configuration read or write memory acccess occurred during functional operation.
		Merged independent error sources at VISS. Refer to Section *VISS Flexible Color Processing (FCP)* for all sources.
16	FCC_OUTIF_OVF_ERR_INTR	FIFO overflow on FIFO for Y12 LSE I/F. Merged all FCC output overflow error sources at VISS. Refer to Section *VISS Flexible Color Processing (FCP)* for all sources.
17	FCC_HIST_READ_ERR_INTR	Host was not able to read the entire histogram mem between VS-VE window (triggered when the first access to histogram has been performed but the last has not been performed). Error won't be reported if Histogram Read starts in Vertical Blanking. Instead use the SW sequence suggested in the spec to read the FCC histogram
18	EE_CFG_ERR	Cinfiguration happened to EE regions causing corruption during frame processing.
19	EE_SYNCOVF_ERR	EE horizontal synchronization FIFO overflow interrupt.
20	LSE_FR_DONE_EVT_INTR	LSE frame done interrupt.
21	LSE_SL2_RD_ERR_INTR	Set whenever there is an error response on VBUSM read command for any input channel.
22	LSE_SL2_WR_ERR_INTR	Set whenever there is an error response on VBUSM write command for any output channel.
23	LSE_CAL_VP_ERR_INTR	Set whenever one of the following input frame errors is detected at VPORT_INPUT.
24	LSE_OUT_FR_START_EVT_INTR	Ouput Frame Start (from VISS top level).
25	RAWFE_X_Y_POINTER_INTR	X,Y pixel position has reached the start of RAWFE pipeline interrupt.
26	VISS_CR_CFG_ERR_INTR	Config read/write access happened to CAC MeshLUT or PCIR IR RemapLUT when CAC or PCID data is active (active window - IN_VS to OUT_VE).
27	RAWFE_DPC_STATS_READ_ERR_INTR	Host was not able to read the entire DPC corrected location memory VS-VE window (Triggered when the first access to statastics memory has been performed but the last has not been performed)

The LDC interrupt events are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_1 through VPAC_INTD_xxx_VPAC_OUT_5_1 registers. Table 7-80 provides more details on the LDC interrupt events. All LDC interrupts are single pulse event signals.

Table 7-80. VPAC Subsystem LDC Interrupt Events

Register Bit	Interrupt Event Name	Description
0	LDC_PIX_IBLK_OUTOFBOUND_INTR	Back mapped pixel co-ordinate goes out of the pre-computed input pixel bounding box.
1	LDC_MESH_IBLK_OUTOFBOUND_INTR	Block mesh co-ordinate goes out of the pre-computed mesh bounding box.
2	LDC_PIX_IBLK_MEMOVF_INTR	Input Pixel block memory overflow.
3	LDC_MESH_IBLK_MEMOVF_INTR	Mesh block memory overflow.
4	LDC_IFR_OUTOFBOUND_INTR	Back mapped input co-ordinate goes out of input frame range.
5	LDC_INT_SZOVF_INTR	Affine and perspective transform precision overflow error.
6	LDC_FR_DONE_EVT_INTR	Frame done LDC.
7	LDC_SL2_WR_ERR_INTR	Error on SL2 VBSUM Write interface.
8	LDC_VBUSM_RD_ERR_INTR	Error on Input VBUSM Read interface.

Copyright © 2026 Texas Instruments Incorporated

The MSC interrupt events are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_2 through VPAC_INTD_xxx_VPAC_OUT_5_2 registers. Table 7-81 provides more details on the MSC interrupt events. All MSC interrupts are single pulse event signals.

Table 7-81. VPAC Subsystem MSC Interrupt Events

Register Bit	Interrupt Event Name	Description
0	MSC_LSE_FR_DONE_EVT_0_INTR	Frame done MSC processing thread 0.
1	MSC_LSE_FR_DONE_EVT_1_INTR	Frame done MSC processing thread 1.
2	MSC_LSE_SL2_RD_ERR_INTR	Error on SL2 VBSUM Read interface.
3	MSC_LSE_SL2_WR_ERR_INTR	Error on SL2 VBSUM Write interface.

The HTS interrupt events are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_3 through VPAC_INTD_xxx_VPAC_OUT_5_3 registers. Table 7-82 provides more details on the HTS interrupt events. All HTS interrupts are single pulse event signals, save for the SPARE_PEND_xxx interrupts, which can be both single pulse events and level events.

Table 7-82. VPAC Subsystem HTS Interrupt Events

Register Bit	Interrupt Event Name	Description
0	PIPE_DONE_0	Pipeline 0 done.
1	PIPE_DONE_1	Pipeline 1 done.
2	PIPE_DONE_2	Pipeline 2 done.
3	PIPE_DONE_3	Pipeline 3 done.
4	PIPE_DONE_4	Pipeline 4 done.
5	PIPE_DONE_5	Pipeline 5 done.
6	PIPE_DONE_6	Pipeline 6 done.
7	TDONE	HWA0 thread done.
8	RESERVED	Reserved.
9	TDONE	HWA2 thread done.
10	RESERVED	Reserved.
11	TDONE	HWA4 thread done.
12	TDONE	HWA5 thread done.
13	TDONE	HWA6 thread done.
14	SPARE_DEC_0	Spare 0 scheduler decrement pulse (ehost mode).
15	SPARE_DEC_1	Spare 1 scheduler decrement pulse (ehost mode).
16	SPARE_PEND_0_PULSE	Spare 0 scheduler pend assertion (ehost mode).
17	SPARE_PEND_0_LEVEL	Spare 0 scheduler pend assertion (ehost mode).
18	SPARE_PEND_1_PULSE	Spare 1 scheduler pend assertion (ehost mode).
19	SPARE_PEND_1_LEVEL	Spare 1 scheduler pend assertion (ehost mode).
20	WATCHDOGTIMER_ERR_0	Watchdog Timeout Error for HWA0.
21	RESERVED	Reserved.
22	WATCHDOGTIMER_ERR_2	Watchdog Timeout Error for HWA2.
23	RESERVED	Reserved.
24	WATCHDOGTIMER_ERR_4	Watchdog Timeout Error for HWA4.
25	WATCHDOGTIMER_ERR_5	Watchdog Timeout Error for HWA5.
26	WATCHDOGTIMER_ERR_6	Watchdog Timeout Error for HWA6.

The UTC interrupt events indicating TR complete are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_4 through VPAC_INTD_xxx_VPAC_OUT_5_4 registers for UTC0, and VPAC_INTD_xxx_VPAC_OUT_0_5 through VPAC_INTD_xxx_VPAC_OUT_5_5 plus

Copyright © 2026 Texas Instruments Incorporated

VPAC_INTD_xxx_VPAC_OUT_0_6 through VPAC_INTD_xxx_VPAC_OUT_5_6 registers for UTC1. Table 7-83 provides more details on the UTC TR complete interrupt events. All UTC TR complete interrupts are single pulse event signals.

Table 7-83. VPAC Subsystem UTC TR Complete Interrupt Events

Register Bit	Interrupt Event Name	Description
31-0	UTC0_COMPLETE_INTR[31:0]	TR complete interrupt.
31-0 plus 31-0	RESERVED	Reserved.

The UTC interrupt events indicating error are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_7 through VPAC_INTD_xxx_VPAC_OUT_5_7 registers for both UTC0 and UTC1. Table 7-84 provides more details on the UTC error interrupt events. All UTC error interrupts are single pulse event signals.

Table 7-84. VPAC Subsystem UTC Error Interrupt Events

Register Bit	Interrupt Event Name	Description
0	UTC0_ERR	UTC0 error.
1	RESERVED	Reserved.
2	UTC0_PROT_ERR_INTR	UTC0 protocol violation.
3	RESERVED	Reserved.

The CTSET interrupt events are mapped to register bits within the VPAC_INTD_xxx_VPAC_OUT_0_7 through VPAC_INTD_xxx_VPAC_OUT_5_7 registers. Table 7-85 provides more details on the CTSET events. All CTSET error interrupts are single pulse event signals.

Table 7-85. VPAC Subsystem CTSET Interrupt Events

Register Bit	Interrupt Event Name	Description
4	CTM_PULSE_INTR	Counter Timer Module (CTM) pulse interrupt.

7.7.2.5 VPAC Subsystem SL2 Memory Infrastructure

The SL2 memory subsystem implements an interconnect between DMA and HWAs initiators. It contains 512KB Shared L2 memory to acts temporary local memory to transfer data between external memory and HWAs.

The SL2 memory does not implement ECC, as it contains mostly pixel data.

7.7.2.6 VPAC Subsystem DMA Infrastructure

Two UTC instances are used inside the VPAC subsystem. UTC0 deals with real time (RT) transfer , and UTC1 is used for non-realtime (NRT) transfer. Both UTC0 and UTC1 data transfer channels are used for DMA transfers in and out of the VPAC SL2 memory. All 32 channels of the UTC0 are mapped on *channel_number = [95:64]* and are recommended to be used for real time data transfer only. All 64 channels of UTC1 are mapped on *channel_number = [63:0]* and are recommended to be used for all other DMA transfer needs of the VPAC subsystem.

All data movement transactions at VPAC subsystem boundary are multiples of aligned 128-byte transfers.

7.7.2.7 VPAC Subsystem Data Formats Support

Table 7-86 summarizes the data formats supported in VPAC Subsystem.

Table 7-86. VPAC Subsystem Data Formats Support

No	Module	Input			Output		
		Bit-depth	Chroma (NV12)	Packing	Bit-depth	Chroma (NV12)	Packing

Copyright © 2026 Texas Instruments Incorporated

Table 7-86. VPAC Subsystem Data Formats Support (continued)

No	Module	Input			Output		
1	VISS	12	RAW	no	12 & 8	420	yes
2		14	RAW	no	12 & 8	420	yes
3		16	RAW	no	12 & 8	420	yes
4		8	RAW	yes	8	420	yes
5		12	RAW	yes	12 & 8	420	yes
5		All of the above			8	422	yes
6	LDC	8	422	yes	8	420	yes
7		8	422	yes	8	422	yes
8		8	420	yes	8	420	yes
9		12	420	yes/no	12	420	yes
10		12	420	yes/no	12 & 8	420	no
11	MSC	12	Planar	yes	12	planar	yes
12		8	Planar	yes	8	planar	yes

7.7.2.8 VPAC Subsystem Debug Features

The VPAC subsystem is a combination of multiple asynchronous pipelines running concurrently. The Counter, Timer and System Event Trace (CTSET) module within the VPAC susbystem provides the following main features:

- Monitoring up to 255 VPAC events.
- ATB interface compliant trace packets (synchronous to VPAC clock domain).
- Counter (count=8) and Timer (count=4).
- All debug events can be configured to be either selected for trace packet, timer or counter features.
- In one of the combination, debug events can be selected to for timer event out to be used to generate external CTI compliant trigger.

For more information on the CTSET module operation, see Chapter *On-Chip Debug*.

Table 7-87 lists the VPAC subsystem events that are mapped on the CTSET module.

Table 7-87. VPAC Subsystem CTSET Event List

Index	Event Name	HWA	Event Type	Event Description
0	viss_err	VISS0	pulse	VISS Related Error message (config, ovf, sync, blank, access, vp) (1)
1	rawfe_h3a_pulse_intr	VISS0	pulse	H3A interrupt
2	rawfe_dbg_ctl_vs_event	VISS0	pulse	Vertical start in the beginning of the RAWFE pipeline
3	rawfe_dbg_ctl_ve_event	VISS0	pulse	Vertical end in the beginning of the RAWFE pipeline
4	rawfe_dbg_ctl_hs_event	VISS0	pulse	Horizontal start in the beginning of the RAWFE pipeline
5	rawfe_dbg_ctl_he_event	VISS0	pulse	Horizontal end in the beginning of the RAWFE pipeline
6	rawfe_dbg_ctl_lse_slv_stall_event	VISS0	pulse	RAWFE is stalling lse target interface due to FCP or MTC stall
7	rawfe_dbg_ctl_lse_mst_stall_event	VISS0	pulse	H3A controller interface to LSE is stalled
8	rawfe_dbg_ctl_lse_intf_stall_event	VISS0	pulse	Within a frame (VS to VE at RFE i/p) LSE is not sending data to RFE
9	rawfe_dbg_ctl_x_y_match_event	VISS0	pulse	X,Y pixel position has reached the start of RAWFE pipeline
10	rawfe_dbg_ctl_dpc_otf_corr_event	VISS0	pulse	DPC OTF corrected a pixel position
11	rawfe_dbg_ctl_pipe_adv_event	VISS0	pulse	Active pipeline advancement

Copyright © 2026 Texas Instruments Incorporated

Table 7-87. VPAC Subsystem CTSET Event List (continued)

Index	Event Name	HWA	Event Type	Event Description
12	nsf4v_in_hs_event	VISS0	pulse	Start of Active Line at NSF4V input
13	nsf4v_in_vs_event	VISS0	pulse	Start of Active Frame at NSF4V input
14	nsf4v_in_he_event	VISS0	pulse	End of Active Line at NSF4V output
15	nsf4v_in_ve_event	VISS0	pulse	End of Active Frame at NSF4V output
16	glbce_filt_start_intr	VISS0	pulse	GLBCE filtering start (generated at the rising edge of filtering signal)
17	glbce_filt_done_intr	VISS0	pulse	GLBCE filtering done (generated at falling edge of filtering signal)
18	glbce_sol_event	VISS0	pulse	Start of Active Line at GLBCE/CAC/ PCID input. See VISS_TOP DBGEVT_CTL register for details.
19	glbce_sof_event	VISS0	pulse	Start of Active Frame at GLBCE/CAC/ PCID input. See VISS_TOP DBGEVT_CTL register for details.
20	glbce_eol_event	VISS0	pulse	End of Active Line at GLBCE/CAC/PCID output. See VISS_TOP DBGEVT_CTL register for details.
21	glbce_eof_event	VISS0	pulse	End of Active frame at GLBCE/CAC/PCID output. See VISS_TOP DBGEVT_CTL register for details.
22	fcfa_sol_event	VISS0	pulse	Start of line processing for CFA input
23	fcfa_sof_event	VISS0	pulse	Start of frame processing for CFA input
24	fcc_stall_event	VISS0	pulse	Stall has occurred on any one of the output LSE interfaces
25	fcc_eol_if_y12_event	VISS0	pulse	End of line on Y12 LSE I/f (only generated for valid lines)
26	fcc_eof_if_y12_event	VISS0	pulse	End of frame on Y12 LSE I/f
27	fcc_eol_if_uv12_event	VISS0	pulse	End of line on UV12 LSE I/f (only generated for valid lines)
28	fcc_eof_if_uv12_event	VISS0	pulse	End of frame on UV12 LSE I/f
29	fcc_eol_if_y8r8_event	VISS0	pulse	End of line on Y8R8 LSE I/f (only generated for valid lines)
30	fcc_eof_if_y8r8_event	VISS0	pulse	End of frame on Y8R8 LSE I/f
31	fcc_eol_if_c8g8_event	VISS0	pulse	End of line on C8G8 LSE I/f (only generated for valid lines)
32	fcc_eof_if_c8g8_event	VISS0	pulse	End of frame on C8G8 LSE I/f
33	fcc_eol_if_s8b8_event	VISS0	pulse	End of line on S8B8 LSE I/f (only generated for valid lines)
34	fcc_eof_if_s8b8_event	VISS0	pulse	End of frame on S8B8 LSE I/f
35	fcc_flexcc_eop_event	VISS0	pulse	End of processing (EOP) for FlexCC
36	lse_fr_done_evt_intr	VISS0	pulse	Frame processing done event
37	lse_out_fr_start_evt_intr	VISS0	pulse	Travelled Frame Start from LSE Core
38	Reserved			
39	ldc_err	LCD0	pulse	LDC processing error (2)
40	mesh_iblk_fetch_start_event	LCD0	pulse	Input mesh block fetch start event
41	mesh_iblk_fetch_done_event	LCD0	pulse	Input mesh block fetch completion event
42	pix_iblk_fetch_start_event	LCD0	pulse	Input pixel block fetch start event (Active for both Y and C)
43	pix_iblk_fetch_done_event	LCD0	pulse	Input pixel block fetch done event (Active for both Y and C)
44	frame_start_event	LCD0	pulse	Frame processing start event

Copyright © 2026 Texas Instruments Incorporated

Table 7-87. VPAC Subsystem CTSET Event List (continued)

Index	Event Name	HWA	Event Type	Event Description
45	lse_stall_event	LCD0	pulse	LSE write stall event
46	coreblk_proc_done_event	LCD0	pulse	LDC Core Block processing done
47	corecblk_wr_done_event	LCD0	pulse	LDC Core block write complete on C12 channel
48	coreyblk_wr_done_event	LCD0	pulse	LDC Core block write complete on Y12 channel
49	vpac_ldc_fr_done_evt_intr	LCD0	pulse	Frame processing done event
50	Reserved			
51	msc_err	MSC	pulse	Error Response for VBUSM read/Write SL2 access (msc_lse_sl2_rd_err_intr || msc_lse_sl2_wr_err_intr)
52	msc_lse_fr_done_evt_0_intr	MSC	pulse	Frame processing done event for MSC Thread0
53	msc_lse_fr_done_evt_1_intr	MSC	pulse	Frame processing done event for MSC Thread1
54	Reserved			
55	Reserved			
56	Reserved			
57	Reserved			
58	cal_vp_stall	CAL	pulse	CAL VP stall
59	Reserved			
60	start	HTS	pulse	HWA-0 (task start)
61	done	HTS	pulse	HWA-0 (task done)
62	init	HTS	pulse	HWA-0 (init)
63	eop	HTS	pulse	HWA-0 (eop)
64	out_ch0_done	HTS	pulse	HWA-0-ch0-done
65	out_ch1_done	HTS	pulse	HWA-0-ch1-done
66	out_ch2_done	HTS	pulse	HWA-0-ch2-done
67	out_ch3_done	HTS	pulse	HWA-0-ch3-done
68	out_ch4_done	HTS	pulse	HWA-0-ch4-done
69	out_ch5_done	HTS	pulse	HWA-0-ch5-done
70	start	HTS	pulse	HWA-2 (task start)
71	done	HTS	pulse	HWA-2 (task done)
72	init	HTS	pulse	HWA-2 (init)
73	eop	HTS	pulse	HWA-2 (eop)
74	out_ch0_done	HTS	pulse	HWA-2-ch0-done
75	out_ch1_done	HTS	pulse	HWA-2-ch1-done
76	out_ch2_done	HTS	pulse	HWA-2-ch2-done
77	out_ch3_done	HTS	pulse	HWA-2-ch3-done
78	start	HTS	pulse	HWA-4 (task start)
79	done	HTS	pulse	HWA-4 (task done)
80	init	HTS	pulse	HWA-4 (init)
81	eop	HTS	pulse	HWA-4 (eop)
82	start	HTS	pulse	HWA-5 (task start)
83	done	HTS	pulse	HWA-5 (task done)
84	init	HTS	pulse	HWA-5 (init)
85	eop	HTS	pulse	HWA-5 (eop)

Copyright © 2026 Texas Instruments Incorporated

Table 7-87. VPAC Subsystem CTSET Event List (continued)

Index	Event Name	HWA	Event Type	Event Description
86	out_ch0_done	HTS	pulse	HWA-4-5-ch0-done
87	out_ch1_done	HTS	pulse	HWA-4-5-ch1-done
88	out_ch2_done	HTS	pulse	HWA-4-5-ch2-done
89	out_ch3_done	HTS	pulse	HWA-4-5-ch3-done
90	out_ch4_done	HTS	pulse	HWA-4-5-ch4-done
91	out_ch5_done	HTS	pulse	HWA-4-5-ch5-done
92	out_ch6_done	HTS	pulse	HWA-4-5-ch6-done
93	out_ch7_done	HTS	pulse	HWA-4-5-ch7-done
94	out_ch8_done	HTS	pulse	HWA-4-5-ch8-done
95	out_ch9_done	HTS	pulse	HWA-4-5-ch9-done
96	Reserved			
97	Reserved			
98	Reserved			
99	Reserved			
100	pipe_done	HTS	pulse	Pipeline-0 Done
101	pipe_done	HTS	pulse	Pipeline-1 Done
102	pipe_done	HTS	pulse	Pipeline-2 Done
103	pipe_done	HTS	pulse	Pipeline-3 Done
104	pipe_done	HTS	pulse	Pipeline-4 Done
105	pipe_done	HTS	pulse	Pipeline-5 Done
106	pipe_done	HTS	pulse	Pipeline-6 Done
107	spare_dec_0	HTS	pulse	Spare 0 sch decrement pulse (ehost mode)
108	spare_dec_1	HTS	pulse	Spare 1 sch decrement pulse (ehost mode)
109	spare_pend_0	HTS	level	Spare 0 sch pend assertion (ehost mode)
110	spare_pend_1	HTS	level	Spare 1 sch pend assertion (ehost mode)
142:111	dbg_event_in_142_111	UTC	pulse	If bit VPAC_CTRL[2].CTSET_HWA_SL2_DBG = '0' -> bits [31:0] = utc_channel_start[31:0]; if CTSET_HWA_SL2_DBG = '1' -> bits [31:12] = 0, bits [11:8] = msc_sl2_access, bits [7:4] = ldc0_sl2_access, bits [3:0] = viss0_sl2_access
174:143	dbg_event_in_174_143	UTC	pulse	If bit VPAC_CTRL[0].CTSET_RT_UTC_IN = '0' -> bits [31:0] = utc0_channel_start[31:0]; if CTSET_RT_UTC_IN = '1' -> bits [31:0] = 0
206:175	dbg_event_in_206_175	UTC	pulse	If bit VPAC_CTRL[4].CTSET_DMA_SOC_DBG = '0' -> bits [31:0] = dru0_ctset_intr[31:0]; if CTSET_DMA_SOC_DBG = '1' -> bits [31:12] = 0, bits [11:8] = ldc0_ext_access, bits [7:4] = utc_nrt_ext_access, bits [3:0] = utc_rt_ext_access
238:207	dbg_event_in_238_207	UTC	pulse	If bit VPAC_CTRL[1].CTSET_RT_UTC_OUT = '0' -> bits [63:32] = dru0_ctset_intr[63:32]; if CTSET_RT_UTC_OUT = '1' -> bits [31:0] = dru0_ctset_intr[31:0]

Copyright © 2026 Texas Instruments Incorporated

Table 7-87. VPAC Subsystem CTSET Event List (continued)

Index	Event Name	HWA	Event Type	Event Description
254:239	dbg_event_in_254_239	SOC	pulse	If bit VPAC_CTRL[3].CTSET_UTC_SL2_DBG = '0' -> bits [15:0] = ext_ctset_event; if CTSET_UTC_SL2_DBG = '1' -> bits [15:8] = 0 , bits [7:4] = utc_nrt_sl2_access, bits [3:0] = utc_rt_sl2_access

(1) The signals combined to generate the viss_err event are: rawfe_cfg_err_intr || glbce_cfg_err_intr || fcfa_cfg_err_intr || fcc_cfg_err_intr || ee_cfg_err || nsfv4_line_cfg_err || rawfe_h3a_buf_ovrflow_pulse_intr || nsf4v_hblank_err_intr || nsf4v_vblank_err_intr || glbce_vp_err_intr || glbce_hsync_err_intr || glbce_vsync_err_intr || fcc_outif_ovf_err_intr || fcc_hist_read_err_intr || ee_syncovf_err || lse_sl2_rd_err_intr || lse_sl2_wr_err_intr || lse_cal_vp_err_intr || viss0_rawfe_dpc_stats_read_err_intr.
(2) The signals combined to generate the ldc_err event: pix_iblk_outofbound_intr || mesh_iblk_outofbound_intr || pix_iblk_memovf_intr || mesh_iblk_memovf_intr || ifr_outofbound_intr || int_szovf_intr || vpac_ldc_sl2_wr_err_intr || vpac_ldc_vbusm_rd_err_intr.

The VPAC subsystem also supports halting an execution pipeline using external or user halt request. Halting a pipeline is achieved by not granting thread start to HWA. Once halted, VPAC waits for an external sync trigger or config resume to restart execution. VISS, and MSC modules halt at line boundary, and LDC halts at block boundary.

The VPAC subsystem supports generic debug capability/features and ARM cross trigger interface for debug triggers:

- Halting the HWA threads on debug request input at logical trigger boundary.
- Continue/restart VPAC from halted state (step/continue) based on external sync trigger and HTS_DBG_CNTL register.
- Debug compliant dbg_attn output after processing external halt request and when VPAC halts debug enabled pipelines.
- CTI (Cross Trigger Interface) compliant out halted trigger after processing external halt request and reaching halted state of VPAC.
- Stalling all pipelines of VPAC (with configurable option to disable impact of debug features on a pipeline).
- Indicate to Host through readable debug_rdy bit in HTS to indicate readiness of HWA resource in case of halted (dbgattn asserted)

The VPAC subsystem debug capability supported is as captured in the HTS_DBG_CAP register. Ext_halt, ext_sync, hwa_halted are CTI (Cross Trigger Interface) mapped async trigger interface (refer to the async protocol of trigger interface in the ARM® CoreSight™ Architecture Specification).

The VISS module, when used in OTF mode, must not be included as debuggable module within VPAC. VISS must be disabled (HTS: debug pipeline disable) to respond to halt request, otherwise behavior is undefined. The VISS module, when used in memory to memory mode, can be halted. But VISS LUTs must not be read during halted state, as it can corrupt the design pipeline.

7.7.2.9 VPAC Subsystem Security Features

The VPAC subsystem implements region based firewall on target endpoints within config CBASS and SL2 CBASS. Both CBASS is driven by independent DMSC FW interface from SoC level. All internal controllers on SL2 SCR are configured with ISC. Figure 7-23 provides FW/ISC summary.

Copyright © 2026 Texas Instruments Incorporated

Module	interconnet	Security component	Parameter	endpoints	FW_ID(dec)	FW/ISC/QoS config address offset(hex)	privID reset value(dec)
VPAC	SCRP	Region FW	2 region	ECC AGG	6048	0	
			1 region	VPAC_TOP	6049	400	
			8 region	INTD	6050	800	
			8 region	HTS	6051	C00	
			1 region	CTSET	6052	1000	
			4 region	VISS0	6053	1400	
			4 region	LDC0	6055	1C00	
			2 region	MSC	6057	2400	
			8 region	UTC0	6059	2C00	
			8 region	UTC1	6060	3000	
	SCRM (SL2)	ISC	1 port x 1 initiator	VISS0		0	213
			1 port x 1 initiator	LDC0		800	215
			1 port x 2 initiator	MSC		1000	217
		Region FW	8 region	SL2 Bank0	6080	0	
			8 region	SL2 Bank1	6081	400	
			8 region	SL2 Bank2	6082	800	
			8 region	SL2 Bank3	6083	C00	

Figure 7-23. VPAC Subsystem FW/ISC Configuration

Copyright © 2026 Texas Instruments Incorporated

7.7.3 VPAC Vision Imaging Subsystem (VISS)

This chapter describes the Vision Imaging Subsystem (VISS).

7.7.3.1 VISS Top Level

This section describes the Vision ISS (VISS) top level details.

7.7.3.1.1 VISS Features

The VPAC includes the following processing and infrastructure sub-modules:

Vision Imaging Sub-System (VISS). The vision imaging pipe does image processing on raw data which includes Wide Dynamic Range (WDR) merge, Defect Pixel Correction (DPC), Lens Shading Correction (LSC), Global/ Local Brightness and Contrast Enhancement (GLBCE), demosaicing , color conversion, and Edge Enhancement (EE). It can operate on sensor data either on-the-fly or in memory-to-memory mode. The VISS provides the following main features:

- Performance is 1 pixel per cycle
- Up to 16-bit input RAW formats
- Support for concurrent 12-bit and 8-bit YUV output
- Support for multiple RAW formats (Bayer, RCCB, RCCC, 4x4 RGB-IR, etc.)
- Flexible to handle any 2x2 RAW format
- Support for other color spaces:
  - RGB output
  - Support for color saturation and gray scale for HSV/HSL, etc. (Hue is not supported)
- Support for multiple WDR /HDR Formats and Up-to 3 exposures merge
- Companded format (12 & 14 bit formats)
- Stagger format of WDR
- Split pixel format of WDR
- Rest of WDR format
- Support of statistics for Auto-Exposure / Auto-white-balance or Auto-Focus (configurable option to select either AE/AWB or AF not both concurrently)
- Advanced Spatial Noise filter with 16-bit support
- Dynamic White balance adaptation
- Histogram on Raw Bayer data
- Support for Flexible CFA for generating multi-plane output for any 2x2 RAW Format sensor configuration
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats
  - Can generate up to 4 independent color planes
  - Support for up two 3 directions per color plane
  - Adaptive Threshold based on intensity measure
- Support for multiple color format output generation
  - Support traditional 12 bit YUV Output
  - Supports RGB output
  - Supports Saturation outputs (8 bits only)
  - Supports Grayscale/Luma/IR/Clear Output
- Support for flexible output format generation including YUV as well as custom
- Support for RGB-IR (4x4) CFA pattern
  - Supports OTF DPC and LSC
  - Supports IR decontamination of the Red, Green Blue channels
  - Supports upto 16-bit IR and YUV outputs
- Edge enhancement Option to bypass visual enhancement functions (NSF4V and GLBCE) for ADAS vision applications
- Concurrent Human Vision (HV) and Mission Vision(MV) outputs
- Chromatic Abaration Correction

Copyright © 2026 Texas Instruments Incorporated

7.7.3.1.2 VISS Block Diagram

The VISS module does on-the-fly processing on raw pixels captured from camera sensor (via CSI RX module). It also does memory to memory processing (via DDR memory) for data captured from other sources at SoC level. Figure 7-24 is simplified block diagram of VISS top level.

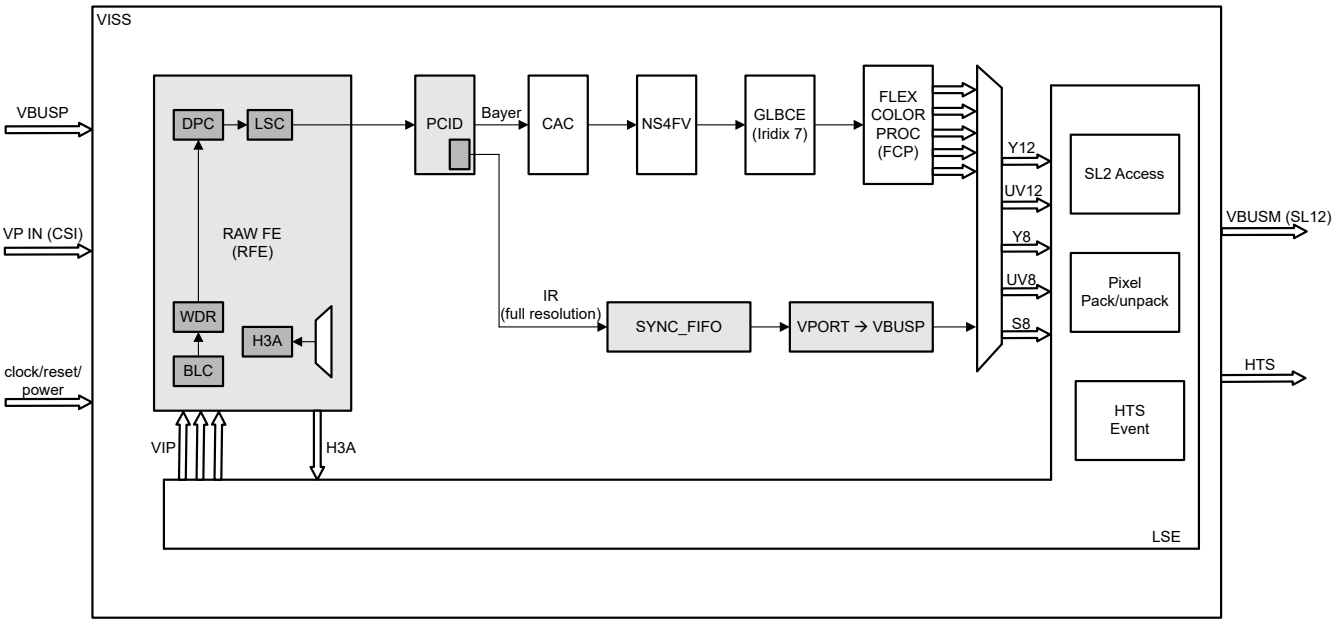


Figure 7-24. VISS Block Diagram

The VISS consists of the following Hardware Accelerators (HWAs):

- RFE (RAW-FE): The RAW Frond End HW block does RAW pixel (that is, Bayer, RCCC, RGBW, etc.) processing on captured image data from sensor and pass-on to NSF4V, GLBCE block, and then to Flexible Color Processing (FCP) HW block for demosaicing and color conversion.
- PCID (Pattern Conversion and IR Demosaicing): Generates full resolution Bayer and IR image from source 4x4 RGB-IR image and performs IR subtraction from the Bayer data. PCID will be enabled in case of when processing 4x4 RGB-IR image.
- CAC: Performs Chromatic aberration correction on Red and Blue pixels. CAC IP can be bypassed in the applications where Chromatic aberration correction is not required.
- NSF4V: Spatial Noise Filter supporting generic 2x2 pixel format with 16-bit pixel size. The NSF4V module can be bypassed in the applications where visual enhancement is not desired.
- GLBCE: The GLBCE module is used for dynamic range control within image for visual quality. If contrast enhancement on input image is required for visual quality, the RFE output is processed by the NSF4v and GLBCE blocks. Otherwise, the RFE output is bypassed to FCP.
- FCP: The Flexible Color Processing HW receives data from the GLBCE and does demosaincing and color conversion. The output of FCP is sent to VPAC shared memory to be written into DDR for the rest of the vision processing by programmable processors (for example, DSP or Arm), or other Vision HW blocks (for example, DMPAC).
- LSE: The Load and Store Engine is and infrastructure block, which performs the following functions:
  - CAL video port: The CSI RX module at SoC level receives CSI-2 sensor data, extracts pixels and drives the result data on its video port interface. The video port is mapped onto one of LSE input interfaces for on-the-fly image processing to reduce DDR bandwidth and latency. The LSE also provides horizontal and vertical blanking cycles to allow core data path to settle at proper boundary of line and frame.
  - SL2 memory access: This module supports load/store data from/to SL2 using VBUSM interface. Loaded data from SL2 are passed on to unpacker function of LSE for RFE. Packed data after FCP/H3A processing is written into SL2.

Copyright © 2026 Texas Instruments Incorporated

- Pixel pack/unpack: Source data (512-bit) for RFE processing is loaded from SL2 and passed onto unpacker function for pixel extraction. Extracted pixels are driven on to the video port of RFE. Similarly, the FCP produced pixels are passed through packer function for eventual write into SL2. The H3A generated data is pseudo mapped as pixels of 32-bit for packing purpose and directly driven by the RFE.
- Event control: The HTS events are generated at line level. These events are routed to LSE to start the processing. For each consumed line, the HTS needs a *task done* indication. For some initial lines of image, there would not be any valid data output. Similarly, the H3A generated data will be at paxel/window height number of lines. For initial lines not producing any valid data, the HTS needs to be indicated with separate mask bits for each output streams. For these initial lines, when there is no valid output due to lines delay inside VISS, the LSE still generates mask output to HTS indicating lack of proper output data.

7.7.3.1.3 VISS Data Flow within VPAC

VISS receives data from sensor through:

- CSI RX video port for OTF processing (Sensor -> CSI RX -> LSE -> RFE -> PCID/CAC/NSF4V/GLBCE -> FCP)
- DMA of VPAC (Sensor -> CAL -> DDR -> VPAC_SL2 -> LSE -> RFE -> PCID/CAC/NSF4V/GLBCE -> FCP)

7.7.3.1.3.1 VISS On-the-fly Processing

The VISS operation is controlled through a single HTS thread. For details of the HTS operation refer to Section *VPAC Hardware Thread Scheduler (HTS)*. The VISS configuration is valid for a single or multiple frame. External-host manages VISS configuration at end of each frame, if required. The HTS controls image processing between lines and handles managing shared memory for VISS inputs/outputs. Sub-frame processing is not natively supported. SW must configure VISS and HTS to start initialization process before enabling sensor capture. After the initialization sequence, the HTS will generate a start trigger to VISS. The generation of the start trigger depends on the availability of H3A data out buffer and FCP data out buffer inside the SL2 memory.

At this stage, the LSE waits for CSI RX video port (VP) to start sending valid pixels. As and when VP data starts streaming in, it is routed to the RFE video port. As RFE, PCID/CAC/NSF4V/GLBCE, and FCP are running in streaming mode, the FCP indicates to LSE about completion of one line of operation . After receiving valid data out, the HTS will trigger UTC for data store into DDR. Each 'thread done' accompanies output mask for all output buffers. Mask bits indicate to HTS about validity of output buffer as for each thread done all output buffers do not need to be produced. The HTS will issue next start after checking all output buffers availability. The streaming data from CSI RX is stallable upto 2KB storage inside VP memory of CSI RX. Any temporary stall can be absorbed, but prolonged stall is buffer overflow inside CSI RX. In VPAC, VISS produced data is transferred using Real Time fabric and low predictable latency needs to be guaranteed.

Copyright © 2026 Texas Instruments Incorporated

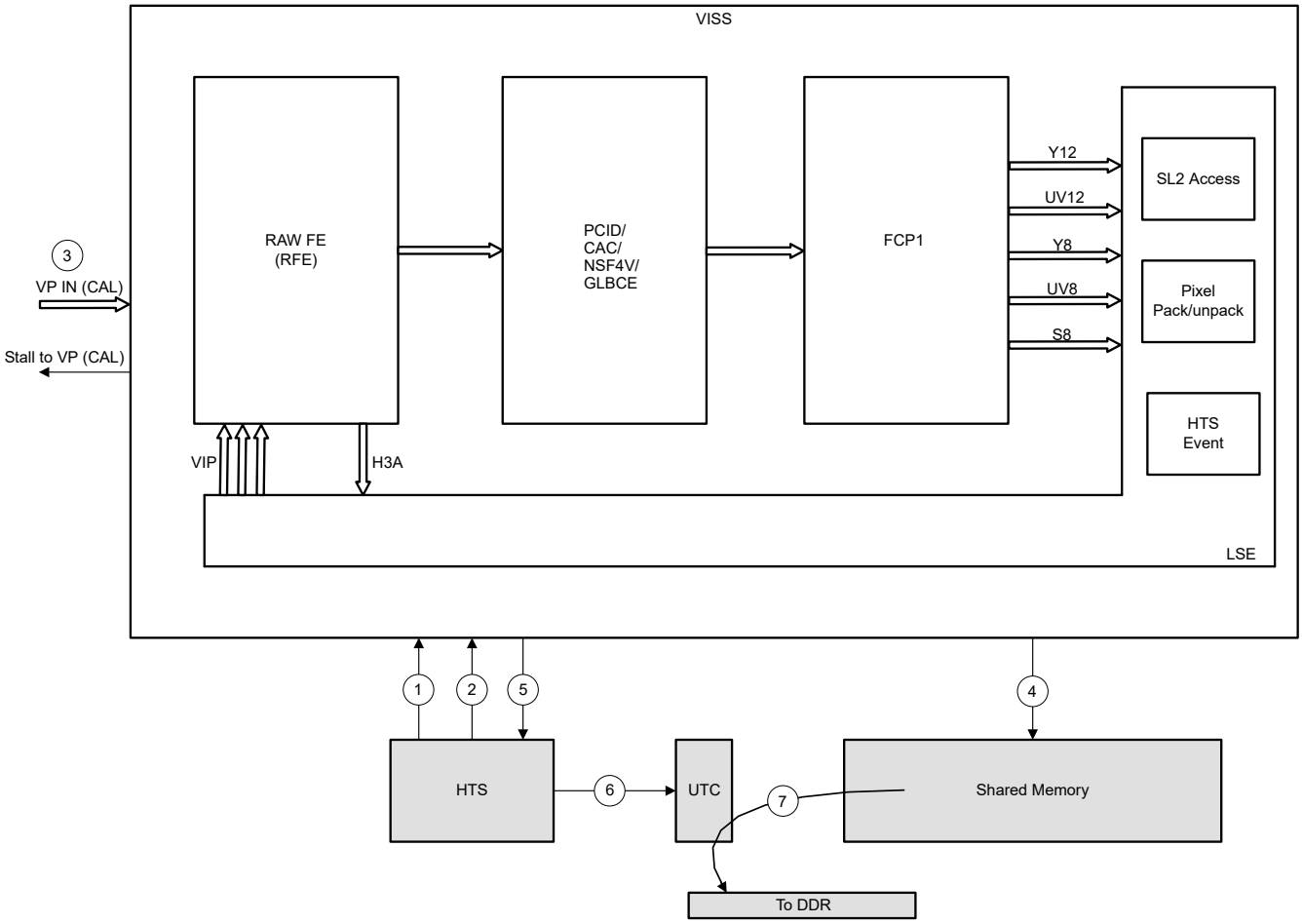


Figure 7-25. VISS On-the-fly Operation

The following OTF operation steps are illustrated in Figure 7-25:

1. *Init* from HTS to initialize VISS.
2. Thread start. Step 1+2 prior to enabling CSI RX + PHY + sensor.
3. Video port data streaming. Unit of operation inside VISS at line level.
4. Output data produced by FCP and H3A at line level. H3A will produce data only at those lines which aligns with paxel_height/window_height.
5. Production of 1 line depends on consumption of 1 line on VP interface. 'Thread done' triggered after completion of 1 processed line write into SL2. In case there is no valid data to be written into SL2, 'thread done' is generated after consumption of 1 line of data from VP. Note: *Horizontal blanking will be configured inside LSE to fecilitate writing complete line out of FCP*.
6. HTS triggers DMA to store output buffer from SL2 to DDR.
7. DMA loads data from SL2 and writes into DDR.

The following assumptions are made for the OTF operation described above:

- UTC write is mapped onto Real Time Thread for guaranteed QoS. Channel done by UTC must be generated after completion of write into DDR. Next thread start depend on availability of minimum one buffer for each output line in SL2. More buffers in SL2 can take care of instantaneous drop in UTC transfer capacity.
- Cycle gap between HTS done and next start must be minimal. CSI RX VP interface will be stalled whenever LSE input buffer on VP interface reaches near full condition.
- Configurable vertical latency times 'Tstart' generated by HTS to let RFE + NSF4V + GLBCE + FCP flush its internal pipe, even though last line of current frame is already fed into RFE.

Copyright © 2026 Texas Instruments Incorporated

- Buffer dependencies:
  - H3A output buffer (AE or AF)
  - FCP output buffer (Y12, YV12, Y8, UV8, S8 or IR)

7.7.3.1.3.1.1 Non-WDR or Companded WDR Sensors

VISS is enabled for single input data when either sensor does not support WDR or sensor does on-chip WDR merge and pass on merged data after companding to 12 bit / 14 bits as shown in Figure 7-26.

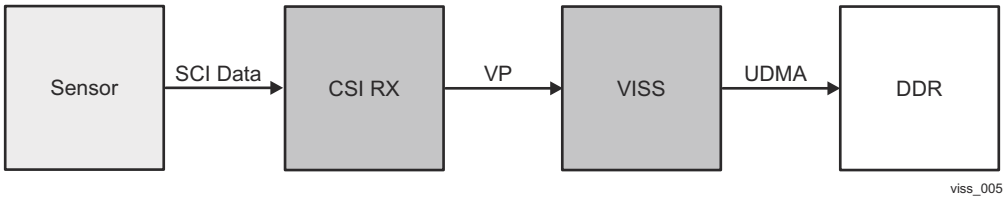


Figure 7-26. VISS Non-WDR/Companded: High Level

7.7.3.1.3.2 VISS Memory to Memory Image Processing

CSI RX writes CSI-2 received pixel data into DDR. Once a frame is written, host is triggered to initiate VISS processing. HTS will go through 'init' sequence to initialize VISS and its SL2 pointers. As UTC is head of pipeline for memory to memory mode of image processing, its scheduler inside HTS needs to be configured for generating "frame height" number of triggers to fetch image lines one by one. This is essential to bring determinism in pipeline as there is no producer for UTC load thread.

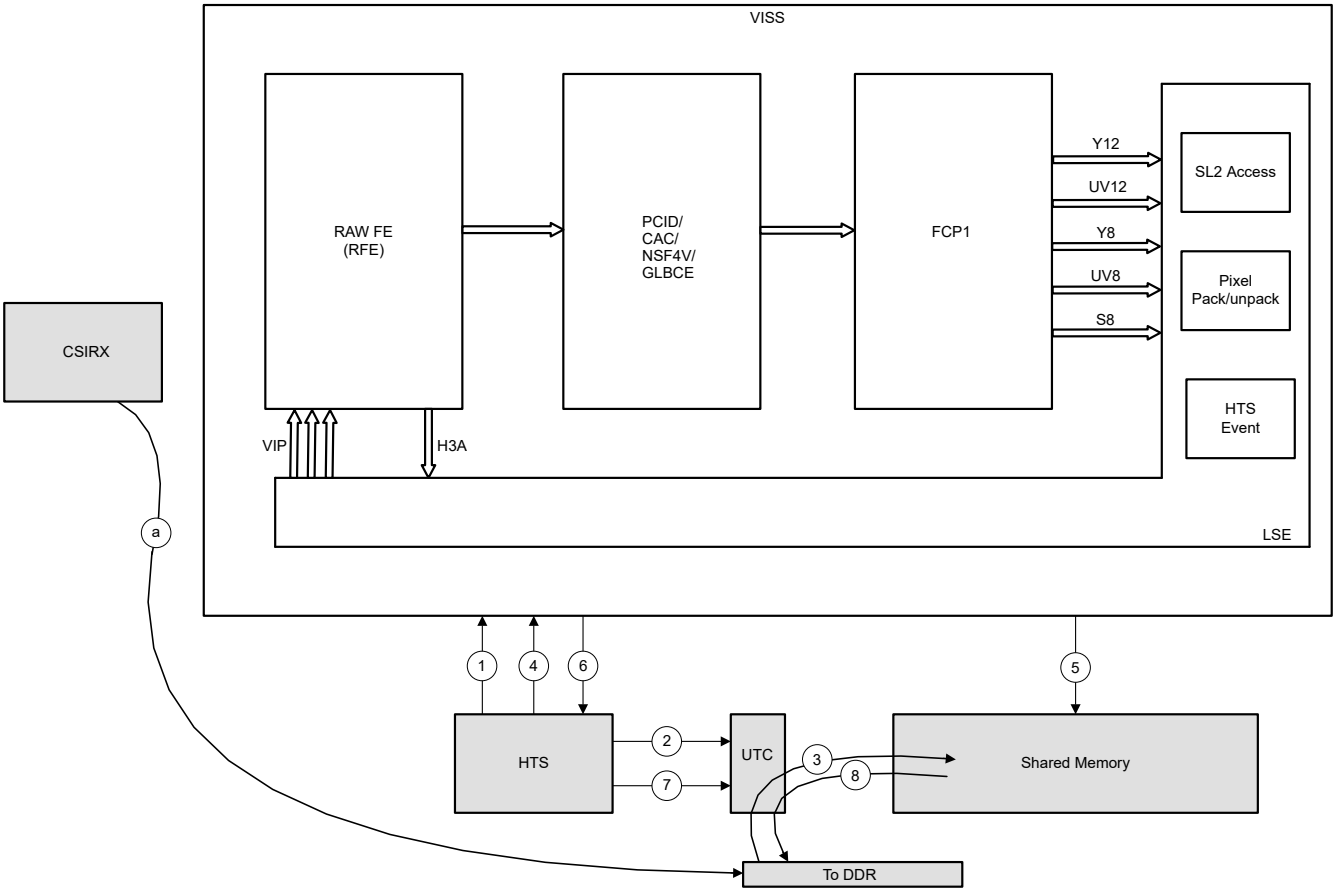


Figure 7-27. VISS Memory to Memory Operation

Copyright © 2026 Texas Instruments Incorporated

The following memory to memory operation steps are illustrated in Figure 7-27:

1. CSI-2 stream captured via CSI RX into DDR. This traffic must be routed through Real Time fabric. CSI RX capture happens independent of VISS processing.

1. *Init* from HTS to initialize VISS.
2. HTS triggers UTC to fetch data for processing. There could be max 3 independent input data.
3. UTC loads the data and stores into SL2.
4. Upon completion of all input buffer loading (at line level) UTC channel 'done' event triggers VISS thread start (still needs to ensure there are buffer available for writing VISS output data).
5. VISS produces line #N corresponding to loading of line #N + vertical_latency.
6. 'Tdone' triggered to HTS.
7. HTS triggers UTC for output buffer transfer into DDR.
8. UTC loads data from SL2 and write into DDR.

The following asumptions are made for the memory to memory operation described above:

- CSI RX write is mapped onto hard real time fabric.
- Configurable vertical blanking times 'Tstart' generated by HTS to let RFE + NSF4V + GLBCE + FCP flush its internal pipe, eventhough last line of current frame is already fed into RFE.
- Buffer dependencies:
  - RFE input buffer (exp0, exp1, exp2)
  - H3A output Buffer (AE or AF)
  - FCP output Buffer (Y12, UV12, Y8, UV8, S8)

7.7.3.1.4 VISS Data Formats Support

Table 7-88 lists the data formats supported by VISS. VISS does not support YUV Input. In case of external YUV Sensor, CSI RX directly writes into DDR and tha data is subsequently processed by the rest of VPAC.

Table 7-88. VISS Data Formats

Direction	Pixel Width	Pixel Container Width (M2M)	Packing	Video port (OTF)	Comments
VISS Input	RAW12/14/16	16b	No	16b / pixel	1. Companded data/External ISP 2. MSB bits are padded with zeros within fixed width pipeline
	RAW8	8b	Yes	16b / pixel	
	RAW12	12b	Yes	16b / pixel	
VISS Output	YUV420 (12b) NV12	12b	Yes	N/A	
	YUV420 (12b) NV12	16b	No	N/A	
	YUV420 (8b) NV12	8b	Yes	N/A	
	YUV422 (8b) UYVY/YUY2/NV16	8b	Yes	N/A	
	RGB (planar)	8b	Yes	N/A	With RGB output, limited YUV data outputs are available (only 12-bit YUV data is available)
	S8	8bit	Yes	N/A	
	IR	8-bit	Yes	N/A	
		10/12-bit	Yes	N/A	

Copyright © 2026 Texas Instruments Incorporated

7.7.3.1.5 VISS VPORT Interface

The video port (VPORT) interface is used to stream the pixel data from up stream to down stream module. In VISS, the VPORT interface is used to receive data from CSI RX (in streaming mode) and in between VISS sub-modules (RFE -> PCID -> CAC -> NSF4V -> GLBCE -> FCP).

The VPORT supports stall signal (VP_STALL) and also either one pixel or 2 pixel data per cycle.

The pixel data (VP_DATA) received on the VPORT is organized in 32 bits as follows:

- VP_DATA[15:0] = pixel (n)
- VP_DATA[31:16] = pixel (n+1). This is ignored, when VPORT_TWO_PIXEL = 0
- MSBs are padded with 0's when less than 16 bits are used.

7.7.3.1.6 VISS Submodule Integration Specifics

7.7.3.1.6.1 LSE Integration

Figure 7-28 captures the channel integration between LSE and rest of VISS submodules. Though digram shows three input channels between LSE to RFE, 3 channels data is transferred using single VBUSP interface.

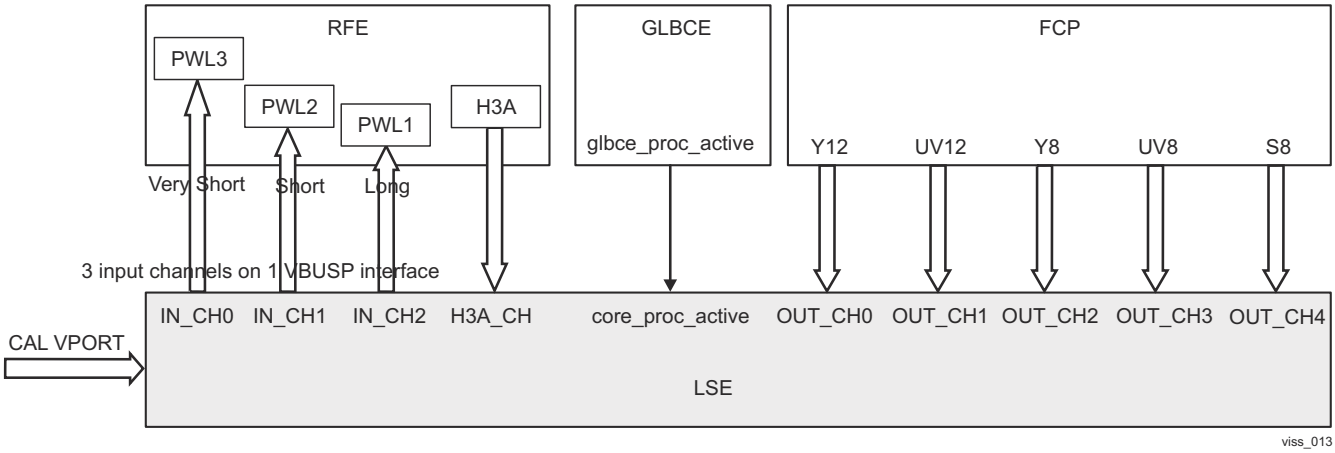


Figure 7-28. VISS LSE Channel Integration

LSE supports YUV422 interleaving only on 8-bit data.

The LSE configuration for VISS can be read in VISS_LSE_STATUS_PARAM register fields.

For more information on LSE module operation, see Chapter *Load and Store Engine (LSE)*.

7.7.3.1.6.2 PCID

Figure 7-24 captures intergration of PCID into VISS. PCID produces both Bayer and IR data concurrently from input RGBIR image. Bayer channel data is fed to rest of the VISS pipe line and IR channel output is directly written to memory using one of LSE channels. VPORT Delay SYNC_FIFO supports programbale horizontal delay to match the delay with FCP output.

7.7.3.1.6.3 Chromatic Aberration Correction

Figure *CAC, NSF4V, and GLBCE Integration* captures the CAC, NSF4V, and GLBCE integration in the VISS pipeline. All CAC, NSF4V, and GLBCE can be enabled or disabled based on use case. By default, all are disabled. General visual use cases require NSF4V and GLBCE (along with FCP.EE) to be enabled. Lenses with purple fringe artifact require CAC to be enabled.

Copyright © 2026 Texas Instruments Incorporated

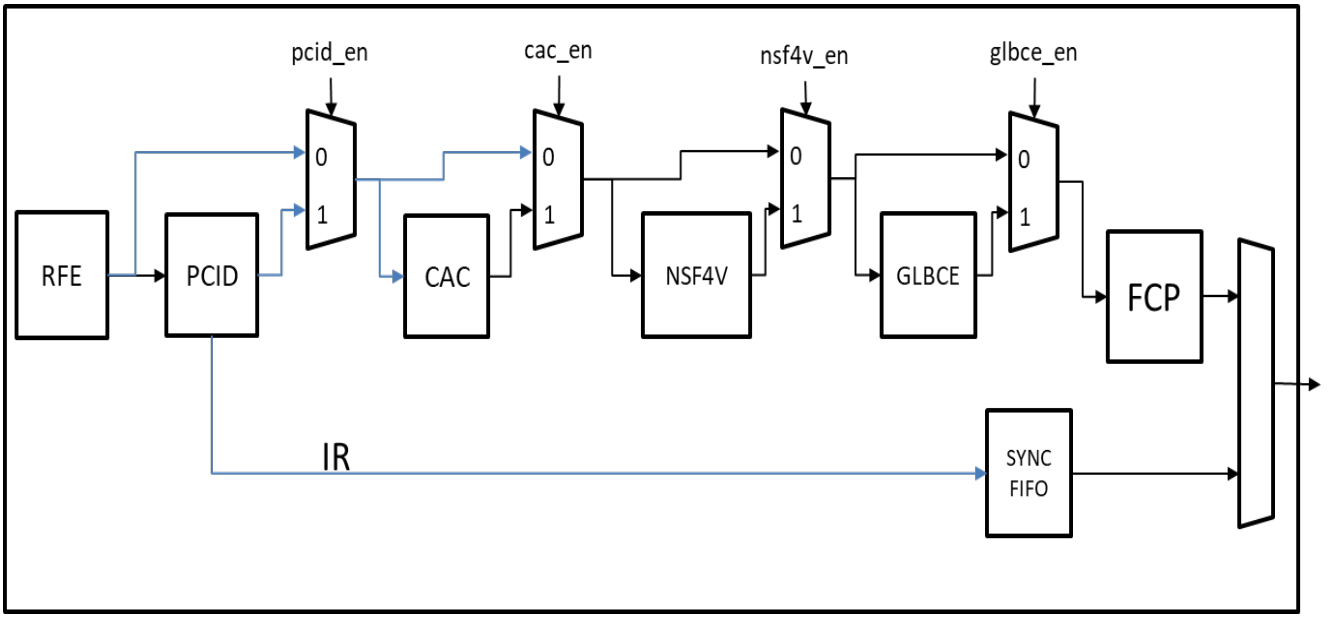


Figure 7-29. CAC, NSF4V, and GLBCE Integration

7.7.3.1.6.4 GLBCE Integration

GLBCE is Global/Local Brightness and Contrast Enhancement module.

The GLBCE module supports the following main features:

- One pixel/cycle operation
- Up to 4096 pixels/line (line size)
- Bayer image
- When GLBCE is off, the RFE/NSF4V output is bypassed to FCP

For more information on GLBCE operation, see Section 7.7.3.7, *VISS Global/Local Brightness and Contrast Enhancement (GLBCE) Module*.

7.7.3.1.6.4.1 GLBCE Startup

GLBCE requires approximately 5400 clock pulses after startup before it could receive data on its video port. Therefore, before RFE -> GLBCE -> FCP path could be used after a GLBCE reset, the SW shall use the following sequence to provide those required clock pulses:

- Prior to enabling rest of VISS pipe, configure VISS clock control to allow free running clock to GLBCE.
- Wait for the 'glbce_filtering_done' event. It indicates that GLBCE has received enough PCLK pulses for internal initialization and that it is now ready to receive pixels.
- Configure GLBCE clock to normal mode. Enable the rest of VISS pipeline.

GLBCE has a minimum input restriction of 480x240 pixels. The input frame must be at least this size.

GLBCE LUTs must not be changed during the active frames. They can only be updated in vertical blanking. GLBCE_LUT_FI may be changed frame by frame depending on the GLBCE tuning method. Other LUTs (GLBCE_REV_PERCEPT_LUT_xx, GLBCE_FWD_PERCEPT_LUT_xx, and GLBCE_WDR_GAMMA_LUT_xx) are not expected to be updated dynamically except in very limited case.

7.7.3.1.6.4.2 GLBCE Bypass

When the GLBCE is disabled (by VISS_CNTL[0] GLBCE_EN = 0), any access targeted to GLBCE registers or GLBCE statistics memory will respond with error status, and 'glbce_cfg_err' interrupt will be generated.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.1.7 VISS Stall Handling

VISS can handle momentarily stall condition created due to lack of enough band width. Streaming mode is described in the following sections.

7.7.3.1.7.1 Stall Handling for Streaming Mode

Streaming stall handling is similar to memory to memory mode, except that final effect will be reflected in generating stall to the VPORT interface. Upon stall assertion, VISS expects no more than 2 VPORT pixel clock cycles.

7.7.3.1.8 VISS Blanking Requirements

Table 7-89 lists some blanking and latency details for VIS submodules.

Table 7-89. VISS Blanking Requirements

IP	Blanking		Latency	
	Vertical (in number of lines)	Horizontal (in pixel clock cycles)	Vertical (in number of lines)	Horizontal (in pixel clock cycles)
RawFE (2x2)	2 - DPC Enable 0 - DPC Disable	8	2 - DPC Enable 0 - DPC Disable	50
RawFE (4x4 RGBIR)	4 - DPC Enable 0 - DPC Disable	10	4 - DPC Enable 0 - DPC Disable	53
PCID	4	27	4	27 (Bayer Out)(3) 30 (IR Out)
CAC	6	20	6	24
NSF4V	15	28	15	44
NSF4V.DWB	0	7	0	7
GLBCE	- (1)	5	1	62
CFA	1	5	3	25
FCC	0	20	1 (2)	25
EE	2	2	2	29

(1) GLBCE requires (1033*7 + 2066*FLOOR(VARIANCEINTENSITY/2,1) + VARIANCESPACE*2066+30) cycles for vertical blanking.
(2) In 420 mode, Chroma is vertically downsampled. For the first line, Chroma channel does not generate any dummy VBUSP requests. Hence, 1 line vertical delay is required.
(3) It is latency of the physical channel not the data type. If IR data is selected to Bayer channel, Bayer channel output latency remain the same (i.e. 27)

The VARIANCEINTENSITY and VARIANCESPACE parameters (in Table 7-89 notes) affect the sensitivity of the transforms across different parts of the frame. Both parameters can be configured through the GLBCE_VARIANCE register.

In Table 7-89, the vertical blanking/latencies are expressed in number of lines (except for GLBCE), and horizontal blanking/latency is expressed in number of pixel clock cycles.

Horizontal blanking, buffer depth and HTS programming parameters (Threshold and PostLoad) depend on VISS frame mode or line mode of operation (selected via LSE_CFG_LSE[5] IN_CH_SYNC_MODE register field). Frame mode is the recommended mode of operation, and line mode is primarly used for debug purposes.

- Frame mode:
  - LSE input line fetch starts once the previous input line plus horizontal blanking cycle are provided to RFE. Horizontal blanking should meet two requirements:
    - Greater than minimum horizontal blanking of each enabled modules
    - Greater than maximum cycle gap between the last line pixel write on all LSE enabled output channel line ends
  - Horizontal Blanking of 35 is expected to be sufficient for most use cases. Horizontal blanking of 55 should be used, if EE is enabled and any of the CFA outputs is directly written out.
- Line mode:

Copyright © 2026 Texas Instruments Incorporated

  - LSE input line fetch start on HTS Tstart
  - Horizontal blanking is addition of all enabled modules horizontal delay (that is, latency)

LSE will automatically generate required vertical blanking until all output channels produce HTS_EOP, and GLBCE processing is complete (when GLBCE is enabled). In the vertical blanking window, LSE generates dummy lines (data = 0x0) following frame width paramter. LSE prematurely ends line w/o HE marker once vertical blanking requirement is met (that is, processing is done and HTS_EOP is generated).

7.7.3.1.9 VISS Interrupts

The VISS submodules generate the interrupt events described in Section *VPAC Subsystem Interrupts*.

7.7.3.1.9.1 Interrupts Merging

The configuration error interrupts are merged at VISS level. The following list describe the configuration error eventss and the corresponding merged interrupts.

- RAWFE_CFG_ERR_INTR: Generated when configuration interface access the RFE end points, while frame processing is active. Merged interrupts from RFE:
  - lut1_cfg_err_intr
  - lut2_cfg_err_intr
  - lut3_cfg_err_intr
  - wdr_lut_cfg_err_intr
  - dpc_line_cfg_err_intr
  - dpc_lut_cfg_err_intr
  - h3a_lut_cfg_err_intr
  - h3a_line_cfg_err_intr
  - h3a_accm_cfg_err_intr
  - lsc_cfg_err_intr
- GLBCE_CFG_ERR_INTR: Generated when the configuration interface access the GLBCE register region or statastics memory, while frame processing is active. Merged interrupts from GLBCE:
  - glbce_statmem_cfg_err_intr
  - glbce_mmr_cfg_err_intr
- FCFA_CFG_ERR_INTR: Generated when the configuration interface access the CFA register region, or LUT or line memory, while frame processing is active. Merged interrupts from CFA:
  - clut_cfg_err_intr
  - dlut_cfg_err_intr
  - lut_cfg_err_intr
  - cfa_mmr_err_intr
  - cfa_pix_err_intr
- FCC_OUTIF_OVF_INTR: FIFO Overflow happened at output interface between FCC and LSE. Merged all FCC output overflow error sources at VISS. Refer to Section VISS Flexible Color Processing (FCP) for all sources.
  - overflow_if_y12_intr
  - dlut_cfg_err_intr
  - lut_cfg_err_intr
  - cfa_mmr_err_intr
  - cfa_pix_err_intr
- FCC_OUTIF_OVF_ERR_INTR: Generated when any of the FCC output interfaces overflows. Merged interrupts from CC:
  - overflow_if_y12_intr
  - overflow_if_uv12_intr
  - overflow_if_y8r8_intr
  - overflow_if_c8g8_intr
  - overflow_if_s8b8_intr
  - overflow_if_irout_intr

Copyright © 2026 Texas Instruments Incorporated

- FCC_CFG_ERR_INTR: Generated when the configuration interface access FCC register region or its LUTs, while frame processing is active. Merged interrupts from FCC:
  - contrast0_cfg_err_intr
  - contrast1_cfg_err_intr
  - contrast2_cfg_err_intr
  - lut_12to80_cfg_err_intr
  - lut_12to81_cfg_err_intr
  - lut_12to82_cfg_err_intr
- EE_CFG_ERR_INTR: Configuration happened to EE regions causing corruption during frame processing. Merged EE sources are:
  - eelut_cfg_err_intr
  - ee_pix_err_intr
- EE_SYNCOVF_ERR_INTR: EE horizontal synchronization FIFO overflow interrupt:
  - ee_hz_align12_intr
  - ee_hz_align8_intr

7.7.3.1.9.2 Handling of Configuration Error Interrupts

Table 7-90 give hints on handling configuration error interrupts.

Table 7-90. VISS Configuration Error Interrupts Handling

Memory	Error Generation Window	HWA-level Register Status Clearing Mechanism
RAWFE		
PWL1_LUT, PWL2_LUT, PWL3_LUT	Config read access during active line OR write access during active frame.	Write ‘1’ to corresponding bit in RAWFE_INT_STAT register.
WDR_LUT		
H3A_LUT		
H3A_ACCM	Config read/write access during active frame at H3A boundary.	
H3A_LINE		
DPC_LUT	Config read/write access during active frame.	
DPC_LINE	Config read access during active line OR write access during active frame.	
LSC Table	Config read/write access during active frame. Active frame in lsc case is VS at pwl to VE at lsc input delayed by 1 cycle.	
NSF4V		
NSF4V_LINE	Config read/write access when datapath is accessing the corresponding memory on the same cycle.	No register status in NSF4V.
GLBCE		
GLBCE non-shadowed registers	Config write access during active frame. Active frame window is from VS_IN to Filter done.	Write ‘1’ to GLBCE_INT_STAT[0] MMR_CFG_ERR register bit.
GLBCE_STAT	Config read/write access during active frame. Active frame window is from VS_IN to Filter done.	Write ‘1’ into GLBCE_INT_STAT[1] STATMEM_CFG_ERR register bit.
GLBCE_LINE	Not mapped to config.	N/A
CFA		
CFA FIR Filter registers	Config write access during active frame.	Write ‘1’ to CFA_INT_STATUS[2] CFA_MMR_ERR register bit.
CFA_LUT	Config read/write access during active frame.	Write ‘1’ into CFA_INT_STATUS[0] LUT_CFG_ERR register bit.
CFA_LINE	Config read/write access during active frame.	Write ‘1’ to CFA_INT_STATUS[1] CFA_PIX_ERR register bit.
FCC		

Copyright © 2026 Texas Instruments Incorporated

Table 7-90. VISS Configuration Error Interrupts Handling (continued)

Memory	Error Generation Window	HWA-level Register Status Clearing Mechanism
LUT_CONTRAST	Config access (read/write) occurs during active line when the LUT is enabled.	Write '1' to corresponding bit in FCC_FLEXCC_INT_STATUS register.
HISTOGRAM	Config access has occurred to the first location but not to the last location till the start of next frame implying that full histogram was not read.	
LUT_COLOR	Config access (read/write) occurs during active line when the LUT is enabled.	
CC_LINE	No error generation logic.	N/A
EE		
EE_LINE	Config accesses during active frame.	Write '1' to EE_INT_STATUS[1] EE_PIX_ERR register bit.
LUT	Config accesses during active frame.	Write '1' to EE_INT_STATUS[0] EELUT_CFG_ERR register bit.

7.7.3.1.10 VISS Error Correcting Code (ECC) Support

ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

One or more memories within VISS are ECC protected using an ECC Memory Wrapper which implements Single Error Correction and Double Error Detection (SECDED).

The ECC wrapper provides Single Error Detection and Correction (SED/SEC). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC wrapper also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC wrapper also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

An ECC Aggregator at the VISS level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. It provides a single EOI-handshake based interrupt to the host (for both single and double error detections) and a standard 32-bit VBUSP interface for configuring and querying the various ECC wrappers via their ECC register set.

The following VISS memories are ECC protected:

- GLBCE statastics memory.
- RAWFE non pixel line buffer memories. See Section *VISS RAW Frond-End (RAWFE)*, for more information.
- FCP non pixel line buffer memories

For more information on the ECC Aggregator operation, see Section *ECC Aggregator* in Chapter *Safety Modules*.

7.7.3.1.11 VISS Programmer's Guide

7.7.3.1.11.1 VISS Initialization Sequence

VISS initialization is part of any other HWA initialization on VPAC. Prior to starting HWA initialization through HTS, VISS must be configured in below order.

1. Configure RFE registers, GLBCE registers, NSF4V registers and FCP registers:
   - The VISS_CNTL[0] GLBCE_EN register bit needs to be set, in order to configure the GLBCE registers.
2. Select input / output stream properties inside LSE.
3. After reset and prior to enabling LSE, apply the following configuration:
   a. Make sure VISS_CNTL[0] GLBCE_EN bit is '1'.
   b. Enable IRQ for interrupt event 'glbce_filtering_done'.
   c. Set VISS_GLBCECONFIG[0] GLBCE_PCLKFREE register bit to '1'.

Copyright © 2026 Texas Instruments Incorporated

d. Wait for interrupt 'glbce_filtering_done'.
e. Clear VISS_GLBCECONFIG[0] GLBCE_PCLKFREE register bit.

4. Enable LSE input and output buffer lines.
5. Enable pipeline and schedulers associated with VISS HTS configuration (pipeline enable must happen after enabling all attached schedulers).
6. Enabling HTS will trigger init sequence which will initialize LSE, RFE and FCP. This will also trigger head of pipe for NAVSS UDMA fetch (in case of memory to memory operation). Otherwise,VISS waits for streaming data from CSI RX.
7. After initializing VISS/VPAC, configure and enable CSI RX, CSI RX PHY and camera sensor in order.
8. Once streaming data starts, it will continue till the end of frame. Blanking (set in VISS_LSE_CFG[31-22] VP_HBLNK_CNT register field) must be configured more than VISS blanking needs (see *VISS Blanking Requirements*).
9. Once a complete frame is fully written into DDR, the HTS module comes back to init state and restart from step #6. For some usesecase, LSE input/out enables after each frame might be considered as well (step #5).
10. After each 'frame_done', if the configuration needs to be changed, it must be done within vertical blanking period for not shadowed registers (refer to Sections *VISS RAW Frond-End (RAWFE)* and *VISS Flexible Color Processing (FCP) Module*, for more details).
11. At any stage, if there is a need to stop the streaming interface, the HTS must first be paused and then aborted, and then reset and re-configure the entire VISS as part or recovery from abort.

7.7.3.1.11.2 VISS Configuration Restrictions

Frame size restriction:

- If GLBCE is enabled in the VISS pipeline, the minimum frame required is 480x240 pixels.
- If GLBCE is disabled in the VISS pipeline, the minimum frame required is 64x16 pixels.
- Frame width and height has to be even.
- Histogram can only be enabled when the frame width is 256 pixels minimum.

LSE H3A channel enabling restriction:

- When the H3A output channel is enabled in LSE, at least one other output channel needs to be enabled.

7.7.3.1.11.3 VISS Real-time Operating Requirements

SW needs to configure UTC/UDMA channel registers for write out buffers.

SW needs to reconfigure RAWFE LSE LUT, if needed, and any other non-shadowed registers during vertical blanking period only. Some control bits would be shadowed, which can be configured during a complete frame.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.2 VISS Load Store Engine (LSE)

For more information on the VISS LSE module operation, see *LSE Integration* and *Load Store Engine (LSE)*.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.3 VISS RAW Frond-End (RAWFE)

This section describes the Vision Imaging Sub-system (VISS) RAW Frond-End (RAWFE) module in the device.

7.7.3.3.1 RAWFE Overview

The RAWFE is a set of raw pixel processing functions to enhance the RAW image signal received from the sensor. The key application of the RAWFE is in the ADAS space where in it is used for both analytics as well as visual use cases.

7.7.3.3.1.1 RAWFE Supported Features

Each RAWFE module implements the following features:

- Support for any 2x2 RAW data pattern.
- Support for 4x4 RGBIR CFA pattern (50% green, 25% IR, 12.5% red and 12.5% blue).
- Support for merge up to 3 separate exposures (non 4x4 RGBIR CFA pattern).
- PWL (piecewise linear) and LUT based de-companding at the front end of each exposure channel.
- Support for WDR merge, including 2 exposure merge, 3 exposure merge and 2 exposure hybrid (12 bit and 16 bit) merge.
- Support for companding LUT from 24 bits down to 12 bits.
  - LUTs support higher precision in lower end.
- Support for LUT as well as adaptive Defective Pixel Correction.
- Support for export of defective pixel coordination in adaptive defect pixel correction mode.
- Support for table based lens shading correction.
- Support for 3A statistics.
  - Statistics can be generated on independent exposures or merged data after tone map.

7.7.3.3.1.2 RAWFE Not Supported Features

The RAWFE does not support the following features:

- Motion compensated WDR Merge
- Simultaneous statistics using H3A on all 3 exposures
- More than 3 exposures for WDR
- Frame size that is not in the 4x4 CFA pattern for RGBIR sensors, or frame height or width which is not integer multiple of 4

7.7.3.3.2 RAWFE Functional Description

7.7.3.3.2.1 RAWFE Functional Operation

The RAWFE processes captured image data from sensor and passes the data to the Flexible Color processing module (FCP) for demosaicing and color conversion.

Figure 7-30 shows a high level conceptual block diagram of the RAWFE module.

The RAWFE consists of the following main components

- Decompanding block: This is used to decompand the sensor compressed raw to native bit depth. The same block can be used for tone mapping as well. The decompanding block can take sensor compressed RAW data and can decompand to up to 24 bits.
- WDR Merge: Each instance of the WDR merge block can take in 2 independent exposures (up to 16 bits) and generate up to 20 bit data.
- LSC Block: The block performs lens shading correction by applying gains stored in a look up memory.
- DPC: The block performs defect pixel correction using either LUT based memory or adaptive defect correction.
- H3A: The H3A block generates 2 different set of statistics, one for AWB and AE and the other for auto focus.
- In addition, when RGBIR sensor is used, H3A can be configured to select PCID output and generate statistics.

Copyright © 2026 Texas Instruments Incorporated

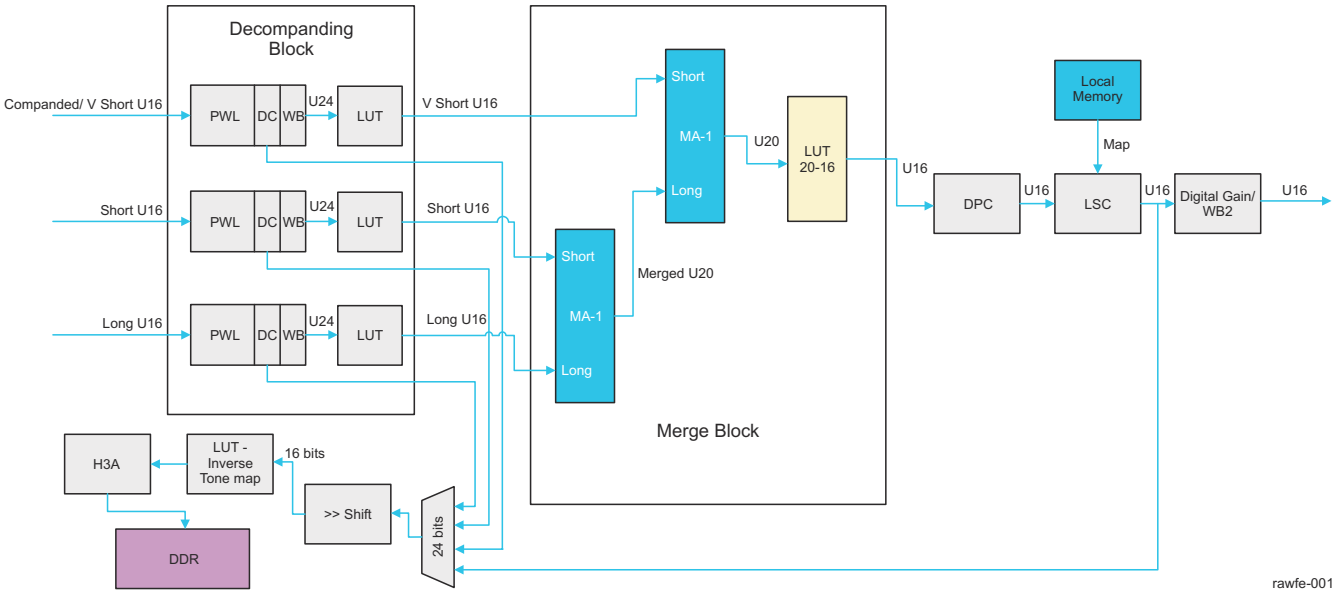


Figure 7-30. RAWFE Block Diagram

7.7.3.3.2.2 RAWFE Integration in VISS

Figure 7-31 show the VISS Block Diagram integrating the RAWFE module.

The VISS Consist of following hardware accelerators

1. RAWFE: The RAWFE block does RAW pixel (e.g. Bayer, Clear Channel, 4x4 RGB-IR) processing on captured image data from sensor and later passes to FCP block for demosaicing and color conversion.
2. FCP (Flexible Color Processing): The FCP receives data from RAWFE and does demosaicing and color conversion. The output of FCP is sent to external memory for vision processing by programmable processors (e.g. DSP or ARM) as well as Vision HW Blocks (e.g. VPAC and DMPAC). If the sensor data is 4x4 RGBIR, output of RAW-FE is first sent to PCID to convert RGB into 2x2 Bayer which is then sent to Flexible Color Processing. Meanwhile, the full resolution IR produced by PCID is sent to external memory.
3. GLBCE (Global Local Brightness and Contrast Enhancement): This is local adaptive tone mapping block for 2x2 RGB either from directly from RAE-FE or converted to 2x2 RGB from 4x4 RGBIR by PCID.

Copyright © 2026 Texas Instruments Incorporated

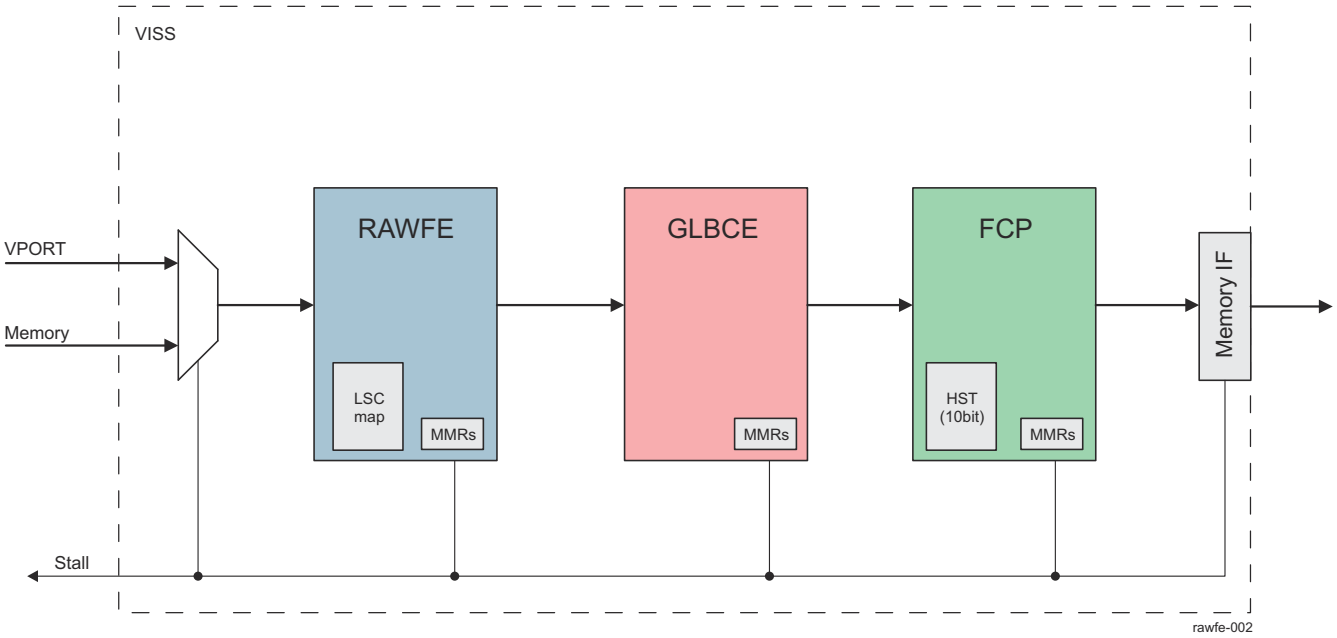


Figure 7-31. RAWFE Integration in the VISS

7.7.3.3.2.3 RAWFE Memory Map

Table 7-91 lists the RAWFE memory mapped RAMs.

Table 7-91. RAWFE Memory Map

Address Offset	Register Space	Size
0x00000	RAWFE registers	
0x00400	H3A registers	
0x00800	V-Short LUT (LUT3-RAM)(1)	639 x 16 bits
0x01000	Short LUT (LUT2-RAM)(1)	639 x 16 bits
0x01800	Long LUT (LUT1-RAM)(1)	639 x 16 bits
0x02000	Merge LUT (WDR_LUT-RAM)(1)	639 x 16 bits
0x02800	H3A LUT (H3a_LUT-RAM)(1)	639 x 10 bits
0x03000	DPC LUT (DPC-RAM)(1)	256 x 29 bits
0x08000	LSC LUT (RAM)(1)	5248 x 39 bits
0x16000	DPC_STAT (DPC_STAT-RAM)	2048 x 26 bits

(1) Partial writes to RAMs are not supported. Software must write the full word. This restriction is an artifact of simplifying ECC support.

Most registers in RAWFE MMR space are shadowed. On frame start, the non shadowed registers will update the shadowed registers and used for the remainder of the frame for data processing. The following registers are not shadowed:

- VISS_RAWFE_INT_STAT
- VISS_RAWFE_DBG_STAT1
- VISS_RAWFE_DBG_STAT2
- VISS_RAWFE_DBG_STAT3
- VISS_RAWFE_DBG_STAT4
- VISS_RAWFE_DBG_CTL

H3A has partial shadowing, refer to Section 7.7.3.3.4.7 for details.

Copyright © 2026 Texas Instruments Incorporated

LUT3 RAM is shadowed. LUT3 and LUT2 will be multiplexed for shadowing capabilities. Software must use a control register to select which LUT is to be used during frame. Software must fully configure LUT2 in this case as well.

Note

Accesses to non shadowed registers or RAMs during active window can cause corruption. During an active phase, non blanking region, any accesses to a RAM, read or write, will corrupt the output data. H3A RAMs, LSC LUT, DPC LUT RAMs are exceptions. These RAMs must be outside the full frame, including horizontal blanking regions, otherwise they will cause corruption. Any write to MMRs that are not shadowed will result in data corruption in non blanking area, the interrupt status and debug registers are exception. MMR reads to any type of register are ok and will not result in frame output corruption. Interrupt events will be asserted if configuration accesses corrupt frame data.

7.7.3.3.2.4 RAWFE ECC for RAMs

Table 7-92 lists the RAWFE ECC RAMs

Table 7-92. RAM ECC Capabilities

Module	# Rams	ECC
PWL LUT1	2	ECC, with single bit correction, including writeback.
PWL LUT2	2	ECC, with single bit correction, including writeback.
PWL LUT3	2	ECC, with single bit correction, including writeback.
WDR LUT	2	ECC, with single bit correction, including writeback.
H3A LUT	2	ECC, with single bit correction, including writeback.
DPC LUT	1	ECC, with single bit correction, including writeback.
DPC STATS	1	ECC, with single bit correction, including writeback.
DPC (line buffer)	8	No ECC
LSC GAIN LUT	1	ECC, with single bit correction, including writeback.
H3A accumulator	1	No ECC
H3A line	1	No ECC

7.7.3.3.3 RAWFE Interrupts

The following section lists the RAWFE interrupts

7.7.3.3.3.1 RAWFE CPU Interrupts

Table 7-93 lists the interrupts generated by the RAWFE sub-modules.

Table 7-93. RAWFE Interrupts

Interrupt	Description
LUT1_CFG_ERR	Config access to LUT1 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
LUT2_CFG_ERR	Config access to LUT2 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
LUT3_CFG_ERR	Config access to LUT3 ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if cfg access occurs during active line. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.

Copyright © 2026 Texas Instruments Incorporated

Table 7-93. RAWFE Interrupts (continued)

Interrupt	Description
WDR_LUT_CFG_ERR	Config access to WDR LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
H3A_LUT_CFG_ERR	Config access to H3A LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any (read/write) cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
H3A_ACCM_CFG_ERR	Config access to H3A accum ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
H3A_LINE_CFG_ERR	Config access to H3A line ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
DPC_LUT_CFG_ERR	Config access to DPC LUT ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame.
DPC_LINE_CFG_ERR	Config access to DPC line ram has corrupted functional operation, config read or write memory access occurred during functional operation. Only asserted if any cfg access occurs during active line OR write occurs in horizontal blanking within a frame.
LSC_CFG_ERR	Config access to LSC ram has corrupted functional operation, config read or write memory access occurred during functional operation. Asserted if any cfg access occurs during active frame. Active frame in lsc case is VS at pwl to ve at lsc input delayed by 1 cycle. LSC_CFG_ERR interrupt may be disabled by setting DISABLE_LUT_CFG_ERR register bit to '1'.
AEW_PULSE	H3A aew pulse interrupt.
AF_PULSE	H3A af pulse interrupt
H3A_PULSE	H3A pulse interrupt
H3A_BUF_OVERFLOW_PULSE	H3A interrupt generated when an overflow occurs in the H3A output buffer.
DPC_STATS_READ_ERR	This interrupt is set when reading by SW to DPC stats is started after frame n is completed but the last valid stats entry is not read by the end of frame n+1. write 1 to clear event status, write 0 has no affect. To avoid dpc_stats of frame n to be overwritten by frame n+1 processing, DPC_STATS_UPDATE_CFG should be programmed to be 2'b01 and 2'b10 alternately for successive frames.

7.7.3.3.3.2 RAWFE Debug Events

Table 7-94 lists the RAWFE debug events. Debug events are used for hardware debug as well as performance measurement. They are not intended to be used as interrupts where the cpu needs to process them.

Table 7-94. RAWFE Debug events

Event	Description
VS_EVENT	ventricle start in the beginning of the rawfe pipeline
VE_EVENT	ventricle end at the end of the rawfe pipeline
HS_EVENT	horizontal start in the beginning of the rawfe pipeline
HE_EVENT	horizontal end at the end of the rawfe pipeline
LSE_SLAVE_STALL	rawfe is stalling lse target interface due to ext or mtc stall
LSE_MASTER_STALL	lse controller interface is stalled
LSE_INTERFACE_IDLE	within a frame lse is not sending data
X_Y_EVENT_MATCH	x-y pixel position has reached the start of rawfe pipeline
DPC_OTF_CORR_EVENT	dpc otf corrected a pixel position

Copyright © 2026 Texas Instruments Incorporated

Table 7-94. RAWFE Debug events (continued)

Event	Description
PIPE_ADV_EVENT	active pipeline advancement

7.7.3.3.4 RAWFE Sub-Modules Details

The following sections describe in detail the sub-modules present within the RAWFE module.

7.7.3.3.4.1 RAWFE Decompanding Block

The decompanding block is responsible for decompanding pwl (or LUT) compressed data back to its linear bit width. The decompanding block can generate up to 24 bits of data, however the data needs to be tone mapped down to 16 bits before additional processing can take place. There are 3 instances of the Decompanding block each implementing the same functionality.

The input interface of the decompanding block is 16 bits.

Figure 7-32 shows a high level block diagram of the decompanding block.

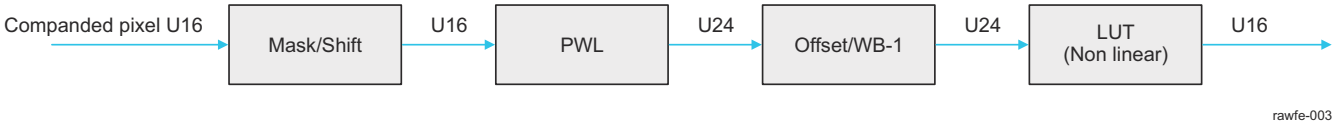


Figure 7-32. RAWFE Decompanding Block Diagram

7.7.3.3.4.1.1 RAWFE Mask & Shift

The mask and shift block is used to mask out any control bits present in the data. The ‘Mask’ is a 16 bit configuration register which will perform a bit wise ‘and’ operation on the incoming data. (For example for sensor data with data in 12 bits and 4 bits of control information, the mask value would be programmed to 0xFFF). The Mask operation is followed by a shift operation. The ‘Shift’ field is a 3 bit configuration register which can perform a right shift (>>) from 0 to 7 bits.

7.7.3.3.4.1.2 RAWFE Piece Wise Linear Operation

The PWL block is used to apply a piece-wise-linear gain on the incoming pixel data and is generally used to uncompress sensor compressed data. The knee points of the PWL curve should be programmed to correspond with the compression knee points used by the sensor configuration.

The PWL block supports an input size from 8 – 16 bits and can support an output up to 24 bits.

Figure 7-33 depicts a PWL curve applied on incoming data

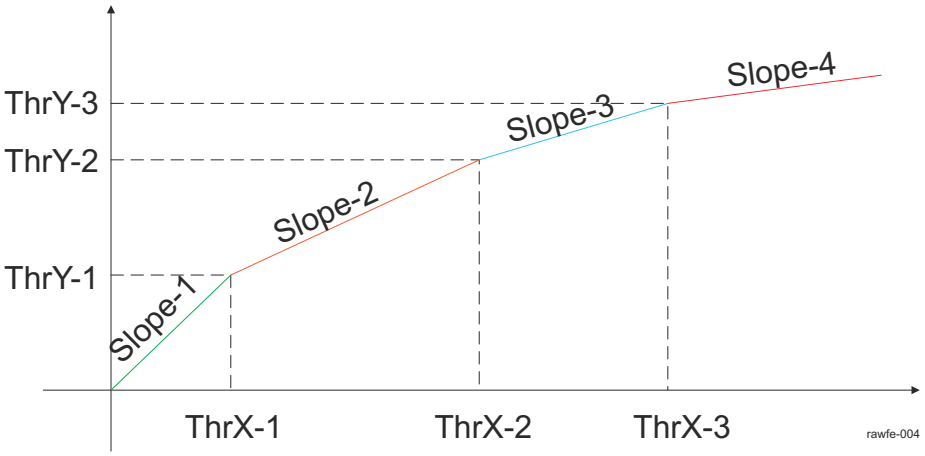


Figure 7-33. RAWFE PWL Curve implementation

Copyright © 2026 Texas Instruments Incorporated

The PWL block implements the following equation assuming the input lies in the nth segment.

$$Out_Pixel = ((X - ThrX_n) \times Slope) + ThrY_n \quad (2)$$

The figure below shows the PWL block architecture

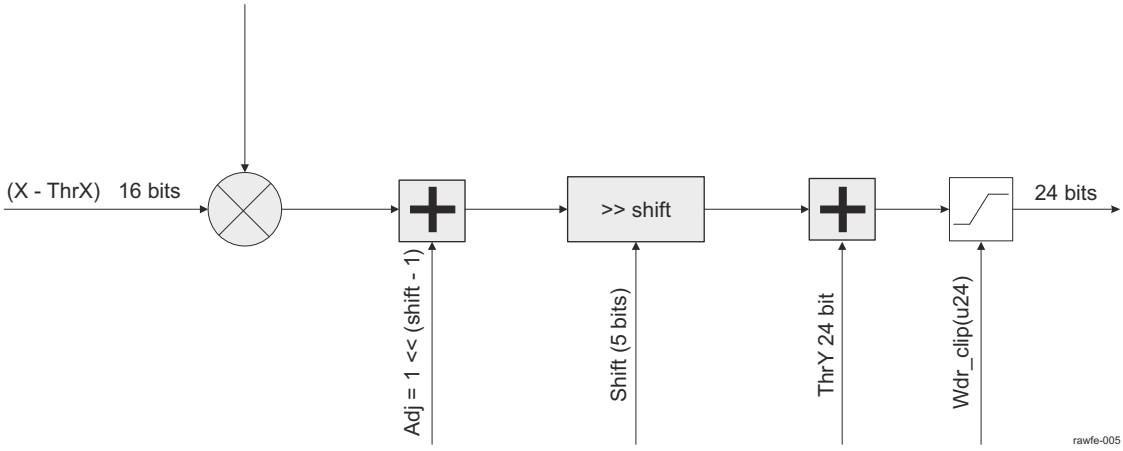


Figure 7-34. RAWFE PWL Block Diagram

The Wdr_clip parameter can be set to clip the output at the desired bit width (typically 20 bits).

7.7.3.3.4.1.3 RAWFE Offset/WB-1 Block

This block is used for applying White balance correction in linear domain as well as for subtracting the DC offset using the signed 24 bit offsets. The functionality is similar to the WB2 block. Note, there is a tap out point to H3A after the offset application, but prior to the gain block. This allows the H3A to receive DC subtracted (but not WB corrected) data.

7.7.3.3.4.1.4 RAWFE LUT Based compression

The LUT based compression is used for addressing multiple use cases.

- The primary function of the LUT is to take the PWL decompounded data (up to 24 bits and after DC subtraction) and compress it using an x^n curve to a bitwidth of 16 bits. This is required since the downstream path is only capable of supporting up to 16 bits of data.
- It can be used to decompand data when the compression curve cannot be addressed using 3 knee points or if the compression is true curve instead of PWL. In this scenario the DC subtraction IP is disabled and the LUT is used to implement both the decompression as well as the DC subtraction functionality.

The LUT compression is modified from the implementation in previous generation ISPs. The previous LUTs had 512 entries leading to a step size of 2^20/512 = 2048. As such there were only 2 LUT points in the critical low range (0-4095) of the image. The LUT implementation is optimized for a 20-bit decompounded data and allows for a step size of 32 in the critical range (lower 12 bits) and a step size of 2048 in the successive ranges.

The Non Linear LUT implementation supports 3 zones for the input range

- For bitWidth < 12 bits, the LUT offers a linear range with a step size of 32. (Only the first 128 locations (plus 129th location for supporting interpolation) are utilized in this mode
- For bitWidths between 13 and 20: This is the most common operating zone for the LUT. In this mode, the range between 0 and 4k is split in 128 locations with a step size of 32. Beyond 4k range, the step is variable and is set to 2k for the highest bit width of 20 and subsequently reduces by half as the bitwidth is reduced (E.g. 1k for 19 bits etc)
- For bitwidths between 21 and 24: In this operation zone, the first 128 locations are used with a varying step size based on bit width such that the step size is 64 for 21 bits of data and doubles with increase in bitWidth.

Copyright © 2026 Texas Instruments Incorporated

(E.g. 128 for 22 bits etc). The remaining locations implement a varying step size between 4k (for 21 bits) and 32k (for 24 bits).

- Example-1: Bitwidth=21 bits:
  - Range 0-8k-> Step size = 64;
  - Range 8k- 2^21 -> Step Size = 4096
- Example-2: Bitwidth = 22 bits
  - Range 0 -16k -> Step Size = 128
  - Range 16k – 2^22 -> Step_Size = 8192
- Example 3: Bitwidth = 24 bits
  - Range 0 – 64K -> Step Size = 512
  - Range 64K - 2^24 -> Step Size = 32k

The LUT logic provides a generic table which can support any bit width and can be used in different regions in the RAWFE as well as other modules.

The LUT3 supports shadowing for very short frame only. If the MMR shadow enable is set then the LUT2 is used on the LUT3 data.

7.7.3.3.4.2 RAWFE WDR Merge Block

The merge block supports merging up to 3 exposures to generate a cumulative 20 bit frame.

The merge block is based on 2 instances of the Motion-Adaptive merge block, each of which is independently capable of supporting the merging of 2 exposures (10 – 16 bits each) to a combined bit width of up to 20 bits. In a more generic scenario the merge block should be designed and verified to support the following scenarios

- Merge 2 exposures of 10 bits each to 16 bits
- Merge 2 exposures of 12 bits each to 16 bits
- Merge 2 exposures of 12 bits each to 20 bits
- Merge 2 exposures (12 bit and 16 bit) to 20 bits.
- Merge 2 exposures of 16 bit each to 20 bits.

Using two instances of the motion adaptive merge block provides the functionality to merge up to 3 independent exposures, each of 12 bits, to a combined bitwidth of 20 bits. The block also provides capability to only merge 2 exposures using a combination of mux and clipping blocks.

The merge block comprises of two instances of the motion adaptive merge (MA1/MA2) as well as an LUT based companding block and mux structures to enable different data flow. To keep the design complexity under control some restrictions are applied to the data connectivity on the merge blocks. A key restriction is that for the two exposure merge only the MA-2 block should be used and the MA-2 should be used along with MA-1 for 3 exposure merge. Further the position of the long and short exposures on each block are fixed. Table 7-95 summarizes some of the possible combinations which can be realized using the logic. (Note, this is not an exhaustive list from a DV perspective).

The MA block when set to bypass/disabled mode, will pass the input stream marked as ‘short’ (refer to Figure 7-30) to the output.

Table 7-95. Figure 7:RAWFE Different merge modes

Merge Mode	MA-1	MA-2		LUT
2 Exposures (12 bit) to 20 bit	Bypassed	Merge Long / Short	Use VS and Short Exposures	Set to 20-16
2 Exposures (12 bit) to 16 bit	Bypassed	Merge Long /Short	Use VS and Short Exposures	Bypassed
2 Exposures (12 bit) to 16 bit with tone mapping	Bypassed	Merge Long /Short	Use VS and Short Exposures	Set to 16-16 mode
2 Exposures (12 & 16) to 20 bit	Bypassed	Merge Long /Short	Use VS and Short Exposures	Set to 20-16
3 Exposures (12 bit each) to 20 bit	Merge Long and Short	Merge VS / MA-1 output	Use all 3 exposures	Set to 20-16
Single exposure companded	Bypass	Bypass	Use VS exposure only	Bypassed

Copyright © 2026 Texas Instruments Incorporated

7.7.3.3.4.2.1 RAWFE WDR Motion Adaptive Merge (MA1 / MA2)

This section provides the detailed description of the two instances of the motion adaptive merge.

Figure 7-35 shows the high level block diagram of the MA merge block

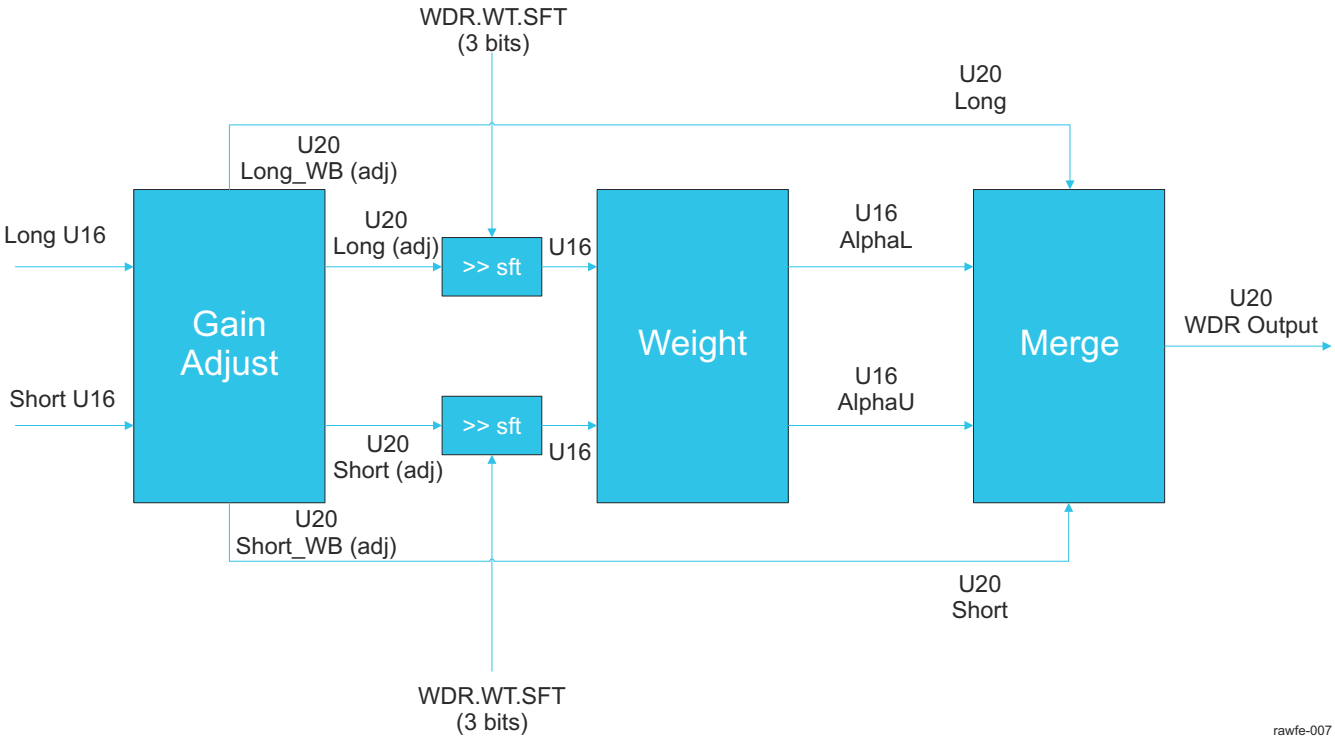


Figure 7-35. RAWFE WDR Motion Adaptive Merge

The MA block receives two inputs each of which can be up to 16 bits. As mentioned earlier, the ordering of the long and the short exposure cannot be changed and should be positioned as it's depicted in the figure above. The MA block consists of 3 main sub-blocks, which are described in the following sub-sections.

7.7.3.3.4.2.2 RAWFE Companding LUT

The companding LUT is used to perform a global tone mapping on the image and reduce the bit depth from 20 bits down to 16 bits. The LUT is implemented as a non uniformly spaced memory, with higher precision in the lower range of the image intensity. The LUT architecture is similar to that used for the decompanding LUT in the previous section.

7.7.3.3.4.3 RAWFE Defective Pixel Correction (DPC) Block for 2x2 Bayer CFA

The defect pixel correction (DPC) block is responsible for correcting defective pixels present on the sensor as an artifact of the manufacturing process. The defect pixel detection can either be LUT based or on the fly (OTF – DPC). For LUT based DPC, the defect map is stored in a memory.

Figure 7-36 shows a high level block diagram of the DPC module. The DPC comprises of two different methods of performing the functionality, each of which is highlighted below. The LUT based defect correction mechanism requires the sensor to be calibrated and the defect positioned stored in memory, whereas the Adaptive DPC automatically detects defects and corrects them. If required the LUT DPC and adaptive DPC can be enabled together in sequence.

Copyright © 2026 Texas Instruments Incorporated

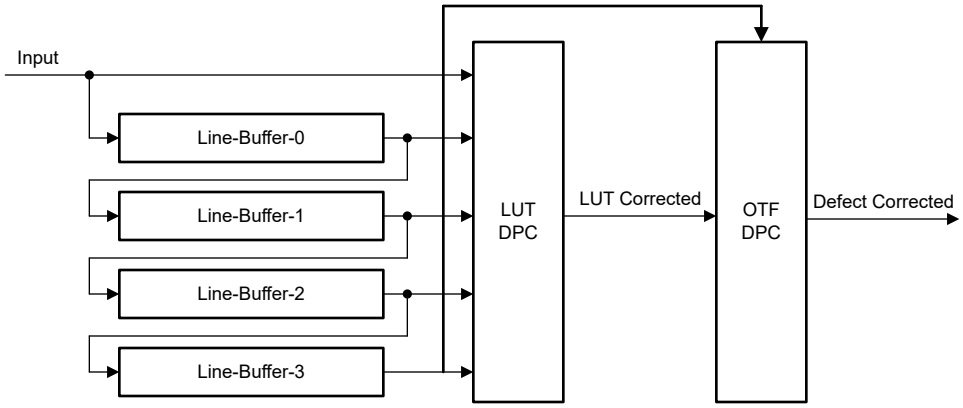


Figure 7-36. RAWFE DPC Block Diagram

7.7.3.3.4.3.1 RAWFE LUT Based DPC

LUT based defect pixel correction provides the highest level of accuracy since all defects can be identified perfectly without any false positives. The LUT based DPC mechanism however suffers from the limitation that the sensor has to be calibrated (each sample has to be individually calibrated) and that can add cost to the testing and qualification process. Some automotive grade sensor vendors provide the defect pixel locations already in an on chip memory and that can additionally be utilized to program the defect LUT without adding the cost. The LUT size for the DPC is constrained to 256 entries, since it is assume for a sensor of 2 -3 Mpix resolution, 256 entries are more than sufficient. Each entry in the LUT is 29 bits with 13 bits for x coordinate, 13 bits for Y coordinate and 3 bits for the method. Figure 7-37 illustrates the organization of the LUT memory

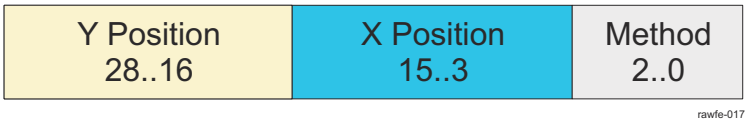


Figure 7-37. RAWFE DPC LUT Memory Organization

A sample table is provided below for 7 entries. The LUT is specified in left to right and top to down fashion (Raster Scan order). As such design only has to look at the current LUT pointer to be able to determine if the current location is a defect or not.

Table 7-96. Figure 14: RAWFE Sample DPC Table

10	1400	0
236	1107	1
236	1200	1
488	10	1
488	100	2
800	138	3
900	1000	1
900	1100	3

The method filed pertains to how the defect pixel is treated once it has been detected. Several possibilities arise including copying from one of the neighbors or using an average of the neighbors. The table below summarizes different methods that can be programmed for treating defect pixels.

Table 7-97. Figure 15:RAWFE LUT DPC Method(s)

Method Value	Functional equivalent
0	Replace with Black or white dot based on config

Copyright © 2026 Texas Instruments Incorporated

Table 7-97. Figure 15:RAWFE LUT DPC Method(s) (continued)

Method Value	Functional equivalent
1	Dout = d4
2	Dout = d5
3	Dout = (d4 + d5)/2
4	Dout = (d2 + d7)/2
5	Dout = d2
6	Dout = d7
7	Dout = ((d2 + d7)/2 + (d4 + d5)/2)/2

The pixel arrangement and numbering convention is depicted in the image below with ‘d0’ being considered the center pixel.

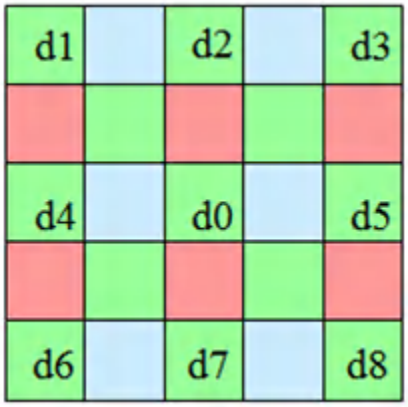


Figure 7-38. RAWFE DPC Pixel numbering convention

When the DPC method is set to ‘0’, the output is either a black or a white pixel (depending on a separate configuration register). With this method, the LUT dpc can be used in conjunction with OTF DPC in a way that the LUT programmed defects are almost always identified by the adaptive DPC. This can be used to accurately correct for all known defects and then using adaptive DPC to correct for temperature or other operating condition based defects. (Defect pixels are strongly correlated with analog gain).

7.7.3.3.4.3.2 RAWFE On-The-Fly (OTF) DPC

The LUT based DPC suffers from the drawback that each sensor die has to be calibrated for defects. The calibration process adds cost to the sensor dies, as such the preferred DPC method relies on automatically detecting the defects based on thresholds.

The OTF DPC algorithm detects defective pixels by comparing the current pixel (d0 on the left of the figure below) against its 8 neighboring pixels with the same color (d1 ~ d8 on the left of the figure below). Let’s use dmax and dmin to denote the maximum and minimum of d1 ~ d8 respectively. If the current pixel d0 is greater than the sum of dmax and a detection threshold T (i.e., d0 > dmax + T) as shown on the right of the figure below, then the current pixel is considered a defect and its value is replaced by dmax. Similarly, if d0 < dmin – T, then the current pixel is replaced by dmin.

Copyright © 2026 Texas Instruments Incorporated

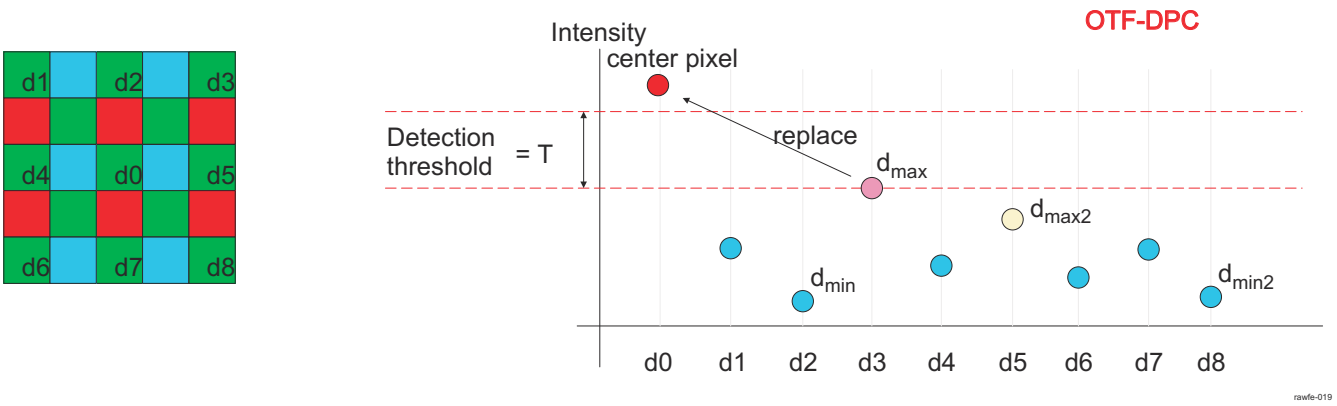


Figure 7-39. RAWFE OTF DPC

The detection threshold T is made adaptive by calculating the local image intensity and using a programmable look-up-table. The local intensity is approximated by the average of the second largest and the second smallest pixel values among d1 ~ d8 (dmax2 and dmin2 respectively in the figure above). With this average value, the threshold T is obtained from the look-up-table with linear interpolation. The look-up-table contains the detection thresholds (U16) at the average values of 0, 512, 1024, 2048, 4096, 8192, 16384, 32768 and the corresponding slope values (S12Q8) for linear interpolation.

7.7.3.3.4.4 RAWFE Defective Pixel Correction (DPC) for 4x4 RGBIR CFA

The DPC algorithm uses the same color neighbors to detect and correct the pixel that is in concern. The 4x4 RGBIR censor data has lower density in red and blue pixels. A larger neighborhood (9x9 pixels) is used to extract 8 same color pixels (d1 to d8) for defective pixel detection and correction for the center pixel d0. The 8 pixels provide the vertical, horizontal, and two diagonal directional neighbors to the center pixel d0.

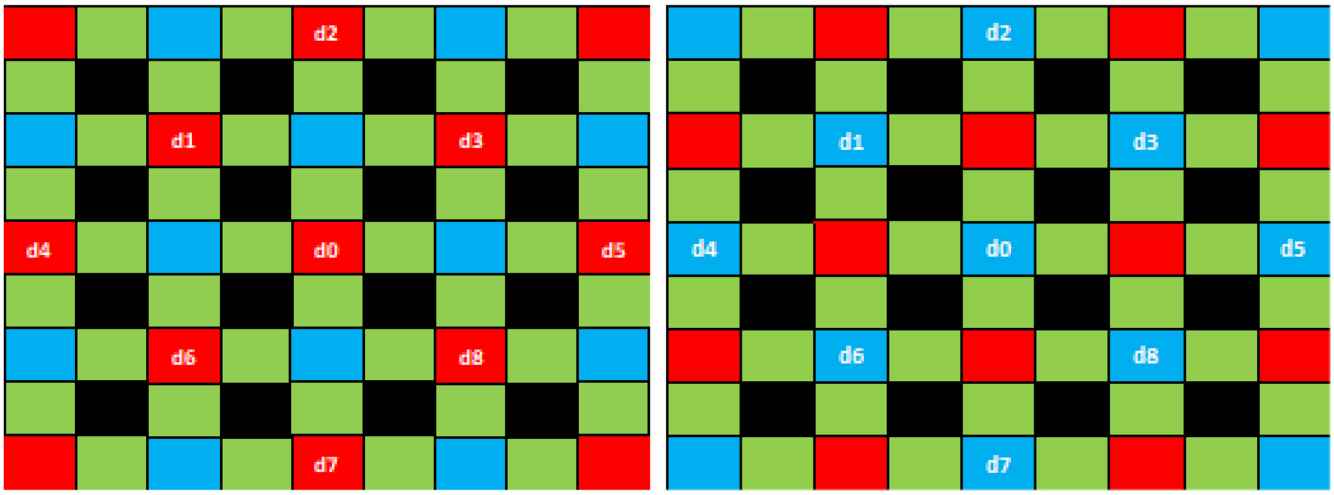


Figure 7-40. Pixels Used in Detecting and Correcting Red and Blue Pixels

Figure 7-40 shows the 8 pixel neighbors (d1 to d8) used to detect and correct center pixel d0 for red and blue pixels. For green and IR pixels, the 8 neighbor pixels are selected in the same way as 2x2 Bayer.

A programmable x-position parameter is added to the threshold and slop LUT to add flexibility in define the piece-wise linear detection threshold curve. The LUT now has 8 entries, each consisting a x-position of threshold curve, a threshold at x-position and a slop at which the threshold changes.

The 2nd LUT, LUT2, is added to allow different color channels to use different threshold curves. For a given color channel, which LUT to use is configured by LUT_MAP register. Since the position of a color channel is

Copyright © 2026 Texas Instruments Incorporated

depending on the pixel color phase in a 4x2 pixel array, OTF DPC uses CFA_PHASE register to determine the pixel positions for a given color channel.

A DPC_COUNT register is used to count the number of detected defective pixels. The count is reset to 0 when a frame starts. It is incremented by one every time when a defective pixel is detected and saturated to 65535.

In OTF DPC, a defective pixel stats memory of 2048 entries is added to record the defective a pixle's x- and y-positions. When enabled, the first 1024 detected defective pixel location is written to the lower 1024 or higher 1024 entries, depending on the configuration. SW can configure OTFDPC_EN_STATS_CFG to update lower or higher stats memory in alternative frames, leaving a whole frame time for SW to read the DPC stats. The defective pixels stats memory is also available in the enhanced 2x2 Bayer mode. The x- and y- positions are organized as in the table below.

Y Position	X Position
25...13	12...0

7.7.3.3.4.5 RAWFE Lens Shading Correction (LSC) and Digital Gain (DG) Block

The lens shading module corrects for the lens fall of (loss of light) at the corners of the lens. Figure 7-41 shows a sample image of this phenomenon and the corresponding result after treating with LSC operation.

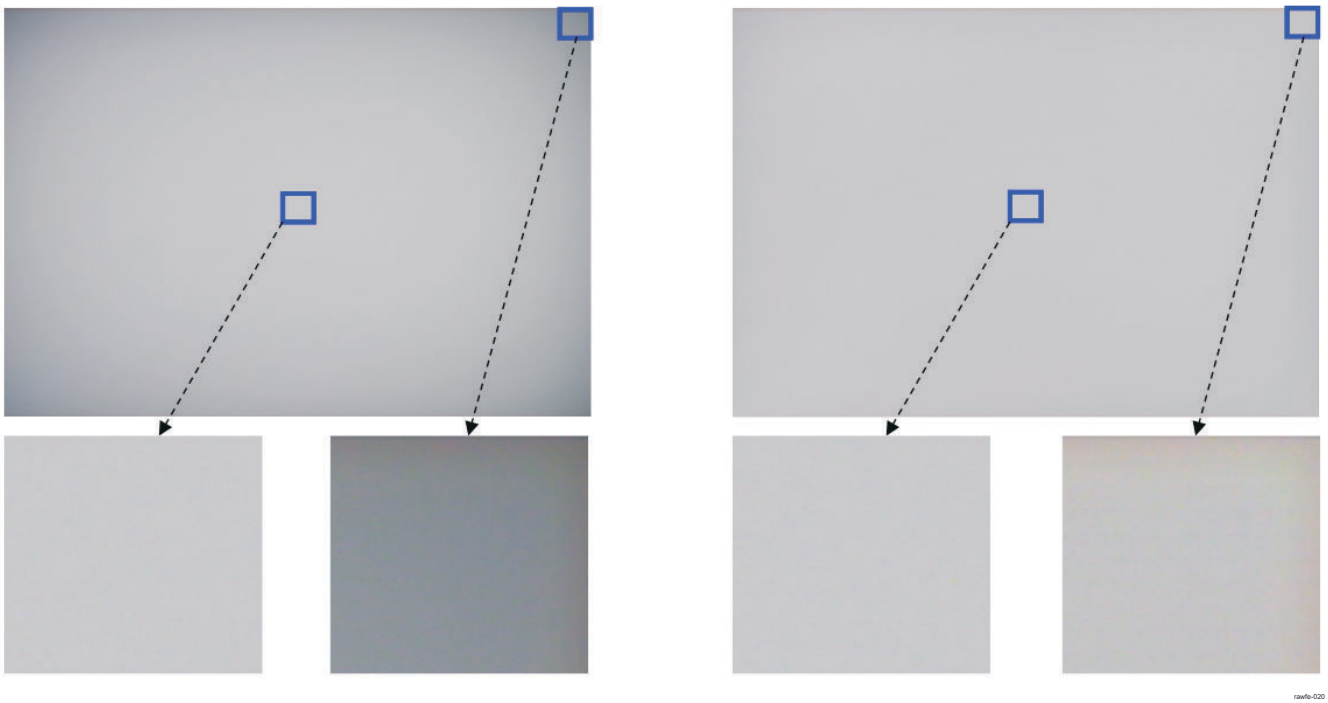


Figure 7-41. RAWFE Before and After LSC

The image on the left shows a block at the center of the lens and a corresponding block at the edge of the lens. Both blocks are equally illuminated, as such should appear equally bright on the image. However, due to lens fall off, the block at the edge appears significantly darker than the block at the center. The image on the right shows the corresponding result after LSC gains have been applied.

This block (2D-LSC) contains lens shading correction by multiplying an image with a gain factor 2-D map, pixel by pixel. The image is conceived to be in Bayer CFA format having a 2x2 color pattern or 4x4 RGBIR pattern. The gain factor map is stored in internal MEMORY down-sampled, and is accessed and up-sampled by the LSC module to pixel resolution before being applied to the pixel data. The key difference from previous generation LSC is that the gain map is stored internally in the memory instead of relying on DRAM fetch.

Copyright © 2026 Texas Instruments Incorporated

The LSC module does not implement any Region of Interest (ROI) functionality and the LSC gain mask is applied on the entire image. Further, only an 8 bit gain is stored in the mask. The gain can be stored in different formats depending on the range. The 8 bits of gain can be used to represent different ranges based on the lsccfg.GAIN_FORMAT register as per the table below

- 0 : U8Q8 -> 0 to 0.996
- 1 : U8Q8 +1 -> 1 to 1.996
- 2 : U8Q7 -> 0 to 1.992
- 3 : U8Q7 +1 -> 1 to 2.992
- 4 : U8Q6 -> 0 to 2.984
- 5 : U8Q6 +1 -> 1 to 3.984
- 6 : U8Q5 -> 0 to 6.968
- 7 : U8Q5 +1 -> 1 to 7.968

Note

The offset field is removed from the implementation of the LUT and only the gain field is retained. This is different than previous implementations.

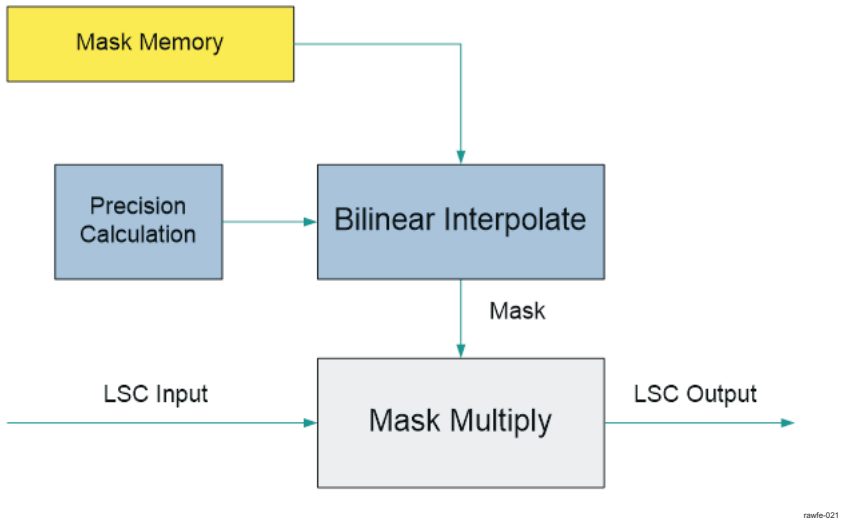


Figure 7-42. RAWFE LSC block diagram

7.7.3.3.4.5.1 RAWFE LSC Features Supported

- Gain map is MxN down-sampled, M being the horizontal sampling factor, N being the vertical sampling factor, M and N being {8,16,32.64,128} independently. For a given frame size, choice of M and N has to be made to ensure the entire LSC LUT to fit into LSC LUT memory.
- The size of the internal memory is based on a 5Mpix sensor (2592x1944) with 32x64 downscaling ratio. As such the size is approximately 20,832 bytes (5248 locations with 4 Bytes/location)
- Every two locations are used to store the channel gains for 8 color channels. Alternatively, the memory can be programmed to support 4 color channels with each location storing channel gains for 4 color channel, Table 4 lists a few example M and N configurations for either 8 or 4 color channels. Note that supporting 8 color channels is only for LSC_CH_MODE = 2.
- Support 8-bit entry in the gain map (in U8Q8, U8Q7, U8Q6, and U8Q5 format with optional base of 1.0 to shift the range up)

7.7.3.3.4.5.2 RAWFE LSC Image Framing with Respect to Gain Map Samples

The gain maps are stored MxN downsampled, and needs to be upsampled by LSC hardware before being applied to the image. It is most straightforward for LSC hardware if the gain map grid is aligned with the input

Copyright © 2026 Texas Instruments Incorporated

image grid. In other words, first pixel corresponds to a source gain map entry, rather than being upsampled through interpolation. Figure 7-43 shows the LSC active region with respect to the gain map grid.

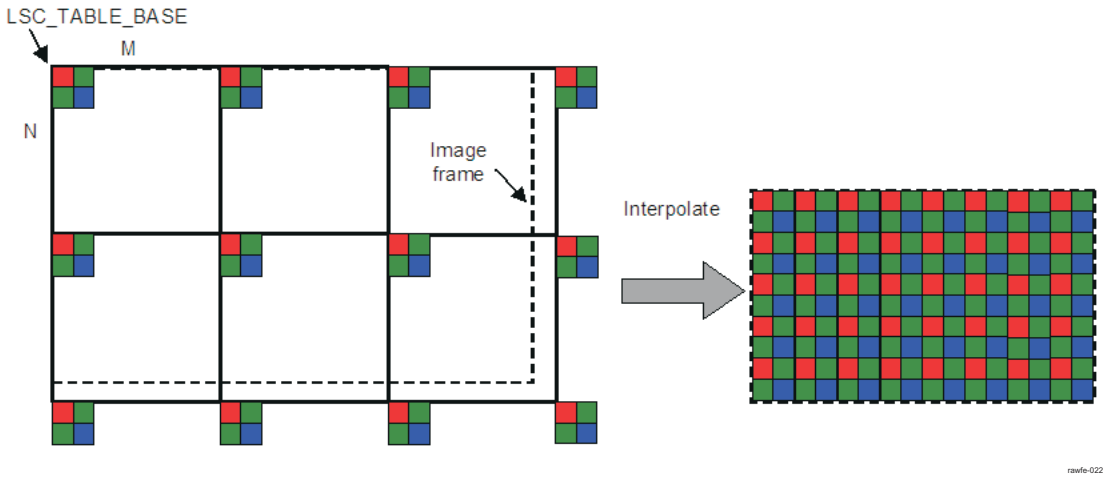


Figure 7-43. RAWFE LSC Active Region with Respect to Gain Map Samples

Figure 7-43 illustrates the LSC active region and map up-scaling feature. (The maps are stored independently per color channel)

For an image frame size of W x H (output and input are the same size), gain map being MxN downsampled, it needs (ceil(W / M) +1) x (ceil(H / N) +1) x 4 bytes of gain map data in internal memory. (NOTE: ceil() represents the ceiling function).

The LSC module is designed to work with Bayer CFA data, having R/Gr/Gb/B color pattern or RGBIR pattern with up to 16 color channels. For the purpose of functional description for Bayer CFA, it is assumed that Red is the starting color, but any other starting color or other 2x2 color pattern can be used by placing color gains in the appropriate order.

Each 2x2 set of samples is stored together in a 32-bit word in internal memory. For R/Gr/Gb/B CFA data and 8x8 downsampled gains, the following order is assumed in the gain map, using conventional (X, Y) coordinate notation (X for horizontal offset and Y for vertical offset):

Line 0: R(0,0) Gr(1,0) Gb(0,1) B(1,1) R(8,0) Gr(9,0) Gb(8,1) B(9,1)...

Line 1: R(0,8) Gr(1,8) Gb(0,9) B(1,9) R(8,8) Gr(9,8) Gb(8,9) B(9,9)...

7.7.3.3.4.5.3 RAWFE LSC Gain Map Upsampling in 2x2 Mode (LSC CHN_MODE = 0)

Before applying to pixel data, the gain maps are upsampled to pixel resolution via bilinear interpolation.

Note that the LSC operation is neutral to color pattern. The starting color of the gain map should be consistent with the starting color of the image, and can be any color. The hardware upsamples each phase of the gain map as a separate plane, and applies the upsampled gain map to the image with the same color phasing.

Upsampling of the pixel-by-pixel gains is performed by locating the 4 same-color anchors for each destination gain value, and applying bilinear interpolation. Gain for a red pixel at destination coordinate (x, y), relative to the starting pixel of the image, is computed as described below.

Let the 4 anchors for a red gain output be denoted by R00, R01, R10, and R11, and the destination coordinate is (k, l) with respect to R00, the red gain is interpolated as:

$$R\left(x,y\right) = \frac{R00 \times (M-k) \times (N-l) + R01 \times k \times (N-l) + R10 \times (M-k) \times l + R11 \times k \times l}{MN} \quad (3)$$

Where:

Copyright © 2026 Texas Instruments Incorporated

- k = x mod M
- l = y mod N
- i = | x / M | × M
- j = | y / N | × N
- R00 = R(i, j)
- R01 = R(i+M, j)
- R10 = R(i, j+1)
- R11 = R(i+M, j+N)

Similarly for the other colors the formulas are:

$$GR\left(x,y\right) = \frac{GR00 \times (M-k) \times (N-l) + GR01 \times k \times (N-l) + GR10 \times (M-k) \times l + GR11 \times k \times l}{MN} \quad (4)$$

$$GB\left(x,y\right) = \frac{GB00 \times (M-k) \times (N-l) + GB01 \times k \times (N-l) + GB10 \times (M-k) \times l + GB11 \times k \times l}{MN} \quad (5)$$

$$B\left(x,y\right) = \frac{B00 \times (M-k) \times (N-l) + B01 \times k \times (N-l) + B10 \times (M-k) \times l + B11 \times k \times l}{MN} \quad (6)$$

In legacy 2x2 mode (LSC_CFG.CHN_MOD register field = 0), however, the relation of (k, l) and (i, j) to (x, y) for other colors is different from the relation for red. For GR, the relations are:

- k = (x-1) mod M
- l = y mod N
- i = | (x-1) / M | × M + 1
- j = | y / N | × N
- GR00 = GR(i, j)
- GR01 = GR(i+M, j)
- GR10 = GR(i, j+N)
- GR11 = GR(i+M, j+N)

In legacy 2x2 mode, the number of supported color channels is fixed at 4.

7.7.3.3.4.5.4 RAWFE LSC Gain Map Upsampling in 4x4 Color Mapping Mode (LSC CNH_MODE = 1 or 2)

In both of the modes, mapping 4x4 color channels to up to 4 or 8 channel gains is configured by the LSC_LUT_MAP07 and LSC_LUT_MAP815 registers. The 16-channel LSC_LUT_MAP is enumerated spatially, independent of the actual pixel color, in each 4x4 CFA as shown in Figure 7-44.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure 7-44. Channel Indexing in 4x4 Pixel Pattern

The mapping is intended for 4x4 RGBIR support. However, it can also be used in 2x2 Bayer sensors by repeating the 2x2 pattern 4 times (2 horizentally and 2 vertically) to form 4x4 pattern and applying 16 LSC_LUT_MAP for corresponding16 pixels to 4 or 8 channel gains in the LUT.

When LSC_CFG.CHN_MODE register field = 1, 4-channel gain is supported. All the channel mapping has to point to gain 0-gain3. When CHN_MODE = 2, 8-channel gain is supported. All the 8 channel gains may be

Copyright © 2026 Texas Instruments Incorporated

used across frames. Alternative, the 8 gains may be configured to a dual 4 channel gain groups (gain0-3 and gain4-7), allowing re-programming of 4 gains for next frame while other 4 gains are used for the current frame. This feature could be useful for multiple sensor support where different sensors may need different gains for shading correction.

Upsampling of the pixel-by-pixel gains is performed by locating the 4 same color channel anchors for each pixel gain value, and applying bilinear interpolation. Let the 4 anchors for a channel gain output be denoted by CH00, CH01, CH10, and CH11, and the pixel coordinate is (k, l) with respect to CH00. Gain for a pixel channel at destination coordinate (x, y), relative to the anchor points, is computed as:

$$\mathrm{CH}\left(\mathrm{x},\mathrm{y}\right) = \frac{\mathrm{CH00} \times (\mathrm{M}-\mathrm{k}) \times (\mathrm{N}-\mathrm{l}) + \mathrm{CH01} \times \mathrm{k} \times (\mathrm{N}-\mathrm{l}) + \mathrm{CH10} \times (\mathrm{M}-\mathrm{k}) \times \mathrm{l} + \mathrm{CH11} \times \mathrm{k} \times \mathrm{l}}{\mathrm{MN}} \quad (7)$$

Where:

- k = x mod M
- l = y mod N
- i = | x / M | × M
- j = | y / N | × N
- R00 = CH(i, j)
- R01 = CH(i+M, j)
- R10 = CH(i, j+1)
- R11 = CH(i+M, j+N)

7.7.3.3.4.6 RAWFE Gain & Offset Block

The gain block allow 2 primary functions

- Allow for digital gain to be applied if the image is too dark even after analog gain and exposure time have been set to the maximum
- Independent gain setting for each color channel in 13 bit format U13Q9 with 9 bits of fraction.
  - Allows a gain value from 0 to 31.996 in steps of 1/256.

The Gain block can be configured to apply an offset on top of the gain. This is to support the log compressed image mode, where in the traditional WB Gain becomes an offset.

7.7.3.3.4.7 RAWFE H3A

7.7.3.3.4.7.1 RAWFE H3A Overview

The H3A module supports the control loops for autofocus, auto white balance, and auto exposure by collecting statistics about the raw image. The metrics are used to adjust parameters for processing the imaging/video data. There are two main blocks in the H3A module:

- Autofocus (AF) engine:

  The AF submodule extracts and filters the red, green, and blue data from input image data and provides the accumulation or peaks of the data in a specified region. The specified region is a 2D block of data referred to as a paxel. The AF engine supports the following features:
  - Peak mode in a paxel: Accumulation of the maximum focus value (FV) of each line in a paxel
  - Accumulation mode in a paxel
  - Accumulation of horizontal and vertical focus value in a paxel
  - Up to 12 paxels in the horizontal direction and up to 12 paxels in the vertical direction with vertical focus
  - Up to 36 paxels in the horizontal direction and up to 128 paxels in the vertical direction with horizontal focus only
  - Programmable width and height for the paxel/window
  - Programmable red, green, and blue position within a 2 × 2 matrix
  - Separate horizontal start for paxel and filtering
  - Programmable vertical and horizontal line increments within a paxel

Copyright © 2026 Texas Instruments Incorporated

  - Horizontal FV uses parallel infinite impulse response (IIR) filters configured in a dual-biquad configuration with individual coefficients (two filters with 11 coefficients each). The filters are intended to compute the sharpness/peaks in the frame on which to focus.
  - Vertical FV uses a 5-tap FIR filter with 8-bit coefficients. With horizontal steps each paxel has up to 32 columns to be maintained for vertical FV calculation.
- Auto exposure and auto white balance (AE/AWB) engine:

  The AE/AWB engine accumulates values and checks for saturated values in a subsampling of the video data. In the case of the AE/AWB, the 2D block of data is referred to as a window. Thus, other than having different names, paxels and windows are essentially the same. However, the numbers, dimensions, and starting positions of AF paxels and AE/AWB windows are programmable separately. AE/AWB supports the following features:
  - Accumulate clipped pixels along with all nonsaturated pixels in each window per color
  - Accumulate the sum of squared pixels in each window per color
  - Minimum and maximum pixel values in each window per color
  - Support for up to 36 horizontal windows with sum + { sum_sq or min+max} output
  - Support for up to 56 horizontal windows with sum output
  - Support for up to 128 vertical windows
  - Programmable width and height for the windows. All windows in the frame are the same size.
  - Separate vertical start coordinate and height for a black row of paxels that is different than the remaining color paxels
  - Programmable horizontal sampling points in a window
  - Programmable vertical sampling points in a window
- Double-buffer for paxel/window accumulation
- H3A data path is 10 bits.
- Maximum input size is 4096 pixels.

7.7.3.3.4.7.2 RAWFE H3A Top-Level Block Diagram

The block diagram in Figure 7-45 shows the process of the AF and AE/AWB data paths through the H3A module.

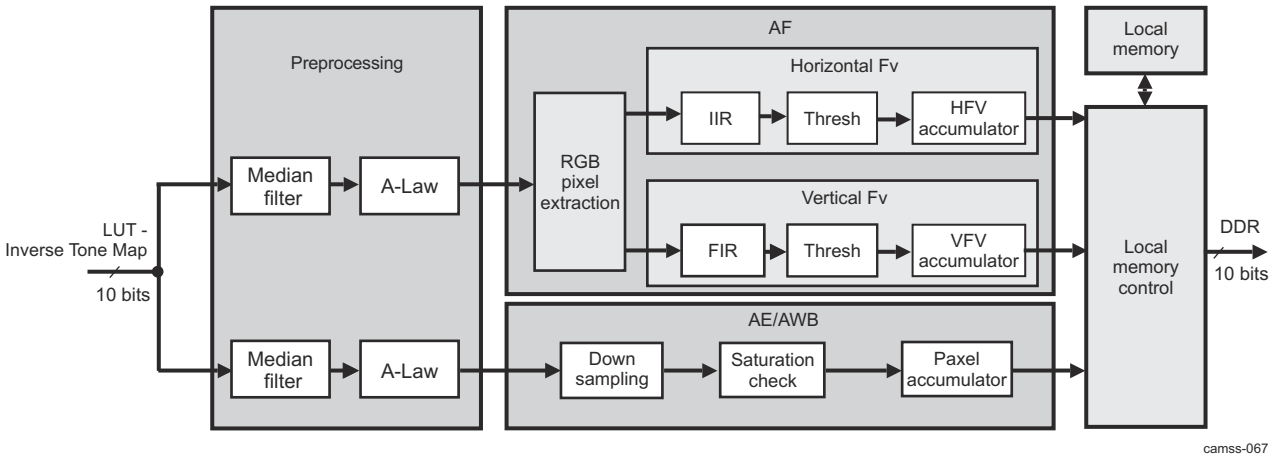


Figure 7-45. RAWFE H3A Top-Level Block Diagram

7.7.3.3.4.7.3 RAWFE H3A Line Framing Logic

In certain cases the number of clock cycles between HD pulses is greater than the line buffer included in the H3A. To solve this problem a framing module was added before the line buffer. The framing module uses the VISS_RAWFE_H3A_LINE_START register to find the position of the first pixel to place into the line buffer. All other registers reference this point as the 0 pixel for their start positions. The line size is 4096 pixels. After 4096 clock cycles the framing logic disables the line buffer and waits until the next HD. If the next HD comes before 4096 clock cycles, then the active region ends immediately and the counter waits

Copyright © 2026 Texas Instruments Incorporated

for the VISS_RAWFE_H3A_LINE_START register count to be reached again. For the vertical position the VISS_RAWFE_H3A_LINE_START[31:16] SLV bit field can be used to determine where the start point of the frame is relative to the rising edge of VD. This logic allows for an active frame to cross VD boundaries and remain in the same frame.

Figure 7-46 shows the RAWFE H3A frame format settings.

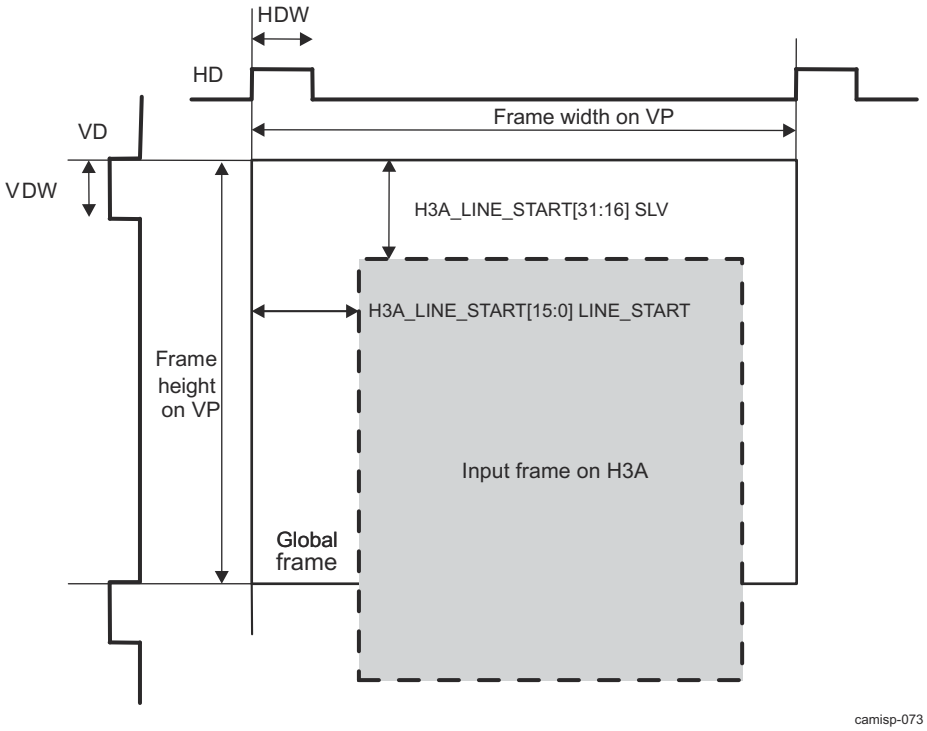


Figure 7-46. RAWFE H3A Frame Format Settings

Note

(Frame width on VP) - (VISS_RAWFE_H3A_LINE_START[15:0] LINE_START) must be less than or equal to 4096, because the H3A memory lines are limited to 4096 pixels.

7.7.3.3.4.7.4 RAWFE H3A Optional Preprocessing

The input to the H3A module is 10-bit RAW data from the IPIPEIF. A 10-bit to 8-bit A-Law compression step can be enabled and disabled separately for the AF engine (the VISS_RAWFE_H3A_PCR[1] AF_ALAW_EN bit) and the AE/AWB engine (the VISS_RAWFE_H3A_PCR[17] AEW_ALAW_EN bit). A-Law compression offers added protection against overflowing the accumulators.

If the A-Law table is enabled, the output is 10 bits, with the upper two bits filled with 0.

For the AF process, a horizontal median filter can be enabled and disabled (the VISS_RAWFE_H3A_PCR[2] AF_MED_EN bit) before A-Law compression. This filter is useful for reducing temperature-induced noise. The horizontal median filter calculates the absolute difference between the current pixel (i) and pixel (i – 2), and between the current pixel (i) and pixel (i + 2). If the absolute difference exceeds a threshold, and the sign of the differences is the same, the average of pixel (i – 2) and pixel (i + 2) replaces pixel (i). The threshold of the horizontal median filter can be set in the VISS_RAWFE_H3A_PCR[10:3] MED_TH bit field.

7.7.3.3.4.7.5 RAWFE H3A Autofocus Engine

The AF engine works by extracting each green (Gr or Gb) pixel from the video stream and subtracts a fixed offset of 128 or 512 (depending of whether A-Law is enabled or disabled) from the pixel value. The offset value

Copyright © 2026 Texas Instruments Incorporated

is then passed through an IIR filter and the absolute value of the filter output is the focus value (FV). Both FV and FV^2 are produced. The FV and FV^2 values can be accumulated or the maximum for each line/column can be accumulated. The following sections describe this process in more detail.

7.7.3.3.4.7.5.1 RAWFE H3A Paxel Extraction

From the paxel starting coordinate (the VISS_RAWFE_H3A_AFPAXSTART[27:16] PAXSH and VISS_RAWFE_H3A_AFPAXSTART[11:0] PAXSV bit fields) specifies the starting point of the paxel grid, with respect to first pixel of the input image frame.

The paxel starting coordinate also indicates which color pixels are extracted if VF is enabled (that is, if VISS_RAWFE_H3A_PCR[20] AF_VF_EN = 1). Normally, either Gr or Gb is used for AF, but it is not important to the hardware whether it is red, green, or blue. If VF is not enabled, then the red, green, and blue pixel extraction is controlled by the VISS_RAWFE_H3A_PCR[13:11] RGBPOS bit field to extract the correct colors from the input stream. Figure 7-47 shows the available options for this bit field. The red and blue pixel positions are interchangeable. For each 2 × 2 grid, the green pixels are summed to create a single value. Because of this, the amplitude of the green output contains 2 pixels, while the red and blue outputs each contain 1 pixel.

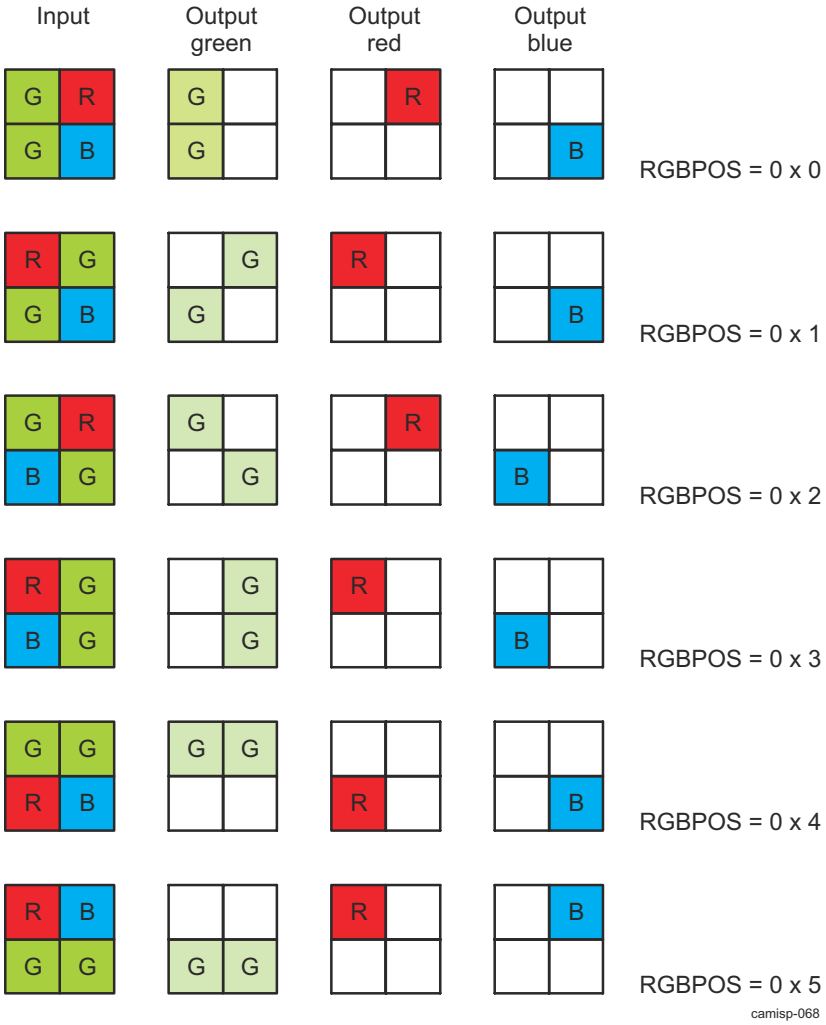


Figure 7-47. RAWFE H3A Red, Green, and Blue Pixel Extraction Examples

Each paxel is VISS_RAWFE_H3A_AFPAX1[23:16] PAXW × VISS_RAWFE_H3A_AFPAX1[7:0] PAXH (width × height) pixels. Inside each paxel, horizontal FV can skip lines, operating on one every VISS_RAWFE_H3A_AFPAX2[16:13] AFINCV lines. Vertical FV can skip columns, operating on one every

Copyright © 2026 Texas Instruments Incorporated

VISS_RAWFE_H3A_AFPAX2[20:17] AFINCH columns. Up to 32 columns are supported for each paxel. If floor (PAXW/AFINCH) >= 32, only the first 32 designated columns are operated on. Because PAXW, PAXH, AFINCV, and AFINCH are all even numbers, AF always operates on the same green color, Gr or Gb. IIR filters for the horizontal FVs start operation at column VISS_RAWFE_H3A_AFIIRSH[11:0] IIRSH.

Figure 7-48 shows the RAWFE H3A horizontal/vertical fv paxel configuration.

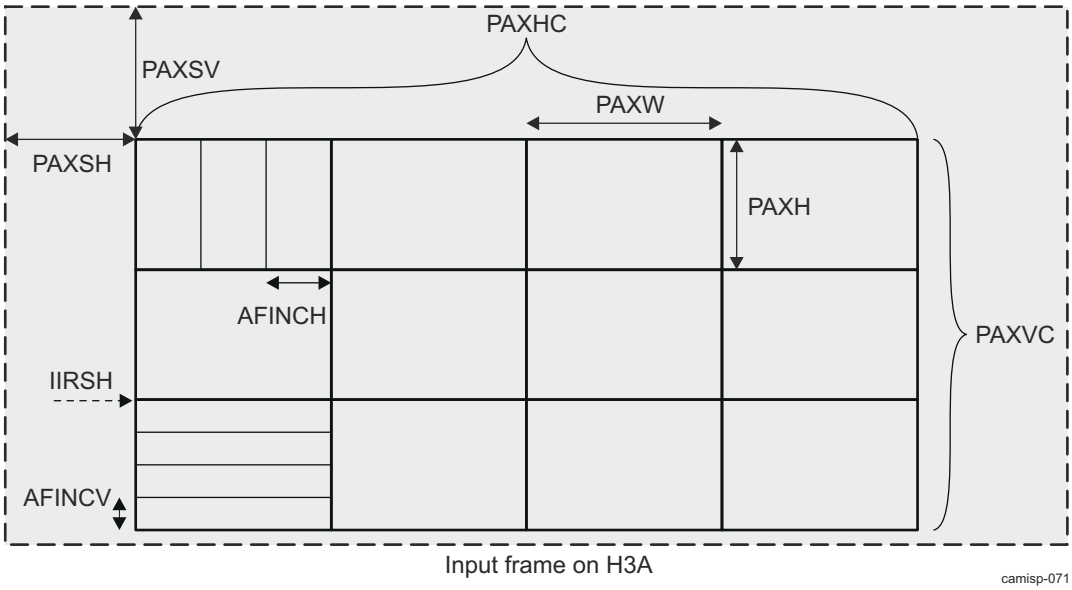


Figure 7-48. RAWFE H3A Horizontal/Vertical FV Paxel Configuration

Note

(VISS_RAWFE_H3A_AFPAXSTART[27:16] PAXSH) + (VISS_RAWFE_H3A_AFPAX2[5:0] PAXHC) × (VISS_RAWFE_H3A_AFPAX1[23:16] PAXW) <= [(Frame width on VP) – (VISS_RAWFE_H3A_LINE_START[15:0] LINE_START)] <= 4096

Table 7-98 lists the bit fields that configure the size and number of paxels.

Table 7-98. RAWFE H3A Paxel Register Field Descriptions

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AFPAX1[23:16] PAXW	8	Paxel width (in pixels)
VISS_RAWFE_H3A_AFPAX1[7:0] PAXH	8	Paxel height (in lines)
VISS_RAWFE_H3A_AFPAX2[5:0] PAXHC	6	Paxel count for horizontal direction
VISS_RAWFE_H3A_AFPAX2[12:6] PAXVC	7	Paxel count for vertical direction
VISS_RAWFE_H3A_AFPAX2[16:13] AFINCV	4	Line increments in a paxel
VISS_RAWFE_H3A_AFPAX2[20:17] AFINCH	4	Column increments in a paxel
VISS_RAWFE_H3A_AFPAXSTART[27:16] PAXSH	12	Paxel start position H
VISS_RAWFE_H3A_AFPAXSTART[11:0] PAXSV	12	Paxel start position V
VISS_RAWFE_H3A_AFIIRSH[11:0] IIRSH	12	IIR filter start position

The H3A AF engine also has an option for an advanced or normal stats collection mode. When 0xCA00 is written to the VISS_RAWFE_H3A_ADVANCED[31:15] ID bit field, then VISS_RAWFE_H3A_ADVANCED[0] AF_MODE can be used to toggle between normal and advanced AF stats collection mode. When the advanced AF stats collection mode is enabled, the ZEROS section of the AF paxel packet is filled with the sum of the maximum FVs, regardless of the color, from HFV_1 and HFV_2.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.3.4.7.5.2 RAWFE H3A Horizontal FV Calculator

The FV calculator takes the unsigned red/green/blue extracted data and subtracts 128 or 512 (depending on whether A-Law is enabled) to place the data in the range –128 to 127 or –512 to 511.

After removing the offset, the data is sent through two parallel IIR filters configured in a dual-biquad configuration. Each filter uses a unique set of 11 programmable coefficients. Each coefficient is 12-bits-wide with 6 bits of decimal, S12Q6 (VISS_RAWFE_H3A_AFCOEF010 to VISS_RAWFE_H3A_AFCOEF0010 for SET0, and VISS_RAWFE_H3A_AFCOEF110 to VISS_RAWFE_H3A_AFCOEF1010 for SET1). The filter-shift registers are cleared on each horizontal line at the position set by the register IIR horizontal start register (the VISS_RAWFE_H3A_AFIIRSH[11:0] IIRSH bit field). The absolute values of the output (16 bits wide with 4 bits of decimal, U16Q4) of both filters are then sent to the AF accumulator module. Signed clipping is performed during the FV calculation. If the input value is m bits (signed) and the required output value is n bits, clipping transforms the input to between -2^1 and 2^1. Values lower than -2^1 are set to -2^1, and values higher than 2^1 are set to 2^1.

7.7.3.3.4.7.5.3 RAWFE H3A HFV Accumulator

The horizontal focus value (HFV) accumulator takes the output of the horizontal IIR filter and accumulates values for each paxel. The size and number of paxels is configurable by registers.

Table 7-98 lists the register fields that configure the size and number of paxels:

- In peak mode (VISS_RAWFE_H3A_PCR[14] FVMODE = 0x1), the maximum value is accumulated.
- In sum mode (VISS_RAWFE_H3A_PCR[14] FVMODE = 0x0), all HFV_n are accumulated in a paxel.

The following equations detail the calculation for:

- Sum of pixel values used in HFV: The pixel values that are used for filtering and accumulation of HFV are also accumulated in this sum of pixel values.
- HFV_n (HFV_n_peak for peak mode or HFV_n_sum for sum mode)
- HFV_count_n
- HFV_sq_n (HFV_sq_n_peak for peak mode or HFV_sq_n_sum for sum mode)

n = 1 or 2 for IIR1 and IIR2, respectively.

For each paxel, these six values are available for each R, G, and B component.

```
for (k=0; k<PAXH) // Loop on paxel rows
{
 rowpeak_n = 0;
 for (l=0; l<PAXW; l++) // Loop on values within a row
 {
 aIIRout_n = ABS(IIRout_n);
 if (aIIRout_n >= threshold_n)
 {
 hfval = aIIRout_n - threshold_n;
 HFV_count_n++;
 }
 else hfval = 0;
 if (hfval > rowpeak_n)
 {
 rowpeak_n = HFV_n;
 }
 HFV_n_sum += hfval;
 HFV_sq_n_sum += (hfval* hfval + RNDADD) >> RNDSHIFT;
 } // Finished looping on values in a row
 HFV_n_peak += rowpeak_n;
 HFV_sq_n_peak += (rowpeak_n * rowpeak_n + RNDADD) >> RNDSHIFT;
}
```

- threshold_n is VISS_RAWFE_H3A_HVF_THR[15:0] HTHR1 and VISS_RAWFE_H3A_HVF_THR[31:16] HTHR2, respectively.
- IIRout_n is the IIRout_1 and IIRout_2 outputs, respectively.
- HFV_count_n and HFV_sq_n are not sent to the DMA interface if VF is disabled.
- RNDADD and RNDSHIFT depend on whether input pixels are 8-bit or 10-bit, and achieves rounding. This is automatically performed by the module.

Copyright © 2026 Texas Instruments Incorporated

- If VF is enabled, only the green color channel values are output to the DMA interface.
- In sum mode:
  - HFV_n = HFV_n_sum
  - HFV_sq_n = HFV_sq_n_sum
- In peak mode:
  - HFV_n = HFV_n_peak
  - HFV_sq_n = HFV_sq_n_peak

7.7.3.3.4.7.5.4 RAWFE H3A VFV Calculator

The VFV calculator takes the unsigned extracted data through two FIR filters, each with a set of five coefficients (VCOEF1_x, where x = 0 to 4, in the VISS_RAWFE_H3A_VFV_CFG1 and VISS_RAWFE_H3A_VFV_CFG2 registers for FIR 1, and VCOEF2_x, where x = 0 to 4, in the VISS_RAWFE_H3A_VFV_CFG3 and VISS_RAWFE_H3A_VFV_CFG4 registers). Each coefficient is 8 bits wide with 4 bits of decimal (S8Q4). The filter outcome is downshifted by 4 bits and takes an absolute value to produce a 16-bit unsigned value. This is then sent to threshold VISS_RAWFE_H3A_VFV_CFG2[31:16] VTHR1 for FIR 1, and VISS_RAWFE_H3A_VFV_CFG4[31:16] VTHR2 for FIR 2, and square logic to produce VFV_n and VFV_sq_n.

7.7.3.3.4.7.5.5 RAWFE H3A VFV Accumulator

The VFV accumulator takes the output of the vertical FIR filters and accumulates values for each paxel. The size and number of paxels is configurable by registers.

Table 7-98 lists the bitfields that configure the size and number of paxels.

The following equations detail the calculation for:

- VFV_n
- VFV_count_n
- VFV_sq_n

n = 1 or 2 for FIR1 and FIR2, respectively.

For each paxel, these six values are available for each R, G, and B component.

```
FIR_coef_n = [VCOEFn_0, VCOEFn_1, VCOEFn_2, VCOEFn_3, VCOEFn_4]; /* coefficient values in S8.4
format */
aFIRout_n = (ABS(inner_product(extracted_G, FIR_coef_n)) + 8) >> 4;
if (aFIRout_n >= threshold_n)
{
 VFV_n = aFIRout_n - threshold_n
 VFV_count_n++;
}
else VFV_n = 0;
VFV_sq_n = (VFV_n * VFV_n + RNDADD) >> RNDSHIFT;
```

- threshold_n is VISS_RAWFE_H3A_VFV_CFG2[31:16] VTHR1 and VISS_RAWFE_H3A_VFV_CFG4[31:16] VTHR2, respectively.
- FIRout_n is the FIRout_1 and FIRout_2 outputs, respectively.
- RNDADD and RNDSHIFT depend on whether the input pixels are 8-bit or 10-bit, and achieves rounding. This is automatically performed by the module.

7.7.3.3.4.7.6 RAWFE H3A AE/AWB Engine

The AE/AWB engine starts by dividing the frames into windows, and then subsamples each window into 2 × 2 blocks. For each subsampled 2 × 2 block, each pixel is accumulated. Also, each pixel is compared to a limit set in a register. If any pixels in a 2 × 2 block are greater than or equal to the limit, the block is not counted in the unsaturated block counter. Pixels greater than the limit are replaced by the limit, and the value of the pixel is accumulated.

The AE/AWB module has three output format modes, which are set through the VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT bit field:

- Sum of square mode: VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x0

Copyright © 2026 Texas Instruments Incorporated

- Min/max mode: VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x1
- Sum-only mode: VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x2

7.7.3.3.4.7.6.1 RAWFE H3A Subsampler

The subsampler partitions the frame into windows using the size, count, and starting location parameters shown on the left in Figure 7-49. Each window is further sampled down to a set of 2 × 2 blocks. The horizontal and vertical distances between the start of blocks within a window is programmable using the parameters shown on the right in Figure 7-49.

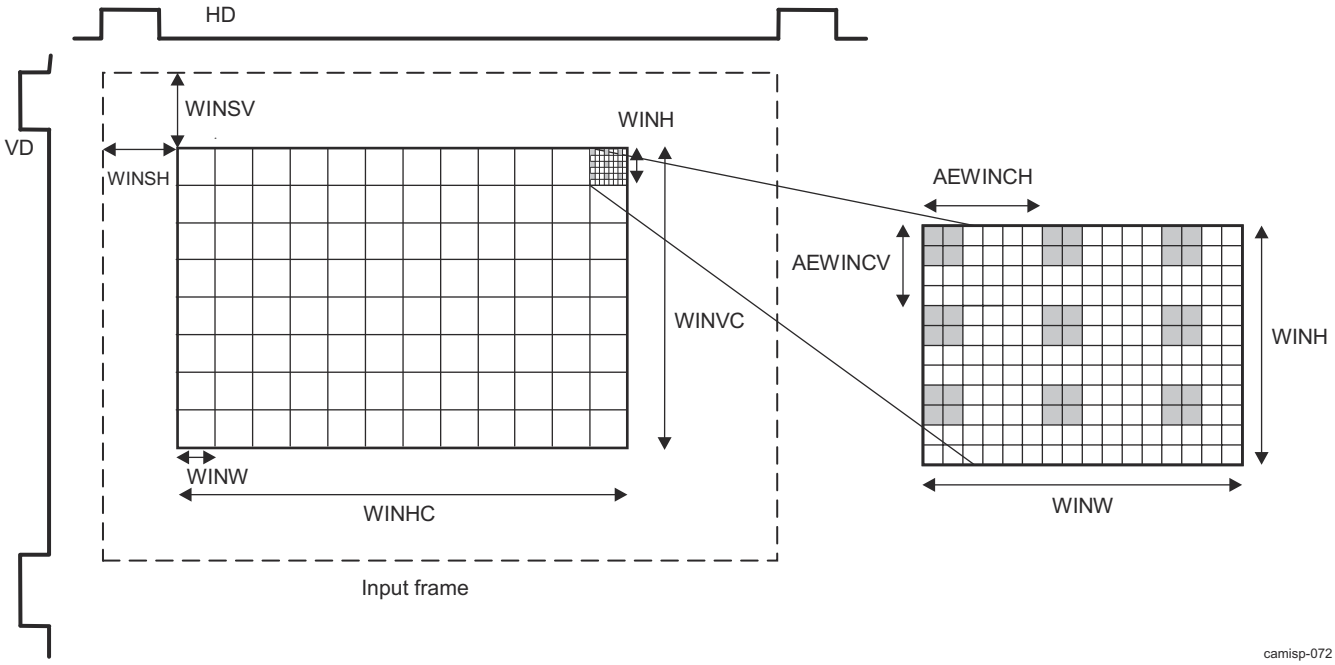


Figure 7-49. RAWFE H3A AE/AWB Window Configurations

Table 7-99 lists the bit fields that configure the window and block sizes, counts, and starting positions.

Table 7-99. RAWFE H3A AE/AWB Window Register Field Descriptions

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AEWWIN1[20:13] WINW	7	Window width (in pixels)
VISS_RAWFE_H3A_AEWWIN1[31:24] WINH	7	Window height (in lines)
VISS_RAWFE_H3A_AEWWIN1[5:0] WINHC	6	Window count for horizontal direction
VISS_RAWFE_H3A_AEWWIN1[12:6] WINVC	7	Window count for vertical direction
VISS_RAWFE_H3A_AEWINSTART[11:0] WINSH	12	Window start position H
VISS_RAWFE_H3A_AEWINSTART[27:16] WINSV	12	Window start position V
VISS_RAWFE_H3A_AEWSUBWIN[3:0] AEWINCH	4	Horizontal distance between subsamples
VISS_RAWFE_H3A_AEWSUBWIN[11:8] AEWINCV	4	Vertical distance between subsamples

7.7.3.3.4.7.6.2 RAWFE H3A Additional Black Row of AE/AWB Windows

In addition to the 128 rows of windows, the AE/AWB module provides support for an additional row of windows for black data. This data may be useful in determining the DC offset noise of the rest of the data. The black row of windows can be before or after the regular rows of windows. The vertical start line for the black row of windows is specified in the VISS_RAWFE_H3A_AEWINBLK[27:16] WINSV bit field, and the height is specified in the VISS_RAWFE_H3A_AEWINBLK[6:0] WINH bit field. The horizontal starting pixel and horizontal width of the black row of windows are the same as for the regular rows of windows.

Figure 7-50 shows a black row of windows before rows of windows.

Copyright © 2026 Texas Instruments Incorporated

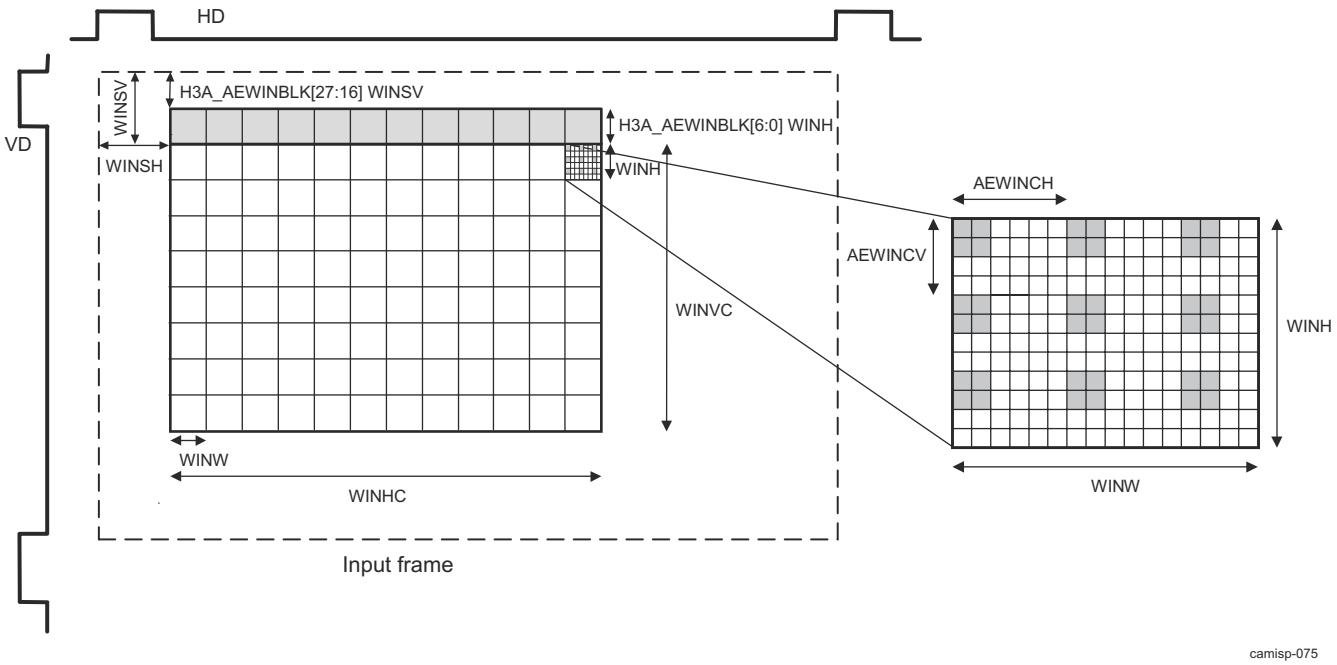


Figure 7-50. RAWFE H3A Black Row of Windows Before Regular Rows of Windows

Table 7-100 lists the bit fields that configure the window and block sizes, counts, and starting positions.

Table 7-100. RAWFE H3A AE/AWB Window With Additional Black Row Register Field Descriptions

Bit Field	Bit Width	Description
VISS_RAWFE_H3A_AEWWIN1[20:13] WINW	7	Window width (in pixels)
VISS_RAWFE_H3A_AEWWIN1[31:24] WINH	7	Window height (in lines)
VISS_RAWFE_H3A_AEWWIN1[5:0] WINHC	6	Window count for horizontal direction
VISS_RAWFE_H3A_AEWWIN1[12:6] WINVC	7	Window count for vertical direction
VISS_RAWFE_H3A_AEWINSTART[11:0] WINSH	12	Window start position H
VISS_RAWFE_H3A_AEWINSTART[27:16] WINSV	12	Window start position V
VISS_RAWFE_H3A_AEWSUBWIN[3:0] AEWINCH	4	Horizontal distance between subsamples
VISS_RAWFE_H3A_AEWSUBWIN[11:8] AEWINCV	4	Vertical distance between subsamples
VISS_RAWFE_H3A_AEWINBLK[27:16] WINSV	12	Window start position H for single black line
VISS_RAWFE_H3A_AEWINBLK[6:0] WINH	7	Window height (in lines) for single black line

7.7.3.3.4.7.6.3 RAWFE H3A Saturation Check

The saturation check module compares the data from the subsampler to the value programmed in the VISS_RAWFE_H3A_PCR[31:22] AVE2LMT bit field. This value is the maximum clipping value. If all 4 pixels in the 2 × 2 block are less than the AVE2LMT value, the value of the unsaturated block counter is incremented. There is one unsaturated block counter per window. The unsaturated block counters are later written to memory.

7.7.3.3.4.7.6.4 RAWFE H3A AE/AWB Accumulators

The output from the saturation check module and the subsampler module are separately accumulated for each pixel in every 2 × 2 pixel block for each window. Therefore, there are eight accumulators per window (one accumulator for each pixel in a 2 × 2 pixel block, times two sets of accumulators: clipped/saturated data and presaturated data). Each of the 4 pixels in the 2 × 2 pixel grid is associated with a color R, Gr, B, Gb); however, the output of these accumulators is referenced by position in the grid, not color.

Copyright © 2026 Texas Instruments Incorporated

The accumulators are 16 bits wide, and the accumulated data is 10 bits wide. Therefore, when a window contains more than 64 pixels of the same color, an overflow risk exists. This risk can be reduced by enabling the A-Law conversion in the preprocessing stage. See Section 7.7.3.3.4.7.4 for details.

The AE/AWB module has a shift value for the accumulation of pixel values that is set in the VISS_RAWFE_H3A_AEWCFG[3:0] SUMSHFT bit field.

7.7.3.3.4.7.7 RAWFE H3A DMA Interface

The DMA interface module takes the data from the AF engine and AE/AWB engine and builds packets to be sent to the memory through the BL module.

The data interface has separate start pointers for the AF and AE/AWB engines.

- The starting address for the AF engine is the VISS_RAWFE_H3A_AFBUFST[31:5] AFBUFST bit field.
- The starting address for the AE/AWB engine is the VISS_RAWFE_H3A_AFBUFST[31:5] AEWBUFST bit field.

The DMA interface module continuously loops through this data as it builds the packets. To optimize the transfer sizes, the DMA interface sends out an AF or AE transfer for each row of paxels or windows. This requires that each horizontal row of paxels or windows starts and ends on a 32-byte boundary. If a horizontal row of paxels or windows ends on a non-32 byte boundary, the hardware packs zeroes. The counts for the AEW that occur every eight windows is sent in the row with the eighth consecutive window.

Table 7-101 shows the packet formats for AF with vertical AF enabled.

Table 7-101. RAWFE H3A AF Packet Format With Vertical AF Enabled

Buffer Start Address (Byte Address) VISS_RAWFE_H3A_AFBUFST	31 16 15 0
	Sum of pixel values used in HFV (Paxel 0)
	HFV_1 (peak or sum) (Paxel 0)
	HFV_sq_1 (peak or sum) (Paxel 0)
	HFV_count_1 (Paxel 0)
	HFV_2 (peak or sum) (Paxel 0)
	HFV_sq_2 (peak or sum) (Paxel 0)
	HFV_count_2 (Paxel 0)
	Zeroes (Paxel 0)
	VFV_1 (Paxel 0)
	VFV_sq_1 (Paxel 0)
	VFV_count_1 (Paxel 0)
	Zeroes (Paxel 0)
	VFV_2 (Paxel 0)
	VFV_sq_2 (Paxel 0)
	VFV_count_2 (Paxel 0)
	Zeroes (Paxel 0)
	Sum of pixel values used in HFV (Paxel 1)
	HFV_1 (peak or sum) (Paxel 1)
	HFV_sq_1 (peak or sum) (Paxel 1)
	HFV_count_1 (Paxel 1)
	...

Copyright © 2026 Texas Instruments Incorporated

Table 7-102 lists the packet formats for AE/AWB for sum of square mode (VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x0) .

Table 7-102. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode

	31 16	15 0	
Buffer address (byte address) VISS_RAWFE_H3A_AEWBUFST	Subsample Accum[1]	Subsample Accum[0]	Window 0 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 32 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 1 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 64 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 2 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 96 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 3 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		

Copyright © 2026 Texas Instruments Incorporated

Table 7-102. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode (continued)

	31 16	15 0	
VISS_RAWFE_H3A_AEWBUFST + 128 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 4 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 160 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 5 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 192 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 6 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 224 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 7 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Sum of squares[0]		
	Sum of squares[1]		
	Sum of squares[2]		
	Sum of squares[3]		
VISS_RAWFE_H3A_AEWBUFST + 256 bytes	Unsaturated count, win 1	Unsaturated count, win 0	Unsaturated block count for the above 8 windows
	Unsaturated count, win 3	Unsaturated count, win 2	

Copyright © 2026 Texas Instruments Incorporated

Table 7-102. RAWFE H3A AE/AWB Packet Format for Sum of Square Mode (continued)

31 16	15 0
Unsaturated count, win 5	Unsaturated count, win 4
Unsaturated count, win 7	Unsaturated count, win 6
Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including the black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.	

Table 7-103 lists the packet formats for AE/AWB in minimum-maximum mode (VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x1).

Table 7-103. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode

	31 16	15 0	
Buffer address (byte address) VISS_RAWFE_H3A_AEWBUFST	Subsample Accum[1]	Subsample Accum[0]	Window 0 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 32 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 1 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 64 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 2 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	

Copyright © 2026 Texas Instruments Incorporated

Table 7-103. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode (continued)

	31 16	15 0	
VISS_RAWFE_H3A_AEWBUFST + 96 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 3 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 128 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 4 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 160 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 5 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 192 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 6 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 224 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 7 data
	Subsample Accum[3]	Subsample Accum[2]	

Copyright © 2026 Texas Instruments Incorporated

Table 7-103. RAWFE H3A AE/AWB Packet Format for Minimum-Maximum Mode (continued)

	31 16	15 0	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
	Minimum[1]	Minimum[0]	
	Minimum[3]	Minimum[2]	
	Maximum[1]	Maximum[0]	
	Maximum[3]	Maximum[2]	
VISS_RAWFE_H3A_AEWBUFST + 256 bytes	Unsaturated count, win 1	Unsaturated count, win 0	Unsaturated block count for the above 8 windows
	Unsaturated count, win 3	Unsaturated count, win 2	
	Unsaturated count, win 5	Unsaturated count, win 4	
	Unsaturated count, win 7	Unsaturated count, win 6	
	Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including the black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.		

Table 7-104 lists the packet formats for AE/AWB in sum-only mode (VISS_RAWFE_H3A_AEWCFG[9:8] AEFMT = 0x2).

Table 7-104. RAWFE H3A AE/AWB Packet Format for Sum-Only Mode

	31 16	15 0	
Buffer address (byte address) VISS_RAWFE_H3A_AEWBUFST	Subsample Accum[1]	Subsample Accum[0]	Window 0 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 32 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 1 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 64 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 2 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	

Copyright © 2026 Texas Instruments Incorporated

Table 7-104. RAWFE H3A AE/AWB Packet Format for Sum-Only Mode (continued)

	31 16	15 0	
VISS_RAWFE_H3A_AEWBUFST + 96 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 3 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 128 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 4 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 160 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 5 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 192 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 6 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 224 bytes	Subsample Accum[1]	Subsample Accum[0]	Window 7 data
	Subsample Accum[3]	Subsample Accum[2]	
	Saturator Accum[1]	Saturator Accum[0]	
	Saturator Accum[3]	Saturator Accum[2]	
VISS_RAWFE_H3A_AEWBUFST + 256 bytes	Unsaturated count, win 1	Unsaturated count, win 0	Unsaturated block count for the above 8 windows
	Unsaturated count, win 3	Unsaturated count, win 2	
	Unsaturated count, win 5	Unsaturated count, win 4	
	Unsaturated count, win 7	Unsaturated count, win 6	
	Data for next eight windows, and so on. If the total number of windows is not a multiple of 8, the unsaturated counters are written immediately following the last window data. For example, if the total number of windows (including black row) is 43, the first 40 windows are written out as per the 272-byte boundary above. Then the remaining three windows are written at +0, +32, and +64 bytes. The counts are written out at +96 instead of +256-byte boundary.		

Copyright © 2026 Texas Instruments Incorporated

7.7.3.3.4.7.8 RAWFE H3A Events and Status Checking

The AF and AEW engines generate an interrupt event at the end of processing each frame. However, these two interrupts are internally tied together so that only one H3A interrupt signal is generated. If the AF engine and AEW engine do not process the same frame concurrently, this should not be an issue. However, if they do run concurrently, one of two outcomes may occur:

- The H3A interrupt may seem to trigger only once for each frame. This can happen when the processing for the AF and AEW engines finishes at or near the same time. The interrupt service routine (ISR) does not have enough time to clear the interrupt flag for the first interrupt before the second interrupt occurs.
- The H3A interrupt may trigger twice for each frame. This can happen when the AF engine or the AEW engine finishes processing the frame much earlier than the other engine. In this case, the ISR does have enough time to clear the interrupt flag for the first interrupt by the time the second interrupt occurs.

The outcome depends on the difference in location of the last paxel/window in the frame (determines when processing is finished), the frequency of the relative clocks in the system, the occurrence and triggering of other interrupts in the system, and the latencies of the context switching and ISR execution.

The VISS_RAWFE_H3A_PCR[15] BUSYAF and/or VISS_RAWFE_H3A_PCR[18] BUSYAEAWB status bits are set when the start of frame occurs (if the VISS_RAWFE_H3A_PCR[0] AF_EN and/or VISS_RAWFE_H3A_PCR[16] AEW_EN bits are 1 at that time). They are automatically reset to 0 at the end of processing a frame. The VISS_RAWFE_H3A_PCR[15] BUSYAF and/or VISS_RAWFE_H3A_PCR[18] BUSYAEAWB status bits may be polled to determine the end of frame status.

7.7.3.3.4.7.9 RAWFE H3A Interface Mux

The H3A logic also implements an LUT or Shift /clip block. The LUT logic is same as the decompanding LUT in the previous section(s) with some minor modifications. The input bit width to the LUT cannot exceed 16 and the LUT entry size is only 10 bits, since the H3A logic works on 10 bit data.

The logic also implements a shift and clipU10 functionality for converting data from up to 24 bits down to 10. The shift parameter has to be programmed to implement the desired shift value. The shift register programming must ensure that the output of the shift is not greater than 16bits when the LUT is enabled and it should not be greater than 10 bits when the LUT is disabled. In addition to the outputs from the 3 PWL/DC and the LSC output, H3A interface mux can also be fed by PCID Bayer output.

7.7.3.3.4.7.10 RAWFE H3A interface to LSE

The H3A interface to LSE is VBUSP. It uses channel id to indicate data or control information. Please refer to LSE functional spec on control information mapping to channel id bits.

7.7.3.3.4.7.11 RAWFE H3A Erratas

The following hardware restrictions apply and need to be taken care of by software:

1. When AF is enabled software can not use the first or last pixels in the window. Start at position 2 end at line -2.
2. When AEW is enabled software can not use the first pixel in a window. Start at position 2.
3. (PAXW+1) % (AFINCH+1) != 1 when VF enabled (PAXW and AFINCH are raw MMR parameters)

7.7.3.3.5 RAWFE Programmer's Guide

7.7.3.3.5.1 RAWFE Core programming details

There are multiple use cases that can be enabled with this module, however broadly the use case are broken in to two categories, one for Companded Data sensors and the other for Separate Exposures sensors. The following fields should be programmed correctly for the IP to function

- PWL Core
- Set slopes and thresholds as per sensor settings only for Companded sensors
- Decompanding LUT

Copyright © 2026 Texas Instruments Incorporated

- This LUT is used for global tone map of decompounded data from 24 bits to 16 bits. (This is typically not used for separate exposures sensor)

- Merge Logic
- Program the merge block based on the exposure rations of the incoming data.
- This block is typically not used for companed sensors.
- LUT after merge should be programmed to tone map from 20 bits down to 16 (Separate exposure sensors only)

- DPC/ LSC
- Set as per tuning parameters for both companded/separate exposures sensors

- H3A LUT and settings
- Use LUT if H3A input comes after LSC. Program LUT to apply inverse global curve.
- Use shift if data is coming from exposures directly.

7.7.3.3.5.2 RAWFE Initialization Sequence

See Section 7.7.2, *VPAC Subsystem* for details.

7.7.3.3.5.3 RAWFE Real-time Operating Requirements

See Section 7.7.2, *VPAC Subsystem* for details.

7.7.3.4 VISS Pattern Conversion and IR Demosaicing (PCID) Module

7.7.3.4.1 Overview and Feature List

The PCID produces both Bayer and IR data concurrently from input RGBIR image. Bayer channel data is fed to rest of the VISS pipe line and IR channel output is directly written to memory using one of LSE channels. VPORT Delay SYNC_FIFO supports programmable horizontal delay to match the delay with FCP output. Figure 7-51 captures integration of PCID into VISS.

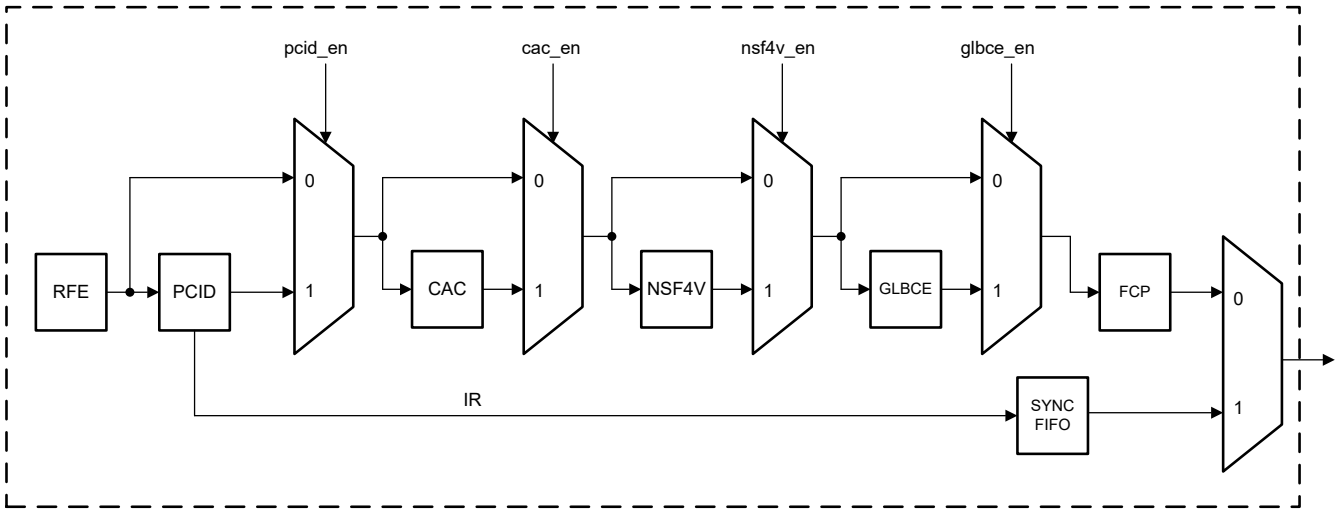


Figure 7-51. PCID Integration in VISS

7.7.3.4.1.1 PCID Features Supported

- Performs pattern conversion on upto 16bits per pixel 4x4 RGBIR sensor data
  - Generates concurrent Bayer and full resolution IR output
  - Supports choice to interpolate the missing R/B pixel values at IR location to create Bayer output
- Can decontaminate Generated Bayer pattern of the IR signal
  - Supports programmable IR subtraction factor value in range [0, 2) for individual color channels

Copyright © 2026 Texas Instruments Incorporated

  - Supports spatial smoothening of the pixel level IR subtraction factors
- Supports 16-bit to upto 16-bit generic Gamma LUT on IR output
- Supports Constant Hue or Color difference method for pattern conversion and IR interpolation

7.7.3.4.2 Functional Operation

The PCID block integrates pattern conversion (re-mosaicing), IR up-sampling and IR subtraction functionality into a single block as illustrated in Figure 7-28. The Pattern conversion and IR up-sampling block performs the task of interpolating missing pixel values, where it uses a two-stage process for re-mosaicing and IR demosaicing.

In the first step, missing green channel information in the input 4x4 RGBIR CFA (at the R, B and IR locations) is interpolated to obtain the green plane to full resolution. In the second step the missing R, B and IR values are interpolated to obtain the re-mosaiced 2x2 Bayer pattern and full resolution IR output. When interpolating the missing pixel values up-sampled green plane acts as a guide for interpolation along the local features such as edges and helps avoid the smoothening. The hardware allows user to choose between constant hue or constant color interpolation schemes along with configurable parameter values that allow precedence for preservation of particular image features during the interpolation process.

The key concerns when subtracting IR signal from the re-mosaiced Bayer pattern are that the level of IR contamination is function of DBF design, the IR contamination of the R, G and B pixels is variable, and subtraction of the IR signal from clipped/saturated pixels can lead to introduction of artifacts. The PCID module can calculates the subtraction factors for every pixel location and used by IR subtraction block for decontamination based on user defined parameters.

7.7.3.4.3 Calculation of pixel level IR subtraction factors

The PCID block enables subtraction of scaled IR signal value from generated 2x2 Bayer CFA pattern as

$$R' = R - Xr \times IR \quad (8)$$

$$G' = G - Xg \times IR \quad (9)$$

$$B' = B - Xb \times IR \quad (10)$$

Where IR subtraction scales {Xr, Xg or Xb } used for a pixels are calculated in a two-step process involving the calculation of pixel level IR subtraction factors and optional spatial smoothening of the calculated IR subtraction factors to suppress noise and obtain smooth transitions.

To avoid subtraction of the IR signal from clipped/saturated pixels and in-turn introduction of color artifacts in the created Bayer pattern, HW allows user to define a cut-off value for R, G and B pixels over which the IR subtraction is disabled. Further to have smoother transition of the IR decontamination scales immediately below the cut-off value HW allows programming of the transition range over which the IR subtraction range gradually increases from no IR subtraction at the cut-off threshold to maximum programmed IR subtraction factor. To enable calculation of IR subtraction factor for Bayer output value at at IR locations in the input CFA (where no R/G/B value is natively available) HW uses nearest maximum R/G/B pixel value for the location to calculate the IR subtraction factor based on this value especially if it is above cut-off or just below it within the transition range. For this user can configure the influence of the nearest maximum R/G/B pixel as function of the L1 distance with values in range [0,1]. When the maximum R/G/B pixel values in a neighborhood are in ranges close to the cut-off value the HW allows user to spatially smoothen the calculated IR subtraction factors in 5x5 region before performing IR subtraction.

Copyright © 2026 Texas Instruments Incorporated

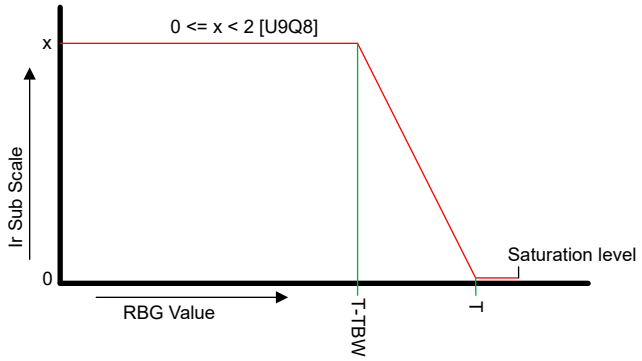


Figure 7-52. Calculated IR Subtraction Scale

Figure 7-52 shows the calculated IR subtraction scale values with regard to the maximum neighborhood value as a function of user-defined cut-off threshold (T)), transition range TBW and the desired maximum IR subtraction factor for the output Bayer channel.

The PCID block when outputting the interpolated IR image allows user to apply a tone curve on the output to compress the bitdepth from 16bit to 16/12/10/8 bits. The Tone curve can be defined using an IR remap LUT defined using 609 16bit knee points stored in32bit containers.

7.7.3.5 VISS Chromatic Aberration Correction (CAC) Module

7.7.3.5.1 Overview and Feature List

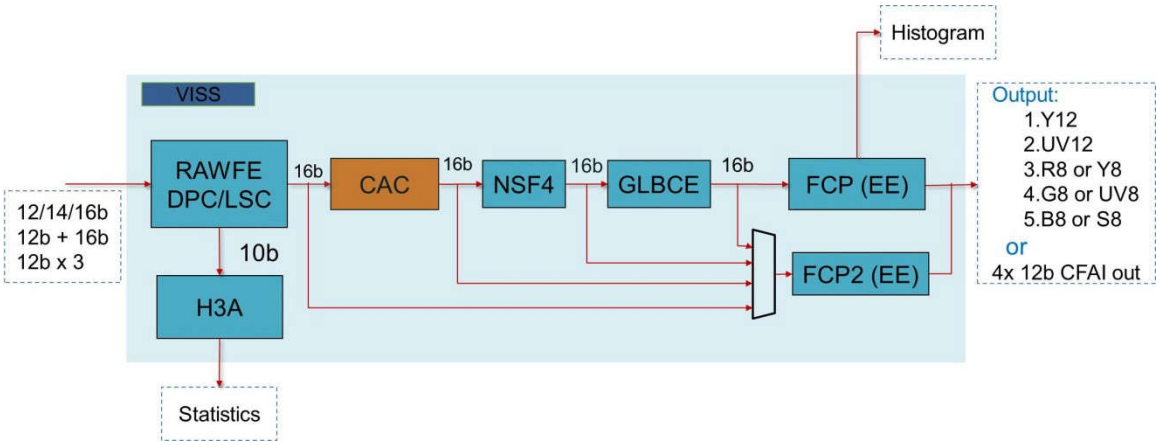
7.7.3.5.1.1 Features Supported

- Chromatic Aberration Correction (CAC) for Red and Green color in Bayer color formats
- Generic mesh based model for CAC
- LUT to store programmable differential correction with 1/8th pixel precision
- Programmable Block size with each block having 2 LUT entries covering entire frame
  - Intermediate correction values are calculated using bilinear interpolation
- Bi-cubic pixel interpolation on corrected back mapped pixel
- Supports Horizontal Correction in the range of -8.0 to +8.0 and Vertical Correction in the range of -4.0 to +4.0
- Performance: up to 1 Pix/cycle (excluding blanking)
- Streaming based processing architecture targeted to integrate with VISS
- Support 16-bit pixel
- Logic support frame dimension of 8Kx8K. Line memory support is limited to 4K line width

7.7.3.5.2 Functional Description

Copyright © 2026 Texas Instruments Incorporated

7.7.3.5.2.1 CAC Integration in VISS



The VISS consist of following hardware accelerators:

- RFE (RAW-FE): The RAW-FRONT HW block does RAW pixel (for example, Bayer, RCCC, RGBW, etc.) processing on captured image data from sensor and pass-on to CAC, NSF4V, GLBCE block, and then to Flexible ColorProc HW block for demosaicing and color conversion.
- CAC (Chromatic Aberration Correction): Captured in this section.
- NSF4V: Spatial Noise filter supporting generic 2x2 pixel format with 16-bit pixel size. NSF4V IP can be bypassed in the applications where visual enhancement is not desired.
- GLBCE: Used for dynamic range control within image for visual quality. If contrast enhancement on input image is required for visual quality RFE/CAC/NSF4V output is processed by GLBCE block. Otherwise GLBCE is bypassed to FCP.
- FCP (FLEXible COLOR PROCessing): The FLEX-COLOR-PROC HW receives data from GLBCE and does demosaincing and color conversion. The output of COLOR-PROC is sent to VPAC shared memory to be written into DDR for rest of vision processing by programmable processors (for example, DSP or Arm) or Vision HW Blocks (for example, VPAC or DMPAC).

7.7.3.5.2.2 Introduction

Chromatic aberration, also known as “color fringing” or “purple fringing”, is a common optical problem that occurs in low quality lens when a lens is either unable to bring all wavelengths of color to the same focal plane (*Longitudinal Chromatic aberration*), and/or when wavelengths of color are focused at different positions in the focal plane (*Lateral Chromatic aberration)* .

Figure 7-53 shows one example of Chromatic Aberration referred as *Lateral Chromatic Aberration* (LCA). In this case, all colors are in focus in the same plane, but the foci are not placed along the optical axis. CAC Hardware accelerator performs *Lateral Chromatic aberration* correction is as part of VISS Pipe line.

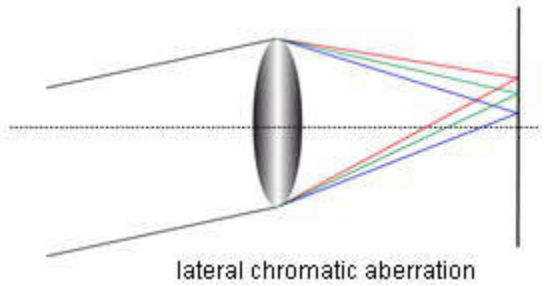


Figure 7-53. Lateral Chromatic Aberration

Copyright © 2026 Texas Instruments Incorporated

7.7.3.5.2.3 Functional Operation

CAC correction requires remapping locations of Red and Blue pixels to align with Green pixel. Green pixel is taken as reference and assumed no correction is needed on Green pixel. CAC HWA is implemented using line based back mapping architecture. Correction involves back-mapping each output pixel to pixel position in the source image, and thus the corrected image is fully populated. As the back mapped pixel locations mostly fall onto fractional coordinates, correction involves bi-cubic interpolation among the nearest available pixels.

As shown in Figure 7-54, CAC consists of Timing control module, 8KB LUT with mesh correction offset table, Mesh LUT bi-linear interpolation block, Pixel interface module, pixel bi-cubic interpolation block and 10 pixel line memories.

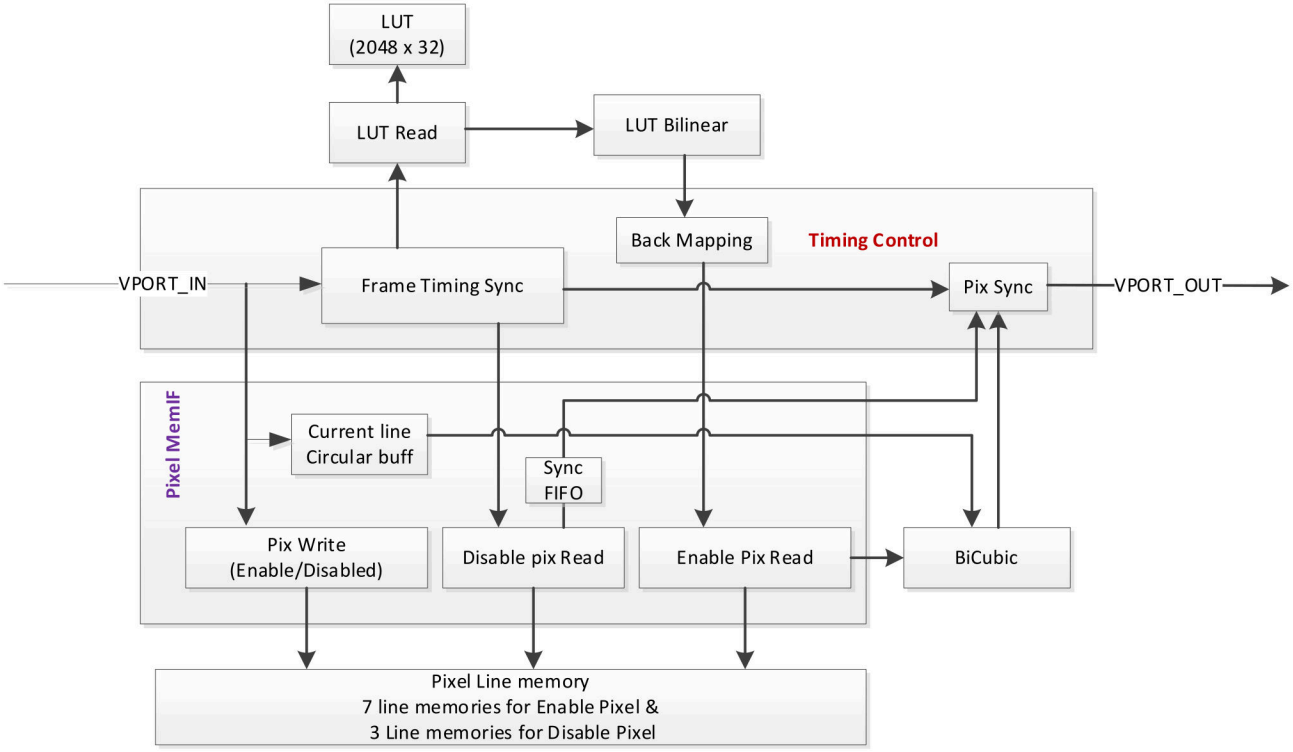


Figure 7-54. CAC Functional Block Diagram

The Timing Control module handles all the timing synchronization aspects of CAC. It generates the Output frame co-ordinates, initiates the Mesh LUT operation on receiving 7th input valid line. It also handles timing synchronization between all the blocks.

7.7.3.5.2.3.1 CAC Back Mapping

Offset table defines a (□x, □y) vector for a regular grid of output points. Each grid will define 2 offset vectors one each for EVEN and ODD line(s), defining how far output pixel needs to be moved in either direction. Final pixel location in input source image is used to compute the output pixel. Offset table can have precise definition and can capture rapidly changing offset tables. However it will require a large amount of memory. Since most offset tables are not expected to change rapidly in a small spatial region, CAC uses a subsampled offset table. Grid is down sampled with down-sampling factor in the range of 8 to 128 with increments of 4 in both horizontal and vertical directions. Subsampling factor is set in the register; BLOCKSZ.SIZE This enables reducing the amount of memory required to describe the offset vectors, but requires hardware to interpolate the missing offset vectors. CAC supports bilinear interpolation to interpolate missing offset vectors.

The mapping procedure is described by the following series of equations. Given an output pixel at location (x_0,y_0), we compute the input image pixel location (x_1,y_1).

Copyright © 2026 Texas Instruments Incorporated

$$ind_x = \frac{X_o}{Blk_{size}};\ f_x = MOD\left(X_o, Blk_{size}\right) \quad (11)$$

$$ind_y = \frac{Y_o}{Blk_{size}};\ f_y = MOD\left(Y_o, Blk_{size}\right) \quad (12)$$

Clip new index to Offset grid boundaries.

$$i_x = CLIP(ind_x, Grid_{HCnt}) \quad (13)$$

$$i_y = CLIP\left(ind_y, Grid_{VCnt}\right) \quad (14)$$

$$\Delta x_p = \frac{\left(\left(blk_{size} - f_y\right) \cdot T_{x0} + f_y \cdot T_{x1} + \frac{square(blk_{size})}{2}\right)}{square(block_{size})} \quad (15)$$

$$\Delta y_p = \frac{\left(\left(blk_{size} - f_y\right) \cdot T_{y0} + f_y \cdot T_{y1} + \frac{square(blk_{size})}{2}\right)}{square(block_{size})} \quad (16)$$

Clip to Horizontally to -8.0 to +8.0 pixel range (-64 to +64 with Q3 precision).

Clip to Vertically to -4.0 to +4.0 pixel range (-32 to +32 with Q3 precision).

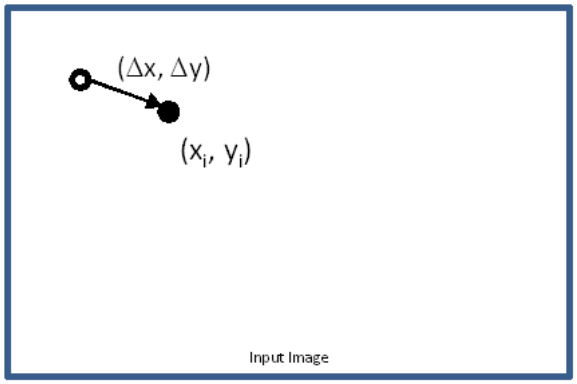


Figure 7-55. Back Mapping Procedure Using Offset Table

The input coordinate (x1,y1) refers to final input frame co-ordinates.

7.7.3.5.2.3.1.1 Offset Table Storage Format

The size of the offset table is dependent on the output frame size. The size must be:

$$\text{Table Width} = CEIL\left(\frac{\text{Frame Width}}{Blk_{size}}\right) + 1 \quad (17)$$

$$\text{Table Height} = CEIL\left(\frac{\text{Frame Height}}{Blk_{size}}\right) + 1 \quad (18)$$

The Offset table should be stored with □x and □y offsets interleaved and stored in raster order. Each offset is S8Q3, so a table entry is 32-bits, including both □x and □y for both Even and Odd line. Each line captures offset for one color either Red or Green.

7.7.3.5.2.3.2 Pixel Interpolation

Copyright © 2026 Texas Instruments Incorporated

As the coordinates calculated by the back mapping function are not integer values in most cases, b-cubic interpolation is applied to the back mapped pixels. Interpolation is done on the pixel of same color (either Red or Blue).

Output pixel is interpolated from the 16 pixels in the 4x4 grid around the back mapped location, as shown in Figure 7-56. *Ri* and *Bi* indicate integer co-ordinates of back mapped pixels. Bi-cubic interpolation is used first along the horizontal direction, then the vertical direction.

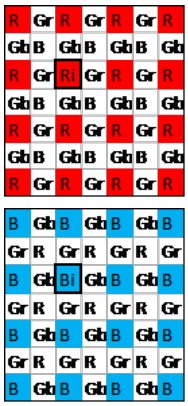


Figure 7-56. Bi-Cubic Interpolation Input Data

7.7.3.5.2.3.3 Bi-cubic Coefficients

The following coefficients are used for bi-cubic operation. Though back-mapped pixel has Q3 precision, there are 16 fractional phases due to Bayer pattern. The filter coefficients should be in S12Q10 format (signed 12-bit with 10 bits of fraction).

Phase	Coefficients (S12Q10)			
0/16	0	1024	0	0
1/16	-28	1014	40	-2
2/16	-49	987	93	-7
3/16	-63	944	158	-15
4/16	-72	888	232	-24
5/16	-76	821	313	-34
6/16	-75	745	399	-45
7/16	-71	663	487	-55
8/16	-64	576	576	-64
9/16	-55	487	663	-71
10/16	-45	399	745	-75
11/16	-34	313	821	-76
12/16	-24	232	888	-72
13/16	-15	158	944	-63
14/16	-7	93	987	-49
15/16	-2	40	1014	-28

Figure 7-57. Bi-cubic Coefficients

7.7.3.5.2.4 Interrupt Conditions

7.7.3.5.2.4.1 Interrupts

Name	Width	Type	Polarity	Category	Clock	Reset	Description
cac_lut_cfg_err	1	pulse	high	func	func_clk	srstn_mod_g_rst_n	CAC cac_lut_cfg_err event

Copyright © 2026 Texas Instruments Incorporated

A ***cac_lut_cfg_err*** interrupt is generated when configuration access happens on LUT memory while CAC is operating on a valid frame. A valid frame is defined as Input Frame Start (VS_IN) to output frame end (VE_OUT).

7.7.3.5.2.4.2 Debug Events

Table 7-105. Debug Events

Event	Description
cac_in_vs_event	Frame Start Input. Optionally can be enabled by setting DBG_CTL.DBG_EN && DBG_CTL.SOF_EN. Status can be observed in DBG_STAT.SOF
cac_in_hs_event	Frame Start Input. Optionally can be enabled by setting DBG_CTL.DBG_EN && DBG_CTL.SOL_EN. Status can be observed in DBG_STAT.SOL.
cac_out_ve_event	Frame End Output. Optionally can be enabled by setting DBG_CTL.DBG_EN && DBG_CTL.EOF_EN. Status can be observed in DBG_STAT.EOF.
cac_out_he_event	Line End Output. Optionally can be enabled by setting DBG_CTL.DBG_EN && DBG_CTL.EOL_EN. Status can be observed in DBG_STAT.EOL.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.6 VISS Spatial Noise Filter (NSF4V)

7.7.3.6.1 NSF4V Introduction

7.7.3.6.1.1 NSF4V Features

For a list of features supported by the NSF4V module, see Section *VPAC Features*.

Some additional NSF4V parameters include:

- Input image format: any 2x2 raw color pattern
- Frame Width: limited to 4096
- Throughput:
  - 1 pixel in/out per clock
- Back Pressure Mechanism:
  - Input clock gating
  - Vert line sync
    - 7 extra lines of clocks plus enough clocks to flush pipeline

7.7.3.6.1.2 NSF4V Not Supported Features

NSF4V does not support the following features:

- Edge Enhancement
- YUV formats
- Module bypass (it is impemented at top subsystem level)

7.7.3.6.2 NSF4V Overview

The advanced noise filter NSF4v decomposes the input image into low-frequency, mid-frequency, and high-frequency bands first and then filter out the noise in frequency domain. NSF4v can adapt its noise filtering strength to local image intensity according to the user programmable noise threshold registers. NSF4 is also able to adjust its strength according to the amount of previously applied lens shading correction gains with a user programmable approximate radial model.

7.7.3.6.2.1 Decomposition Kernel Representation

With combination of just level 1, levels 1+2, and levels 1+2+3, horizontally and vertically, the result feature detection filter kernels are shown in Figure 7-58. Note that all coefficients are powers of 2, and therefore do not require multiplication.

Copyright © 2026 Texas Instruments Incorporated

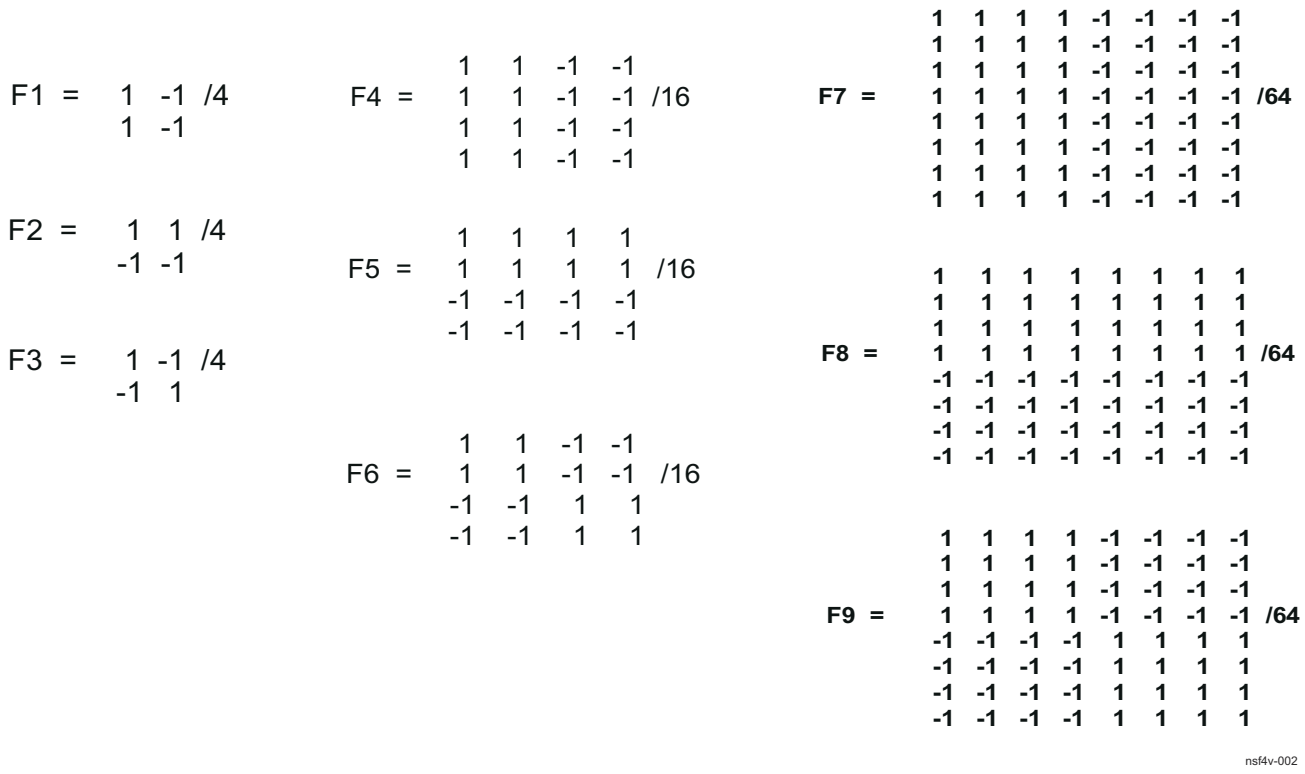


Figure 7-58. NSF4V Kernel Representation for Decomposition Filters

G1...9 are the counter-part reconstruction kernels. Each of G1...9 is a row-wise and column-wise mirror of the counterpart F function.

The kernel representation in Figure 7-59 illustrates a filter alignment.

Copyright © 2026 Texas Instruments Incorporated

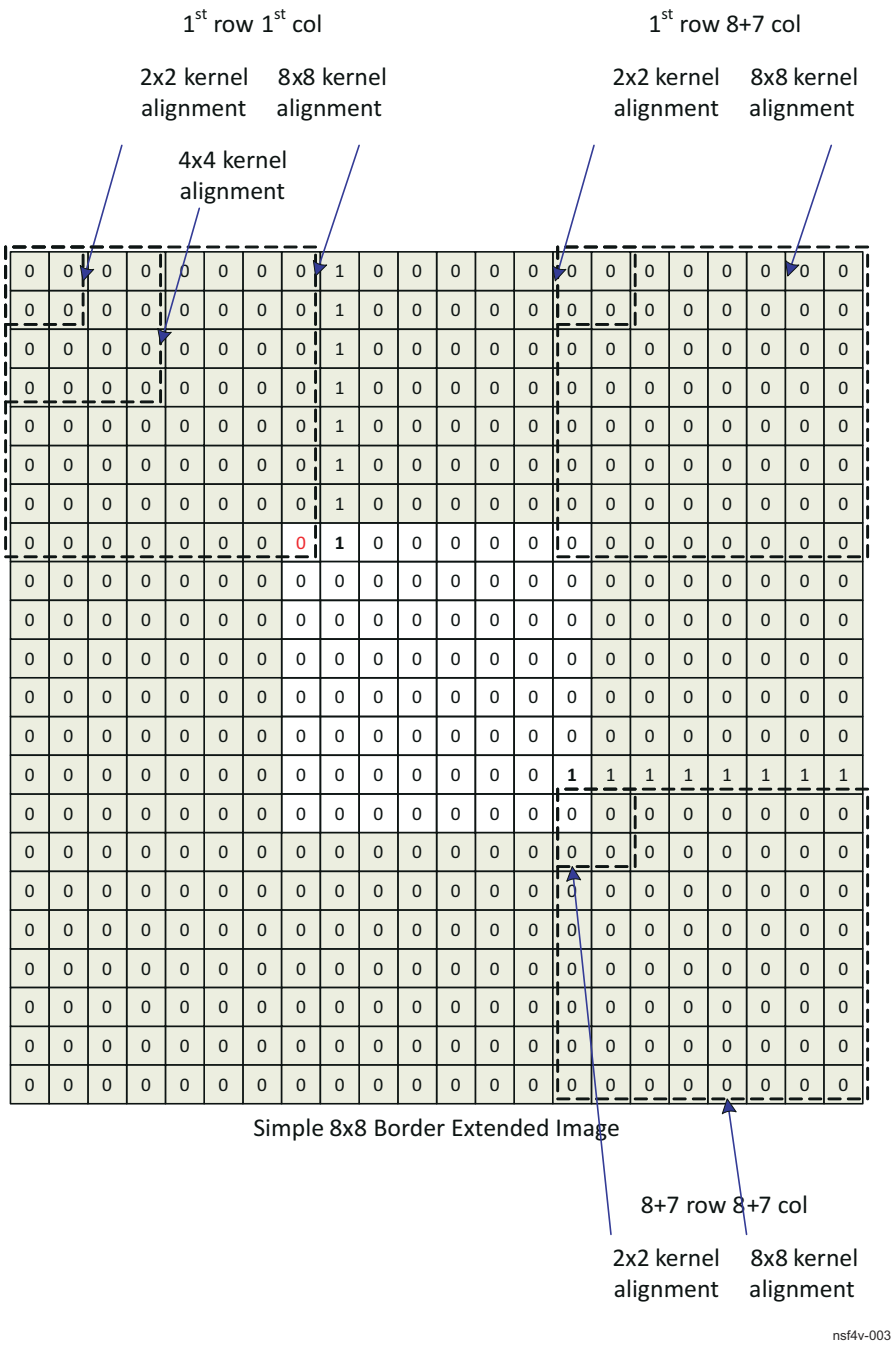


Figure 7-59. NSF4V Kernel Representation Filter Alignment

Figure 7-59 is showing an academic 8x8 pixel image frame which has been border extended on all 4 edges. Superposed onto this border extended frame (shown in dotted lines) is the kernel alignment for all 3 different kernel sizes for each of the three levels:

- Level 1: 2x2 kernel
- Level 2: 4x4 kernel
- Level 3: 8x8 kernel

For each of the 9 different kernels, a matrix of (width + 7) x (height +7) decomposition values result. All of these result matrixes are exactly aligned.

Copyright © 2026 Texas Instruments Incorporated

0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-2**	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	**-1**	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

nsf4v-004

Figure 7-60. NSF4V Example F1 Kernel Result

The kernel alignment is top-left aligned for the first pixel. Which translates in the recursive approach as a X-Y shift. In order to be mathematically aligned in the Recursive approach, the Level 1 output needs to be shifted 6 positions in both the X and Y direction relative to the Level 3 output. Additionally, the Level 2 output needs to be shifted 4 positions in both the X and Y direction relative to the Level 3 output. These shifts are critical such that Level 1, 2, and 3 values are cycle aligned when presented to the thresholding circuit.

One can reach the same conclusion by looking at the figure and thinking about which line delay or which horizontal pipelined pixel position is used in order for the kernels to output their results in the same pipeline cycle.

7.7.3.6.3 NSF4V Lens Shading Correction Compensation

The Lens Shading Correction (LSC) is implemented prior to NSF. This is not an optimal order of operation and therefore an adjustment factor is implemented to compensate for the order of operations. The LSCC (Lens Shading Correction Compensation) first approximates the gain of LSC and calculates both that gain and it's inverse. Within the T_n calculation, the inverse gain is applied to the LL2 average, then the T_n is calculated. Last, the result is multiplied by the gain. In this way, the LSC is approximately reversed temporarily in order that T_n is calculated properly.

LSCC calculates the radius (distance) from the center of the image, then uses a programmable curve to look up the gain value.

While calculating the radius, the center on the frame is programmable. Also a correction gain (kh and kv) separately for X and Y are programmable which allows for elliptical shaped lenses. Because the SQRT is such an expensive function, the resulting radius is used in the SQR form and as such has a very large dynamic range. A programmable shift value effectively selects which portion of this dynamic range is to use for the curve lookup.

The curve lookup is a 16 segment curve. The SQR(Radius) is lookup on the X axis and the resulting gain is supplied on the Y axis.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.6.4 NSF4V Noise Threshold Adaptation to Local Image Intensity

Figure 7-61 shows the local image intensity to noise threshold T_n mapping via the 12-segment interpolated lookup. Twelve sets of (X, Y, S), S = slope, are provided per curve, one set per color plane, with the first X coordinate fixed at zero.

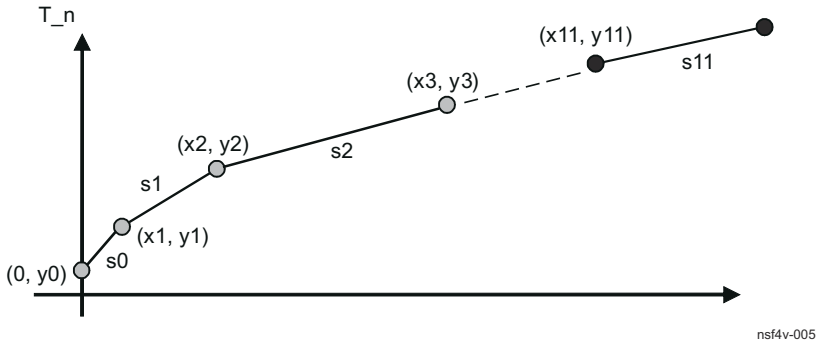


Figure 7-61. NSF4V Local Image Intensity to Noise Threshold Piece-wise Linear Function

The slopes should be implemented with signed value, as some image sensors might require higher noise threshold at the low intensity range than high intensity range.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.7 VISS Global/Local Brightness and Contrast Enhancement (GLBCE) Module

7.7.3.7.1 GLBCE Overview

The Global Local Brightness Contrast Enhancement module performs dynamic range compression or tone mapping of an input image based on a model of the human visual system. It is used to produce most natural images under a wide range of capture conditions, typically by revealing shadow detail which would otherwise be under-exposed in high contrast situations.

7.7.3.7.2 GLBCE Interface

Data flow control of GLBCE module is done by PCLK. GLBCE processes the image cycle by cycle, and output the data after certain cycles of latency.

7.7.3.7.3 GLBCE Core

The GLBCE Core is an iridix® core which performs "dynamic range compression" or "tone mapping" on an input image based on a model of the human visual system. It is used in a camera pipeline to produce the most natural images under a wide range of capture conditions, typically by revealing shadow detail which would otherwise be under-exposed in high contrast situations. The purpose of the transform is to map the image content from an input source such that it remains fully visible on an output display without loss of content. Note that iridix is specifically designed to preserve color, sharpness and boost contrast of the source image.

The iridix core is based on space-variant or pixel-by-pixel processing algorithms which construct a family of transforms based on the analysis of image content. The algorithms inside iridix are adaptive, meaning that a single set of parameters can be set to process any source image or any video sequence under different lighting conditions.

Iridix generates effective tone curves by analyzing the regions surrounding each pixel. The tone curves generated by iridix are based on an asymmetry curve which is then shaped and controlled by a number of parameters that control the gain limitation weighting towards shadows, mid tones and highlights.

7.7.3.7.3.1 GLBCE Core Key Parameters

Table 7-106 shows the key parameters within the iridix core. It is recommended that all parameters be set statically to their recommended values other than the iridix strength parameter, which should be controlled dynamically with scene content. The calculation of iridix strength is explained in Section 7.7.3.7.3.2.

Table 7-106. GLBCE Core Key Parameters

Parameter	Description
iridix strength	This register sets iridix processing strength.
	0x00: Video data will not be processed at all, and will pass to the output unchanged.
	0xFF: Maximum processing strength.
Variance space	This register affects the sensitivity of the transform to different areas of the image, and can be increased in order to emphasize small regions (e.g. faces). If this parameter is set to zero, the sensitivity to small areas is maximized (i.e. the transform becomes more local). When this parameter is set to 0xF, the transform deemphasizes small local details, and the transform becomes more global
Variance intensity	For a high-contrast image with shadows and highlights, a small value will cause iridix to adjust shadows and highlights almost independently. A large variance causes iridix to become progressively more global, meaning that the presence of highlights affects the processing of shadows and vice-versa.
Slope min limit	This register is used to restrict the slope of the tone curve generated by iridix. When this parameter is set to 0x00, the tone curve slope generated by iridix is not limited.
	When this value is set to 0xFF, iridix will not generate a tone curve with a slope less than 3/4. Intermediate values of slope limit are linear interpolation between minimum and maximum values.
	The recommended range of this parameter is 0-128
Slope max limit	This register is used to restrict the slope of the tone-curve generated by iridix. When this parameter is set to 0xFF, the tone curve slope generated by iridix is not limited.
	When this value is set to 0x00, iridix will not generate a tone-curve with slope greater than 16/15 (48°). Intermediate values of slope limit are linear interpolation between minimum and maximum values.
	The recommended range of this parameter is 0-160. For noisy images, lower values should be used.

Copyright © 2026 Texas Instruments Incorporated

Table 7-106. GLBCE Core Key Parameters (continued)

Parameter	Description
Black level	The value stored in this register will be used as the zero level for iridix processing in all data channels. Data below this value will not be processed and remain unchanged.
White level	The value stored in this register will be used as the white level for iridix processing in all data channels. Data above this value will not be processed and remain unchanged.
Asymmetry LUT	This look-up-table is 33 words, each of which is 16-bits. The asymmetry function is used to balance the iridix effect between the dark and bright regions of the image.
Forward perceptual LUT	The iridix core works in a perceptually uniform space and thus when linear data is presented to the core, this look-up-table must be used to map the linear data to the perceptual space so that iridix can perform tis perceptual functions.
Reverse perceptual LUT	The iridix core works in a perceptually uniform space and thus when linear data is required from the output of the core, this look-up-table must be used to map the data from the perceptual space back to linear space for further processing in the pipeline.

7.7.3.7.3.2 GLBCE Iridix Strength Calculation

This section describes how the dynamic parameter, iridix strength, is calculated.

Note

Dynamic range of a digital camera can be described as the ratio of maximum light intensity measurable (at pixel saturation), to minimum light intensity measurable (but above read-out noise). This is a very important concept to consider when computing iridix strength, because the gain applied by iridix is also proportional to the visible boost of noise in the image. The maximum gain applied by iridix should be determined by the dynamic range of the camera system. This is governed by the equation presented in this section.

With iridix strength is zero, all tone curves are linear for all images. As iridix strength is increased, the variation between curves in shadows, mid tones and highlights increases, resulting in increasing compression of global dynamic range.

Iridix strength is calculated as follows:

$$\text{iridix strength} = \text{Max}\left(0, \text{Min}\left(\frac{\text{IR_Gain_Target} - 1}{\text{IR_Gain}_{\text{Max}} - 1}, \text{f}\left(\text{analog gain}\right)\right)\right) \times 255 \quad (19)$$

Where:

- IR_Gain_Target is the desired iridix gain to be applied to the image. This is calculated from the image's histogram on a per-frame basis by the auto exposure algorithm (in RAWFE H3A module). The histogram is analysed to compute an exposure ratio for the desired global histogram shift. This computed exposure ratio is then used to feed a look-up-table which in turn generates the IR_Gain_Target parameter.
- IR_GainMax is the maximum iridix gain. If the fwd_percep_lut is disabled, then the IR_GainMax should be set to 64, and if the fwd_percep_lut is enabled, then the IR_Gainmax should be set to 16.
- f(analog gain) is a function that determines the maximum iridix strength in relation to noise. f(analog gain) is characterised as follows:

Copyright © 2026 Texas Instruments Incorporated

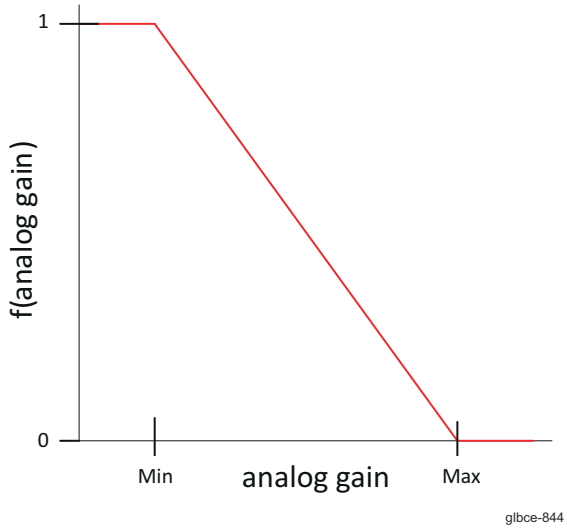


Figure 7-62. GLBCE Iridix f(Analog Gain) Function

f(analog gain) should only be used when the RAWFE is in linear mode. Otherwise, it should be set to 1

7.7.3.7.3.3 GLBCE Iridix Configuration Registers

7.7.3.7.3.3.1 GLBCE Iridix Frame Width

Frame Width is the number of pixels in an active line and is controlled from GLBCE_FRAME_WIDTH[15:0] VAL. Recommended value 2560 (decimal)

7.7.3.7.3.3.2 GLBCE Iridix Frame Height

Frame Height is the number of active lines in a frame and is controlled from GLBCE_FRAME_HEIGHT[15:0] VAL. Recommended value 1920 (decimal)

7.7.3.7.3.3.3 GLBCE Iridix Control 0

This is the main control register for the iridix core is GLBCE_CONTROL0 where GLBCE is enabled/disabled and the type of processing algorithm is selected.

7.7.3.7.3.3.4 GLBCE Iridix Control 1

This register is reserved for debugging purposes. Recommended value 6 (decimal)

7.7.3.7.3.3.5 GLBCE Iridix Strength

GLBCE_STRENGTH_IR[7:0] VAL register sets the processing strength of the iridix core. Recommended value 128 (decimal)

7.7.3.7.3.3.6 GLBCE Iridix Variance

The GLBCE_VARIANCE parameter affects the sensitivity of the transform to different areas of the image, and can be increased in order to emphasize small regions (e.g. faces). GLBCE_VARIANCE[7:4] VARIANCEINTENSITY recommended value 12 (decimal). GLBCE_VARIANCE[3:0] VARIANCESPACE recommended value 7 (decimal).

7.7.3.7.3.3.7 GLBCE Iridix Dither

When the number of color gradations of a display device is small (for example 6 bits per color), pixel dithering can make gradients look smother. Dithering of the least significant bits is applied and then these bits can be later truncated. The three least significant bits (D2, D1, D0) of this GLBCE_DITHER register are responsible for the strength of dithering. There are 4 possible levels of dithering. Recommended value 0 (decimal)

Copyright © 2026 Texas Instruments Incorporated

7.7.3.7.3.3.8 GLBCE Iridix Amplification Limit

The parameters dark amplification limit and bright amplification limit are used to restrict the luminance space in which iridix can adaptively generate tone curves for each pixel. GLBCE_LIMIT_AMPL[7:4] BRIGHTAMPLIFICATIONLIMIT recommended value 0 (decimal). GLBCE_LIMIT_AMPL[3:0] DARKAMPLIFICATIONLIMIT recommended value 0 (decimal).

7.7.3.7.3.3.9 GLBCE Iridix Slope Min and Max

GLBCE_SLOPE_MIN[7:0] SLOPEMINLIMIT register is used to restrict the slope of the tone curve generated by iridix. Recommended value 64 (decimal).

GLBCE_SLOPE_MAX[7:0] SLOPEMAXLIMIT register is used to restrict the slope of the tone curve generated by iridix. Recommended value 72 (decimal).

7.7.3.7.3.3.10 GLBCE Iridix Black Level

The value stored in the Black Level register GLBCE_BLACK_LEVEL[15:0] VAL will be used as the zero level for iridix processing in all data channels. Data below the Black level will not be processed and remain unchanged. Recommended value 0 (decimal).

7.7.3.7.3.3.11 GLBCE Iridix White Level

The value stored in White Level register GLBCE_WHITE_LEVEL[15:0] VAL will be used as white level for iridix processing in all data channels. Data above White level will not be processed and stay unchanged. Recommended value 65535 (decimal).

7.7.3.7.3.3.12 GLBCE Iridix Asymmetry Function Look-up-table

The Asymmetry function is used to balance the iridix effect between the dark and bright regions of the image. The Asymmetry Function Lookup Table geometry is 33 words, each of which is 16-bits wide.
GLBCE_LUT_FI_00
GLBCE_LUT_FI_01
... repeat until...
GLBCE_LUT_FI_32
Recommended values (decimal):
0:5377:10218:14600:18585:22224:25561:28631:31466:34092:36530:38801:40921:42904
:44764:46511:48156:49706:51171:52557:53870:55116:56299:57425:58498:59520:60497
:61429:62322:63176:63995:64781:65535

The C-code for generating the Asymmetry LUT is shown below:

```
#include <math.h>
int AsymmetryTable[33];
void GenerateAsymmetry(int Asymmetry, int SecondPole)
{
    double x  = (double(Asymmetry)+1)/257 * 2 - 1;
    int ai = (int)floor(0.5 + 255 * (1- 1/(1000*x*x*x) + x - ((x >= 0)*2)));
    double as = fabs(double(ai) / 255);
    double dp = double(SecondPole) / 255;
    int ii;
    for (ii = 0 ; ii < 33 ; ++ii)
    {
            if(ai >= 0)
            {
                    x = double(ii) / 32;
            }
            else
            {
                    x = double(32 - ii) / 32;
            }
            int y = (int)floor((dp+(1-dp)*pow((fabs(1-dp-x)/dp), 3)) * (x*(as+1)/(as+x) ) *
                 65535.0 + 0.5);
            y = y < 0 ? 0 : y > 65535 ? 65535 : y;

            if (ai < 0)
            {
```


Copyright © 2026 Texas Instruments Incorporated

```
                    y = 65535 - y;
                }
                AsymmetryTable[ii] = y;
            }
        }
```

7.7.3.7.3.3.13 GLBCE Iridix Forward and Reverse Perceptual Functions Look-up-tables

The iridix core works in a perceptually uniform space and thus when linear data is presented to the core it needs to be processed with special perceptual functions. These are implemented as two perceptual LUTs namely: Forward Perceptual LUT and Reverse Perceptual LUT. Each LUT is made up of sixty-five 32-bit words.
GLBCE_REV_PERCEPT_LUT_00
GLBCE_REV_PERCEPT_LUT_01
... repeat untill ...
GLBCE_REV_PERCEPT_LUT_64

Recommended values (in decimal) for Reverse LUT:
0:228:455:683:910:1138:1369:1628:1912:2221:2556:2916:3304:3717:4158:4626:5122:
5645:6197:6777:7386:8024:8691:9387:10113:10869:11654:12471:13317:14194:15103
:16042:17012:18014:19048:20113:21210:22340:23501:24696:25922:27182:28475
:29800:31159:32552:33977:35437:36930:38458:40019:41615:43245:44910:46609
:48343:50112:51916:53755:55630:57539:59485:61466:63482:65535

GLBCE_FWD_PERCEPT_LUT_00
GLBCE_FWD_PERCEPT_LUT_01
... repeat untill ...
GLBCE_FWD_PERCEPT_LUT_64

Recommended values (in decimal) for Forward LUT:
0:4622:8653:11684:14195:16380:18335:20118:21766:23304:24751:26119:27422:28665
:29857:31003:32108:33176:34209:35211:36185:37132:38055:38955:39834:40693:41533
:42355:43161:43951:44727:45488:46236:46971:47694:48405:49106:49795:50475:51145
:51805:52456:53099:53733:54360:54978:55589:56193:56789:57379:57963:58539:59110
:59675:60234:60787:61335:61877:62414:62946:63473:63996:64513:65026:65535

7.7.3.7.3.3.14 GLBCE Iridix WDR Look-up-table

If an image sensor with built-in WDR functionality is used, and that sensor outputs data in a companded format, this LUT can be used to reverse the companding function in the sensor and feed linear data into the iridix core. This LUT has 257 entries.

GLBCE_WDR_GAMMA_LUT_00
GLBCE_WDR_GAMMA_LUT_01
... repeat untill ...
GLBCE_WDR_GAMMA_LUT_256

Recommended values (in decimal):
0:5887:10263:13117:15287:17058:18564:19881:21055:22117:23088:23984:24818:25597:26330
:27021:27677:28301:28896:29466:30011:30536:31040:31527:31997:32451:32891:33318:33733
:34135:34527:34908:35280:35643:35996:36342:36680:37010:37333:37650:37960:38264:38562
:38854:39141:39423:39700:39972:40239:40502:40761:41016:41266:41513:41756:41996:42232
:42465:42694:42921:43144:43364:43582:43797:44009:44218:44425:44629:44831:45030:45228
:45423:45615:45806:45995:46181:46366:46548:46729:46908:47085:47260:47434:47606:47776
:47945:48112:48277:48441:48604:48765:48924:49083:49239:49395:49549:49702:49854:50004
:50153:50301:50448:50593:50738:50881:51024:51165:51305:51444:51582:51719:51855:51990
:52124:52257:52389:52521:52651:52781:52909:53037:53164:53290:53415:53540:53663:53786
:53908:54030:54150:54270:54389:54507:54625:54742:54858:54974:55089:55203:55316:55429
:55541:55653:55764:55874:55984:56093:56202:56310:56417:56524:56630:56736:56841:56945
:57049:57153:57256:57358:57460:57561:57662:57763:57863:57962:58061:58159:58257:58355

Copyright © 2026 Texas Instruments Incorporated

:58452:58548:58645:58740:58835:58930:59025:59118:59212:59305:59398:59490:59582:59673
:59764:59855:59945:60035:60124:60213:60302:60390:60478:60566:60653:60740:60827:60913
:60999:61084:61169:61254:61338:61422:61506:61589:61673:61755:61838:61920:62002:62083
:62164:62245:62326:62406:62486:62566:62645:62724:62803:62882:62960:63038:63115:63193
:63270:63347:63423:63500:63576:63651:63727:63802:63877:63952:64026:64101:64175:64248
:64322:64395:64468:64541:64613:64685:64757:64829:64901:64972:65043:65114:65185:65255
:65325:65395:65465:65535

7.7.3.7.4 GLBCE Embedded Memory

GLBCE needs 2 lines of line memories and 1 set of cache memories.

Table 7-107. GLBCE Memories

Memory Type	Qty	Size	Clock	ECC Supported	Description
DELAY LINE	1	2122 x 64 (GLBCE_LINE_SIZE /2+10)x64	FCLK	No	GLBCE line memory
STATMEM	8 banks	16 bit x 1024	FCLK	Yes	GLBCE Cache memory (computes and stores the statistics from the frame)

7.7.3.7.5 GLBCE General Processing

GLBCE block needs appropriate statistics information to be stored to its cache memory in prior to processing incoming frame. Usually, this takes a two-pass process.

1. GLBCE is configured for input frame-A
2. Frame-A is given to GLBCE
3. GLBCE processes the input frame-A
4. GLBCE generates statistic (statistics-A) data based on frame-A in vertical blanking period
5. (Optional process) statistics-A is read from cache memory and moved to SDRAM
6. GLBCE is configured for input frame-B
7. Frame-B is given to GLBCE
8. GLBCE processes the input frame-B with the statistics-A generated from frame-A
9. GLBCE generates statistics-B based on frame-B in vertical blanking period
10. (Optional process) statistics-B is read from cache memory and moved to SDRAM

Alternatively, GLBCE can process in the following manner

1. Pre-calculated Statistics-C is copied to GLBCE cache memory from SDRAM
2. GLBCE is programmed for input frame-D
3. Frame-D is given to GLBCE
4. GLBCE processes frame-D based on statistics-C
5. GLBCE generates statistics-D based on frame-D in vertical blanking period
6. (Optional process) statistics-D is read from cache memory and moved to SDRAM

In Section 7.7.3.7.6, some examples closer to real world use cases are explained based on the above two processes.

7.7.3.7.6 GLBCE Continuous Frame Processing

This is a very basic case, which does not need transferring statistics between cache memory and SDRAM. The most common example is video capture. In this case, the most straight forward way to run GLBCE, is to generate statistics from N-th frame, and apply it to (N+1)-th frame. The effect from using statistics of different frame should be small if the difference between frames is small. For example, it is expected this does not generate artifact, if the video frame rate is 30fps or larger. Below this rate, the single frame processing programming model explained in Section 7.7.3.7.7 should be considered.

In the case of continuous frame processing, GLBCE is operated in the following manner:

Copyright © 2026 Texas Instruments Incorporated

1. GLBCE is programmed for incoming frame
2. Input frame comes in
3. GLBCE processes Input frame based on the statistics from the previous frame
4. GLBCE generates statistics based on the current input frame in the vertical blanking period
5. Repeat Step 1 through 4 as required

For the first frame of a set of sequential frames, Strength parameter should be 0, so GLBCE does not affect the image based on wrong statistics. SW may need to increase the strength value gradually after the first frame.

7.7.3.7.7 GLBCE Single Image Processing

For a single image, GLBCE needs to process the image twice, once for statistics generation, and once for actual GLBCE process.

1. GLBCE is configured for the input image.
2. The image is given to GLBCE. (Output image from GLBCE should be surpressed, for example, by disabling the next module in the path)
3. Statistics is generated in vertical blanking period.
4. The input image is given to GLBCE again. Output image with right GLBCE is generated.

   Also, GLBCE can take in small size image in the first pass to reduce processing time.
5. A small size version (IMG-S) of the input image (IMG-IN) is generated.
6. GLBCE is configured for the small size image (IMG-S).
7. The small size image (IMG-S) is given to GLBCE.
8. Statistics for the small size image is generated in vertical blanking period.
9. GLBCE is configured for the original input image (IMG-IN).
10. The original input image (IMG-IN) is given to GLBCE.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.8 VISS Flexible Color Processing (FCP) Module

7.7.3.8.1 FCP Overview

The Flexible Color Processing module is responsible for performing CFA Interpolation as well as performing a host of color processing functions for generating different color outputs.

7.7.3.8.1.1 FCP Features Supported

The FCP include the following main features:

- Flexible CFA for generating multi-plane output for any 2x2 raw format sensor configuration.
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats
  - Can generate up to 4 independent color planes
  - Support for up to 3 directions per color plane
  - Adaptive Threshold based on intensity measure
- Support for multiple color format output generation:
  - Support 12-bit YUV output
  - Supports RGB output
  - Supports Saturation outputs (8 bits only)
  - Supports Grayscale/Luma/IR/Clear output.
- Support for 16 to 12 compression (LUT based) at the entry of the block
- Support for flexible output format generation including YUV as well as RGB, SV, and so forth.
- Single cycle/pixel performance

7.7.3.8.2 FCP Functional Description

The Flexible Color Processor (FCP) receives data from RAW-FE and does demosaicing and color conversion. The output of FCP is sent to external memory.

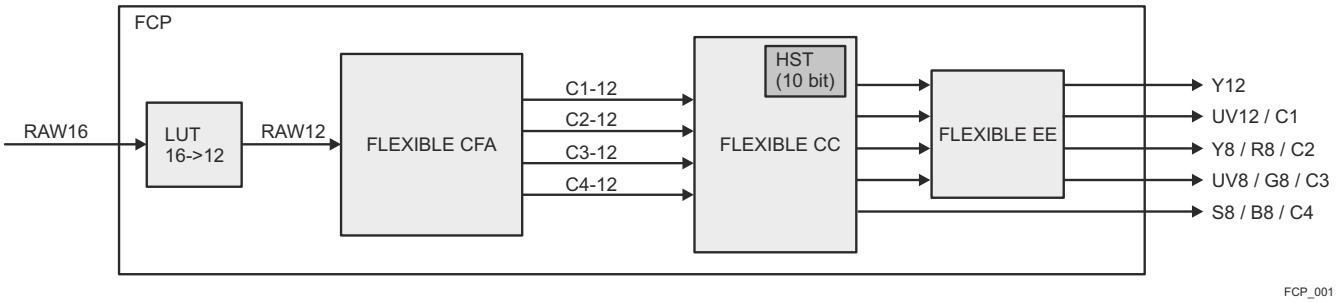


Figure 7-63. FCP Block Diagram

The FCP consists of the following major sub-blocks for supporting different sensor and output formats:

- LUT based compression: used to reduce the bit width from 16 bits to 12 bits while still preserving the dynamic range.
- Flexible CFA: this block takes in as input any 2x2 raw sensor pattern and is capable of generating up to 4 full resolution color planes.
- Flexible Color Conversion (Flexible CC): this block takes in the output of the Flexible CFA and generates multiple standard as well as custom data formats.
- Flexible Edge Enhancer (Flexible EE): this block can take the output of the Color Conversion and align the Y and UV channels as well as enhance the edges in the luma channel.

7.7.3.8.3 FCP Submodule Details

7.7.3.8.3.1 Flexible CFA / Demosaicing

7.7.3.8.3.1.1 Feature-set

The Flexible CFA include the following main features:

- Support for Flexible CFA for generating multi-plane output for any 2x2 raw format sensor configuration.
  - Supports RCBC, RCCC, Bayer, RGBC as well as other formats.

Copyright © 2026 Texas Instruments Incorporated

  - Can generate up to 4 independent color planes.
  - Supports Clear (C) pixel.
  - Supports Infrared (IR) pixels.
- Edge aware CFA interpolation.
- Software programmable filter for interpolation up to 4 color channels.
- Support for up to 4 phases and 3 directions per color channel.
- Software controlled algorithm for direction selection.
- Adaptive threshold calculation for direction selection.
- Support of up to 2 sets of programmable mask based gradient selection

7.7.3.8.3.1.2 Block Diagram of Flexible CFA

The Flexible CFA module follows the VPORT (internal) interface for both the input/output to the RAW-FE or Flexible CC module.

7.7.3.8.3.1.2.1 Gradient/Threshold Calculation

The Flexible CFA incorporates 2 sets of gradient and direction selection logic. Typically for symmetric 2x2 patterns like Bayer, only a single set is sufficient; however in sensors wherein one of the channels is working on a different data type, the second set can be used for that channel. An example of such sensors is the RGB-IR sensor, wherein the 2x2 pattern consists of one pixel of each color channel and one pixel of the IR channel. In this case, mixing the gradients and intensity of the color channels with the IR channel could be meaningless.

The gradient selection uses bit masks to select the desired pixel position for calculating the horizontal and the vertical gradient.

Figure 7-64 shows the notation for the bit-mask values. The same notation is used for vertical masks as well and the notation implies that setting a ‘1’ in any field of the bit Mask enables the absolute difference of those 2 pixels to be summed in the gradient calculation.

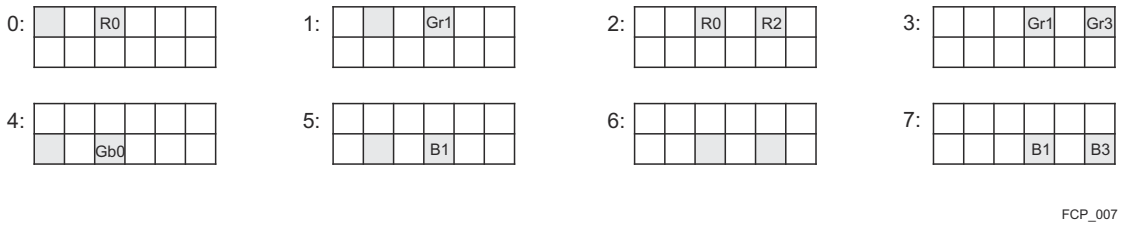


Figure 7-64. Gradient Bit Masks

Similar masks are also used to calculate the intensity (see Figure 7-65), however instead of taking the absolute difference, the bit masks are used to sum up the pixels in those locations.

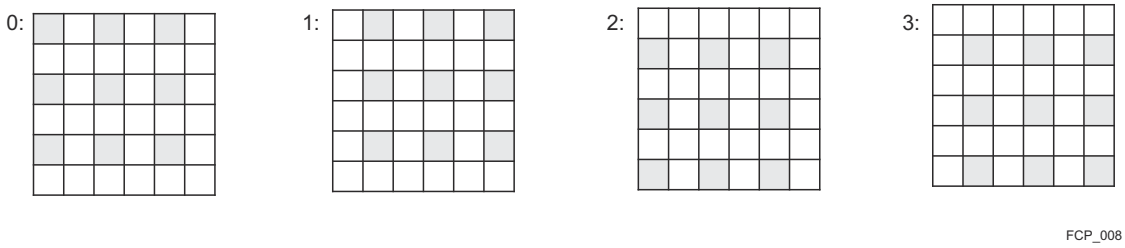


Figure 7-65. Intensity Bit Mask

The 4 bit Intensity mask is used to choose which pixels in the 6 rows should be used for creating the intensity sum. The same masks are used for every 2x2 window within the 6x6 buffer. (Thus setting all 4 bits to ‘1’ will be equivalent to summing up the entire 6x6 window.) A programmable shift is then used to bring the intensity down to the 0-16k range. The maximum value of intensity can be 6x6 x4k = 144k, so the maximum value of shift is 4 (divide by 16). The shift value can be programmed to any value between 0 and 4 and should be chosen based

Copyright © 2026 Texas Instruments Incorporated

on the mask. Note that dependent on the shift value, it's possible that the full range of 0-16k of the intensity may not be reached, hence the adaptive threshold should be programmed accordingly.

The adaptive threshold calculation uses the calculated intensity. For more information, see Section 7.7.3.8.3.1.2.2, *Software Controlled Direction Selection*.

7.7.3.8.3.1.2.2 Software Controlled Direction Selection

Software can control or guide direction selection at frame or sequence level as shown in Figure 7-66.

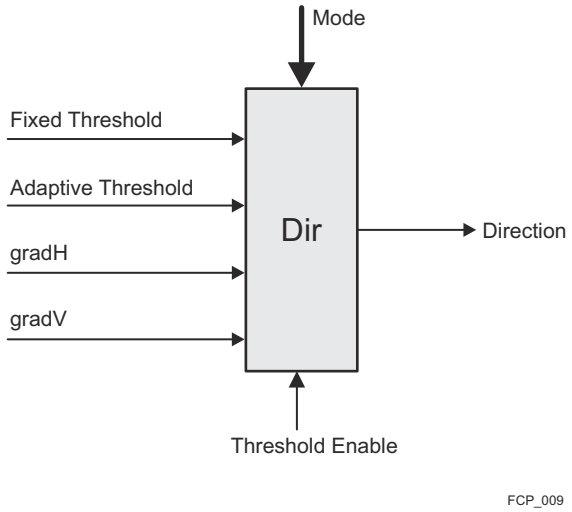


Figure 7-66. Software Controlled Direction Selection

Table 7-108 summarizes number of option available to software.

Table 7-108. Summary of HSX Spaces

Mode	Name	Details
0	Horizontal	Dir = 0
1	Vertical	Dir = 1
2	HV	Dir = 2
3	HV_Threshold (Fixed threshold is MMR)	in Dir = 0: gradH ≥ gradV + threshold in Dir = 1: gradV ≥ gradH + threshold in Dir = 2: Otherwise
4	HV_AdaptiveThreshold	in Dir = 0: gradH ≥ gradV + threshold in Dir = 1: gradV ≥ gradH + threshold in Dir = 2: Otherwise

For calculation of adaptive threshold, 7 options are used. Each one defines a threshold to be used as a different intensity level as it follows:

- Entry-0 -> Intensity = 0
- Entry-1 -> Intensity = 512
- Entry-2 -> Intensity = 1024
- Entry-3 -> Intensity = 2048
- Entry 4 -> Intenisty = 4096
- Entry 5 -> Intensity = 8192
- Entry 6 -> Intensity = 16384

The adaptive threshold can then be calculated as a linear interpolation between 2 successive entries based on the given intensity.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.8.3.2 Edge Enhancer Module Wrapper (WEE)

Figure 7-67 shows a high level block diagram of the Edge Enhancer (EE) module wrapper. For more information about the Edge Enhancer algorithm, see *Edge Enhancer (EE)*. The wrapper places the Edge Enhancer within the FCP structure along with assotiated muxes.

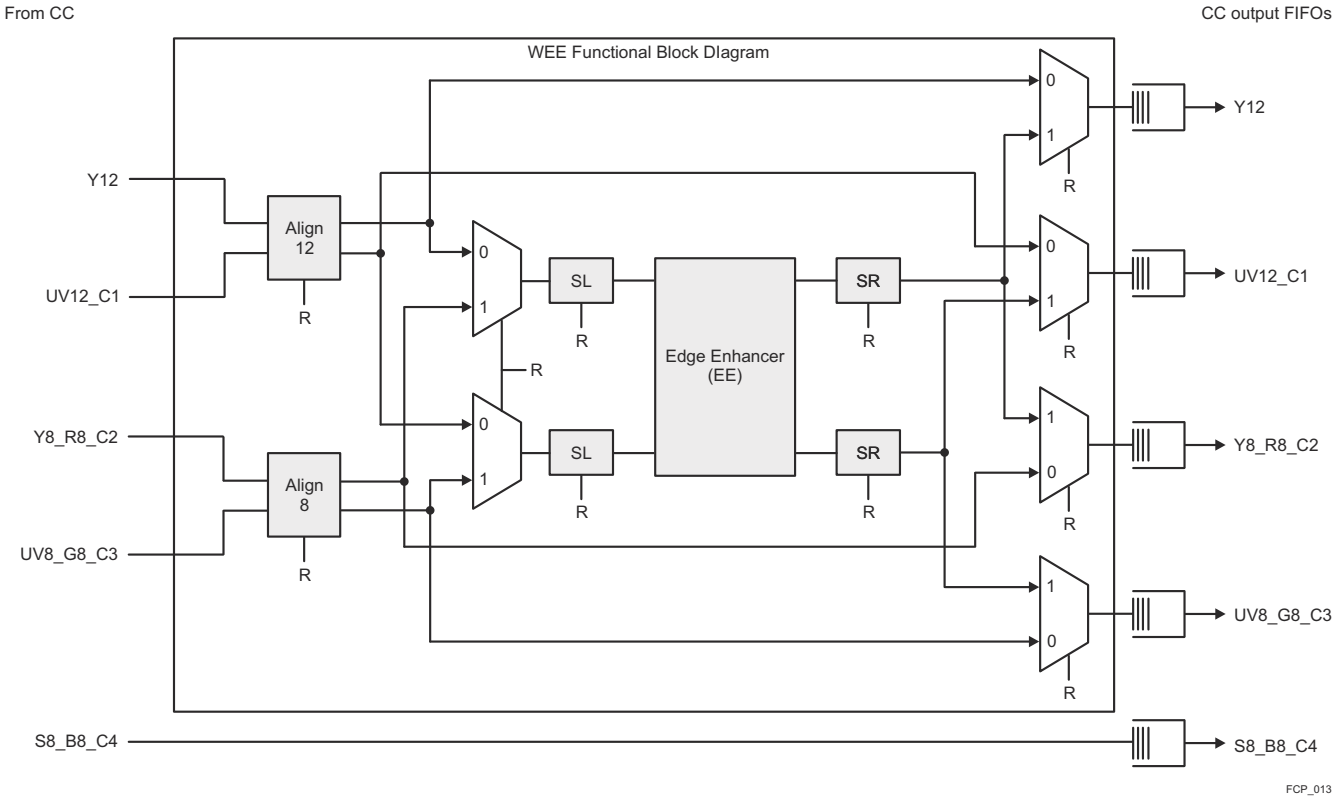


Figure 7-67. Block Diagram of Edge Enhancer Module Wrapper

7.7.3.8.3.2.1 EE - Edge Enhancer Block

The Edge Enhancer module is used to enhance the visual quality of the image by increasing its sharpness. The module only works on Y (Luma) data and uses a combination of 2D High-Pass Filtering to detect edges and then apply them on the input image.

7.7.3.8.3.3 Flexible Color Conversion (CC)

Figure 7-68 shows a high level block diagram of the Flexible Color Conversion (CC) module.

Copyright © 2026 Texas Instruments Incorporated

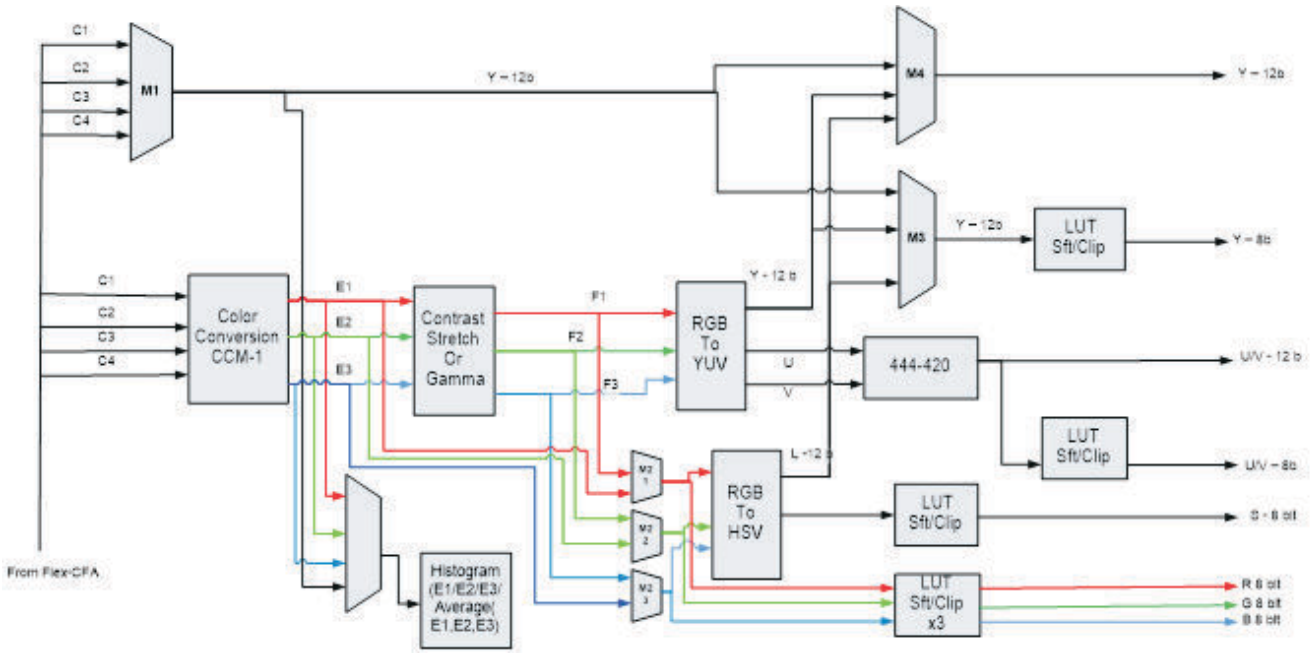


Figure 7-68. Flexible CC Block Diagram (Logical View)

The Flexible CC enables multiple use cases and allows different color formats to be generated.

The Flexible CC module follows the VPORT (Internal) Interface for both the input to the Flexible CC module.

7.7.3.8.3.3.1 Interface Mux

The Flexible CC logical block diagram has 5 primary outputs connecting to the LSE modules in the VISS. For more information about LSE modules, see *Load Store Engine (LSE)*. A multiplex scheme is used to connect multiple outputs on to 5 interfaces using the interface Mux. The Flexible CC output can optionally also allow data from Flexible CFA to be tapped out on to one of the primary interfaces. Note that 12-8 LUTs are shared between the RGB generation and the Y/UV 8-bit generation logic.

Figure 7-69 shows the details of the interface mux. There are only 3 LUTs shared in different mode and only 5 concurrent output streams can be activated at any point of time. For 8-bit outputs, the data is stored in the MSB 11-4 bits of the 12-bit bus. Further, the Y12 output is dedicated and not multiplexed with other interface signals.

Copyright © 2026 Texas Instruments Incorporated

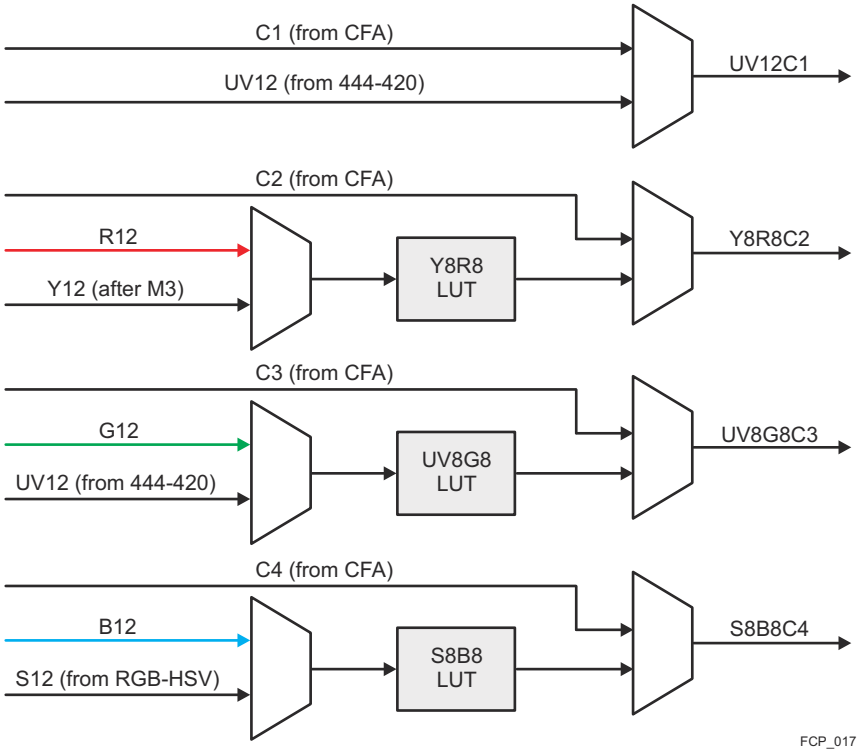


Figure 7-69. Flexible CC Interface Mux

7.7.3.8.3.3.2 Color Conversion (CCM-1)

The Color Conversion (CCM-1) is defined using the mathematical equations from Figure 7-70.

$$E1(x,y) = W11 \times C1(x,y) + W12 \times C2(x,y) + W13 \times C3(x,y) + W14 \times C4(x,y) + Offset_1$$
$$E2(x,y) = W21 \times C1(x,y) + W22 \times C2(x,y) + W23 \times C3(x,y) + W24 \times C4(x,y) + Offset_2$$
$$E3(x,y) = W31 \times C1(x,y) + W32 \times C2(x,y) + W33 \times C3(x,y) + W34 \times C4(x,y) + Offset_3$$

Figure 7-70. CCM-1

The CCM matrices weights (W**) are 12 bit signed each (S12Q8) representing a range from -8 to +7.996 with 8 bits of fraction. The offsets are 13 bit signed notation (S13Q11) representing a range of -4096 to +4095. The offset is effectively shifted by one (left shifted/ multiplied by 2) before being applied. The effective range of offset is -2 to +1.995 with 11 bits of fraction.

The 4 component CCM allows for a power combination of color format generation to be addressed.

7.7.3.8.3.3.3 RGB to HSX Conversion

The RGB-HSV module is used for converting color spaces and creating custom luminance data plane. This way the 8-bit Saturation and 12-bit Luminance (L or V) plane are generated.

Figure 7-71 shows the block which generates the Luminance / Gray Scale Plane as well as the Saturation (8-bit) plane.

Copyright © 2026 Texas Instruments Incorporated

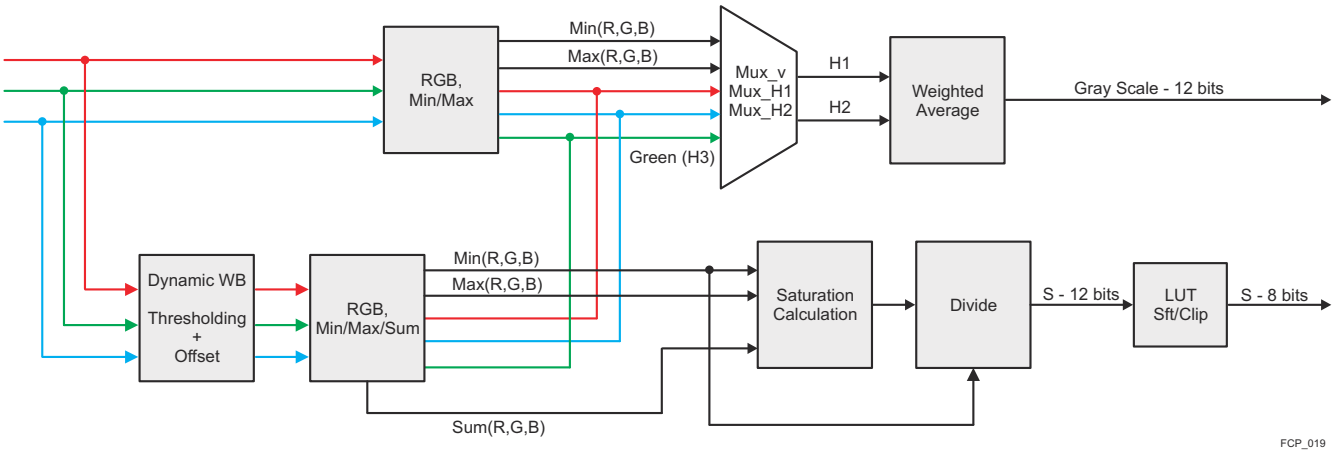


Figure 7-71. S & V Generation

7.7.3.8.3.3.3.1 Weighted Average Block

The Weighted Average block implements the equation shown on Figure 7-72.

$$GrayScale(x,y) = W1 \times H1(x,y) + W2 \times H2(x,y) + W3 \times H3(x,y) + Offset_1$$

Figure 7-72. Gray Scale Computation

In the equation on Figure 7-72 the Weights (W1*) are 12 bits signed with a range of -8 to +7.9996 in S12Q8 format. The offset is Signed 13 bits in S13Q11 format with a range of -2 to +1.995.

Figure 7-73 depicts the archtiectue of the Weighted Average block:

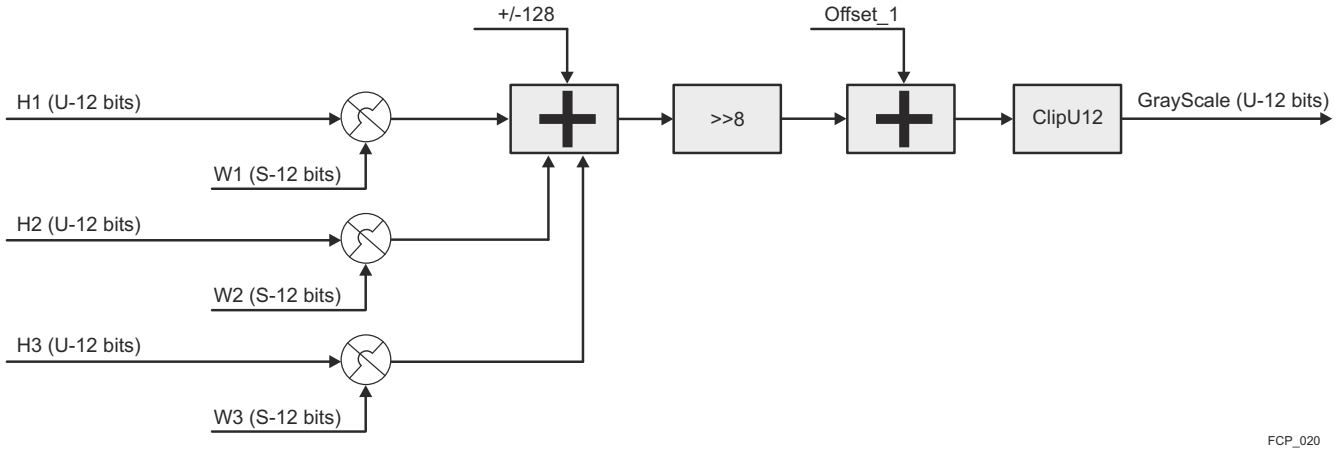


Figure 7-73. Weighted Average Block

The mux before the matrix chooses whether the native R/B channels are transmitted or alternatively the Min/Max (RGB) from the Min-Max block are transmitted. Further, the calculation of V can work on either the non-WB corrected data or the WB corrected version. For more information about WB correction, see Section 7.7.3.8.3.3.3.2, *Saturation Block*.

The combination of the Min-Max block, the mux and the Weighted Matrix allows any of the Gray Scale calculation as shown in Table 7-109.

Copyright © 2026 Texas Instruments Incorporated

Table 7-109. Summary of HSX Spaces

No	Formula	Configuration
Grey scale (HSI)	(R+G+B) / 3	H1 = R, H2 = B; Offset_1 = 0; W1 = W2 = W3 = 1/3 = 85;
Gray Scale (HSV)	Max(RGB)	H1 = Max(RGB); H2 = NA Offset_1 = 0; W2/W3 = 0; W1 = 1 (256);
Gey Scale HSL	(Max(RGB) + Min (RGB)) / 2	H1 = Max(RGB), H2 = Min(RGB); W1 = W2 = 0.5 (128) Offset_1 = 0; W3 = 0;
		H1 = R, H2 = G; W1 = W3 = 0.25 (64); W2 = 0.5 (128) Offset_1 =0

Table 7-110. Grey Scale and Saturation Calcuation

No	HSI Color Space	HSV Color Space	SHL Color Space	Customer Requirements
Grey Scale Computation	(R+G+B) / 3	Max(R,G,B)	(Max(R,G,B) + Min(R,G,B)) / 2	(R+2*G+B) / 3
Saturation (S) Computation	1-(Min(R,G,B) / (R+G+B))	(Max(R,G,B) - Min(R,G,B)) / Max(R,G,B)	(Max(R,G,B) - Min(R,G,B)) / 255-Gray value	S = Max(R,G,B) - Min(R,G,B)
Hue (H) Computation	Complex formula involving (R-G) (R-B) & (G-B)	Division formula livolving (R-G) (R-B) & (G-B)	Complex formula involving all component	Separate output for (R-G) and Yellowness

7.7.3.8.3.3.3.2 Saturation Block

The Saturation block first applies a dynamic white balance offset to correct the saturation plane when the image is in log domain and is used for analytics data flow. Once the WB offset is applied, the saturation calculation proceeds as described below. Note that the White balance offset is only applied if the independent pixel values are below a threshold (VISS_FCP_FCC_RGBHSV_WB_LINLOGTHR_1/2). Further, even the minimum of RGB is compared against a threshold (VISS_FCP_FCC_RGBHSV_WB_LINLOGTHR_2[27-16] SATMINTHR bit field) and the higher of the two is used. The saturation calculation always uses the data with the WB correction applied, however the V calculation can choose between WB corrected or uncorrected data using VISS_FCP_FCC_CFG_1[26] MUXRGBHSV_MUX_V.

Table 7-110 shows the condensed form for the Grey Scale and Saturation calculation. In the Flexible CC the saturation calculation is supported using two modes based on the VISS_FCP_FCC_CFG_2[6] HSVSATMODE bit as follows.

- HSVSATMODE = 0: Max(RGB) - Min(RGB)
- HSVSATMODE = 1: SUM(RGB) - Min(RGB)

Similarly the denominator for the dvision is chosen using the following selection on the VISS_FCP_FCC_CFG_2[5-4] HSVSATDIVMODE bit field.

- HSVSATDIVMODE = 0: No division, denominator = 1
- HSVSATDIVMODE = 1: Max(RGB)
- HSVSATDIVMODE = 2: 4095 – Gray value (computed in Figure 7-72)
- HSVSATDIVMODE = 3: Sum(RGB)

7.7.3.8.3.3.3.3 Division Block

The division operation is implemented to calculate 1/x followed by a multiplication with the value. The input to the LUT is 12-bit unsigned value and the output is 8-bits fraction representing 1/x calculation. The divide LUT is implemented as a non-linear ROM with 1216 (1280 for aligning to 128 size boundary) entries and up to 2 segments to reduce error. Ideally to implement a full ROM for an input space of 12 bits would require 4k entires, however to save space and reduce error, the ROM is partitioned such that it is full resolution for the first 256 entries where the curve is highly non linear, whereas the next 4k - 256 entires have a step size of 4. Each ROM entry has a bit size of 20 bits in U20Q20.

Copyright © 2026 Texas Instruments Incorporated

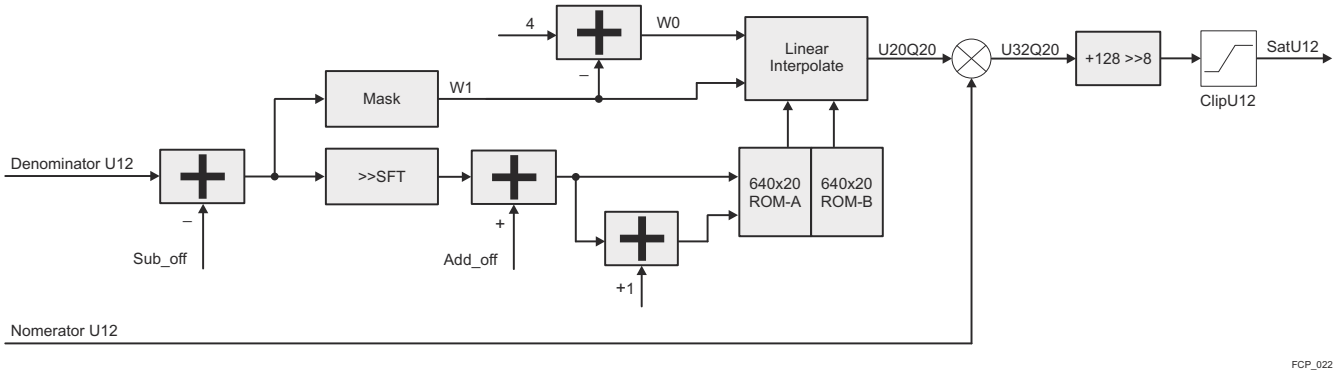


Figure 7-74. Division LUT

These are the parameters that need to be calculated based on the range of the input.

- SFT: '0' if input <256, 2 otherwise
- Mask: '0' if input <256, 4 otherwise
- Add_off: '0' if input <256, 256 otherwise
- Sub_off: '0' if input <256, 256 otherwise

The 'shift' parameter at the end is a configuration register which needs to be programmed. Typical programming value is 8 to generate a result in Q12 format (such that the range is 0 to1). This is usually the case when the denominator is greater than the numerator.

7.7.3.8.3.3.3.4 LUT Based 12 to 8 Downsampling

In the final step the generated saturation value (12 bit) or the incoming RGB values (12 bit) needs to be scaled down to 8 bits using the linear LUT with 513 entries.

Figure 7-75 is a high level block diagram of the LUT block. It shows the weight calculation for the LUT assuming the incoming data can be of any bit depth between 8 and 12 bits.

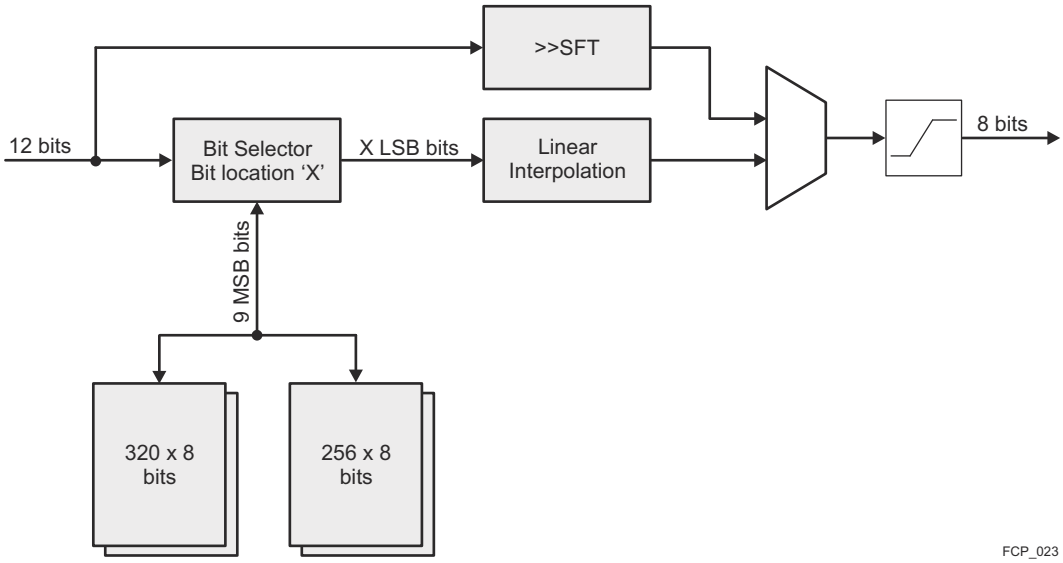


Figure 7-75. LUT Based 12-8 Compression

The LUTs are sized to provide 513 locations of data, to enable linear interpolation for all locations. The internal weights are sized as 3 bits since the maximum delta between 2 steps is only 4095/512 = 8.

Figure 7-76 shows the high level block diagram of the interpolation process. The logic is designed to scale for bit width and can support anything from 8 to 12 bits of input to support different usecases. However, since the

Copyright © 2026 Texas Instruments Incorporated

step size is different depending on the input bit width, the bit selection and weight calculation logic is designed to account for that.

A shift operation can be performed instead of LUT to reduce bitwidth from 12 down to 8 bits.

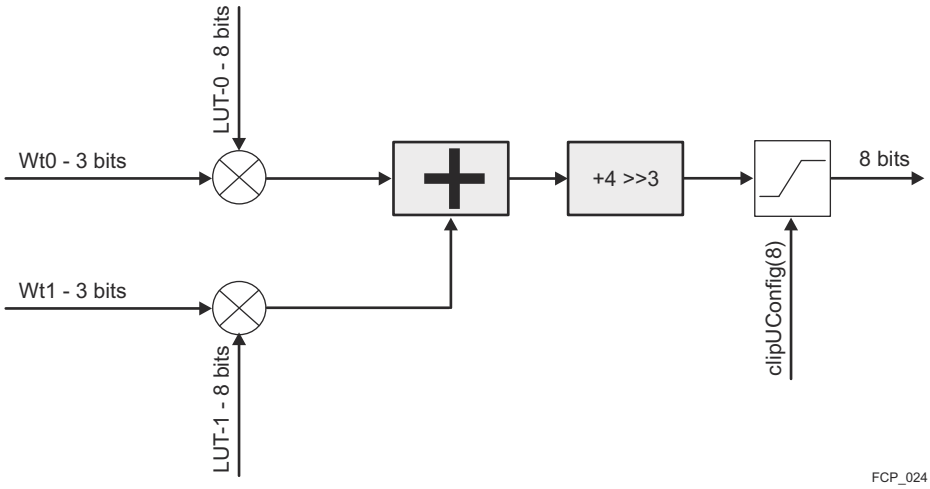


Figure 7-76. Linear Interpolation on LUT

The code below shows the addressing for the LUT as well as the weight calculation logic for supporting multiple bit widths at the input from 8 – 12 bits.

12-8 Bit LUT Curve

```
CASE BIT_SEL
12 (12 bit data): Addr = Inp[3:11]; Wt1 = Inp[2:0]
11 (11 bit data): Addr = Inp[2:10]; Wt1 = Inp[1:0] & '0'
10 (10 bit data): Addr = Inp[1:9]; Wt1 = Inp[0] & '00'
9 (9 bit data): Addr = Inp[0:8]; Wt1 = '000'
8 (8 bit data): Addr = Inp[0:7]; Wt1 = '000';
//Wt0 is always calculated at 8 - Wt1
Wt0 = 8 - Wt1
```

The input bit width (BIT_SEL) is specified using a dedicated register for generating Y8 output. However for other paths (RGB/UV/Sat) the 8 bit conversion is done using either 12 bits (if data is tapped prior to contrast block) as the bitwidth or the clipping value after contrast block (VISS_FCP_FCC_CFG_2[12-9] CONTRASTBITCLIP, if data is tapped after contrast block).

7.7.3.8.3.3.4 Histogram

The histogram can work on either one of the color components after the CCM-1 or the fourth color component routed directly from the Flexible CFA. The Histogram can also work on internally generated Luma value using simple averaging of RGB components. The Histogram module supports a local memory which can be read by the host processor. The Histogram provides data which can be used by the software to generate tone curves for the Contrast Stretch LUT.

The input to the Histogram module is calculated based on the control MMR as it follows:

- 0: R
- 1: G
- 2: B
- 3: MuxC1_4
- 4: (R+2*G+B) / 4

Copyright © 2026 Texas Instruments Incorporated

The histogram module downsamples the image in the horizontal direction by a factor of 2 using simple averaging filter. This ensures that the memory access requirements of the module are limited to 1 read + 1 write access every 2 cycles (1 access/cycle) and thus can be achieved using a simple memory architecture.

The histogram module requires a ping-pong memory which is mapped to the configuration bus. This allows for histogram operation to run in concurrent with the Read operation from the host. For ease of software complexity, both the Ping/Pong memories map to the same address and it is the responsibility of the design to ensure that the muxing logic always take care of the correct access from both the host and the module. To detect error conditions, if the host is not able to read fast enough from the memory, an error interrupt should be generated whenever the host has initiated a read transfer from the first location of the memory but has not initiated a read transfer from the last location of the memory.

The Histogram module has 2 banks of memores each of 256x20 bits which are used in ping pong fashion. The input pixel has to be downshifted to match the bin size. The Histogram module has separate registers for creating an ROI using horizontal/vertical start postions as well as horizontal/vertical size.

Typically the ROI registers (VISS_FCP_FCC_CFG_HIST_1 and VISS_FCP_FCC_CFG_HIST_2) have to be set that at least one line is skipped for histogram calculation. The one line of time can be used by the histogram module to set the value of each of the bins to zero. This also puts a constraint on the histogram ROI to have a minmum width of 256 pixels.

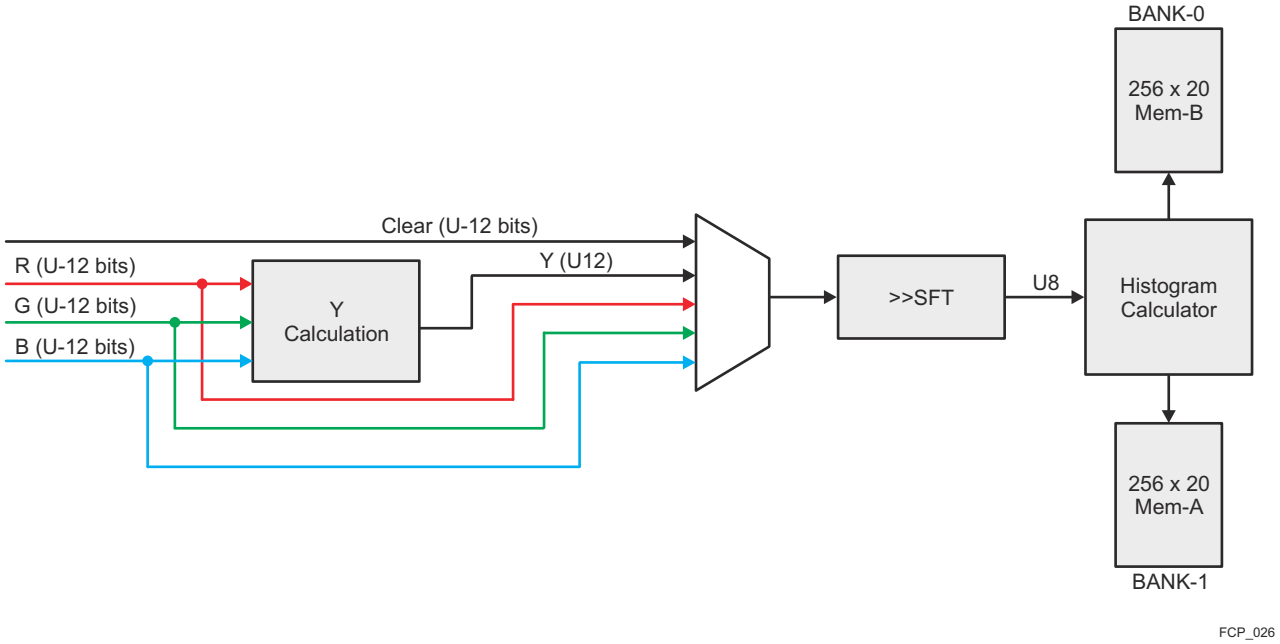


Figure 7-77. Histogram Module

7.7.3.8.3.3.5 Contrast Stretch / Gamma

The logic of the Contrast/Gamma unit is based on the same as that of the 12-8 bit conversion LUTs. There are 3 copies of the logic, one for each color channel. The standard data flow allows for a 12 bit input and 12 bit unsigned output, however different bit width combinations are possible using the select bit of the LUT and the clip value. Each entry of the LUT table is 12 bits.

Figure 7-78 and Figure 7-79 show the block diagram and implementation of linear interpolation for the Contrast Stretch/Gamma module.

Copyright © 2026 Texas Instruments Incorporated

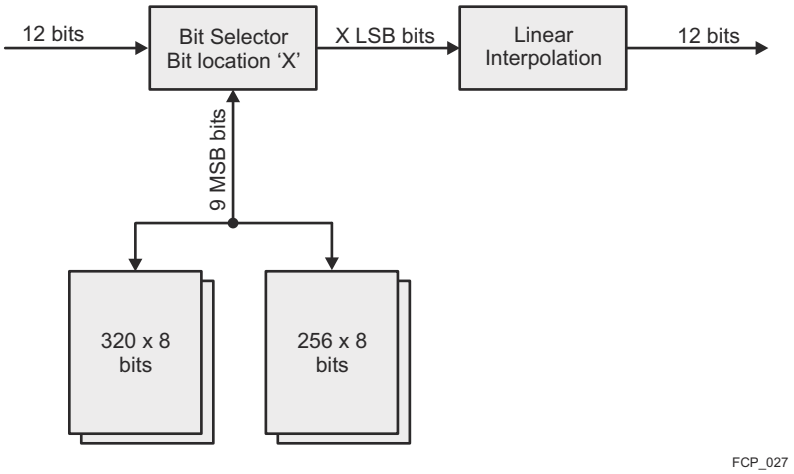


Figure 7-78. Contrast Enhancement Module

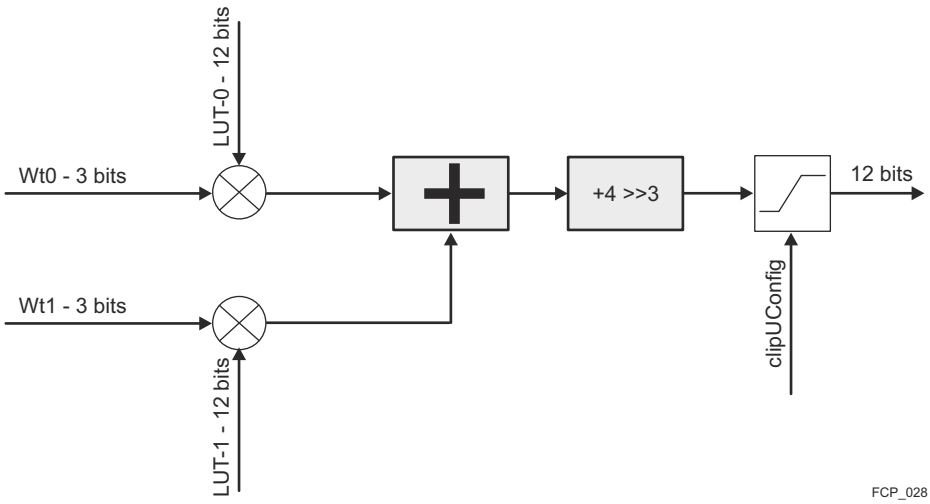


Figure 7-79. Contrast Enhancement Linera Interpolation

The code below shows the LUT addressing scheme which is similar to the 12-8 bit conversion module. The configurable clip at the end of processing allows for less than 12 bit output to be supported directly from the module.

Contrast Enhancement, LUT Addressing

```
CASE BIT_SEL
0 (12 bit data): Addr = Inp[3:11]; Wt1 = Inp[2:0]
1 (11 bit data): Addr = Inp[2:10]; Wt1 = Inp[1:0] & '0'
2 (10 bit data): Addr = Inp[1:9]; Wt1 = Inp[0] & '00'
3 (9 bit data): Addr = Inp[0:8]; Wt1 = '000'
4 (8 bit data): Addr = Inp[0:7]; Wt1 = '000';
//Wt0 is always calculated at 8 - Wt1
Wt0 = 8 - Wt1
```

7.7.3.8.3.3.6 RGB-YUV Conversion

The RGB-YUV conversion is a matrix based conversion from 12-bit RGB to 12-bit YUV. The output after conversion is YUV444.

The conversion is carried out using the equation on Figure 7-80.

Copyright © 2026 Texas Instruments Incorporated

$$Y(x,y) = WRY \times R(x,y) + WGY \times G(x,y) + WBY \times B(x,y) + Offset_1$$
$$Cb(x,y) = WRCb \times R(x,y) + WGCb \times G(x,y) + WBCb \times B(x,y) + Offset_2$$
$$Cr(x,y) = WRCr \times R(x,y) + WGCr \times G(x,y) + WBCr \times B(x,y) + Offset_3_$$

Figure 7-80. RGB-to-YUV Conversion

In this equation, the weights are signed 12 bits (S12Q8) with 8 bit of fraction precision, representing a range of -8 to +7.996. The offsets are signed 13 bits.

Figure 7-81 shows the implementation of the RGB to YUV module. The output after conversion is YUV444, as such it is followed by a chroma downsampling stage where in the Chroma resolution is reduced by half in both the horizontal and vertical direction.

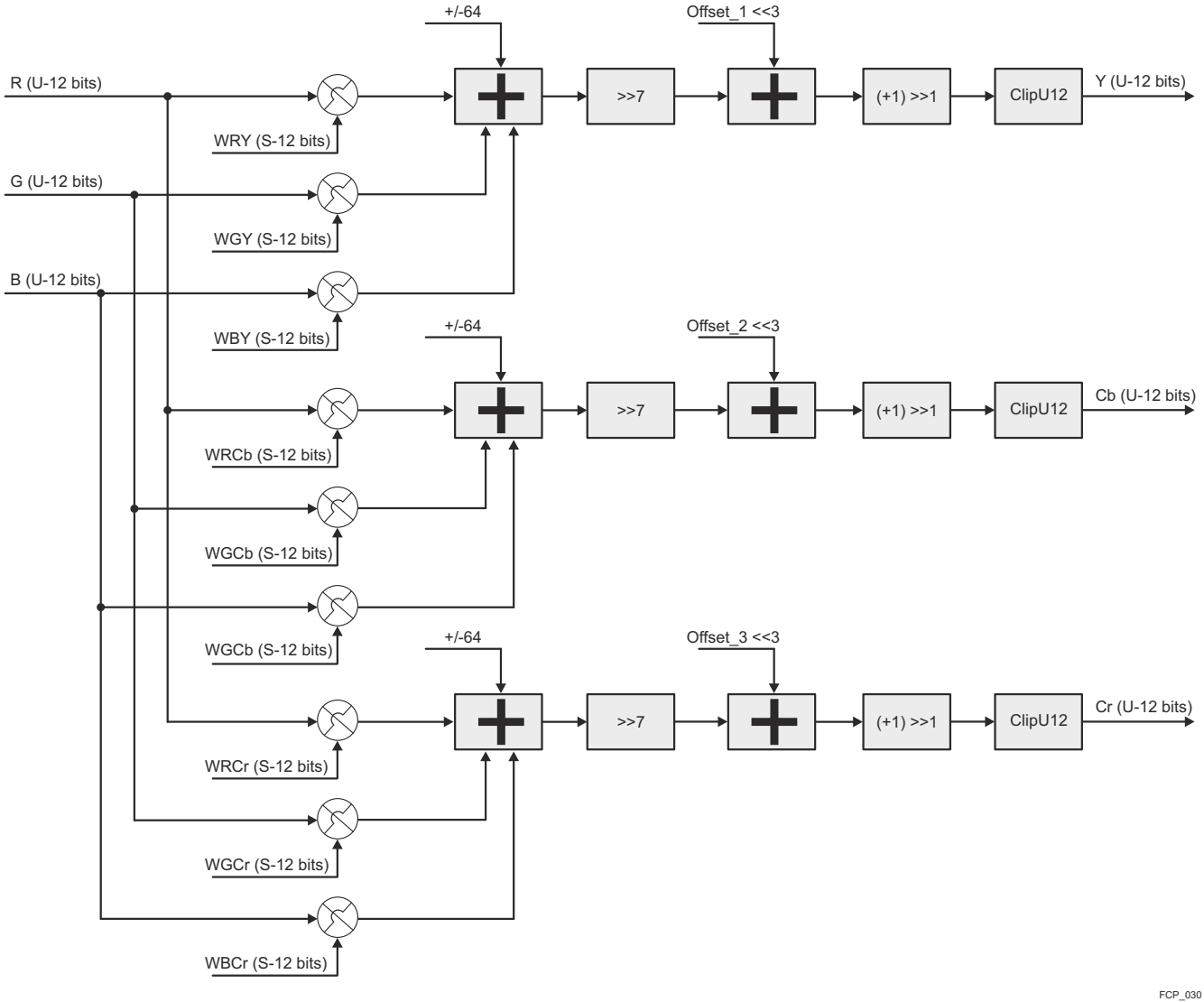


Figure 7-81. RGB to YUV

Copyright © 2026 Texas Instruments Incorporated

7.7.3.8.3.4 444-422/420 Chroma Down-sampler

This module down-samples the chroma components from 444-420 format; effectively reducing the chroma resolution to 50% in both the horizontal and vertical direction.

The downsampling is performed using a 3 Tap fitering [1 2 1] followed by decimation. The conversion logic generates chroma in co-sited/co-located format in both the horizontal and vertical direction.

Figure 7-82 shows the block diagram of the format conversion module. The Chorma output can be optionally downshifted from 12 bits to 8 bits using the LUT/SFT module. Only one copy of the Horizontal averaging filter is required since there is a decimation step following the filtering. Only even locations of both Cb/Cr are retained while the odd locations are decimated, however from design perspective the Hz Averager module works on Cb on even cycles and Cr on odd cycles.

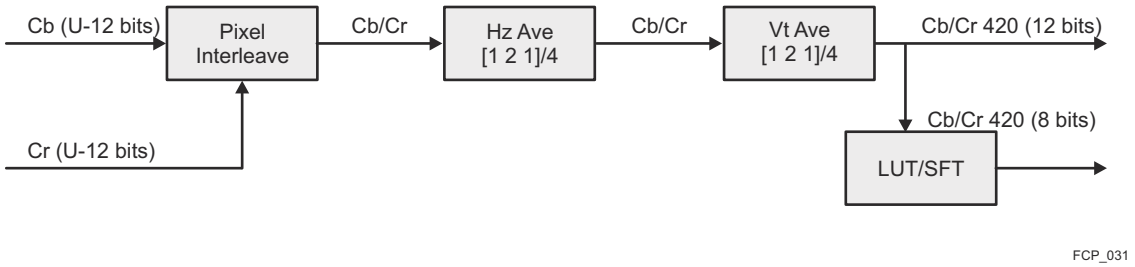


Figure 7-82. 444 to 420 Module

Additionally another mode of operation is to support planar YUV422 mode. In this case the chroma is only downsampled in the horizontal direction and the vertical averaging/downsampling is disabled.

The mode is referred to pseudo 422 since actual 422 mode requires the data to be interleaved between the Y and the chroma channels. In this case the interleaving of Y & C is handled by LSE whereas the Flexible CC module will only generate Y & Cb/Cr (interleaved) planes separately.

7.7.3.8.4 FCP Interrupts

The FCP module has three interrupts, as it follows:

- LUT_CFG_ERR - 16-12 LUT is written during active window
- CFA_PIX_ERR - Access to line memories by CFG during active window
- CFA_MMR_ERR - Non-Shadowed registers are written during active window

There are two additional events:

- SOL_EVENT - Start of line processing for Flexible CFA, triggered when the line enters the Flexible CFA
- SOF_EVENT - Start of frame processing for Flexible CFA, triggered when the frame enters the Flexible CFA

For information about the FCP interrupt events, see Section *VISS Top Level Functional Description*.

7.7.3.8.5 FCP Programmer's Guide

7.7.3.8.5.1 HWA Core Programming Details

The following fields should be programmed correctly for the FCP module to function.

- Input LUT
  - Use the LUT to reduce the bitwidth of the data to 12 bits so that it can pass through the Flexible CFA pipe.
  - If the GLBCE module is disabled, set the bit width and enable the LUT.
- Flexible CFA
  - Program the Flexible CFA kernels based on the input data pettern. Program only 3 channels for Bayer sensors and up to 4 for other non-standard formats like RGB-IR.
  - Set up the correct thresholds (use only one set for Bayer type sensors and two sets for non-standard formats like RGB-IR).
- Flexible CC
  - Set up the Flexible CC for either visual or analytics data flow.

Copyright © 2026 Texas Instruments Incorporated

- For visual, program the CCM-1 to be used as RGB2RGB and Contrast to be used as Gamma with clip at 10 bits. Only the 8-bit outputs are used and the Y8/C8 data is generated using shift down from 10 to 8 (instead of LUT).
- For analytics, CCM-1 is progrmmed depending on sensor input format and the hisogram is used to populate the contrast table. 12-bit outputs are used however Y8/C8 can be generated using 12 to 8 LUT based downshifting. The correct Saturation generation parameters should also be set.

7.7.3.8.5.2 HWA HTS Programming Details

For details of programming, see *HWA Thread Scheduler (HTS)* and *VPAC Subsystem*.

7.7.3.8.5.3 HWA Data Transfer Programming Details

For details of programming for PARAM space, see Section *UDMA Specification*.

7.7.3.8.5.4 Initialization Sequence

For information, see Section 7.7.2, *VPAC Subsystem*.

7.7.3.8.5.5 Real-time Operating Requirements

For information, see Section 7.7.2, *VPAC Subsystem*.

7.7.3.8.5.6 Power Up/Down Sequence

For information, see Section 7.7.2, *VPAC Subsystem*.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.9 VISS Edge Enhancer (EE)

This section describes the Edge Enhancer module.

7.7.3.9.1 Edge Enhancer Introduction

The Edge Enhancer module is used to enhance the visual quality of the image by increasing the sharpness of the image. The module only works on Y (Luma) data and uses a combination of 2D HPF (High-Pass Filter) filtering to detect edges and then apply them on the input image.

The Edge Enhancer consists of two separate blocks internally, the edge enhancer filter and the edge sharpener filter. The module can be programmed to either select the output of one of the filters or to use a blend of both.

Figure 7-83 below shows a conceptual block diagram of the edge enhancer filter. The input data is assumed to be a 12 bits/pixel. The EE module is tuned to process 8-bit images (MSB aligned) but has support to process 10-bit and 12-bit images as well.

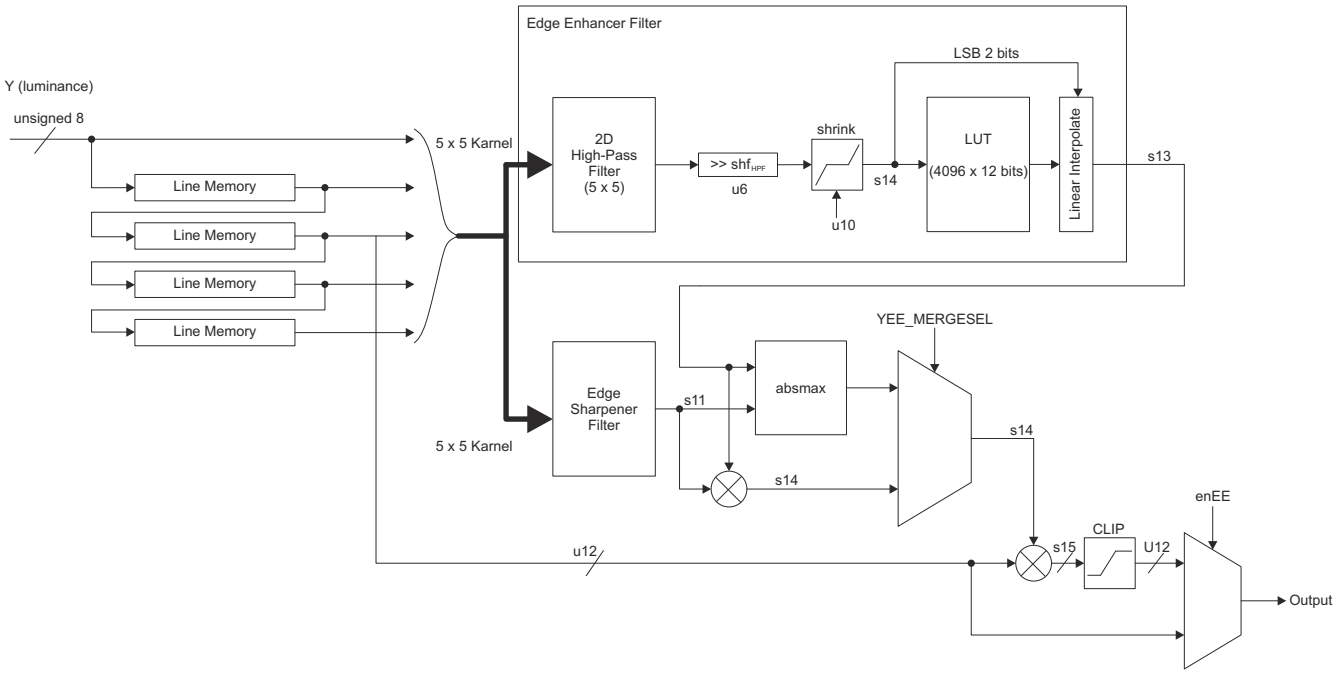


Figure 7-83. Edge Enhancer Block Diagram

7.7.3.9.1.1 Edge Enhancer Filter

The edge enhancer filter works on a 5 × 5 HPF filter (M) with programmable coefficients. The filter is symmetric as such only 3 × 3 coefficients are programmable.

Here, M is a 5 × 5 matrix with programmable coefficients specified by VISS_FCP_EE_YEE_COEF_Ri_Cj (i and j are 0, 1, or 2). The shift value (shf_{HPF}) is specified by VISS_FCP_EE_YEE_SHIFT[5-0] YEE_SHIFT.

Copyright © 2026 Texas Instruments Incorporated

$$HPF(h,v)=\left(\sum_{j=-2}^{2}\sum_{i=-2}^{2}M_{i,j}Y(h+i,v+j)\right)>>shf_{HPF}$$

$$M=\begin{pmatrix} M_{2,2} & M_{1,2} & M_{0,2} & M_{1,2} & M_{2,2} \\ M_{2,1} & M_{1,1} & M_{0,1} & M_{1,1} & M_{2,1} \\ M_{2,0} & M_{1,0} & M_{0,0} & M_{1,0} & M_{2,0} \\ M_{2,1} & M_{1,1} & M_{0,1} & M_{1,1} & M_{2,1} \\ M_{2,2} & M_{1,2} & M_{0,2} & M_{1,2} & M_{2,2} \end{pmatrix}$$

Figure 7-84. Edge Enhancer Linear Filter

The HPF value is shrink by a threshold value, $threshold_{HPF}$ (u6 = 10), specified by VISS_FCP_EE_YEE_E_THR register, and clipped to signed 14 bits to get the index for the LUT. The MSB 12 bits of the index are used for the LUT address (4k locations) whereas the LSB 2 bits acts as weight to the linear interpolation logic.

$$index=\text{clip}(\text{shrink}(HPF,threshold_{HPF}),-8k,8k)$$

$$\text{shrink}(x,threshold)=\begin{cases} x+threshold & x<-threshold \\ 0 & -threshold\le x\le threshold \\ x-threshold & threshold<x \end{cases}$$

$$\text{clip}(x,limit_{LOW},limit_{HIGH})=\begin{cases} -limit_{LOW} & x<-limit_{LOW} \\ x & -limit_{LOW}\le x\le limit_{HIGH} \\ limit_{HIGH} & limit_{HIGH}<x \end{cases}$$

Figure 7-85. Edge Enhancer Indexing

The edge enhancement intensity is looked up from the LUT, where A is the MSB 12 bits of the index.

$$E_{\text{int}}=Interpolate(LUT[A],LUT[A+1])$$

Figure 7-86. Edge Intensity LUT Formula

The 5 × 5 HPF filter is not fully programmable, rather only the top left quadrant is programmable coefficients. The rest of the kernel is symmetric along the y and the x axis.

Figure 7-87 below depicts how the kernels are implemented, with the kernel locations marked inside the boxes. The green color shows the kernels that are programmable.

Copyright © 2026 Texas Instruments Incorporated

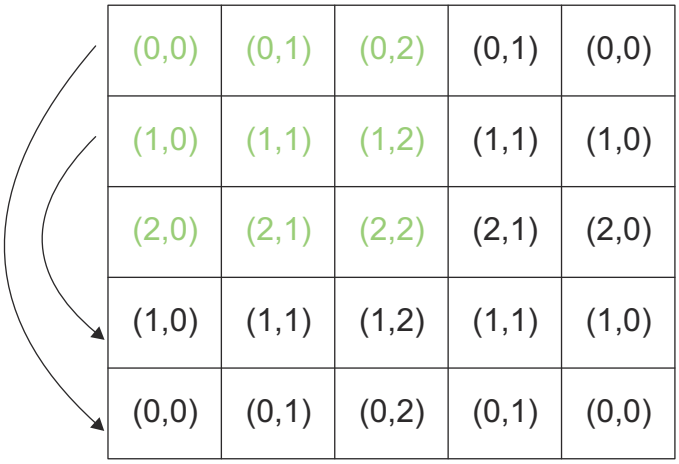


Figure 7-87. 5 × 5 HPF Filter

The border conditions are handled by data duplication on each of the borders when the desired pixel go out of range.

The output of the LUT is the final output of the edge enhancer filter.

7.7.3.9.1.2 Edge Sharpener Filter

Figure 7-88 below shows a high level block diagram of the edge sharpener function.

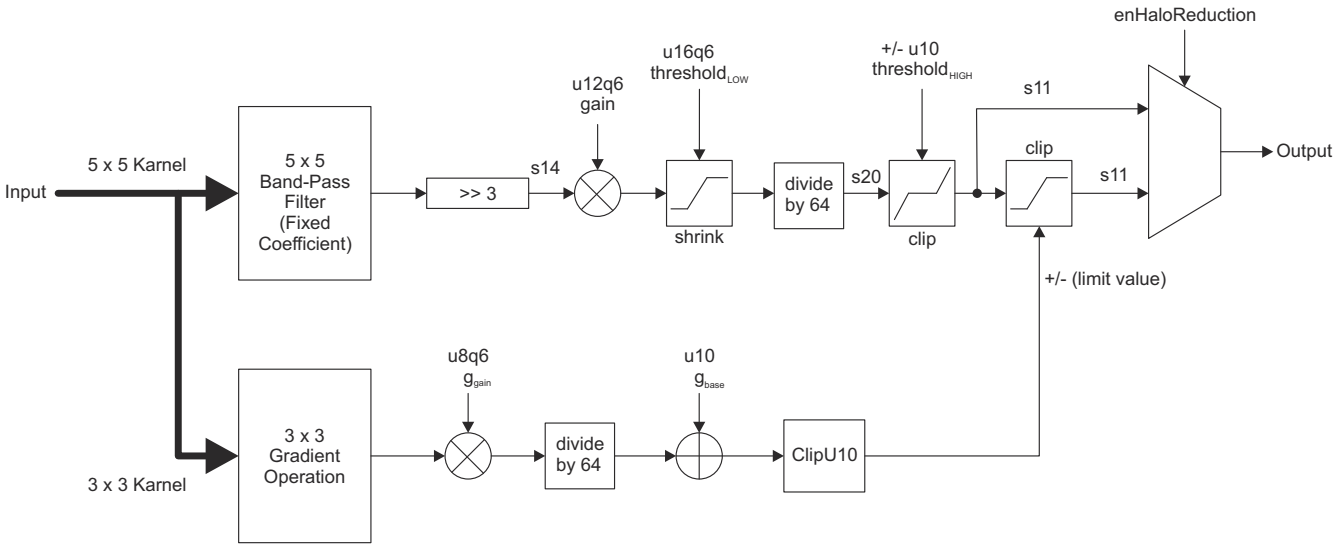


Figure 7-88. Edge Sharpener Function

In edge sharpener filter module, enabled when VISS_FCP_EE_YEE_MERGESEL[0] YEE_MERGESEL = 1, edge clarity is enhanced without producing a halo artifact. In this module, edge intensity is derived by the following 2D linear filter with fixed coefficients shown in Figure 7-89.

Copyright © 2026 Texas Instruments Incorporated

$$S_{i,j} = \begin{pmatrix} 0 & -1 & -2 & -1 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -2 & 2 & 8 & 2 & -2 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & -1 & -2 & -1 & 0 \end{pmatrix}$$

$$sharpness(h,v) = \text{clip}\left(\text{shrink}\left(g \times \left(\left(\sum_{j=-2}^{2} \sum_{i=-2}^{2} S_{i,j} Y(h+i, v+j) \right) >> 3 \right), threshold_{LOW} \right) >> 6, -threshold_{HIGH}, threshold_{HIGH} \right)$$

Figure 7-89. Edge Sharpener Details

The gain (*g*) and threshold values for shrink/clip function (*threshold* $_{LOW}$, *threshold* $_{HIGH}$) are determined by register values (VISS_FCP_EE_YES_E_GAIN, VISS_FCP_EE_YES_E_THR1, VISS_FCP_EE_YES_E_THR2). The precision of *g* is in U12Q6, *threshold* $_{LOW}$ is in U16Q6 and that of *threshold* $_{HIGH}$ is in U10.

This edge intensity is then clipped by a threshold value in the formula shown in Figure 7-90.

$$S_{\text{int}} = \begin{cases} clip(sharpness, -grad, grad) & \text{Halo reduction on} \\ sharpness & \text{Halo reduction off} \end{cases}$$

Figure 7-90. Edge Intensity Clipping Formula

The threshold value (grad) is a function of the activity around the target pixel, which is derived from gradient values. Gain (g_{grad}) and offset (g_{base}) are specified by VISS_FCP_EE_YES_G_GAIN and VISS_FCP_EE_YES_G_OFT.

$$grad = g_{grad}\left(\min(gx_L, gx_R) + \min(gy_T, gy_B)\right) >> 6 + g_{base}$$

$$gx_L = abs\left(\sum_{j=-1}^{1} \sum_{i=-1}^{1} G_{xL}(i,j) \cdot y(h+i, v+j) \right)$$

$$gx_R = abs\left(\sum_{j=-1}^{1} \sum_{i=-1}^{1} G_{xR}(i,j) \cdot y(h+i, v+j) \right)$$

$$gy_T = abs\left(\sum_{j=-1}^{1} \sum_{i=-1}^{1} G_{yT}(i,j) \cdot y(h+i, v+j) \right)$$

$$gy_B = abs\left(\sum_{j=-1}^{1} \sum_{i=-1}^{1} G_{yB}(i,j) \cdot y(h+i, v+j) \right)$$

$$GxL = \frac{1}{4}\begin{pmatrix} 1 & -1 & 0 \\ 2 & -2 & 0 \\ 1 & -1 & 0 \end{pmatrix}$$

$$GxR = \frac{1}{4}\begin{pmatrix} 0 & 1 & -1 \\ 0 & 2 & -2 \\ 0 & 1 & -1 \end{pmatrix}$$

$$GyT = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ -1 & -2 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$GyB = \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ -1 & -2 & -1 \end{pmatrix}$$

Figure 7-91. Edge Threshold Value Formula

Capping with gradient value prevents overly enhancing edges, and suppresses halo artifacts around edges.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.9.1.3 Merge Block

The output from edge enhancer filter and edge sharpener filter are merged with the following function.

$$E_{merge} = \begin{cases} E_{\text{int}} + S_{\text{int}} & VISS_FCP_EE_YEE_MERGESEL[0]YEE_MERGESEL = 0 \\ \text{absmax}(E_{\text{int}}, S_{\text{int}}) & VISS_FCP_EE_YEE_MERGESEL[0]YEE_MERGESEL = 1 \end{cases}$$

$$\text{absmax}(x, y) = \begin{cases} x & abs(y) \le abs(x) \\ y & otherwise \end{cases}$$

Figure 7-92. Edge Enhancer and Sharpener Merger Formula

The E_{merge} value is added to the Y input value to make the final output.

The contribution of each of the filters can be modified. The edge sharpener filter block can be turned off or reduced in strength using the gain fields. The edge enhancer filter block can be disabled using the LUT.

Copyright © 2026 Texas Instruments Incorporated

7.7.3.9.2 Edge Enhancer Programming Model

Table 7-111 captures the programming parameters and the programming model of the Edge Enahncer module.

Note

UMQN implies M bits with N bits of fraction and M-N bits of Integer.

For more information, see *Edge Enhancer Registers*.

Table 7-111. Edge Enhancer Programming Parameters

Parameter Name	New Precision	Tuning Methodology
VISS_FCP_EE_EE_CFG_0[12-0] WIDTH	13 bits	Width of the image
VISS_FCP_EE_EE_CFG_0[28-16] HEIGHT	13 bits	Height of the image
VISS_FCP_EE_EE_ENABLE[0] YEE_ENABLE	1	Enable Module
VISS_FCP_EE_YEE_SHIFT[5-0] YEE_SHIFT	U6	-
VISS_FCP_EE_YEE_COEF_R0/1/2_C0/1/2[9-0] YEE_COEF_R0/1/2_C0/1/2	Signed 10	-
VISS_FCP_EE_YEE_E_THR[9-0] YEE_E_THR	U10	Shrink Threshold before LUT, scale by 16×
VISS_FCP_EE_YEE_MERGESEL[0] YEE_MERGESEL	1 bit	Mergesel: Off(0), On(1)
VISS_FCP_EE_YES_E_HAL[0] YES_E_HAL	1 bit	Halo Reduction; Off(0)/ On(1)
VISS_FCP_EE_YES_G_GAIN[7-0] YES_G_GAIN	U8Q6	Usually 1.5 times YES_E_GAIN
VISS_FCP_EE_YES_G_OFT[9-0] YES_G_OFT	U10	Scale by 16
VISS_FCP_EE_YES_E_GAIN[11-0] YES_E_GAIN	U12Q6	Same as previous
VISS_FCP_EE_YES_E_THR1[15-0] YES_E_THR1	U16Q6	Scale by 16
VISS_FCP_EE_YES_E_THR2[9-0] YES_E_THR2	U10	Scale by 16

Copyright © 2026 Texas Instruments Incorporated

7.7.4 VPAC Lens Distortion Correction (LDC) Module

This section describes the Lens Distortion Correction (LDC) module.

7.7.4.1 LDC Overview

The LDC module deals with lens geometric distortion issues in the camera system. The distortion can be common optical distortions, such as barrel, pin-cushion, or fisheye distortion. LDC is not limited to just these types of distortions. Affine transformation support is also added to support the image and video stabilization application, where multiple images of the same scene need to be aligned. Perspective warp is an extension to affine transform and offers additional capabilities. Perspective transformations can align two images that are captured from different camera viewpoints or locations. This can also be used to rectify the left and right images of a stereo camera to reduce the complexity of a disparity computation. Perspective transformations can also generate new viewpoints.

7.7.4.1.1 LDC Features

- Autonomous memory-to-memory operation
  - Tile based processing
- Generic mesh based distortion model to correct multiple distortion types
- Perspective warp transformation for perspective correction; affine transform is a subset of perspective warp and supports scaling and rotation
- Support multiple input and output YUV data formats:
  - Among the supported data formats are UYVY (YCbCr 422), NV12 (YCbCr 420) and NV21 (YCbCr 420)
- Support up to 8192 x 8192 image dimension
- Pixel Interpolation
  - Bi-cubic interpolation for Y and bilinear interpolation for Cb/Cr
  - Bilinear interpolation mode to offer double throughput
- Performance
  - 1 cyc/pixel (bilinear)
  - 2 cyc/pixel (Bi-cubic)
- Support to process either Luma only or Chroma only modes in YUV420 input data format to save bandwidth
- Support for independent block size in multiple regions (up to 9 regions)
- Dual output channels for programmable output pixel size
- ECC support on Mesh Data internal storage memories and width conversion LUTs on dual output channel
- Interface to HTS module for block level synchronization with other IPs
- Circular addressing support for Output Write (for SL2 interface) and Input Read
- Event for Block/Frame completion and Error scenarios

7.7.4.2 LDC Functional Description

7.7.4.2.1 LDC Block Diagram

Most digital cameras suffer from some degree of nonlinear geometric distortion. A spatial transformation is required to correct the distortion. In automotive applications, cameras use wide angle lenses, including fisheye lenses to provide 180+° field of view. To visually present the scene to the user in an easy-to-consume representation, these distortions need to be corrected

Back mapping gives coordinates of the distorted image as a function of coordinates of the undistorted output image. Correction involves back-mapping each output pixel to a location in the source distorted image, and thus the corrected image is fully populated. As the distorted pixel locations mostly fall onto fractional coordinates, correction involves interpolation.

As shown in Figure 7-93, the LDC consists of a back mapping block, a x/y offset table, image buffer interface, buffer, an interpolation block, and SL2 interface.

Figure 7-93 is a block diagram of the LDC.

Copyright © 2026 Texas Instruments Incorporated

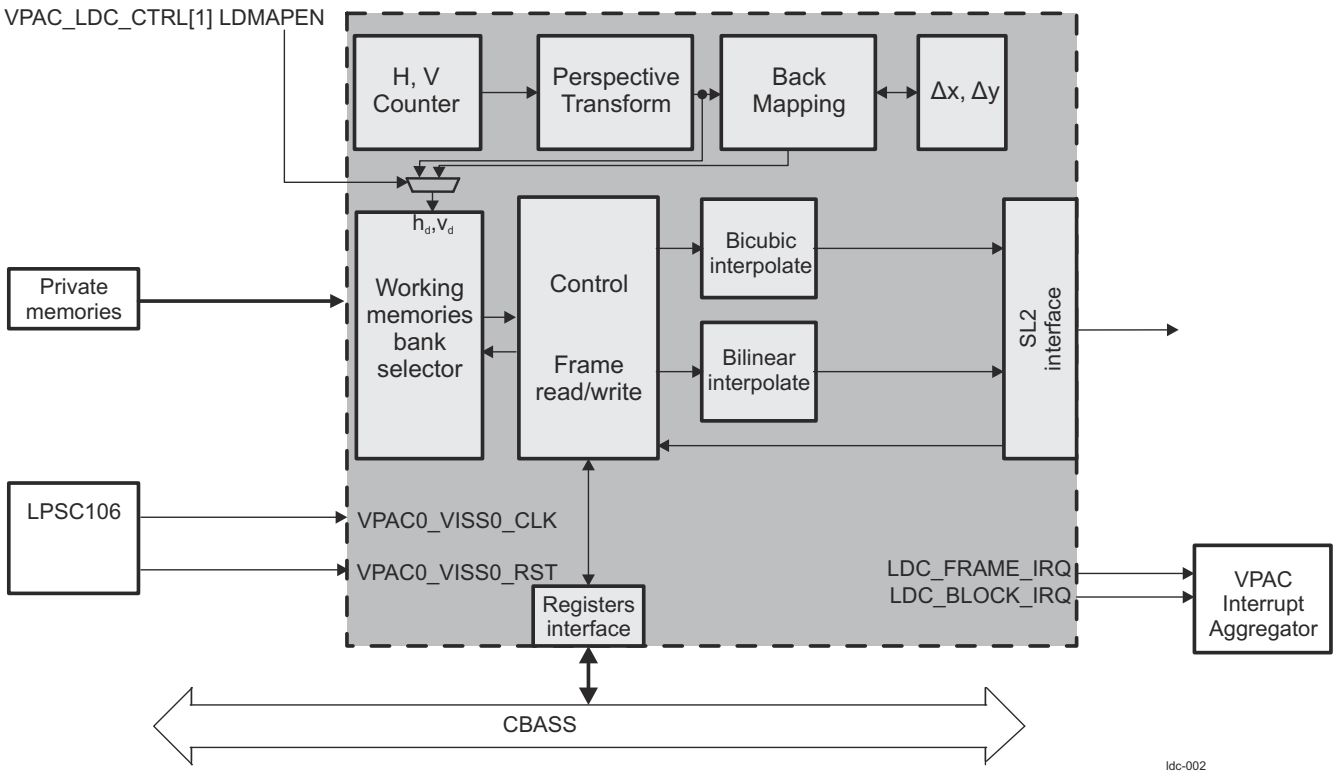


Figure 7-93. LDC Block Diagram

Given the coordinates of the undistorted image, the corresponding coordinates of the distorted image are calculated by combining the output coordinates and the offsets from the offset table. Distorted pixels are read from the image buffer, and buffered for the bilinear interpolation. After the interpolation, corrected image is written back to the SL2 memory.

The LDC processes the image in small two-dimensional (2D) blocks. The software configures appropriate parameters, then initiates the LDC function by writing to an LDC register. The LDC controls the sequencing through 2D blocks, DMA transfers, and computation to process an entire image autonomously. Interrupt, if enabled, is asserted at the completion of the image.

The LDC write is controlled by HTS at block level.

To start the LDC, software must set the LDC_CORE_CTRL[0] LDC_EN bit to 1. The LDC_CORE_CTRL[2] BUSY bit is a status that reflects LDC activity.

7.7.4.2.2 LDC Clocks

Complete VPAC LDC operates on single clock (that is, VPAC0_LDC0_CLK). Except the signals that are coming from LPSC, all other LDC sub-block clock domains are synchronous to VPAC0_LDC0_CLK.

7.7.4.2.3 LDC Interrupts

LDC generates interrupt events and these are made available to the VPAC. VPAC implements compatible interrupt function via the interrupt aggregator, see chapter Section 7.7.2 *VPAC Subsystem*.

All the interrupts are active high and pulsed for one VPAC0_VISS0_CLK cycle. Following interrupts events are generated. Out of these only ECC interrupt should be treated as real interrupt and others are used as events for the interrupt aggregator at VPAC level.

Table 7-112. LDC Interrupt Events

Interrupt Event	Type	Category	Description
PIX_IBLK_OUTOFBOUND	pulse	func	PIX_IBLK_OUTOFBOUND pulse interrupt event

Copyright © 2026 Texas Instruments Incorporated

Table 7-112. LDC Interrupt Events (continued)

Interrupt Event	Type	Category	Description
MESH_IBLK_OUTOFBOUND	pulse	func	MESH_IBLK_OUTOFBOUND pulse interrupt event
IFR_OUTOFBOUND	pulse	func	IFR_OUTOFBOUND pulse interrupt event
INT_SZOVF	pulse	func	INT_SZOVF pulse interrupt event
VPAC_LDC_FR_DONE_EVT	pulse	func	LSE frame done event
VPAC_LDC_SL2_WR_ERR	pulse	func	LSE SL2 VBUSM Write Error interrupt event
PIX_IBLK_MEMOVF	pulse	func	Input pixel block internal memory overflow event
MESH_IBLK_MEMOVF	pulse	func	Input mesh block internal memory overflow event
VPAC_LDC_VBUSM_RD_ERR	pulse	func	Error event on VBUSM Read Interface
UNCORR_PULSE	pulse	error	ECC Aggregator uncorrectable error, pulse
UNCORR_LEVEL	level	error	ECC Aggregator uncorrectable error, level
CORR_PULSE	pulse	error	ECC Aggregator correctable error, pulse
CORR_LEVEL	level	error	ECC Aggregator correctable error, level

7.7.4.2.3.1 LDC Interrupt Events Description

7.7.4.2.3.1.1 PIX_IBLK_OUTOFBOUND

Generated when back mapping of block non-corner pixel co-ordinate goes out of the pre-computed input bounding box (i.e. data required for processing is not available in private pixel storage). This event is generated for the first occurrence of error in each frame. Upon this error, PIX_PAD setting needs to be reviewed. Upon pixel out of bound, pixel co-ordinate will be clipped to the pre-fetched block boundary. Usually no special treatment is required on this error event. Only few pixels are expected to be have this condition..

7.7.4.2.3.1.2 MESH_IBLK_OUTOFBOUND

Generated when mesh data required for non-corner mesh co-ordinate goes out of the pre-computed mesh bounding box (i.e. mesh data required for processing is not available in internal mesh storage). This event is generated for the first occurrence of error in each frame. This error condition is handled inside design by clipping the mesh data location to the pre-fetched block boundary. No special treatment is expected on this error event.

7.7.4.2.3.1.3 IFR_OUTOFBOUND

Generated when back mapped input pixel co-ordinate goes out of valid input frame boundary or co-ordinates post Perspective warping goes out of valid Mesh frame size (Frame size for which mesh data is available). This event is generated for the first occurrence of error in each frame. In the event of error condition, design will clip the co-ordinates to valid frame boundary. It is recommended to review the source of the error to make sure that it is expected.

7.7.4.2.3.1.4 INT_SZOVF

Generated when intermediate variables of affine or perspective transform operation goes above what hardware supports (U16Q3). This event is generated for the first occurrence of error in each frame. In the event of error condition, design will clip the variables the size to what HW supports. Upon this error, Affine and Perspective coefficients need to be reviewed.

7.7.4.2.3.1.5 VPAC_LDC_FR_DONE_EVT

Generated on once complete output frame is written into SL2.

Copyright © 2026 Texas Instruments Incorporated

7.7.4.2.3.1.6 VPAC_LDC_SL2_WR_ERR

This event is generated whenever the LDC SL2 Output VBUSM Interface write status is something other than 0 (success).

7.7.4.2.3.1.7 PIX_IBLK_MEMOVF

Generated when input pixel block memory storage requirement is more than internal pixel memory or each side of input block size goes above 1023. This event will be generated once per block in case of error. In the event of error condition, one or more output pixel blocks corruption. Upon this error, Output Block size setting needs to be reviewed.

7.7.4.2.3.1.8 MESH_IBLK_MEMOVF

Generated when Mesh block internal storage requirement is more than internal Mesh memory or either side of input block size goes above 1010. This event will be generated once per block in case of error. In the event of error condition, one or more output pixel blocks corruption. Upon this error, Output Block size or Affine/Pwarp parameters or mesh sub-sampling factor setting needs to be reviewed.

7.7.4.2.3.1.9 VPAC_LDC_VBUSM_RD_ERR

This event is generated whenever the LDC Read VBUSM Interface status is something other than 0 (success).

7.7.4.2.4 LDC Affine Transform

The LDC first performs an affine transform, which can be used for rotations and scaling, as shown in Figure 7-94. The output of the pixel coordinate is used as the input address to the affine transform.

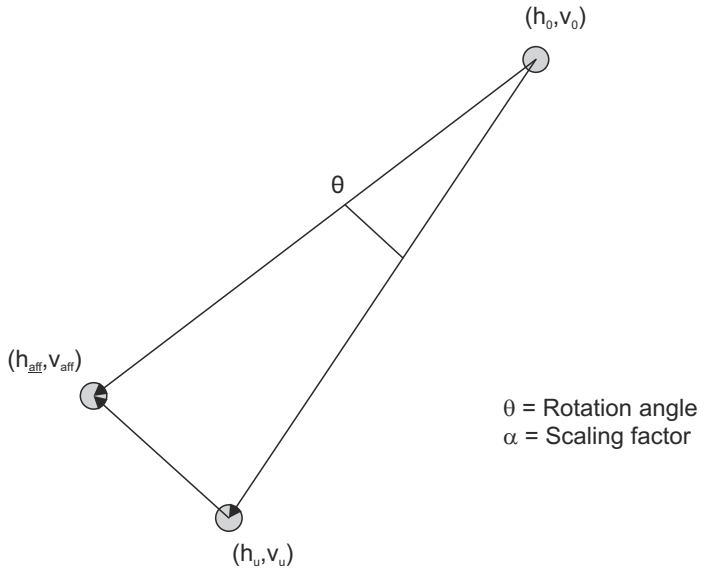


Figure 7-94. LDC Affine Transformation

The mapping from destination coordinate to the source coordinate is expressed as:

$h_{aff} = a \times h_u + b \times v_u + c$

$v_{aff} = d \times h_u + e \times v_u + f$

This transform allows full rotation, expansion, contraction, and translation. The parameters {a, b, d, e} are in S16Q12 (signed number on 14 bits with 2 bits for integer and 12 bits of fraction) format, and parameters {c, f}

Copyright © 2026 Texas Instruments Incorporated

are in S16Q3 format (signed number on 16 bits with 13 bits for integer and 3 bits of fraction). These parameters must be set in following bit fields:

- a = LDC_CORE_AFF_AB[15:0] A
- b = LDC_CORE_AFF_AB[31:16] B
- c = LDC_CORE_AFF_CD[15:0] C
- d = LDC_CORE_AFF_CD[31:16] D
- e = LDC_CORE_AFF_EF[15:0] E
- f = LDC_CORE_AFF_EF[31:16] F

7.7.4.2.5 LDC Perspective Transformation

When a camera is viewing a scene from two different positions or when multiple cameras are viewing the scene from different positions, a transformation between the two viewing angles is needed to align the images. Under specific conditions, the class of geometric transformations known as homography, or planar-perspective transformation, will capture the geometric relationship between the images accurately. Common applications of homography transforms are to align (or stitch) multiple frames of the same scene to compute a panoramic output image. A second application is the alignment of planar surfaces in the world. Finally, perspective transforms are also useful in computing depth maps from a stereo image pair. By rectifying the two views, the search to compute disparity between the two views is simplified to a 1-D search problem. The homography is defined by a 3x3 transformation matrix, as in

$$h_{aff} = a * h_u + b * v_u + c \quad (20)$$

$$v_{aff} = d * h_u + e * v_u + f \quad (21)$$

$$z = \max(0, g * h_u + h * v_u + 1) \quad (22)$$

$$h_p = h_{aff} / z \quad (23)$$

$$v_p = v_{aff} / z. \quad (24)$$

The affine transform is a subset of the perspective transformation. By setting g = h = 0, $h_p = h_{aff}$ and $v_p = v_{aff}$.

In image alignment applications, the homography matrices are computed by locating corresponding points in the two frames and estimating the matrix parameters to transform the set of points in one frame onto the corresponding points in the second frame. In the stereo rectification application, the matrix is determined (pre-computed) at the calibration step and remains fixed.

When LDC is requested to provide a change in the frame size between the input and output, the affine/perspective parameters must be programmed to implement the coordinate scaling (see for programming details). In the following sections, the mesh table is used to program distortion correction. The table sizes are defined based on the total output frame size.

7.7.4.2.6 LDC Lens Distortion Back Mapping

In YCbCr mode, the offset table defines a (x,y) vector for a regular grid of output points. The grid can be fully sampled or down sampled. A fully sampled grid will define an offset vector for every output pixel, defining exactly where to fetch the input data to compute the output pixel. This is the most precise definition and can capture rapidly changing offset tables. The drawback is that it will require a large amount of memory bandwidth as the LDC engine will be reading offset values for every output pixel. Since most offset tables are not expected to change rapidly in a small spatial region, LDC supports reading a subsampled offset table. Offset tables can be subsampled by powers of two in both horizontal and vertical directions and the subsampling factor is set in the register, LDC_CORE_MESHTABLE_CFG[2-0] M. This mode conserves memory bandwidth by reducing the amount of data read to describe the offset vectors, but requires more hardware to interpolate the missing offset vectors. LDC supports bilinear interpolation to interpolate missing offset vectors.

The mapping procedure is described by the following series of equations. Given an output pixel at location, we compute the input pixel location.

Copyright © 2026 Texas Instruments Incorporated

x_c=CLIP(x_o,0,8×W-1), W: Mesh Frame Width (25)

y_c=CLIP(y_o,0,8×H-1), H: Mesh Frame Height (26)

Mask=(1 << (3+M)) -1 (27)

i_{x0} = (x_c >> (3 + M)), f_x=x_c & Mask (28)

i_{y0} = (y_c >> (3 + M)), f_y=y_c & Mask (29)

Clip new locations to previously fetched mesh block boundaries.

i_x=CLIP(i_{x0},block_startmx,block_endmx-1) (30)

i_y=CLIP(i_{y0},block_startmy,block_endmy-1) (31)

Copyright © 2026 Texas Instruments Incorporated

$$\text{Ofst0} = \text{READ32}\left(\text{OfstTable}\left[i_x, i_y\right]\right)$$
$$\Delta x_0 = \text{MSB16}(\text{Ofst0})$$
$$\Delta y_0 = \text{LSB16}(\text{Ofst0})$$

$$\text{Ofst1} = \text{READ32}\left(\text{OfstTable}\left[i_x + 1, i_y\right]\right)$$
$$\Delta x_1 = \text{MSB16}(\text{Ofst1})$$
$$\Delta y_1 = \text{LSB16}(\text{Ofst1})$$

$$\text{Ofst2} = \text{READ32}\left(\text{OfstTable}\left[i_x, i_y + 1\right]\right)$$
$$\Delta x_2 = \text{MSB16}(\text{Ofst2})$$
$$\Delta y_2 = \text{LSB16}(\text{Ofst2})$$

$$\text{Ofst3} = \text{READ32}\left(\text{OfstTable}\left[i_x + 1, i_y + 1\right]\right)$$
$$\Delta x_3 = \text{MSB16}(\text{Ofst3})$$
$$\Delta y_3 = \text{LSB16}(\text{Ofst3})$$
$$T_{x0} = \left(\left(1 \ll (M+3)\right) - f_x\right) \cdot \Delta x_0 + f_x \cdot \Delta x_1$$
$$T_{x1} = \left(\left(1 \ll (M+3)\right) - f_x\right) \cdot \Delta x_2 + f_x \cdot \Delta x_3$$

$$T_{y0} = \left(\left(1 \ll (M+3)\right) - f_x\right) \cdot \Delta y_0 + f_x \cdot \Delta y_1$$
$$T_{y1} = \left(\left(1 \ll (M+3)\right) - f_x\right) \cdot \Delta y_2 + f_x \cdot \Delta y_3$$

$$\Delta x = \left(\left(\left(1 \ll (M+3)\right) - f_y\right) \cdot T_{x0} + f_y \cdot T_{x1} + (1 \ll 2M+5)\right) \gg (2M+6)$$
$$\Delta y = \left(\left(\left(1 \ll (M+3)\right) - f_y\right) \cdot T_{y0} + f_y \cdot T_{y1} + (1 \ll 2M+5)\right) \gg (2M+6)$$

$$x_i = x_o + \Delta x$$
$$y_i = y_o + \Delta y$$

x_o, y_o: U16Q3
x_i, y_i: U16Q3
$\Delta x, \Delta y$: S16Q3

ldc-023

Figure 7-95. LDC Lens Distortion Back Mapping Offset Calculation

Graphically, this procedure is shown in Figure 7-96.

Copyright © 2026 Texas Instruments Incorporated

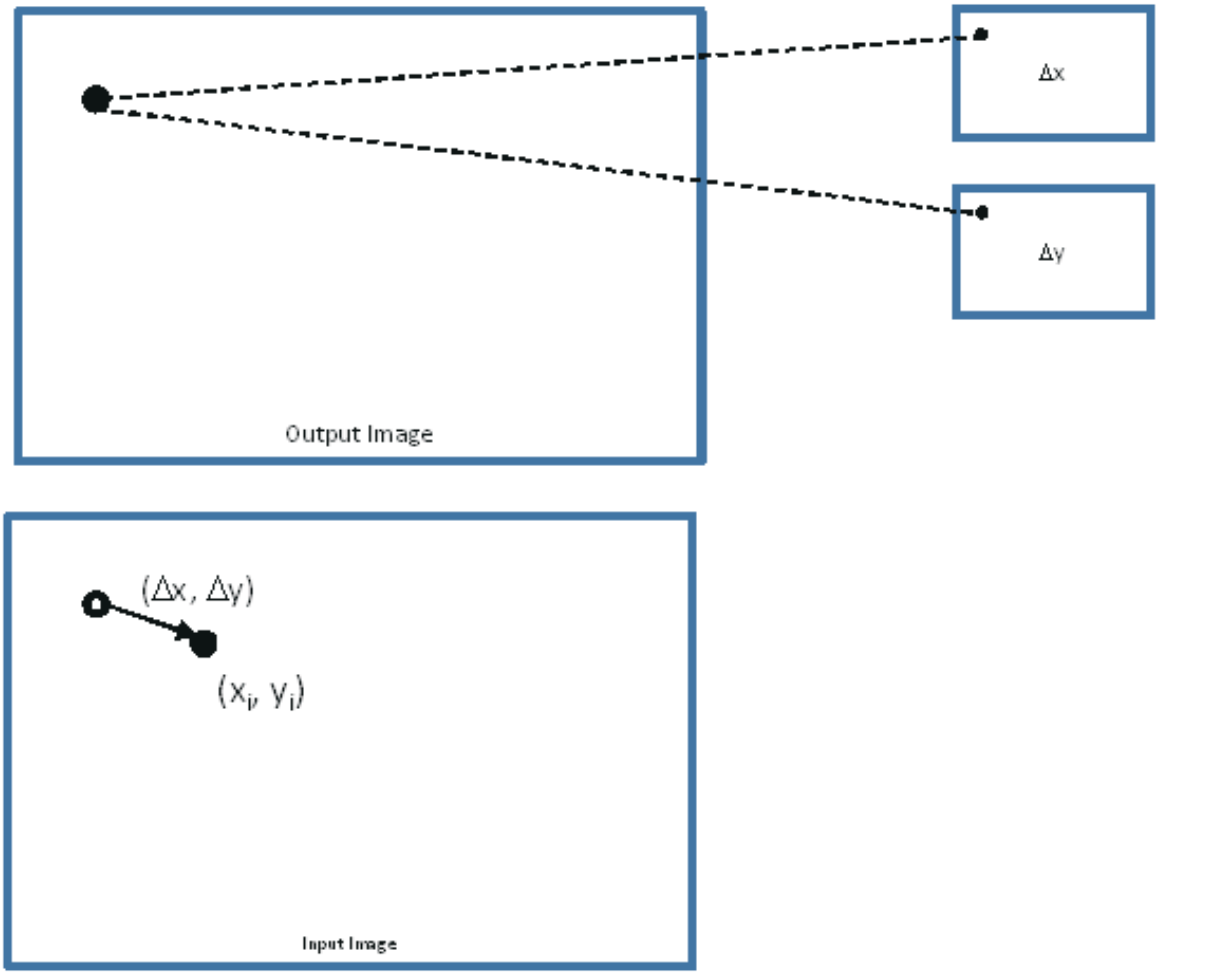


Figure 7-96. LDC Back Mapping Procedure Using Offset Table

The input coordinate (x_i,y_i) refers to final input frame co-ordinates and is used to fetch the appropriate input pixels for interpolation. The interpolation procedure is described in . Before fetching data(x_i,y_i), is clipped to the input frame boundary.

7.7.4.2.6.1 LDC Mesh Table Storage Format

The size of the mesh table is dependent on the Total output frame size. The size must be:

$$\text{Table Width}=\text{CEIL}((\text{Mesh Frame Width})/2^M)+1 \quad (32)$$

$$\text{Table Height}=\text{CEIL}((\text{Mesh Frame Height})/2^M)+1. \quad (33)$$

Mesh frame width and height are specified by registers: LDC_CORE_MESH_FRSZ[13-0] W and LDC_CORE_MESH_FRSZ[29-16] H. The downsampling factor, M, is specified by register LDC_CORE_MESHTABLE_CFG[2] M.

The mesh table should be stored with delta(x) and delta(y) offsets interleaved and stored in raster order. Each offset is S16Q3, so a table entry is 32-bits, including both delta(x) and delta(y). The location of the mesh table is configured by registers LDC_CORE_MESH_BASE_H and LDC_CORE_MESH_BASE_I. The buffer width (>= Table Width), is provided by LDC_CORE_MESH_OFST[15-0] OFST. The mesh table must be aligned on a 16-byte boundary.

Copyright © 2026 Texas Instruments Incorporated

.

7.7.4.2.7 LDC Pixel Interpolation

In UYVY and NV12 mode, the pixel interpolation type can be set to bi-cubic interpolation for best quality or bilinear interpolation for faster performance.

As the coordinates (h_d,v_d) calculated by the back-mapping function are not generally integer values, bi-cubic or bilinear interpolation is applied to the distorted pixels.

Depending on register configuration, bi-cubic or bilinear interpolation is used to interpolate the output Y pixels:

- LDC_CORE_CFG[6] YINT_TYP = 0, bi-cubic interpolation for Y, bilinear for other components
- LDC_CORE_CFG[6] YINT_TYP = 1, bi-linear interpolation for all components

In the case of bi-cubic interpolation, the distorted pixel is interpolated from the 16 Y pixels in the 4 x 4 grid around the distorted location, as shown in Figure 7-97. Bi-cubic interpolation is used first along the horizontal direction, then the vertical direction.

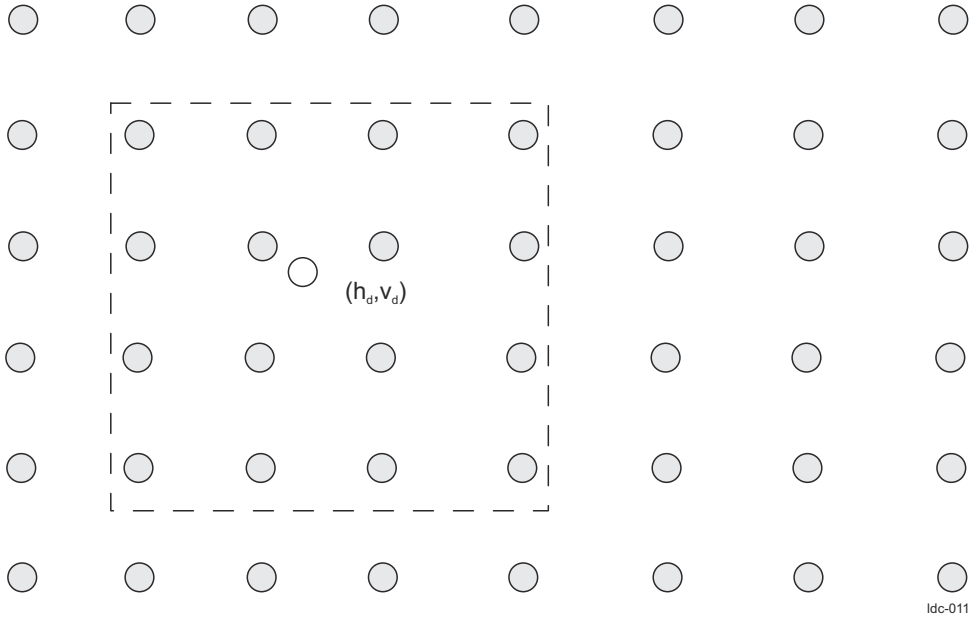


Figure 7-97. LDC Bi-cubic Interpolation for Y

If bi-linear interpolation is selected, the distorted pixel is interpolated from the four Y pixels in the 2 x 2 grid around the distorted location, as shown in Figure 7-98.

Copyright © 2026 Texas Instruments Incorporated

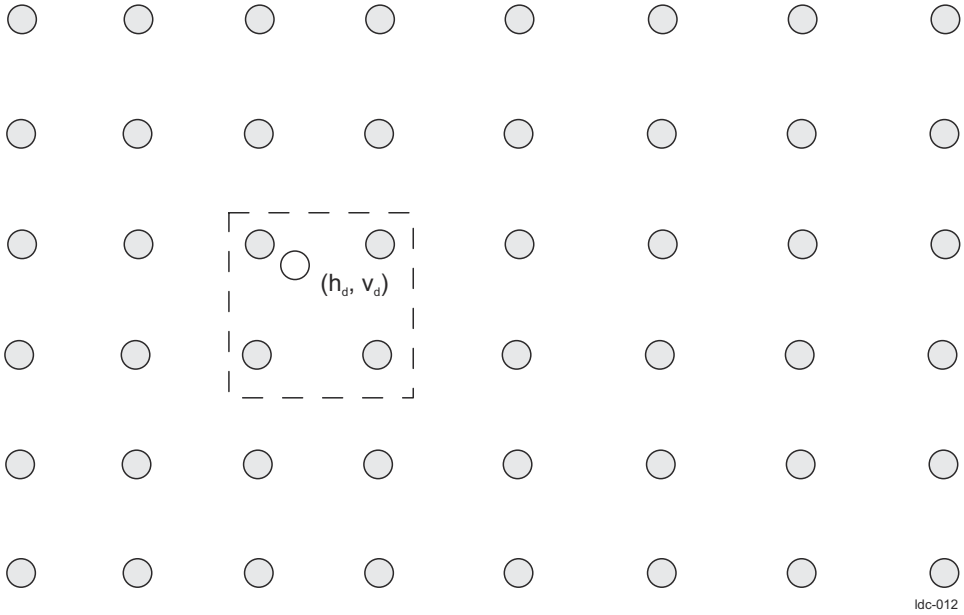


Figure 7-98. LDC Bi-linear Interpolation for Y

For Cb and Cr components, simple bilinear interpolation is used. Each distorted pixel is interpolated from the four same-color pixels on the 2 x 2 grid around the distorted location. For UYVY data, the Cb/Cr grid is not square, but is 2x wider compared to the height. This is shown in Figure 7-99.

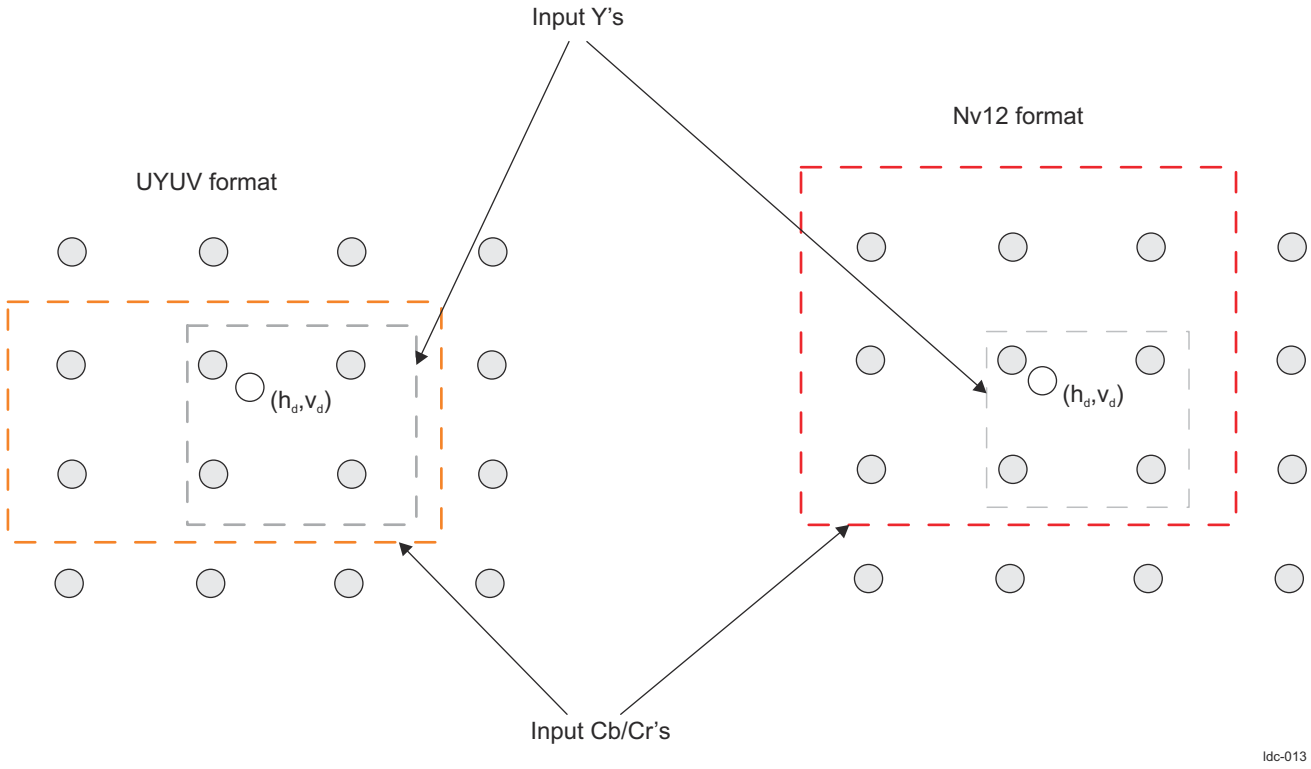


Figure 7-99. LDC Bilinear Interpolation for CB/Cr in UYVY and NV12 Format

Copyright © 2026 Texas Instruments Incorporated

7.7.4.2.8 LDC Buffer Management

Nonlinear nature of the Back Mapping suggests 2-D processing to make more efficient use of external memory bandwidth. Since the mapping computation is non-trivial, the hardware requires the software to provide a number of data management parameters to assist hardware with data transfers.

7.7.4.2.8.1 LDC Buffer Management

Once mesh block fetch is completed, final input co-ordinates are calculated by applying back mapping on previously calculated perspective warp corner pixel co-ordinates. Additional padding is applied on top of these back mapped corner co-ordinates based on the interpolation type.

The LDC processes a block of LDC_CORE_OUT_BLKSZ[7:0] OBW x LDC_CORE_OUT_BLKSZ[15:8] OBH (output block width by output block height) pixels at a time in a raster-scan order throughout the image. OBW is constrained to be a multiple of a base width, depending on the operating mode. This is described in Table 7-113. OBH should be even. The LDC uses four corners of the output block plus some padding to fetch the input block. Due to extreme scenarios, there will be scenarios where above bounding box calculated by hardware doesn't cover all the input data required to generate particular output block. To cover those cases, software needs to supply additional PixelPad, the amount of padding in input block in all directions. Software must supply OBW, OBH, and LDC_CORE_OUT_BLKSZ[19:16] PIXPAD, the amount of padding in input pixels. Software must guarantee that, for EVERY output pixel in the OBW x OBH output block, the input pixels required are bounded by back mapping of the four corners plus/minus the padding. More precisely, the input block is determined by:

- IBX_start = min(truncate(distortx(corner1)), truncate(distortx(corner3))) - HwPad - PixelPad
- IBX_end = max(truncate(distortx(corner2)), truncate(distortx(corner4))) + HwPad + PixelPad
- IBY_start = min(truncate(distorty(corner1)), truncate(distorty(corner2))) - HwPad - PixelPad
- IBY_end = maxn(truncate(distorty(corner3)), truncate(distorty(corner4))) + HwPad + PixelPad

where corner1, corner2, corner3, and corner4 are upper-left, upper-right, lower-left, and lower-right corners of the OBW x OBH output block, and distortx(.), distorty(.) are X and Y distorted coordinates of the corners. Distorted coordinates computed by the back-mapping process described in Section 7.7.4.2.6, *LDC Lens Distortion Back-Mapping*. Figure 7-100 shows the input block bound calculation. Table 7-113 lists the OBW requirements.

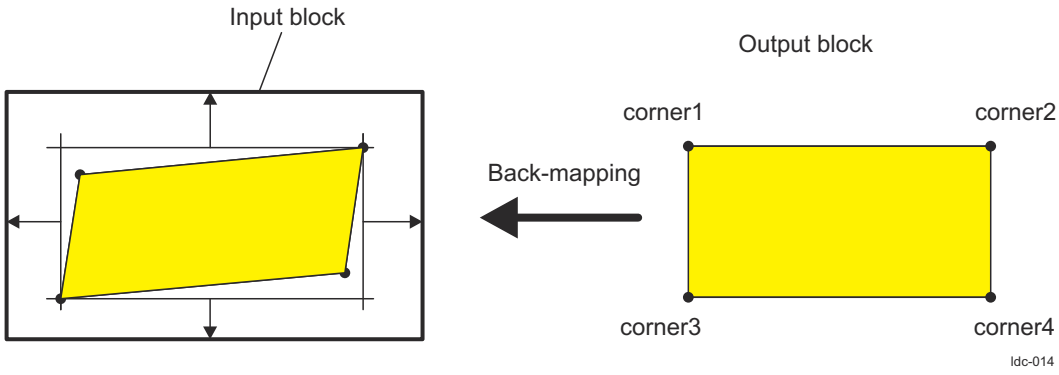


Figure 7-100. LDC Input Block Bound

Table 7-113. LDC OBW Requirements

Data Format	OBW must be a multiple of
YUV422	8
YUV420	8

Typically for geometric distortion, the LDC_BLOCK[19:16] PIXPAD bit field must be 4 to accommodate neighbor sets for all colors. Software must set the PIXPAD bit field correctly.

Copyright © 2026 Texas Instruments Incorporated

OBH and OBW must be large enough for efficient operation of the lens distortion operation. As shown in Table 7-113, OBW is constrained to ensure efficient external memory write. Another constraint is that input block size, (IBX_end – IBX_start + 1) x (IBY_end - IBY_start + 1), for EVERY input block of the image, must fit the allocated input buffer. These parameters must not be specified pessimistically (OBH, OBW too small, or PixelPad too large). Pessimistic parameter setting degrades performance and incurs unnecessary external memory transfer.

The LDC can process a portion of the image, rather than the entire image. This saves time by letting an image process through multiple software/LDC interactions to correct only a portion of the image (See Multi-pass Frame processing).

An intermediate event, *hts_tdone*, is also provided on completion of each macro block output write. This allows the LDC operation to be pipelined with other tasks. The LDC output write stall after this event, waiting for a pulse on the hts_tstart signal to begin writing the next macro block.

7.7.4.2.9 LDC Multi Region with Variable Block size

Distortion in wide angle lenses causes output block size to be very small due to extreme scaling in particular regions of the Image. This could be caused by high field-of-view or viewing position. The range of “Ratio of Output block to input pixel block size” is very high across the frame.

In order to solve this, LDC supports dividing the frame into multiple regions and output block size can be programmed independently for each region. Frame can be divided up to 9 regions. Frame can be divided into maximum of three slices horizontally and maximum of three slices vertically as shown in Figure 7-101. All the regions in a vertical slice share the common horizontal start location and all the regions in a horizontal slice share the common vertical start location. Width of vertical slices can be configured using LDC_CORE_REGN_W12_SZ[13-0] W1, LDC_CORE_REGN_W12_SZ[29-16] W2 and PAC_LDC_REGN_W3_SZ[13-0] W3 register fields. Height of horizontal slices can be configured using LDC_CORE_REGN_H12_SZ[13-0] H1, LDC_CORE_REGN_H12_SZ[29-16] H2 and LDC_CORE_REGN_H3_SZ[13-0] H3 register fields.

Copyright © 2026 Texas Instruments Incorporated

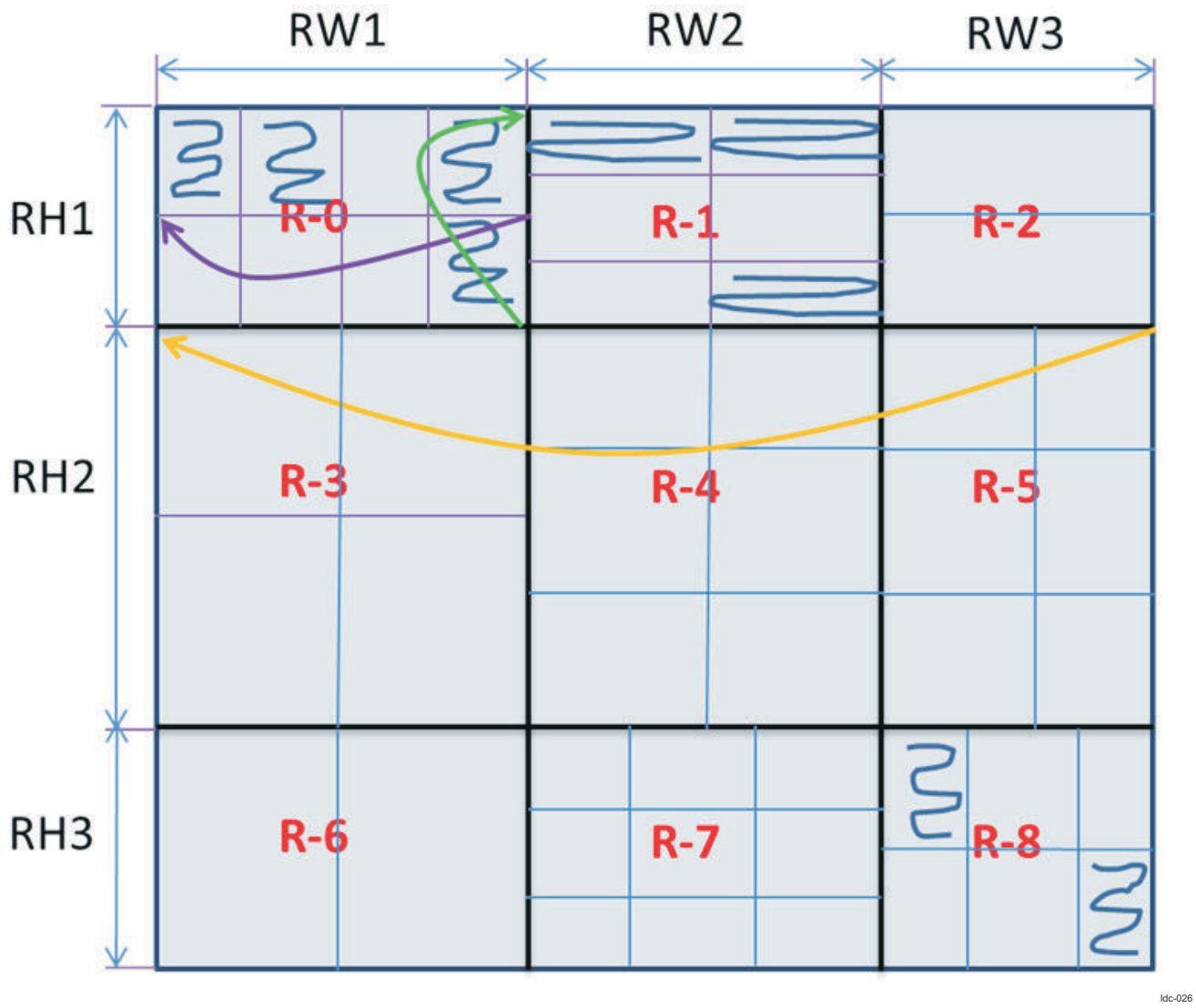


Figure 7-101. LDC Variable Block Size with Multiple Regions

Hardware will process the each region as independent frame, hence processing of the pixels will in 2D sequence with in the region. Up on completing processing of one region, it will process next region in raster direction.

7.7.4.2.9.1 LDC Region Skip Feature

In surround view use case, some regions of the frame won't be used for stitching of final frame. By programming VPAC_LDC_REGION_CTRL_j[0] ENABLE to low, hardware will skip the processing of particular region.

Note

Hardware has to make sure that FRAME_DONE information to LSE is generated on last pixel of last valid region.

As an example shown in Figure 7-102, Region-5, 7 and 8 are skipped for processing. FRAME_DONE is passed on last pixel of Region-6. Ordering and numbering of the regions will remain the same even in the case of skipping some regions.

Copyright © 2026 Texas Instruments Incorporated

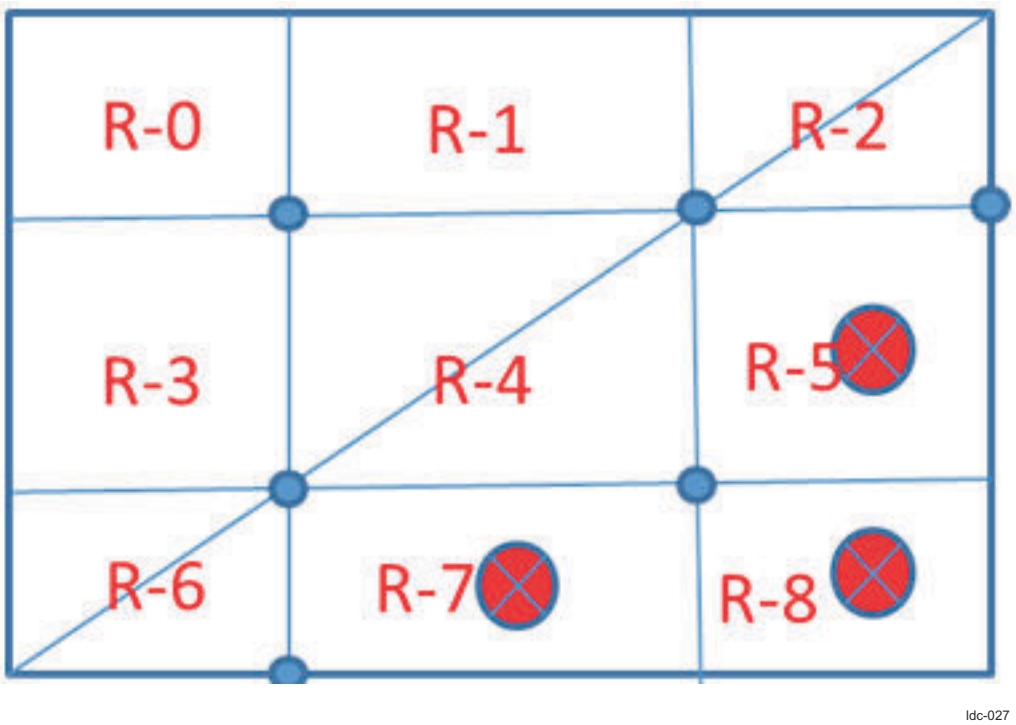


Figure 7-102. LDC Region Skipping Example

7.7.4.2.9.2 LDC Support for sub-set of 3x3 regions

HW supports dividing the frame into sub-set of 3x3 regions. Following combinations of regions are supported.

- 3 vertical slices
  - RW1 = x , RW2 = y and RW3 = z
- 2 vertical slices
  - RW1 = x , RW2 = y and RW3 = 0
  - Last region width has to be zero
- 1 vertical slice
  - RW1 = x , RW2 = 0 and RW3 = 0
  - Last two regions width has to be zero
- 3 horizontal slices
  - RH1 = x , RH2 = y and RH3 = z
- 2 horizontal slices
  - RH1 = x , RH2 = y and RH3 = 0
  - Last region height has to be zero
- 1 horizontal slice
  - RH1 = x , RH2 = 0 and RH3 = 0
  - Last two regions height has to be zero

As shown in Figure 7-103, 2x2 region partitioning can be done by programming RW3=RH3=0.

Note

Region numbering remains as in 3x3 region partitioning with R3, R6-R8 being null regions.

Copyright © 2026 Texas Instruments Incorporated

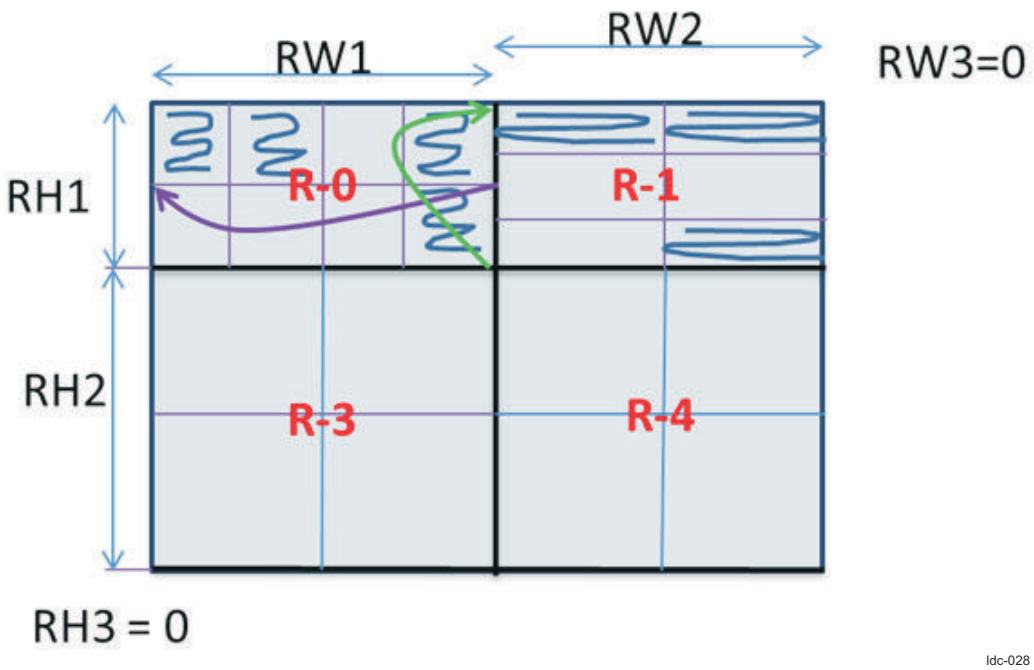


Figure 7-103. LDC 2x2 Region Partitioning Example

7.7.4.2.9.3 LDC Limitations of Multi Region Scheme

Following features can't be supported when multi region processing is enabled.

- Circular buffer support can't be enabled on Input data fetch
- HTS synchronization can't be enabled on Input data fetch
- LDC can't be joined with other HWAs in the VPAC. LDC output needs to be written to external memory.
- HTS can't make use of multiple BPR buffers available in SL2 circular buffer

7.7.4.2.9.4 LDC Multi Region Block Constrains

- RW1+RW2+RW3 must be equal to OFW
- RH1+RH2+RH3 must be equal to OFH
- SW to program the LSE buffer Stride, at-least the size of largest output block width among all regions.
- SW to program buffer height at-least twice the maximum block height among all regions

7.7.4.2.10 LDC Multi-pass Frame processing

The LDC can process a portion of the image, rather than the entire image. This saves time by letting an image process through multiple software/LDC interactions to correct only a portion of the image.

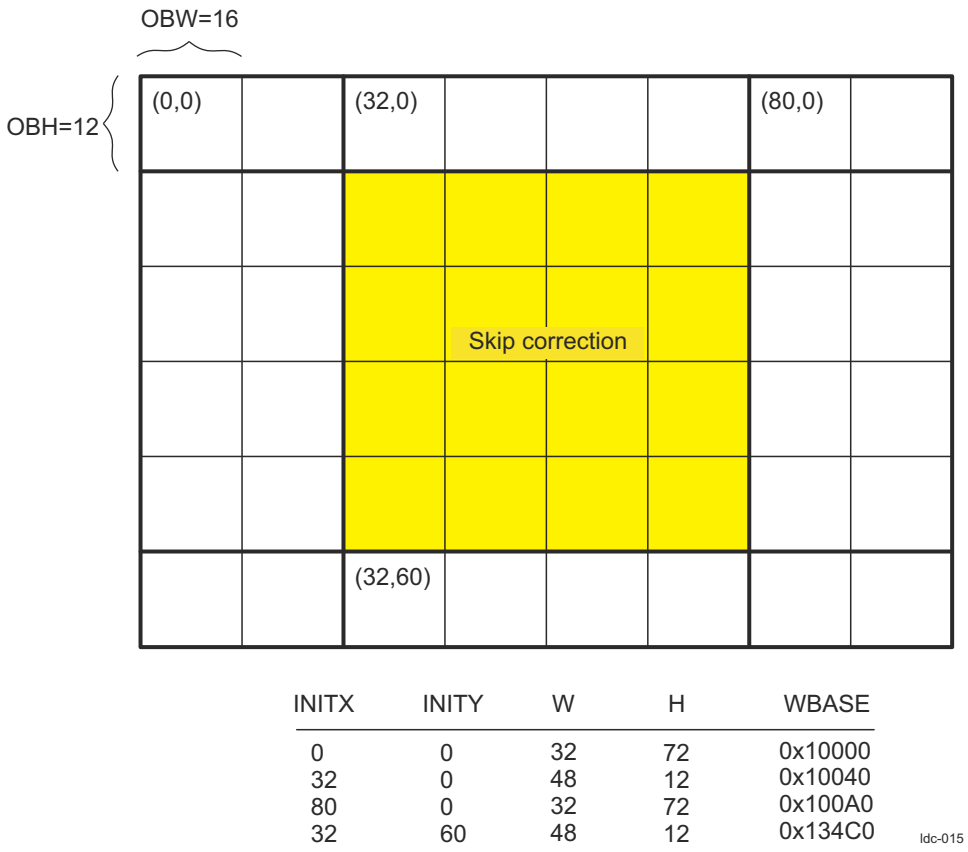
An intermediate interrupt can occur after processing each macro block. This lets the LDC operation be pipelined with other tasks.

To process only a portion of the image, the following parameters are required:

- LDC_CORE_INITXY[12:0] INITX: X coordinate of upper-left corner of output frame
- LDC_CORE_INITXY[28:16] INITY: Y coordinate of upper-left corner of output frame
- LDC_CORE_INPUT_FRSZ[13:0] W: Width of output frame
- LDC_CORE_INPUT_FRSZ[29:16] H: Height of output frame
- FrameBase SDRAM address of upper-left corner of output frame (Y and Cb/Cr base needed in case of YCbCr420)
- FrameOfst SDRAM frame width (in bytes)

Figure 7-104 is an example of multiple-pass processing with middle-of-the-image skipped.

Copyright © 2026 Texas Instruments Incorporated



INITX	INITY	W	H	WBASE
0	0	32	72	0x10000
32	0	48	12	0x10040
80	0	32	72	0x100A0
32	60	48	12	0x134C0

ldc-015

Figure 7-104. LDC Multiple-Pass Correction Example

The LDC does not copy skipped blocks from the input frame to the output frame (necessary task unless output frame = input frame); software must set up and initiate this memory copy.

A note on in-place operation to conserve external memory: Depending on the offset table and configuration parameters, it may not be feasible to point the output image at the input image and have corrected pixels overwriting the source image. Software must determine whether it is feasible to overwrite, and allocate a separate frame buffer when it is not feasible. The LDC does not care if the input and output image pointers coincide, nor does it check any dependency violations.

The starting address of the input frame, corresponding to input coordinate (0, 0), is specified in the LDC_RD_BASE[31:0] RBASE bit field and offset in the LDC_RD_OFST[15:0] ROFST bit field ((Y and Cb/Cr base required for YCbCr4:2:0: LDC_420C_RD_BASE[31:0] RBASE)). The LDC does not clips input block to input frame size if any of the input block falls outside the input frame.

Note

- Any number of input blocks CAN fall partially or totally outside the input frame by assuming coordinates that are outside valid range (negative or exceeding maximum image width defined by the line offset).
- If any output pixel refers to any input pixel outside the input frame, the computed value for that output pixel would be same as neighbor pixel which falls into valid input frame. The other output pixels can still have correct outcomes.

7.7.4.2.11 LDC Input/Output Data Formats

Table 7-114 captures valid number of LDC input and output data mode/format combinations.

Copyright © 2026 Texas Instruments Incorporated

Table 7-114. LDC Supported Data Format Combinations

Input		Output	
Data Mode	Data format	Data Mode	Data format
420 420_Y 420_UV	8-bit	Same as input	8bit
			12-bit - packed
	12-bit - packed		12-bit - packed
	12-bit - unpacked		12-bit - unpacked
			8-bit (on channel[2/3] using LUT)
	12-bit - packed		8-bit (on channel[0/1] with shift) 8-bit (on channel[2/3] using LUT)
422	8-bit	420	8-bit
			12-bit – packed
		422	8-bit

Output data mode will be same as input data mode except when input data is 422, output data can also be converted to 420. In the case of input data format is 8-bit and output data format is 12-bit packed, valid data range is only 8-bit. Output data formatted into 12-bit packed format with 4 MSB bits being all zeros.

The following list contains the register fields used to configure data modes and formats:

- Input data mode - LDC_CORE_CTRL[4-3] IP_DATAMODE
- Input data format - LDC_CORE_CTRL[6-5] IP_DFMT
- Output Luma data format – LDC_CORE_LSE_BUF_CFG_j (j = 0, 2) PIX_FMT_PW
- Output Chroma data format – LDC_CORE_LSE_BUF_CFG_j (j = 1, 3) PIX_FMT_PW

LDC supports YCbCr 422 in the UYVY input data format. The sample grid for YCbCr 422_UYVY is shown in Figure 7-105.

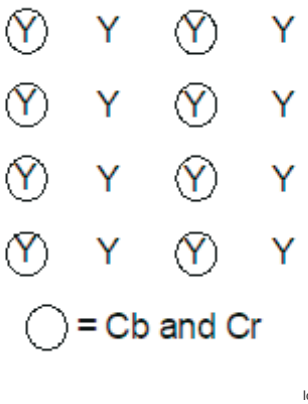


Figure 7-105. LDC Sample Grids for UYVY

UYVY data are stored in 8-bit per color components as shown in Table 7-115.

Table 7-115. LDC UYVY 8-bit Per Color Components

31			0
Y(1)	Cr(0)	Y(0)	Cb(0)

The sample grid for YUV420 NV12 is shown in Figure 7-106.

Copyright © 2026 Texas Instruments Incorporated

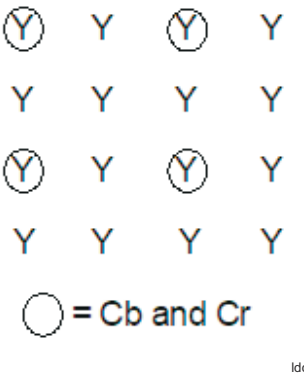


Figure 7-106. LDC Sample Grids for NV12

YCbCr 420 data are stored 8-bit per color components, and in two planes, Y by itself and Cb/Cr interleaved as shown in Table 7-116

Table 7-116. LDC YCbCr 420 data stored 8-bit per color components

31			0
(Y plane) Y(3)	Y(2)	Y(1)	Y(0)
(Cb/Cr plane) Cr(2)	Cb(2)	Cr(0)	Cb(0)

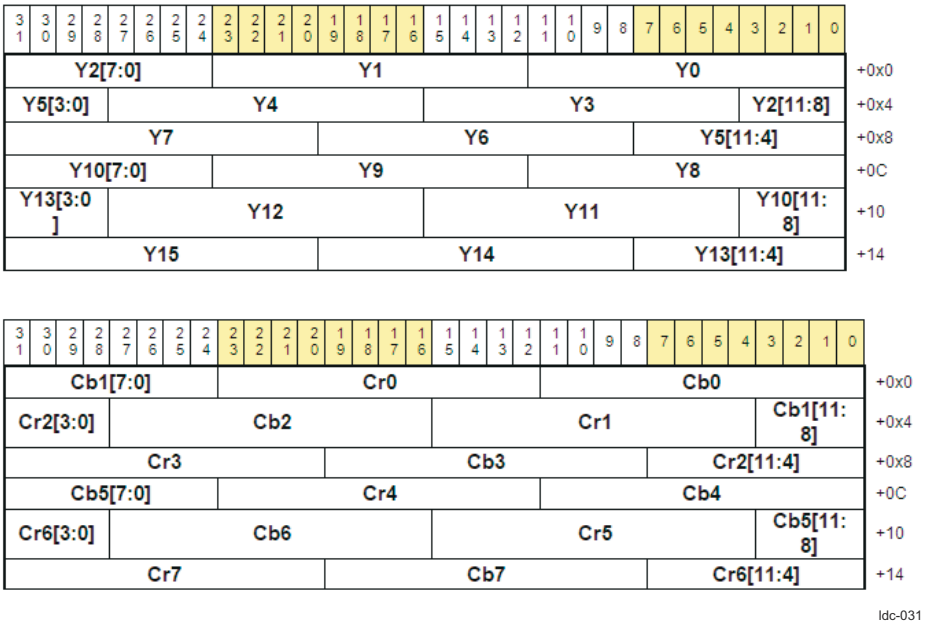


Figure 7-107. LDC YUV420 (2-plane 12-bit Fully Packed Format)

7.7.4.2.12 LDC YUV422 to YUV420 Conversion

Supporting input YUV422 format enables external sensors providing the YUV data directly. YUV422 to YUV420 conversion is done after pixel interpolation and achieved by dropping odd Chroma lines in the block. This retains the Chroma co-sited relation with Luma as at LDC input.

Copyright © 2026 Texas Instruments Incorporated



Figure 7-108. LDC YUV422 to YUV420 Conversion

7.7.4.2.13 LDC SL2 Interface (LSE)

LDC Output write interface is handled by LSE sub-module. LSE (Load Store Engine) module is a common component integrated in VPAC to perform data load and store tasks on the SL2 memory for the HWA algorithm core.

VBUSP interface is be used to transfer the data from LDC Core to LSE.

For more LSE details, see *Load Store Engine (LSE)*.

7.7.4.2.13.1 LDC PSA (Parallel Signature Analysis)

LSE integrates a CRC signature capture mechanism on each output channel data to verify the integrity of the HWA's internal paths. When enabled, every valid pixel data from the CORE is sampled by a CRC module to update the signature for the channel before the data is packed and sent to the SL2 buffer. Note that data used for PSA is raw data from the CORE rather than data written into SL2 with container alignment. For LDC, PSA signature is calculated in the same order as CORE data transfer to LSE which is 3D block order as against frame order. At the end of frame condition, the final signature is saved in a separate read-only signature register (VPAC_LDC_LSE_PSA_SIGNATURE_y y=0..#_of_Output Channel-1) till the next end of frame condition for software to read and compare it against the golden reference signature. The read-only register keeps the last saved value until a reset (set to 0) or is replaced by another frame data signature.

7.7.4.2.14 LDC LUT Mapped Dual Output

LDC produces parallel 8-bit and 12-bit outputs to support surround view and ADAS applications concurrently. LDC produces parallel 8-bit output from 12-bit input using LUT based implementation. This 8-bit output is written into SL2 using LSE output channels (channel[2] for Luma and channel[3] for Chroma). Though use case requires 8-bit output, hardware is designed to be generic which supports any 8-12bit to 8-12bit conversion.

12 or 8 bit generated by data interpolation block is mapped to 8 to 12 bits using a LUT based mapping. The LUT is similar to the LUTs used in VISS and is a linear LUT with 513 entries.

The LUTs are sized to provide 513 locations of data, to enable linear interpolation for all locations. The internal weights are sized as 3 bits since the maximum delta between 2 steps is only 4095/512 = 8. The logic is designed to scale for bit width and can support anything from 8 to 12 bits of input to support different use cases. However, since the step size is different depending on the input bit width, the bit selection and weight calculation logic is designed to account for that. LSE Shift operation can be instead of LUT to reduce bit width from 12 down to 8.

The input bit width (IN_BITDEPTH explained above) is specified using a dedicated register. Output is clipped to programmable clip value ($2^{BitClip\ -1}$). Combination of IN_BITDEPTH, LUT values and Clip value programation allows any 8-12bit to 8-12bit conversion.

Copyright © 2026 Texas Instruments Incorporated

7.7.4.2.15 LDC Band Width Controller

A Bandwidth Limiter is required to limit the mean BW that LDC can request over the VBUSM Read interface. The software sets the maximum allowed bytes per cycle in the register, LDC_CORE_VBUSMR_CFG[27-16] BW_CTRL.

To determine the maximum allowed bytes per cycle, the software needs to know the configuration of LDC, which determines the maximum input block size. Given a specific configuration, the ldc_findMaxInputBuffer utility provided with the functional C model, can be used to give the maximum input block size. The performance requirements determine the maximum number of cycles allowed per block.

7.7.4.2.16 LDC Input Block Fetch Limit

In order to limit the band width hogging by LDC in case of wrong mesh table data, maximum amount of input block pixel data being fetched will be capped at maximum internal memory available for respective data (Note: Luma and Chroma has different size internal buffers).

In YUV420* modes, limit is imposed by Luma data. Pixel data fetch is stopped on detecting the overflow condition and interrupt event is generated. Maximum of 2 requests can be issued upon overflow detection. In case of mesh data, only interrupt event is generated on detecting overflow condition and there is no stalling of fetch. These events are generated for the first overflow case per frame.

7.7.4.2.17 LDC HTS Interface

The HTS interface is used to synchronize between blocks in VPAC Subsystem. In the case of LDC, it controls the start of LDC operation by issuing hts_init. LDC will be waiting for hts_init once it is enabled, so LDC is expected to be enabled before issuing hts_init.

HTS also controls the LDC output write which will indirectly control all LDC operations (that is, fetch and processing). LDC will start writing output pixel data when only it receives hts_tstart for that particular block. Once all output data is written, LDC will generate hts_tdone (also hts_eop if it is last block of the frame).

7.7.4.2.18 LDC VBUSM Read Interface

VBUSM Read interface has following features

- Data Width – 128 Bit
- Address width – 48 Bit
- Maximum of number of Tags – 32
  - Design can be constrained to use lesser number of Tags
- Maximum Command Size: Programmable
  - 32/64/128/256 Bytes
  - Command will split at maximum command size address boundary
- Chanid Encode: chanid[1:0] identify the source/type of the data interface is requesting. Chanid[11:2] is tied low.
  - Chanid[1:0] decoding
    - 00 – Mesh Data.
    - 01 – Luma data in 420 mode/422 data.
    - 10 – Chroma data in 420 mode.

7.7.4.3 LDC Programmers Guide

7.7.4.3.1 LDC Programming Geometric Distortion Mode

Geometric distortions, barrel and pincushion distortion, can be corrected on UYVY and NV12 data formats. The Mesh based back mapping is used in this mode. An additional affine transform can be configured at the same time.

1. Check and wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0).
2. Set LDC_CORE_CTRL[4-3] IP_DATAMODE for UYVY (0) or NV12 data (2). Program LDC_CORE_CTRL[6-5] IP_DFMT to required data storage format.
3. Set LDC_CORE_CTRL[1] LDMAPEN = 1 to enable lens distortion back mapping.

Copyright © 2026 Texas Instruments Incorporated

4. Set the input frame base address in LDC_CORE_RD_BASE_H / LDC_CORE_RD_BASE_L registers. Note that the frame base address must be aligned on a 16-byte boundary. In case of YUV420, YUV422SP, Y1_Y2 and Y1_Y2Y3 modes, configure LDC_CORE_RD_420C_BASE_H / LDC_CORE_RD_420C_BASE_L registers for Chroma Data.
5. Set the input frame line offset in LDC_CORE_RD_OFST[15-0] OFST.
6. If reading the input image from a circular buffer:
   a. Set LDC_CORE_CTRL[9] IP_CIRCEN = 1.
   b. LDC computes the row number and applies modulo operation. The absolute address in the buffer is computed from this wrap-around row and the column number and then data is fetched from the circular buffer. Set the circular buffer size in LDC_CORE_RD_OFST[29-16] MOD register field. In YUV420 modes, the number of rows in the buffer is MOD/2. In others modes, the number of rows in the buffer is MOD for both Y buffer and Cb/Cr buffer.
7. Set the tile size in VPAC_LDC_REGION_OUT_BLKSZ_j[15-8] OBH and VPAC_LDC_REGION_OUT_BLKSZ_j[7:0] OBW. Note the constraints on OBW in Table 7-113.
8. Set the pixel pad in VPAC_LDC_REGION_OUT_BLKSZ_j[19-16] PIXPAD.
9. Set the input frame size in LDC_CORE_INPUT_FRSZ[29-16] H and LDC_CORE_INPUT_FRSZ[13-0] W.
10. Set the output frame size in LDC_CORE_MESH_FRSZ[29-16] H and LDC_CORE_MESH_FRSZ[13:0] W. Mesh data has to be present for entire frame after transform.
11. Set the output compute frame size in LDC_CORE_COMPUTE_FRSZ[29-16] H and LDC_CORE_COMPUTE_FRSZ[13-0] W.
12. Set the starting output point in LDC_CORE_INITXY[12-0] INITX and LDC_CORE_INITXY[28-16] INITY.
13. Configure SL2 Interface programming registers. Details of LSE configuration is captured in Section 7.7.4.3.4.
14. Set the mesh offset table pointer to the correct address (LDC_CORE_MESH_BASE_H / LDC_CORE_MESH_BASE_I[31-0] ADDR and LDC_CORE_MESH_OFST[15-0] OFST). Set the table down sampling factor for MxM down sampling in LDC_CORE_MESHTABLE_CFG[2-0] M.
15. Set the Y plane interpolation type to bilinear or bi-cubic in LDC_CORE_CFG[6] YINT_TYP.
16. If also using affine transform, set the six affine transform parameters in LDC_CORE_AFF_AB.A, LDC_CORE_AFF_AB.B, LDC_CORE_AFF_CD.C, LDC_CORE_AFF_CD.D, LDC_CORE_AFF_EF.E, and LDC_CORE_AFF_EF.F. If affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, and F = 0.
17. If also using perspective transform, enable LDC_CORE_CTRL[7] PWARPEN and set LDC_CORE_PWARP_GH[15-0] G and LDC_CORE_PWARP_GH[31-16] H. If perspective transform is not used, disable LDC_CORE_CTRL[7] PWARPEN and set LDC_CORE_PWARP_GH[15-0] G = 0 and LDC_CORE_PWARP_GH[31-16] H = 0.
18. Set LDC_CORE_CTRL[0] LDC_EN = 1 to start the LDC operation.
19. Start the HTS init sequencing. LDC fetch/processing is gated with hts_init.
20. Wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

7.7.4.3.2 LDC Programming Rotational Video Stabilization (Affine Transformation)

In the rotational video stabilization use case, consecutive frames capturing the same scene are aligned to minimize camera shake. Camera motions cause the frames to have translation or rotation differences. Thus, the frames need to be aligned to maintain a stable output video. The affine transform offers control to account for scaling, rotation, and translation transforms. For example, the rotation use case can be parameterized with the following affine transform parameters:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} h_o - h_o \cos(\theta) - v_o \sin(\theta) \\ v_o - v_o \cos(\theta) - h_o \sin(\theta) \end{bmatrix}$$

ldc-035

Copyright © 2026 Texas Instruments Incorporated

where theta is the angle of rotation. In the scaling case, the affine transform uses:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} h_o(1-\alpha) \\ v_o(1-\alpha) \end{bmatrix}$$

ldc-036

where alpha is the scale factor. Translation uses an identity matrix along with the translation vector as the affine transform parameters, as in:

$$\begin{bmatrix} h_{aff} \\ v_{aff} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_d \\ v_d \end{bmatrix} + \begin{bmatrix} T_h \\ T_v \end{bmatrix}$$

ldc-037

where Th and Tv are the translation parameters. More complex transforms can be derived and the affine transformation is general so that optimization routines may be used to provide the best parameters to align two frames. In all affine transforms, the perspective warp parameters, g and h, should be set to 0.

The following steps need to be performed by the application to initialize and run LDC to perform affine transform.

1. Check and wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0).
2. Set LDC_CORE_CTRL[4-3] IP_DATAMODE and LDC_CORE_CTRL[6-5] IP_DFMT to required data mode and format.
3. If lens distortion correction is performed at the same time as the affine transform, set LDC_CORE_CTRL[1] LDMAPEN = 1. Otherwise, set LDC_CORE_CTRL[1] LDMAPEN = 0.
4. Set the input frame base address in LDC_CORE_RD_BASE_H / LDC_CORE_RD_BASE_I. Note that the frame base address must be aligned on a 16-byte boundary.
5. Set the input frame line offset in LDC_CORE_RD_OFST[15-0] OFST.
6. If reading the input image from a circular buffer:
   a. Set LDC_CORE_CTRL[9] IP_CIRCEN = 1.
   b. If using row address, LDC computes the row number and applies a modulo operation. The absolute address in the buffer is computed from this wrap-around row and the column number and then data is fetched from the circular buffer. Set the circular buffer size in LDC_CORE_RD_OFST[29-16] MOD. In YUV420 modes, the number of rows in the Y buffer is MOD and the number of rows in the Cb/Cr buffers is MOD/2. In others modes, the number of rows in the buffer is MOD for both Y buffer and Cb/Cr buffer.
7. Set the tile size in LDC_CORE_OUT_BLKSZ[15-8] OBH and LDC_CORE_OUT_BLKSZ[7-0] OBW. Note the constraints on OBW in Table 7-113.
8. Set the pixel pad in LDC_CORE_OUT_BLKSZ[19-16] PIXPAD.
9. Set the input frame size in LDC_CORE_INPUT_FRSZ[29-16] H and LDC_CORE_INPUT_FRSZ[13-0] W.
10. Set the output frame size in LDC_CORE_MESH_FRSZ[29-16] H and LDC_CORE_MESH_FRSZ[13-0] W. Mesh data has to be present for entire frame after transform.
11. Set the output compute frame size in LDC_CORE_COMPUTE_FRSZ[29-16] H and LDC_CORE_COMPUTE_FRSZ[13-0] W.
12. Set the starting output point in LDC_CORE_INITXY[12-0] INITX and LDC_CORE_INITXY[28-16] INITY.
13. Configure SL2 Interface programming registers. Details of LSE configuration is captured in Section 7.7.4.3.4.
14. If the data format is NV12, set the 420 UV input plane base address in LDC_CORE_RD_420C_BASE_H / LDC_CORE_RD_420C_BASE_I. This address must be 16-byte aligned.
15. Set the mesh offset table pointer to the correct address (LDC_CORE_MESH_BASE_H / LDC_CORE_MESH_BASE_I and LDC_CORE_MESH_OFST. Set the table down sampling factor for MxM down sampling in MESHTABLE_CFG.M.
16. Set the Y plane interpolation type to bilinear or bicubic in LDC_CORE_CFG[6] YINT_TYP.

Copyright © 2026 Texas Instruments Incorporated

17. Set the six affine transform parameters in LDC_CORE_AFF_AB.A, LDC_CORE_AFF_AB.B, LDC_CORE_AFF_CD.C, LDC_CORE_AFF_CD.D, LDC_CORE_AFF_EF.E, and LDC_CORE_AFF_EF.F. If affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, and F = 0.
18. Disable perspective warp transform with LDC_CORE_CTRL[7] PWARPEN = 0. Set the two perspective transform parameters LDC_CORE_PWARP_GH[15-0] G = 0 and LDC_CORE_PWARP_GH[31-16] H = 0.
19. Set LDC_CORE_CTRL[0] LDC_EN = 1 to start the LDC operation.
20. Start the HTS init sequencing. LDC fetch/processing is gated with hts_init.
21. Wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

7.7.4.3.3 LDC Programming Perspective Transformation

Perspective transforms are used to align the image data of two different cameras viewing the same scene from different positions. It can also be used in the case of stereo image rectification to align the epipolar lines of the left/right image pair with their scan lines.

The following steps need to be performed by the application to initialize and run LDC to perform perspective transform.

1. Check and wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0).
2. Set LDC_CORE_CTRL[4-3] IP_DATAMODE and LDC_CORE_CTRL[6-5] IP_DFMT to required data mode and format.
3. If lens distortion correction is performed at the same time as the perspective transform, set LDC_CORE_CTRL[1] LDMAPEN = 1. Otherwise, set LDMAPEN = 0.
4. Set the input frame base address in LDC_CORE_RD_BASE_H / LDC_CORE_RD_BASE_I. Note that the frame base address must be aligned on a 16-byte boundary.
5. Set the input frame line offset in LDC_CORE_RD_OFST[15-0] OFST.
6. Set the tile size in LDC_CORE_OUT_BLKSZ[15-8] OBH and LDC_CORE_OUT_BLKSZ[7-0] OBW. Note the constraints on OBW in Table 7-113.
7. Set the pixel pad in LDC_CORE_OUT_BLKSZ[19-16] PIXPAD.
8. Set the input frame size in LDC_CORE_INPUT_FRSZ[29-16] H and LDC_CORE_INPUT_FRSZ[13-0] W.
9. Set the output frame size in LDC_CORE_MESH_FRSZ[29-16] H and LDC_CORE_MESH_FRSZ[13-0] W. Mesh data has to be present for entire frame after transform.
10. Set the output compute frame size in LDC_CORE_COMPUTE_FRSZ[29-16] H and LDC_CORE_COMPUTE_FRSZ[13-0] W.
11. Set the starting output point in LDC_CORE_INITXY[12-0] INITX and LDC_CORE_INITXY[28-16] INITY.
12. Configure SL2 Interface programming registers. Details of LSE configuration is captured in Section 7.7.4.3.4.
13. If the data format is NV12, set the 420 UV input plane base address in LDC_CORE_RD_420C_BASE_H / LDC_CORE_RD_420C_BASE_I. This address must be 16-byte aligned.
14. Set the mesh offset table pointer to the correct address (LDC_CORE_MESH_BASE_H / LDC_CORE_MESH_BASE_I and LDC_CORE_MESH_OFST. Set the table down sampling factor for MxM down sampling in MESHTABLE_CFG.M.
15. Set the Y plane interpolation type to bilinear or bicubic in LDC_CORE_CFG[6] YINT_TYP.
16. Set the eight perspective transform parameters in LDC_CORE_AFF_AB.A, LDC_CORE_AFF_AB.B, LDC_CORE_AFF_CD.C, LDC_CORE_AFF_CD.D, LDC_CORE_AFF_EF.E, and LDC_CORE_AFF_EF.F. If affine transform is not used, then these need to be set to the following values: A = 4096, B = 0, C = 0, D = 0, E = 4096, F = 0,G = 0 and H = 0. Enable LDC_CORE_CTRL[7] PWARPEN = 1.
17. Set LDC_CORE_CTRL[0] LDC_EN = 1 to start the LDC operation.
18. Start the HTS init sequencing. LDC fetch/processing is gated with hts_init.
19. Wait for LDC_CORE_CTRL[2] BUSY to become IDLE (0) or wait for the end of frame completion interrupt.

7.7.4.3.4 LDC Programming LSE

Following provides the LSE general programming guideline.

It also captures an additional secondary specific use case configuration values for some registers.

- Frame Size = 1920x1080

Copyright © 2026 Texas Instruments Incorporated

- Output Block size = 64x32
- Luma data being consumed by MSC and Chroma is written out by UDMA
  - Block to line conversion done for Luma data via SL2.
  - Chroma data is written out at block level.
- Circular buffer size just to do ping-pong between LDC – MSC(Y) and LDC – UDMA(C)
- 12-bit pixel data

1. Pixel Data Format
   a. LDC_CORE_LSE_BUF_CFG_j (PIX_FMT_PW, PIX_FMT_CNTRSZ, and PIX_FMT_ALIGN) parameters define the pixel output data format for this output channel.
   b. Common data formats:
      i. Fully packed 12-bit : PIX_FMT_PW=1, PIX_FMT_CNTRSZ=1, PIX_FMT_ALIGN=0
      ii. 12-bit unpacked (MSB aligned) : PIX_FMT_PW = 1, PIX_FMT_CNTRSZ=2, PIX_FMT_ALIGN=1

2. Output SL2 Circular Buffer Configuration
   a. LDC_CORE_LSE_BUF_ATTR0_j (buf_stride) – define the line stride size (must be 64 byte multiple)
      i. LDC_CORE_LSE_BUF_ATTR0_j[15-6] BUF_STRIDE = 2880 (1920 * 1.5)
      ii. LDC_CORE_LSE_BUF_ATTR0_j[15-6] BUF_STRIDE = 96 (64 * 1.5)
   b. LDC_CORE_LSE_BUF_ATTR0_j (cbuf_size) – define the size of the circular buffer in the SL2 (number of lines)
      i. LDC_CORE_LSE_BUF_ATTR0_j[24-16] CBUF_SIZE = 68 (2*OBH+4)
      ii. LDC_CORE_LSE_BUF_ATTR0_j[24-16] CBUF_SIZE = 64 (2*OBH)

3. Output SL2 Circular Buffer 2D mode Configuration (applicable for LDC only - TBD)
   a. LDC_CORE_LSE_BUF_ATTR1_j (buf_blk_width) – defines the number of bytes equivalent to OBW (output block width) pixels.
      i. LDC_CORE_LSE_BUF_ATTR1_j.buf_blk_width = 96 (64*1.5)
   b. LDC_CORE_LSE_BUF_ATTR1_j (cbuf_bpr) – defines the number of 2D blocks per row in the SL2 circular buffer.
      i. LDC_CORE_LSE_BUF_ATTR1_j[25-16] CBUF_BPR_CHAN = 30
      ii. LDC_CORE_LSE_BUF_ATTR1_j[25-16] CBUF_BPR_CHAN = 1

4. Base Address and Channel Enable:
   a. LDC_CORE_LSE_BUF_BA_j[23-6] ADDR – defines the SL2 base address of the Circular buffer for this output channel.
   b. LDC_CORE_LSE_BUF_BA_j[31] ENABLE – enables this output channel.

7.7.4.3.5 LDC Programming Restrictions and Special Cases

The following list of LDC programming restrictions should be considered:

1. LDC_CORE_LSE_BUF_ATTR1_j[25-16] CBUF_BPR_CHAN supports 1023 number of block per row. Hence, with output block width of 8, maximum line size supported in SL2 memory is 8184 pixels.
2. In 422 mode, always program as data format as 8-bit (design does not select data format automatically based on mode)
3. Due to hardware limitations, make sure that input block size (height and width) does not cross 1023 (pixels/lines).
   - The hardware generates either PIX_IBLK_MEMOVF or MESH_IBLK_MEMOVF events in this case.
4. LDC_CORE_CTRL[0] LDC_EN should be high before issuing hts_init.

Copyright © 2026 Texas Instruments Incorporated

7.7.5 VPAC Multi-Scaler (MSC)

7.7.5.1 MSC Overview

VPAC_MSC (VPAC Multi-Scaler, hereinafter simply referred to also as MSC) is a Hardware Acceleration (HWA) module in Vision Pre-processing Accelator (VPAC) . MSC supports two sets of multi-output scaling filters to accelerate generation of pyramid (Octave) and intra-octave scales required by Optical Flow processing and DSP vision algorithms.

7.7.5.1.1 MSC Features

- 10 scaled outputs independently mapped to 1 or 2 processing threads (any number up to 10 scaler outputs can be mapped to one thread, while the remaining scaler outputs can be mapped to the other thread)
- Programmable scaling engine:
  - Pyramid or intra-octave scale generation
  - 1x ~ x/4 image downscaling
  - 1x generic separable convolution filtering
  - On-the-fly (OTF) or memory-to-memory (M2M) operation with line buffers in SL2 (thus, frame width not constrained by internal line buffer width).
- Polyphase filters:
  - 5-tap separable filter kernel structure
  - Programmable kernel sizes for vertical/horizontal filter (3, 4, or 5)
  - Polyphase implementation with support of 64 or 32 phases
  - Independent programming of vertical/horizontal filters with initial phase programming option
- Filter Coefficients (shared by all scalers):
  - Coefficients in signed 10-bit number (typically 1-bit integer with 8 fractional bits)
  - Programmable Coefficient shift size to set precision of coefficient number (shift size represents the fractional bit width)
  - 2 dedicated sets of 5-tap single phase filter coefficients (for example, Gaussian) for Pyramid (Octave) generation or custom convolution filtering.
  - 4 sets of 5-tap x 32 phase coefficients
    - Two can be combined to support 5-tap x 64-phase filters
    - First can be configured to hold additional 5-tap single-phase convolution filters for non-resizing application.
- Input/output format support:
  - Each thread can support the following input source formats:
    - One plane (uniform or interleaved format - Y, CbCr interleaved, R)
  - Each Input channel supports 8/12-bit pixel data in 8/12/16-bit pixel container with programmable LSB/MSB alignment
  - Each Output channel supports 8/12-bit pixel data in 8/12/16-bit pixel container with programmable LSB/MSB alignment
- Performance:
  - 1 cycle/pixel per plane
  - Input line skip for thread performing only pyramid-generation (50% less processing time)
- Other features:
  - Programmable Input and Output Regio of Interest (ROI) (or clipping) for each scaler
  - Support for Vertical and Horizontal edge Padding (replication) for both luma/chroma data plane inputs
- Functional Safety Support: CRC frame data signature capture on all output channels for fault detection in the data path.

7.7.5.1.2 MSC Not Supported Features

- More than 2 input threads or 10 outputs
- Block based operation for scalar
- Multi-component (RGB) Scaling
- Error checks for out of bound parameters

Copyright © 2026 Texas Instruments Incorporated

7.7.5.2 MSC Functional Description

7.7.5.2.1 MSC Functional Overview

VPAC_MSC consists of 10 programmable resizers performing multi-thread/multi-scaling operations (1-input to N-outputs and 1-input to M-outputs where N+M is 10 or less). Each processing thread of MSC HWA reads its input plane data from a Shared Memory buffers (SL2) circular line buffers, performs multi-scaling operations (ratios between X and 0.25X) on the selected thread channel input, and writes out results to SL2 circular line buffers. In case of OTF operation, the source data is generated from another HWA. In case of M2M operation, the source data is read from the DDR memory. Data transfers from/to SL2 to/from external memory (or another HWA) are handled by VPAC level DMA controller with transfer request events coming from VPAC top level HWA Thread Scheduler (HTS), see *HWA Threads Scheduler (HTS)*.

A typical configuration of the MSC module uses one input plane data to generate 7 intra-octave scales and the next octave image. Multi-pass processing is then utilized to generate up to next 6 or 7 sets of scales/octaves. Since the resolution after each octave is essentially reduced by 4x, the performance requirement for generating an infinite number of scales is bounded by 1.33x of the base input image pixel rate. The second input can be used to enable a pyramid generation processing for another input data or a set of scales/pyramid generation for chroma data as long as the total number of scales needed by both threads is 10 or less. Each resizer in a processing thread can access any coefficient set.

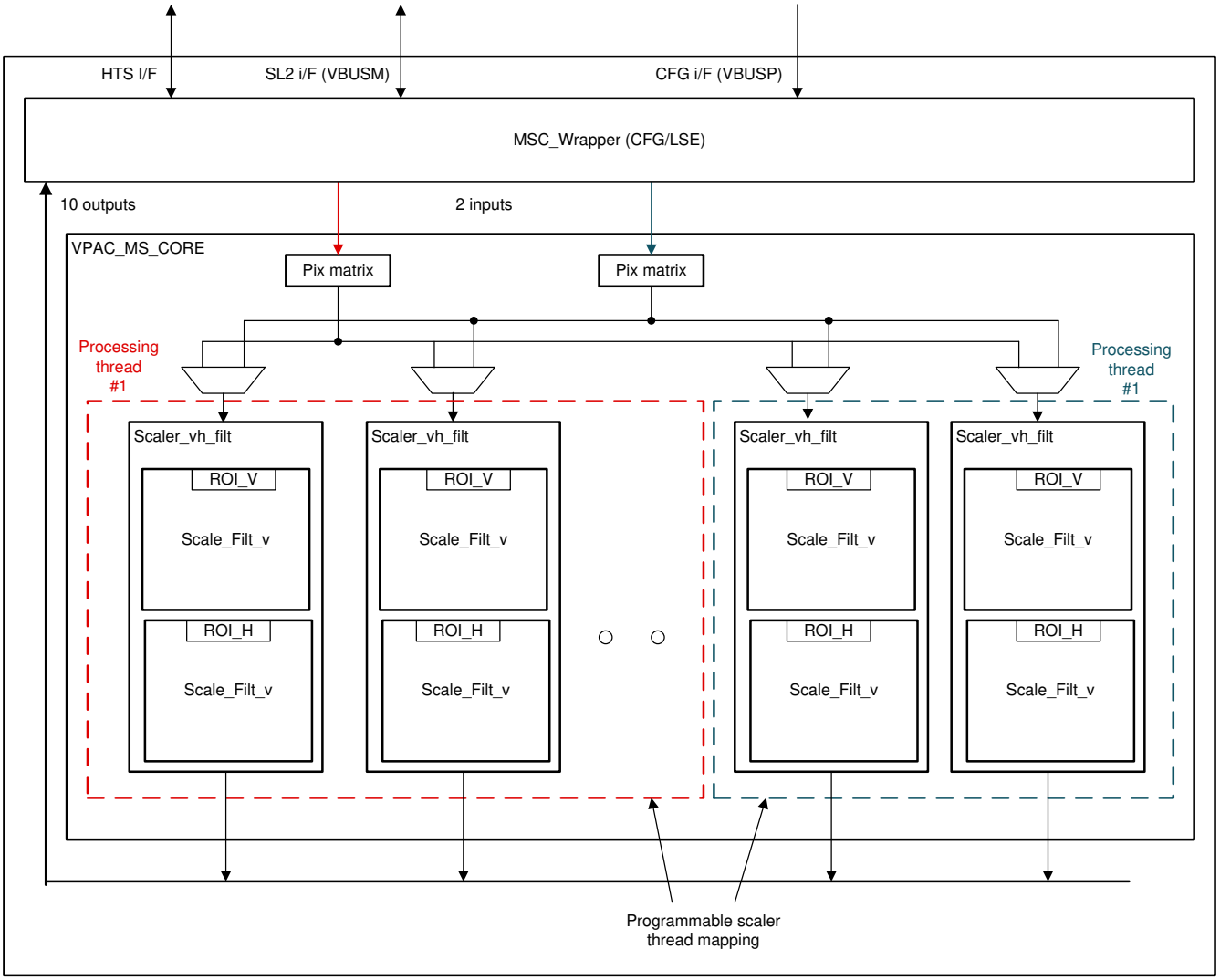


Figure 7-109. Block Diagram of Multi-Scaler Filter Core

Copyright © 2026 Texas Instruments Incorporated

For details on MSC integration in VPAC, see Section 7.7.2,*VPAC Subsystem Level*.

7.7.5.2.1.1 MSC Submodule Details

The MSC_LSE (Load Store Engine) handles all interfacing to the SL2 memory space and performs following functions:

- Provides interface to SL2 circular buffers via a common VBUSM controller interface
- Manages input and output channel updates in sync with HTS
- Provides data unpacking for core and packing for SL2 interface
- Manages LSE specific configuration registers

The MSC_CORE performs all filtering operations for various 2-in/10-out multi-scaling/multi-thread configurations. It has no direct connection to external interface and it works on a frame data coming in and out in raster scan order.

The internal data transfers between the MSC_LSE and MSC_CORE are done over VBUSP write-only streaming interfaces.

7.7.5.2.1.1.1 MSC Load Store Engine (MSC_LSE)

7.7.5.2.1.1.1.1 MSC_LSE Overview

Figure 7-110 shows the MSC_LSC block diagram.

Copyright © 2026 Texas Instruments Incorporated

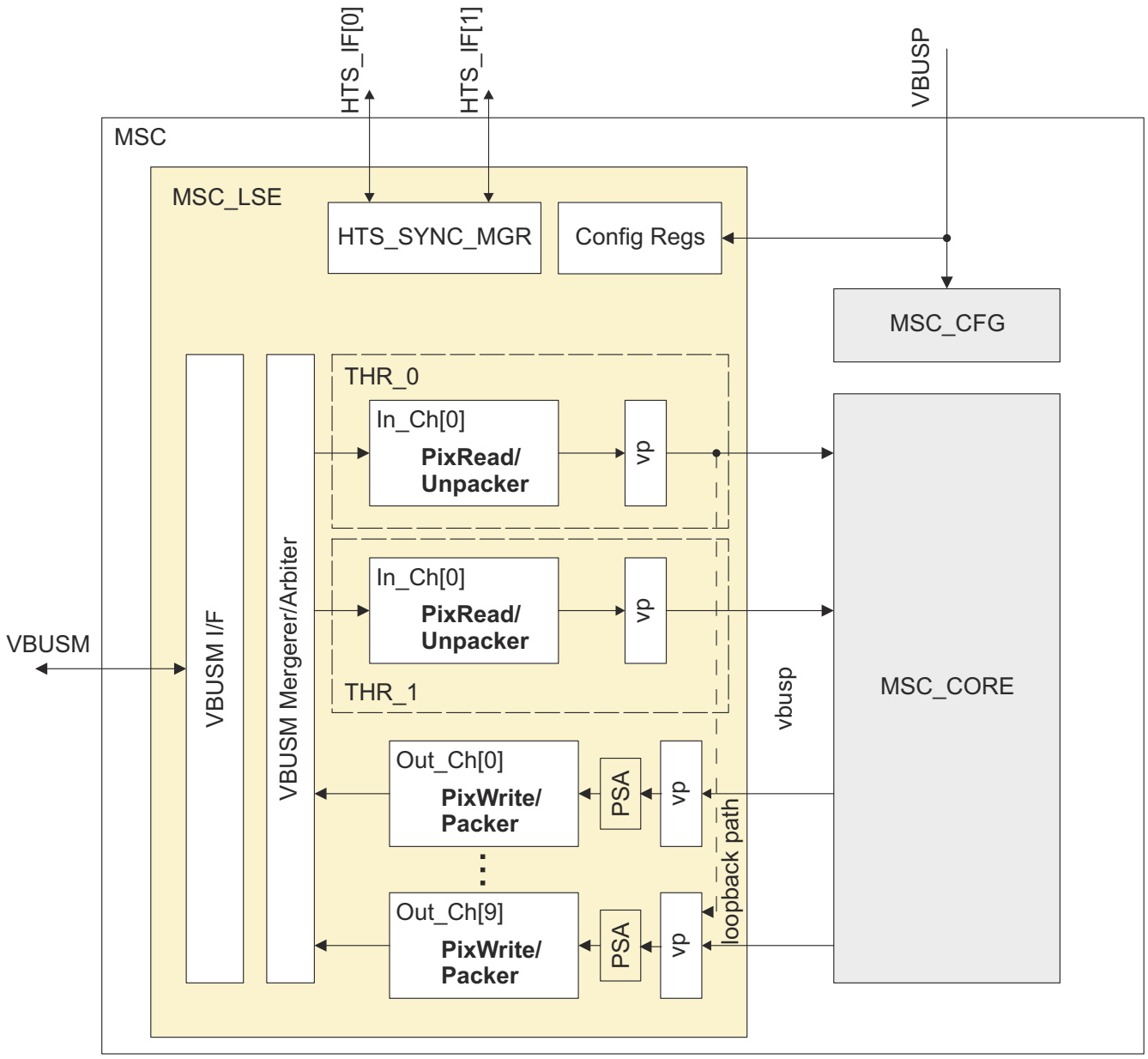


Figure 7-110. MSC_LSC Block Diagram

7.7.5.2.1.1.1.1.1 MSC_LSE Features

The MSC_LSE supports following VPAC_LSE features:

- Up to 2 processing threads supported (a thread is a processing chain with its own HTS start/done).
- Input channel features:
  - One SL2 input channel per thread with following features for the input channel:
    - Single data plane support
    - 8/12 packed and 12-bit unpacked source format support
    - Up to 5 lines - input kernel height support
    - 1D addressing mode only with CBUF address handling
    - Input Line Skip Support
    - Vertical boundary Edge Padding (Replication only) support
    - Channel enable
    - Video frame data with frame sync signals over a common vbusp streaming write-controller interface

Copyright © 2026 Texas Instruments Incorporated

- Output Channel Features:
  - Up to 10 SL2 output channels (pixel data output only)
  - Single data plane support for each output channel
  - Programmable thread mapping for each output channel
  - 8/12 packed and 12-bit unpacked output pixel data format support
  - 1D addressing mode support with CBUF address handling
  - Channel enable
  - Video frame data with frame sync signals (along with optional in-band control signals) over a separate vbusp streaming write-target interface
- HTS synchronization support
- LSE internal data bypass mode support
  - Input channel 0 can be configured to bypass/loopback mode the data to the core and/or directly to the output channel 9 (bypassing the core).

7.7.5.2.1.1.1.2 MSC_LSE Internal Data Loopback Channel

The MSC_LSE provides a data loopback channel.

When enabled (VPAC_MSC_LSE_CFG_LSE[0] LOOPBACK_EN), the LSE sends the “loopback-enabled” input channel unpacked data directly to the designated output channel packing logic without going through the HWA_CORE. The looped back data can be packed in a different format in the output channel to perform a data format conversion or can be packed in the same format as the input.

When the loopback mode is enabled, the “loopback” designated output channel is no longer available for HWA_CORE output data transfer. But, the input channel can still operate normally to read in the data for the HWA_CORE (provided that the other output channel is used for transferring out the HWA_CORE output data) by setting VPAC_MSC_LSE_CFG_LSE[1] LOOPBACK_CORE_EN. In this mode, the input channel flow is throttled by both the core and the loopback data path.

The data flow of the loopback mode is also controlled by the HTS synchronization signals.

7.7.5.2.1.1.1.3 MSC_LSE PSA Support

As a safety function, LSE integrates a CRC signature capture mechanism on each output channel data to verify the integrity of the HWA internal paths. When enabled (VPAC_MSC_LSE_CFG_LSE[8] PSA_EN), every valid transfer frame data from the MSC is passed through a CRC module to update the signature for the channel before the data is packed and sent to the SL2 buffer. At the end of frame condition, the final signature is saved in a separate read-only signature register (VPAC_MSC_LSE_PSA_SIGNATURE_y[31-0] VAUE = 0,...,Number_of_Output Channel-1) till the next end of frame condition for software to read and compare it against the golden reference signature.

The PSA signature register is based on the following IEEE 802.3 standard 32-bit CRC polynomial:

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$

The PSA module is inserted in the LSE data path as shown in Figure 7-110.

7.7.5.2.1.1.1.4 MSC_LSE Feature Detailed Description

For detailed LSE feature description, see *Load Store Engine (LSE)*.

7.7.5.2.1.1.2 MSC_CORE (HWA Core)

The MSC_CORE (HWA_CORE) is configured as two multi-scaling engines each of which can perform 1-to many (multi) scaling operations on an independent input source. The number of outputs (scaler filters) assigned to each multi-scaling engine is user-configurable.

7.7.5.2.1.1.2.1 MSC_CORE Overview

Figure 7-111 shows the architectural overview of the MSC_CORE. Each reconfigurable multi-scaling engine in the MSC_CORE receives its input frame data from its own dedicated VP (vbusp write only target) interface. VP_IN_0 port is used for the thread #0 (multi-scaling engine-0) and VP_IN_1 port is used for the thread

Copyright © 2026 Texas Instruments Incorporated

#1 (multi-scaling engine-1). The THREAD_MAP parameter of the MSC_LSE output channel configuration register (VPAC_MSC_LSE_DST_BUF_CFG_j[7] THREAD_MAP) controls which input port is connected to which scaling filter (filter_0 ~ filter_9).

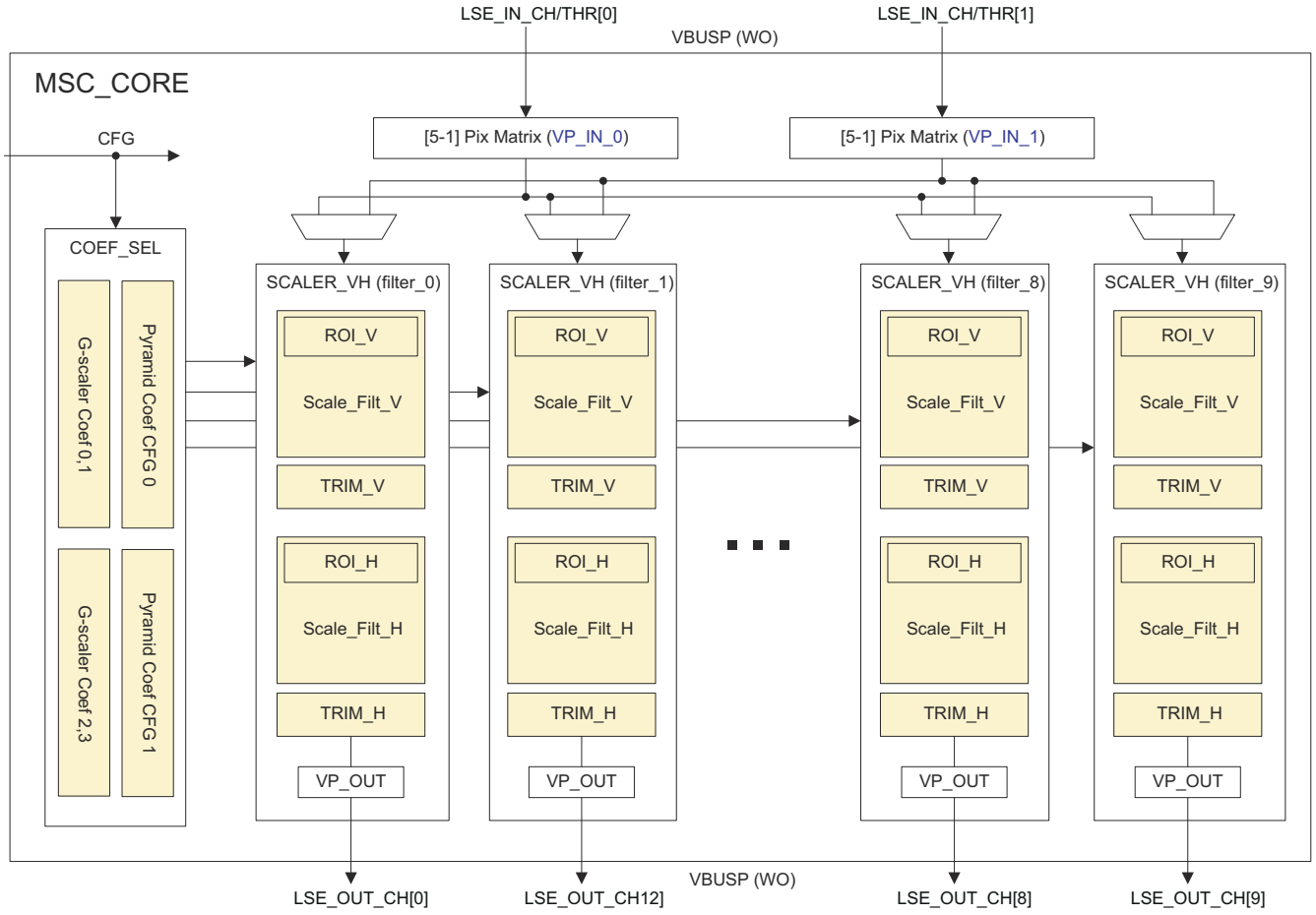


Figure 7-111. MSC_CORE Block Diagram

The scaling filter implements the vertical-before-horizontal architecture in order to avoid intermediate result line buffers that would be needed in the horizontal-before-vertical configuration. This architecture also allows the source frame width to be not restricted by internal line buffer sizes.

Each scaling filter in the MSC_CORE consists of 5-tap (32/64-phase) Polyphase filter based vertical and horizontal resizers. At the input of each resizer, a ROI checker determines the input start position (that is, marks the input data as valid). Output of each resizer is then trimmed to the configured output size of the scaling filter in each respective direction.

For MSC, the vertical input edge padding is performed in the MSC_LSE while the horizontal input boundary padding is done within the MSC_CORE before the horizontal resizer.

7.7.5.2.1.1.2.2 Polyphase Filter of Vertical/Horizontal Resizers

7.7.5.2.1.1.2.2.1 Filter Data Path Logic

Figure 7-112 shows the micro-architecture of the polyphase filter data path. The 5-tap filter structure can be programmed to perform a 5-tap Gaussian filter (for Octave generation) or a 4-tap bi-cubic downscaling filter (for scale generation). Or, it can be configured to perform a custom convolution filter without resizing. Filter coefficients for any unused taps should be set to 0. For example, in case of a 4-tap filtering, C2(Fi) filter coefficient should be set to 0. For 3-tap filtering, C-2(Fi) and C2(Fi) filter coefficients should be set to 0.

Copyright © 2026 Texas Instruments Incorporated

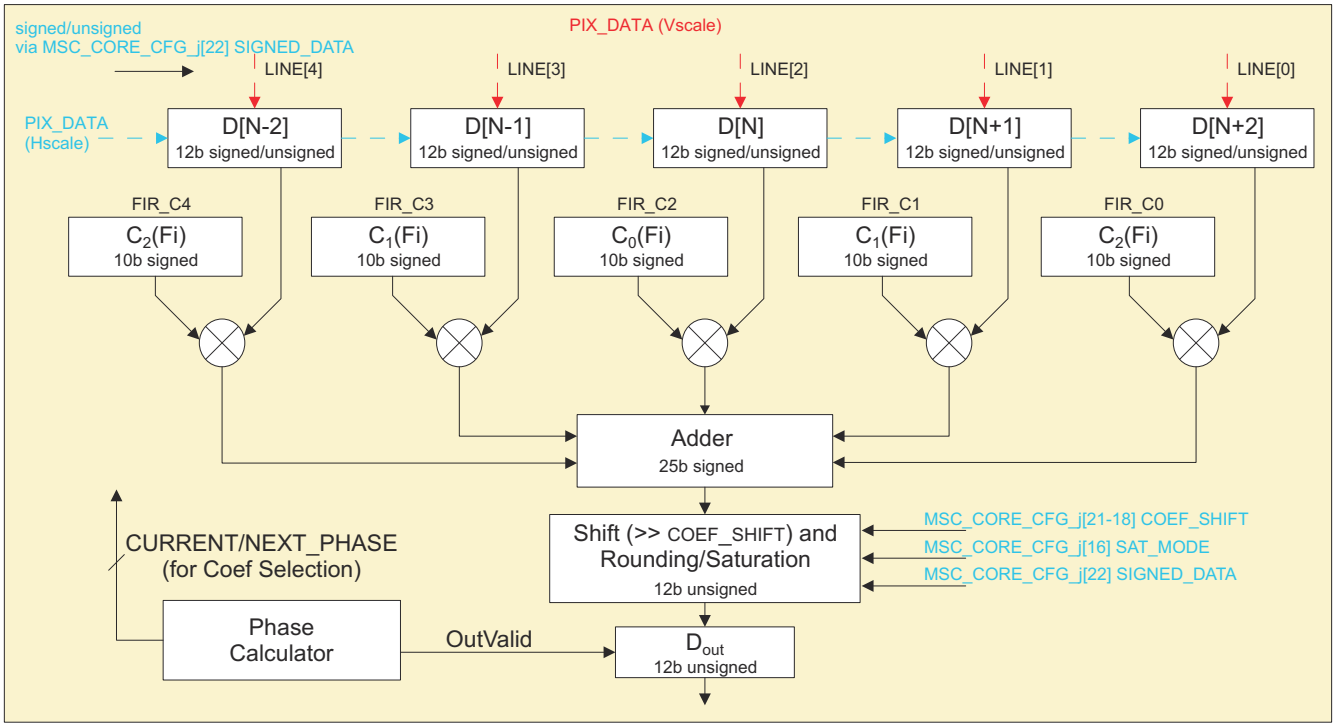


Figure 7-112. Polyphase Filter Micro-Architecture

Kernel Size Configuration

There is no separate configuration required to set the filter kernel size (number of taps) within the polyphase filter. The filter always works as a 5-tap filter but requires unused taps to be masked using the zero coefficients.

Coefficient Precision Selection

The precision of the coefficients is user-configurable using VPAC_MSC_CORE_CFG_j[21-18] COEF_SHIFT which determines the number of fractional bits of the coefficient.

Unsigned/Signed Data Type support

Typically for image resizing, the input and output data are in unsigned (positive) integer numbers in the range of [0..4095]. But, in some non-image convolution filtering (for example, Sobel filter), the input and output data may be in signed integer data format. The MSC supports a user-selectable data type mode bit (VPAC_MSC_CORE_CFG_j[22] SIGNED_DATA) per resizer to specify whether the input/output data is in signed or unsigned data format. The selection determines the input range and output clipping limits as shown below:

VPAC_MSC_CORE_CFG_0[22] SIGNED_DATA = 0 ->[0..4095]

VPAC_MSC_CORE_CFG_0[22] SIGNED_DATA = 1 -> [-2048..2047]

Note that all filters in the same processing thread should have the SIGNED_DATA bit set to a same value.

Rounding Logic

The rounding is done by “adding 0.5 and dropping fractional bits”. This achieves the following:

For data >= 0: “round to nearest, ties away from zero”

For data < 0: “round to nearest, ties toward zero”

Output Saturation Logic

For some non-image unsigned data filtering, the result of the final adder could be negative numbers. To avoid clipping all negative numbers to 0, the MSC filter data processing supports a mode

Copyright © 2026 Texas Instruments Incorporated

(VPAC_MSC_CORE_CFG_j[22] SAT_MODE = 1) to offset the result by 2048 and then clip the final result to [0..4095] in the output saturation logic to preserve the full data range.

Edge Pixel Replication

Pixel replication in vertical direction (edge line replication) is done in the MSC_LSE.

Pixel replication in horizontal direction (edge pixel replication) is performed in the horizontal filter. For chroma data plane, Cb/Cr pixel replication is done independently with edge Cb/Cr data respectively.

7.7.5.2.1.1.2.2.2 Filter Parameters

Table 7-117 lists the parameters used to define the configuration of the resizing (downsampling) vertical/ horizontal filters.

Table 7-117. Parameters of the Resizing (Downsampling) Vertical/Horizontal Filters

Parameter	Bit Precision	Description
FIRINC_V VPAC_MSC_CORE_FIRINC_j[30-16] VS	U15Q12 (Unsigned 15b number with 12-bit fraction)	Vertical Resizing Ratio value (vertical filter increment) Valid Range: 4096 (1x scaling) < FIRINC_V < 16384 (1/4x scaling) +1 in the above equation is optional. +1 forces non-repeating coefficients in integer ratio scaling cases.
FIRINC_H VPAC_MSC_CORE_FIRINC_j[14-0] HS	U15Q12 (Unsigned 15b number with 12-bit fraction)	Horizontal Resizing Ratio value (Horizontal filter increment)) Valid Range: 4096 (1x scaling) < FIRINC_V < 16384 (1/4x scaling)
ACC_INIT_V VPAC_MSC_CORE_ACC_INIT_j[27-16] VS	U12Q12 (Unsigned 12b number with 12b fraction)	Initial Vertical Resizer Phase Valid Range: 0 < ACC_INIT_V < 4095
ACC_INIT_H VPAC_MSC_CORE_ACC_INIT_j[11-0] HS	U12Q12 (Unsigned 12b number with 12b fraction)	Initial Horizontal Resizer Phase Valid Range: 0 < ACC_INIT_V < 4095

Additionally, Table 7-118 lists the mode parameters defined the filter operation and coefficient selection.

Table 7-118. Filter Mode and Coefficient Selection via VPAC_MSC_CORE_CFG_j Registers

Parameter	Bit Width	Description
SIGNED_DATA VPAC_MSC_CORE_CFG_j[22] SIGNED_DATA	1-bit	Integer type of input and output frame data: 0 : Unsigned 12-bit (default) 1 : Signed 12-bit
UV_MODE VPAC_MSC_CORE_CFG_j[17] UV_MODE	1-bit	Source data interleave format: 0 - non-interleaved (Y data) 1 - interleaved (UV data)
SP_VS_COEF_SEL VPAC_MSC_CORE_CFG_j[15-12] SP_VS_COEF_SEL	4-bits	Single Phase – Vertical Filter Coef Selection N : 5-tap/32-phase Filter Coef Set #0 (Entry N) where N=0-9 10 : Gaussian Filter #0 11 : Gaussian Filter #1 12-15 : Invalid When FilterMode=0 and N <10, then the multi-phase coef set #0 is dedicated for one or more single phase filters.

Copyright © 2026 Texas Instruments Incorporated

Table 7-118. Filter Mode and Coefficient Selection via VPAC_MSC_CORE_CFG_j Registers (continued)

Parameter	Bit Width	Description
SP_HS_COEF_SEL VPAC_MSC_CORE_CFG_j[10-7] SP_HS_COEF_SEL	4-bits	Single Phase – Horizontal Filter Coef Selection N : 5-tap/32-phase Filter Coef Set #0 (Entry N+10) where N=0-9 10 : Gaussian Filter #0 11 : Gaussian Filter #1 12-15 : Invalid When FilterMode=0 and N <10, then the multi-phase coef set #0 is dedicated for one or more single phase filters.
VS_COEF_SEL VPAC_MSC_CORE_CFG_j[5-4] VS_COEF_SEL	2-bits	Multi-phase Vertical Coef Selection 00 : 5-tap/32-phase Filter coef set #0 01 : 5-tap/32-phase Filter coef set #1 10 : 5-tap/32-phase Filter coef set #2 11 : 5-tap/32-phase Filter coef set #3
HS_COEF_SEL VPAC_MSC_CORE_CFG_j[3-2] HS_COEF_SEL	2-bits	Multi-phase Horizontal Coef Selection 00 : 5-tap/32-phase Filter coef set #0 01 : 5-tap/32-phase Filter coef set #1 10 : 5-tap/32-phase Filter coef set #2 11 : 5-tap/32-phase Filter coef set #3
PHASE_MODE VPAC_MSC_CORE_CFG_j[1] PHASE_MODE	1-bit	Filter Phase mode selection 0 - 64 phases 1 - 32 phases
FILTER_MODE VPAC_MSC_CORE_CFG_j[0] FILTER_MODE	1-bit	Filter Mode 0 – Single Phase Filter (for example, Gaussian Filter for Pyramid) 1 – Multi-phase Scaling Filter
FILT_SAT_MODE VPAC_MSC_CORE_CFG_j[16] SAT_MODE	1-bit	Filter Output Saturation Mode 0 - [0..4095] clipping 1 - [-2048.. 2047] clip followed by +2048 This parameter is used only when signed_data =0 (unsigned data type).
COEF_SHIFT VPAC_MSC_CORE_CFG_j[21-18] COEF_SHIFT	4-bits	Coef Shift Size: configures the precision of the 10-bit signed filter coefficients (valid Shift range: 5~9) : 5: Shift by 5 (5-bit fraction) 6: Shift by 6 (6-bit fraction) 7: Shift by 7 (7-bit fraction) 8: Shift by 8 (8-bit fraction) 9: Shift by 9 (9-bit fraction) Integer size = 9 - #_of_fraction_bits

7.7.5.2.1.1.2.2.3 Single-Phase Filter Parameters

For a single-phase filter configuration (VPAC_MSC_CORE_CFG_j[0] FILTER_MODE = 0), software must ensure that FIRINC parameters (VPAC_MSC_CORE_FIRINC_j[30-16] VS and VPAC_MSC_CORE_FIRINC_j[14-0] HS) are programmed with one of the following values only:

- 0x4000 (1/4 X resizing)
- 0x2000 (1/2 X resizing)
- 0x1000 (1X – no scaling)

Copyright © 2026 Texas Instruments Incorporated

VPAC_MSC_CORE_ACC_INIT_j[27-16] VS and VPAC_MSC_CORE_ACC_INIT_j[11-0] HS configuration values are ignored by the MSC hardware (internally set to 0x0) when VPAC_MSC_CORE_CFG_j[0] FILTER_MODE = 0. If any phase offsets are needed, the starting pixel/line locations of the frame and the filter coefficient values should be adjusted to implement the desired phase shift.

7.7.5.2.1.1.2.2.4 Interleaved Mode Handling

When the input is in the interleaved data format (for example, YUV420 Chroma data), the horizontal filter supports two sets of 5-input buffers (Cb and Cr buffers) and alternates filtering operation between two. Phase calculation and coefficient updates are done every other sample. Edge replications at the start of the line and at the end of the line are handled separately for Cb and Cr data.

There is no change in how the vertical filter handles an interleaved data format source.

7.7.5.2.1.1.2.2.5 Input Skip Line Support

When a processing thread is only performing an octave generation (1/2x single phase resizing), the next input lines can be offset by 2 lines since the output is only generated every other input line times. By skipping lines (VPAC_MSC_LSE_SRC_CFG_j[7] SRC_LN_INC_2), the SL2 access is reduced in half for this mode and the cycle time to complete an Octave generation is equal to ½ of the input frame size.

Line N: Read Line N-2, N-1, N, N+1, N+2

Line N+1: Skip

Line N+2: Read Line N, N+1, N+2, N+3, N+4

Line N+3: Skip

Even though “N+1” line processing is skipped, all source lines are still required for proper filtering. Therefore, the DMA transfer from DDR to SL2 should include all lines. VPAC_MSC_CORE_FIRINC_j[30-16] VS should be set to 4096 (1x resizing) to compensate for source line skip (if it is doing a ½ resizing).

Horizontal Skip is done normally as ½ x resizing by the core HScale Filter.

7.7.5.2.1.1.2.3 Scaler Filter Thread Mapping

The scaling filter_# (# = 0..9) is enabled and its input is connected to one of the processing thread source VP ports (VP_IN_0 or VP_IN_1) by configuring the output channel #n buffer configuration:

VPAC_MSC_LSE_DST_BUF_CFG_j[7] THREAD_MAP: 0 maps the filter to VP_IN_0, 1 maps to the VP_IN_1.

VPAC_MSC_LSE_DST_BUF_CFG_j[TBD] ENABLE: enables both filter and output channel

By mapping each output to a thread explicitly in the output channel configuration register, MSC guarantees that a filter can only be connected to one processing thread.

7.7.5.2.1.1.2.4 Filter Coefficients

Since each resizer needs to access the common coefficients independently, the coefficients are stored in registers (instead of memory). Each resizer selects directly from these register using the coefficient selection scheme as shown in Figure 7-113. The same mux is used to select either vertical or horizontal coefficients.

Copyright © 2026 Texas Instruments Incorporated

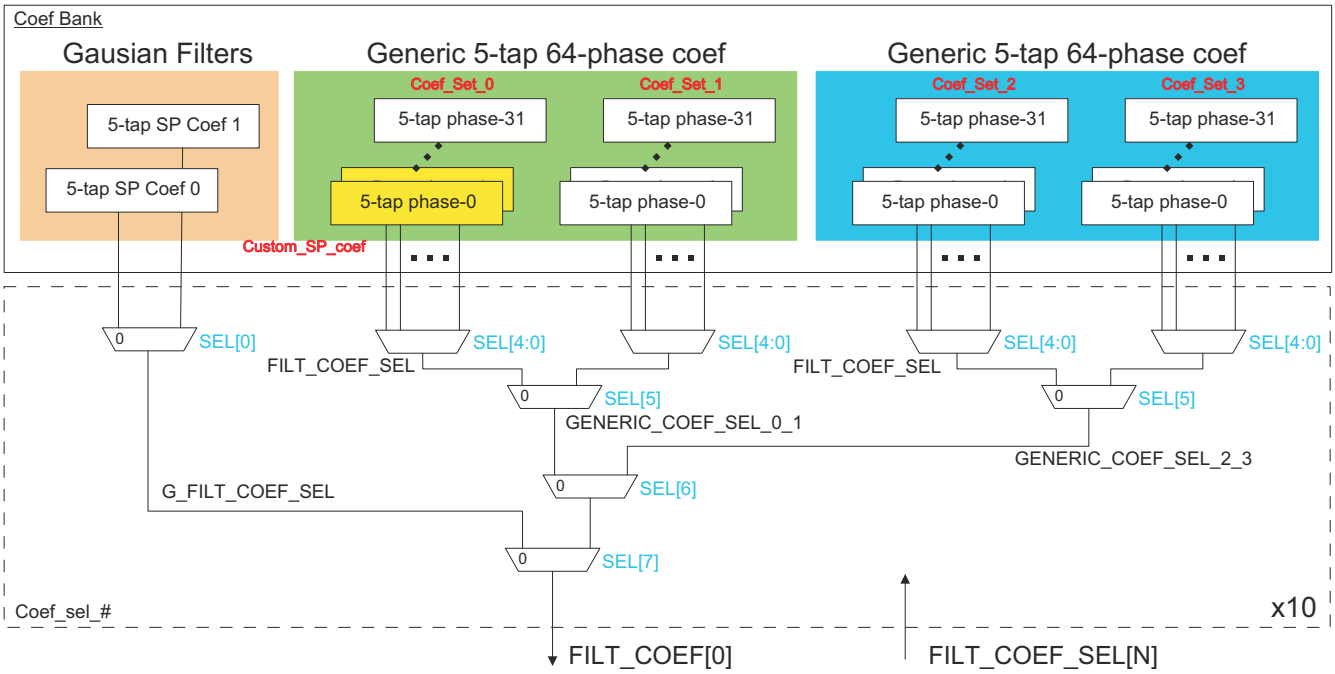


Figure 7-113. Coefficient Selection

7.7.5.2.1.1.2.4.1 Filter Coefficient Parameter Configuration

Each coefficient table entry consists of 5 coefficients stored in two memory mapped configuration registers - VPAC_MSC_CORE_C210_j and VPAC_MSC_CORE_C43_j. FIR_C4-0 parameters map to C-2, C-1, C0, C1, and C2 coefficients (see Figure 7-111).

7.7.5.2.1.1.2.4.2 3/4/5-Tap Filter Configuration

All filters operate in 5-tap filter mode – requiring 5-tap coefficients regardless of the filter usage. For less-than-5 tap filter configurations, coefficients corresponding to the unused taps should be set to 0.

For example, FIR_C0 and FIR_C4 will need to be set to 0x0 for a 3-tap configuration.

For 4-tap configuration, FIR_C0 will need to be set to 0x0.

7.7.5.2.1.1.2.5 Input/Output ROI Trimmers

Each scaler filter implements the ROI scaling as shown below with input and output trimmers. The input trimmer simply marks the beginning of the ROI to indicate where the valid source pixels and lines start in the source image. Doing the source ROI marker allows the exact specification of a sub-image in a common source image (no position or size restriction) while keeping the original boundary pixels (that is, not use the replicated pixels/ lines).

The output trimmers then uses destination size parameters (VPAC_MSC_CORE_OUT_SIZE_j[28-16] HEIGHT and VPAC_MSC_CORE_OUT_SIZE_j[12-0] WIDTH) to trim out the exact final output image sizes.

VPAC_MSC does not perform output size correction by stuffing extra pixels in the case the programmed VPAC_MSC_CORE_OUT_SIZE_j[28-16] HEIGHT and VPAC_MSC_CORE_OUT_SIZE_j[12-0] WIDTH exceeds the actual output size of the scaler. In this case, VPAC_MSC simply outputs the actual size.

7.7.5.2.2 Resizer Algorithm Details

This section describes the algorithm details of the Horizontal and Vertical Down-Scaling engines.

7.7.5.2.2.1 Multiple Scales Generations

For a high number of scales needed by vision image processing algorithm (for example, 32 or more), there are many options in generating the multiple scales as shown in Figure 7-114. Among the options, the “linear rescale

Copyright © 2026 Texas Instruments Incorporated

from octave resolution" gives the best tradeoff in video quality and design simplicity. The MSC implements 1-to-N multi-rescaling (that is, from one base image to multi-intra-scales and the next octave image generation) as shown as option B in Figure 7-114.

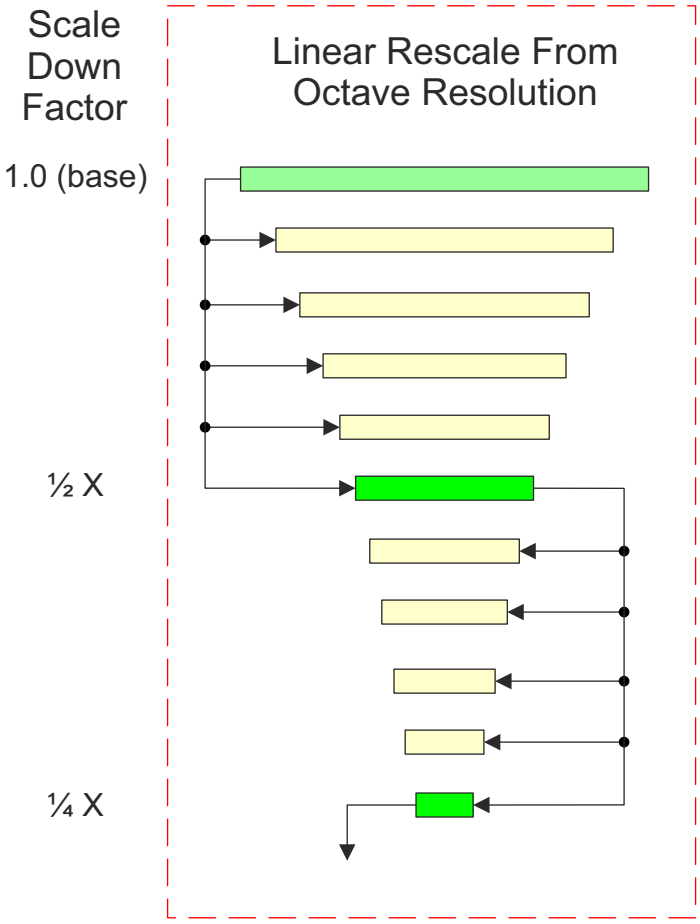


Figure 7-114. Multiple Scale Generation

7.7.5.2.2.2 Polyphase Filter

Both vertical and horizontal resizers in MSC hardware are implemented using a programmable Polyphase filter structure supporting 5-tap kernel with either 64 or 32 phases.

7.7.5.2.2.2.1 Interpolation/Resampling

Figure 7-115 illustrates the interpolation principle. The assumptions are that the input pixels are evenly spaced. The distance between each input pixel is assumed to be 1. The magnification ratio is given by 64/N and p/64 is the initial phase of the output data. The output pixels are evenly spaced as well. The distance between each output pixel is given by N/64 (in the example below, N > 64). The position of the n-th output pixel is given by (n x N + p) / 64.

Copyright © 2026 Texas Instruments Incorporated

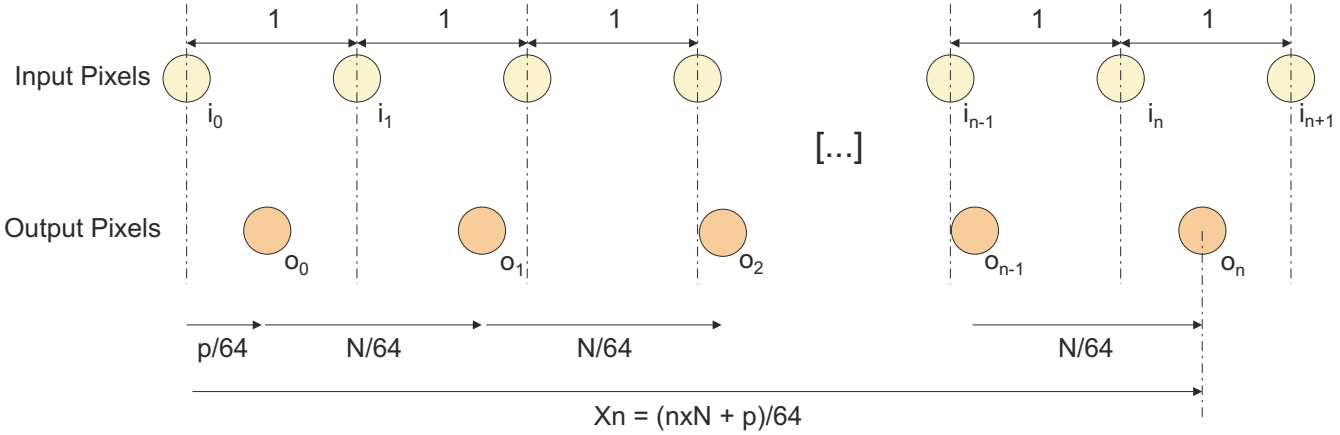


Figure 7-115. Interpolation Principle

Figure 7-116 illustrates the interpolation principle at the n-th output pixel (on) at position x_n. Interpolation method with m and d parameters, are described as:

m = floor((n x N + p) / 64)

d = ((n x N + p) / 64) - m

Figure 7-116 illustrates the concept when the filters are configured in 4-tap mode for non-integer rescaling (the first coefficient corresponding to O_{m-2} is set to zero).

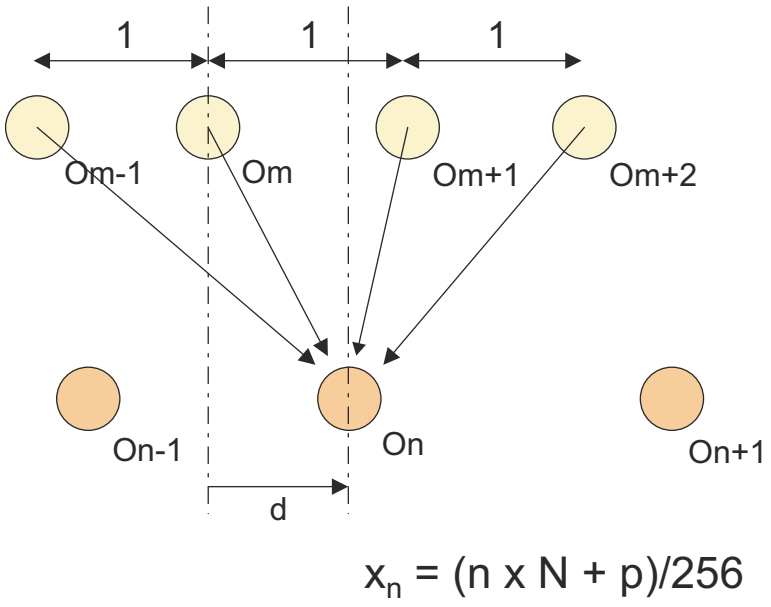


Figure 7-116. Interpolation Process

Figure 7-117 illustrates another explanation of the concept of a Polyphase filter using a simple 2-tap linear filter in a design which supports 5 phases.

Copyright © 2026 Texas Instruments Incorporated

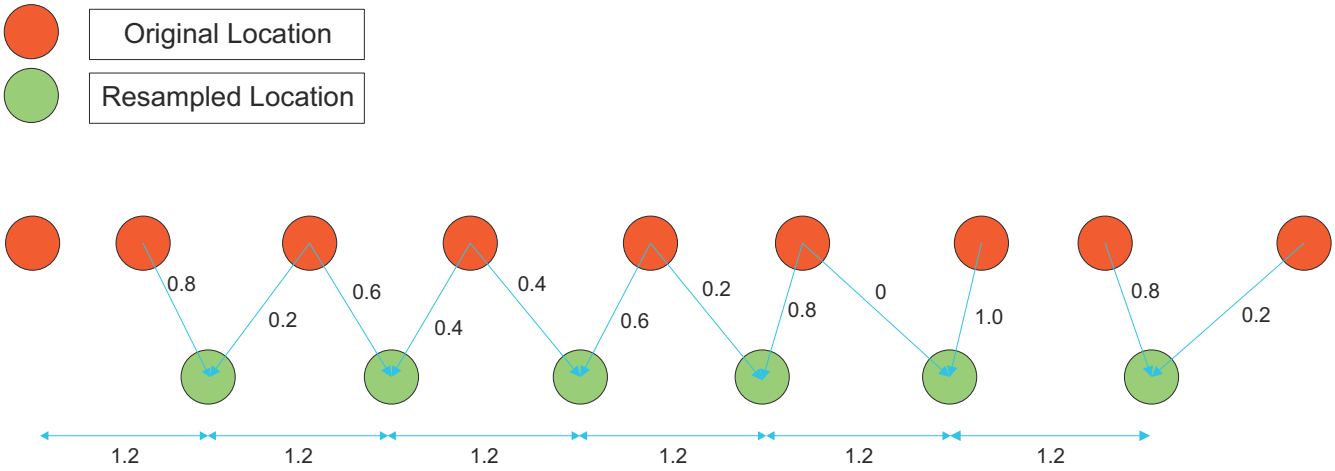


Figure 7-117. Polyphase Filtering

For each one of the 5 phases, the weights (or filter coefficients) are different and after the 5 phases the weights start repeating, see Figure 7-117.

Figure 7-118 depicts the architecture for the independent polyphase filter unit.

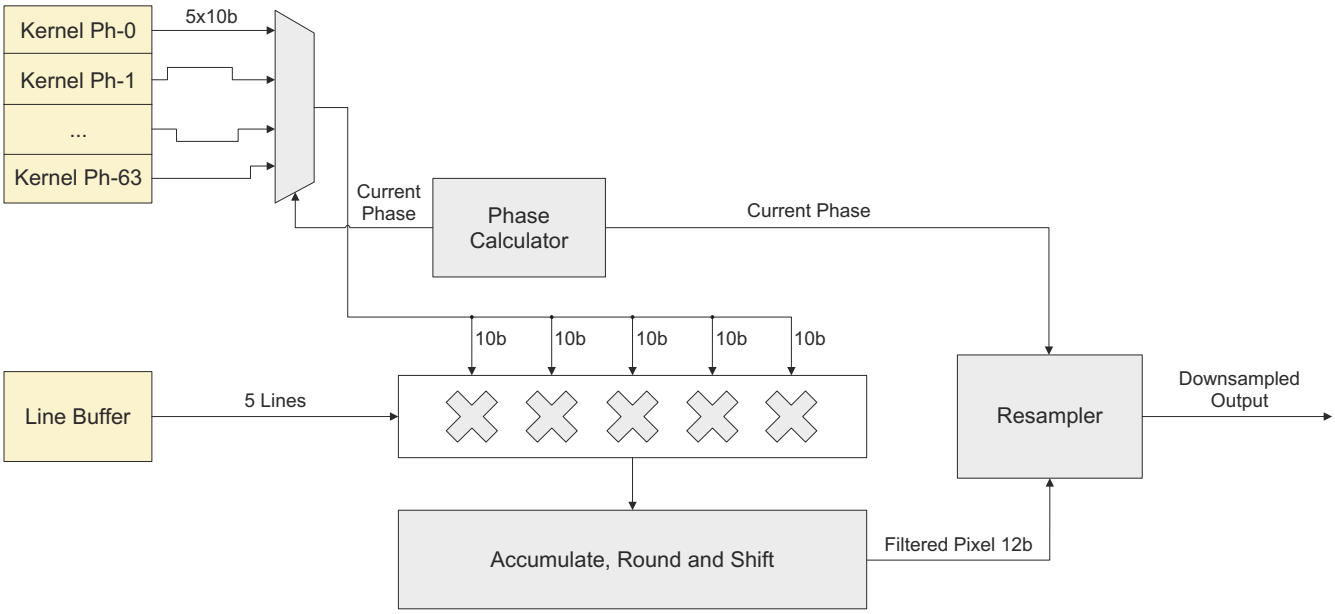


Figure 7-118. Polyphase Filter Unit

Both the horizontal and vertical filter described earlier are an instance of the Polyphase filter unit, though only the vertical filter requires line memories (implemented in SL2 memory space). In all, there will be 20 instances of the Polyphase filter unit, 2 filters per channel and 10 total channels in the VPAC_MSC module.

7.7.5.2.2.2.2 Phase Calculation and Re-sampler

The phase calculation block calculates the current phase which in turn is used for selecting the right coefficient set and for the output qualifier in the Re-sampler block. The Re-sampler engine then decides whether or not the output created by the interpolation unit is a valid output or not.

Copyright © 2026 Texas Instruments Incorporated

7.7.5.2.2.2.3 Shared Coefficient Buffers

The MSC module supports up to 4 sets of multi-phase coefficients for generic non-integer resizing and up to 10 sets of single-phase coefficients (5x1 entries) for Octave generation and integer resizing applications. The multi-phase coefficients can also be configured in one of two following options:

- 4 sets of 5-tap x 32 phase coefficients
- 1 set of 5-tap x 64 phase coefficients and 2 sets of 5-tap x 32 phase coefficients
- 2 sets of 5-tap x 64 phase coefficients

The 32 phase configuration option allows using independent coefficients for vertical and horizontal resizer to achieve aspect ratio change during the scaling.

7.7.5.2.2.2.4 Border Pixel Padding

The MSC module replicates missing lines and pixels at top/bottom and left/right sides of the input frame for non-interleaved (luma) and interleaved (chroma) data.

The missing lines at the top/bottom are replicated in the LSE sub-module prior to the vertical scaler and the missing pixels at the left/right sides of each line are replicated in the core prior to the horizontal scaler. For more information about LSE module, see *Load Store Engine (LSE)*.

Replication scheme simply repeats the edge line or pixel to create missing lines/pixels. Figure 7-119 shows horizontal pixel replication.



Figure 7-119. Horizontal Pixel Replication

If a scaling is performed on a sub-frame (ROI) within the input frame, no replicated lines/pixels will be used since the lines/pixels outside the sub-frame are already present in the input frame.

For interleaved data format, the horizontal replication is done separately for U and V data. For U-data, edge U-pixel will be replicated. For V-data, edge V-pixel will be replicated. There is no distinction in the vertical edge padding.

7.7.5.2.2.3 ROI Handling

- ROI supported during scaling generation
  - Only one ROI at Input Frame and one ROI at Output frame per scale
  - ROI definitions (for input as well as all output scales) can be changed at every input frame under software control.
  - ROI for each scale (for output) can be different among scales (as well as with respect to the input). It is set under software Control.
  - The key purpose of ROI is DDR bandwidth reduction. There is additional help by reduction on processing power on DSP and hardware.
- The input ROI is a superset area considering following two scenarios.
  - Multiple ROIs in given Input Frame, if there are multiple ROI for input frame
  - Input Frame area from Overlapped ROI across scales

Figure 7-120 shows an example of multiple ROIs in the source superset ROI area.

Copyright © 2026 Texas Instruments Incorporated

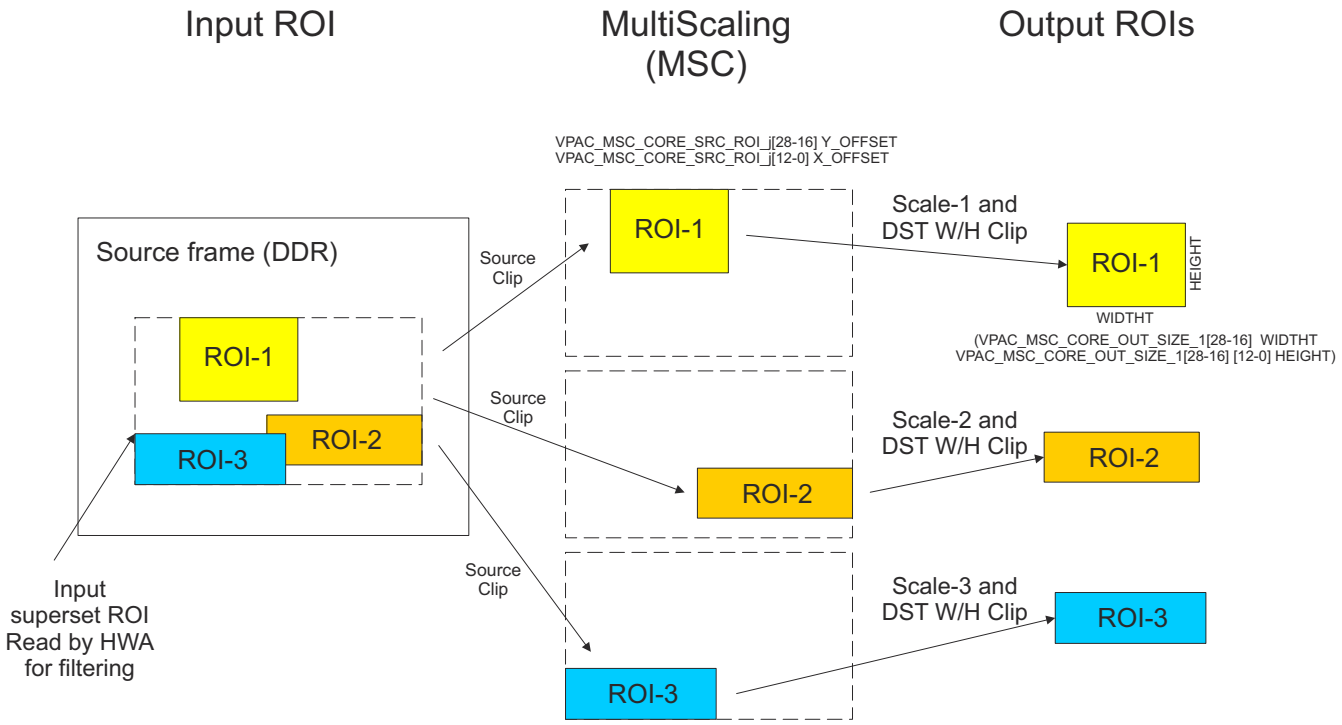


Figure 7-120. Multiple-ROI Support Illustration

Figure 7-121 illustrates another application of generating output ROIs with different scales from a common source region.

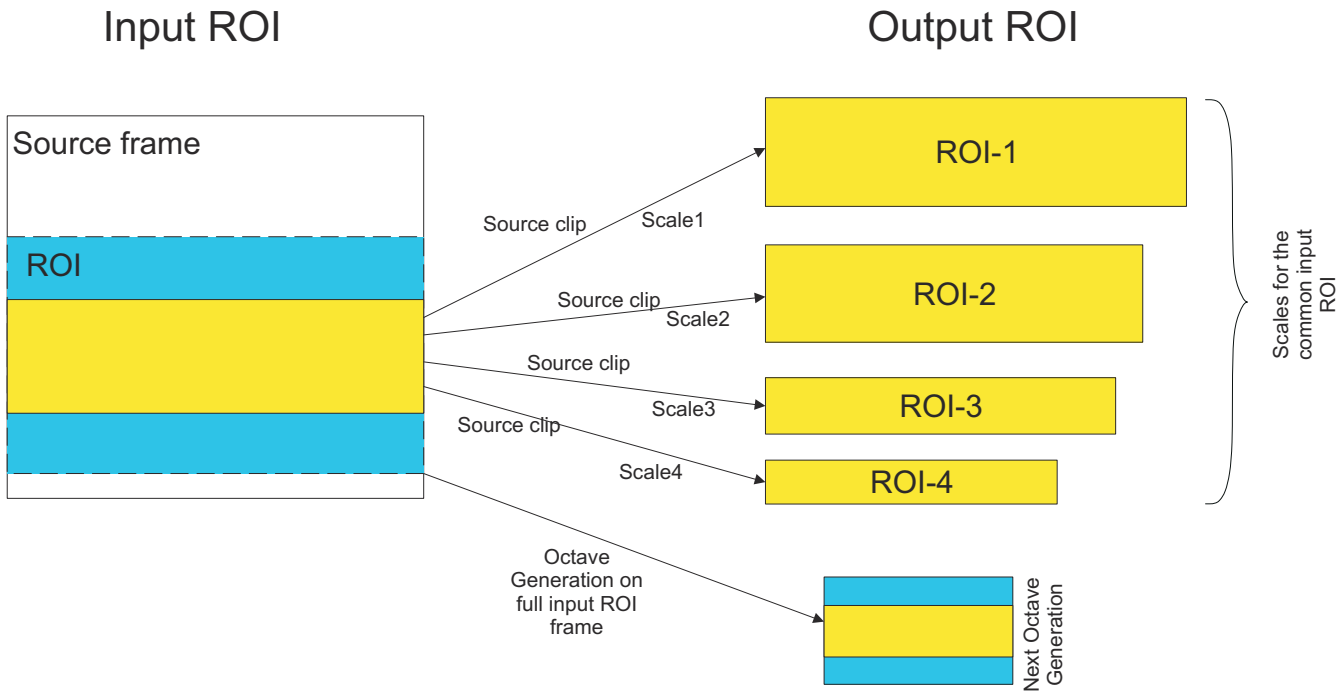


Figure 7-121. Multiple-ROI From a Common Source ROI

By defining output ROIs, only the data within ROI in the “superset ROI” scaled output image is saved thus reducing the DDR traffic significantly.

Copyright © 2026 Texas Instruments Incorporated

7.7.5.2.3 MSC Data Formats Supported

Table 7-119 summarizes data format supported by MSC.

Table 7-119. MSC Input/Output Data Formats

Module (IO)	Bit-Depth	Chroma Format	Packing
MSC Input	8-bit	YUV420	Fully Packed
	12-bit	YUV420	Fully Packed
	12-bit	YUV420	Unpacked in 16-bit
MSC Output	8-bit	YUV420	Fully Packed
	12-bit	YUV420	Fully Packed
	12-bit	YUV420	Unpacked in 16-bit

Table 7-120 and Table 7-121 show the pixel data memory organization for YUV420 (2-plane 12-bit fully packed format).

Table 7-120. YUV420 (2-plane 12-bit Fully Packed Format), First Plane

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Y2[7:0] Y1 Y0	+0x0
Y5[3:0] Y4 Y3 Y2[11:8]	+0x4
Y7 Y6 Y5[11:4]	+0x8
Y10[7:0] Y9 Y8	+0C
Y13[3:0] Y12 Y11 Y10[11:8]	+10
Y15 Y14 Y13[11:4]	+14

Table 7-121. YUV420 (2-plane 12-bit Fully Packed Format), Second Plane

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Cb1[7:0] Cr0 Cb0	+0x0
Cr2[3:0] Cb2 Cr1 Cb1[11:8]	+0x4
Cr3 Cb3 Cr2[11:4]	+0x8
Cb5[7:0] Cr4 Cb4	+0C
Cr6[3:0] Cb6 Cr5 Cb5[11:8]	+10
Cr7 Cb7 Cr6[11:4]	+14

Table 7-122 and Table 7-123 show the pixel data memory organization for YUV420 (2-plane 8-bit fully packed format)

Table 7-122. YUV420 (2-plane 8-bit Fully Packed Format), YUV 4:2:0 – NV12, First Plane

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Y3 Y2 Y1 Y0	+0x0
Y7 Y6 Y5 Y4	+0x4
Y11 Y10 Y9 Y8	+0x8
Y15 Y14 Y13 Y12	+0xC

Table 7-123. YUV420 (2-plane 8-bit Fully Packed format), YUV 4:2:0 – NV12, Second Plane

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Cr1 Cb1 Cr0 Cb0	+0x0
Cr3 Cb3 Cb2 Cb2	+0x4
Cr5 Cb5 Cr4 Cb4	+0x8
Cr7 Cb7 Cb6 Cb6	+0xC

Copyright © 2026 Texas Instruments Incorporated

Note

MSC supports both NV12 and NV21 chroma component ordering. Since there is no color processing, the ordering of Cb and Cr is irrelevant. Simply, if a NV12 chroma plane is the input, the output will be a NV12 chroma plane.

12-bit unpacked format (16-bit container) with LSB/MSB alignments is shown below

Table 7-124 and Table 7-125 show the pixel data memory organization for YUV420 (2-plane 12-bit unpacked format – LSB aligned)

Table 7-124. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
unused										Y1							unused								Y0							+0x0
unused										Y3							unused								Y2							+0x4

Table 7-125. YUV420 (2-plane 12-bit Unpacked Format – LSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
unused										Cr0							unused								Cb0							+0x0
unused										Cr1							unused								Cb1							+0x4

Table 7-126 and Table 7-127 show the pixel data memory organization for YUV420 (2-plane 12-bit unpacked format – MSB aligned)

Table 7-126. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), First Plane

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Y1													unused								Y0											+0x0
Y3													unused								Y2											+0x4

Table 7-127. YUV420 (2-plane 12-bit Unpacked Format – MSB aligned), YUV 4:2:0 – NV12 (12-bit) (0x3D), Second Plane

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Cr0													unused								Cb0											+0x0
Cr1													unused								Cb1											+0x4

7.7.5.3 MSC Interrupt Conditions

7.7.5.3.1 CPU Interrupts

Table 7-128 lists the interupts generated by the MSC module.

Table 7-128. Interrupts

Interrupt	Type	Description
VPAC_MSC_LSE_FR_DONE_EVT_0	Pulse	Frame Processing complete for all filters in the processing thread 0.
VPAC_MSC_LSE_FR_DONE_EVT_1	Pulse	Frame Processing complete for all filters in the processing thread 1.
VPAC_MSC_LSE_SL2_RD_ERR	Pulse	Set whenever there is an error response on VBUSM read command request for any input channel
VPAC_MSC_LSE_SL2_WR_ERR	Pulse	Set whenever there is an error response on VBUSM write command request for any output channel

Copyright © 2026 Texas Instruments Incorporated

All interrupts are single pulse event signals that are mapped to the VPAC level interrupt aggregation logic. The MSC has no mask/set/clear registers for these events. For more information see Section 7.7.2, *VPAC Subsystem*.

7.7.5.3.2 Interrupt Event Description

7.7.5.3.2.1 VPAC_MSC_LSE_FR_DONE_EVT_0/1 Events

These events are generated when all filters associated with the thread 0 or 1 complete the frame processing. This is set when input EOF (end of frame) event and output EOF (end of frame) events are detected.

7.7.5.3.2.2 VPAC_MSC_LSE_SL2_RD_ERR Interrupt Event

This event is generated whenever the VBUSM read status is something other than 0 (success).

The VPAC status register VPAC_MSC_STATUS_ERROR[4-0] VM_RD_ERR stores the last non-zero RSTATUS value and the input channel number for user to see what type of error has occurred on which channel.

Table 7-129. Encoding for RSTATUS

RSTATUS	Meaning
0	Success
1	Addressing error
2	Protection error
3	Timeout error
4	Data error
5	Unsupported Addressing Mode error
6	RESERVED
7	Exclusive read fail

7.7.5.3.2.3 VPAC_MSC_LSE_SL2_WR_ERR Interrupt Event

This event is generated whenever the VBUSM write status is something other than 0 (success).

The VPAC status register VPAC_MSC_STATUS_ERROR[14-8] VM_WR_ERR stores the last non-zero WSTATUS value and the output channel number for user to see what type of error has occurred on which channel.

Table 7-130. Encoding for WSTATUS

WSTATUS	Meaning
0	Success
1	Addressing error
2	Protection error
3	Timeout error
4	Data error
5	Unsupported Addressing Mode error
6	RESERVED
7	Exclusive write fail

7.7.5.4 MSC Performance

Since the resizer performs only downscaling operations, the throughput of scaler is mostly dictated by the input processing rate – 1 input pixel per 1 clock cycle. Once the data is made available to the scaler filter after the initial set up and data fetching, the scaler should be fully active except for few cycles at the end of each line to flush the line.

If a processing thread is performing a pyramid generation only, the vertical line skip feature can be enabled to reduce the frame processing time approximately by ½.

Copyright © 2026 Texas Instruments Incorporated

7.7.5.5 MSC Clocking

Complete VPAC_MSC module operates on single clock - VPAC0_MSC_CLK.

7.7.5.6 MSC Reset

VPAC_MSC has one synchronous active low reset input. The entire module is reset by this reset.

7.7.5.7 MSC Programmer's Guide

7.7.5.7.1 Programming Model

7.7.5.7.1.1 MSC Programming Guidelines

MSC does not support shadow registers for its configuration registers. All changes must be done prior to the HTS.INIT request for a frame processing.

Input and output channels are individually enabled. It is expected that at least one input and output channels are enabled for each HTS thread enabled.

The MSC does not check for proper programming of all configuration parameters. It is strictly software responsibility to ensure programming of MSC (Core and LSE) registers do not cause conflict with each other.

7.7.5.7.1.2 MSC_Core Programming Details

All filters enabled for a multi-scaling thread are programmed via:

- VPAC_MSC_CORE_CFG_j[0] FILTER_MODE; Coefficient Set Selection - VPAC_MSC_CORE_CFG_j[3-2] HS_COEF_SEL, VPAC_MSC_CORE_CFG_j[5-4] VS_COEF_SEL, VPAC_MSC_CORE_CFG_j[10-7] SP_HS_COEF_SEL, VPAC_MSC_CORE_CFG_j[15-12] SP_VS_COEF_SEL; and filter tap size parameters VPAC_MSC_CORE_CFG_j[11] SP_VS_COEF_SRC and VPAC_MSC_CORE_CFG_j[6] SP_HS_COEF_SRC.
- Coefficients set in VPAC_MSC_CORE_C210_j, VPAC_MSC_CORE_C43_j, VPAC_MSC_CORE_C210_j_k, VPAC_MSC_CORE_C43_j_k registers.
- ROI source offset - VPAC_MSC_CORE_SRC_ROI_j[28-16] Y_OFFSET and VPAC_MSC_CORE_SRC_ROI_j[12-0] X_OFFSET, and size. The size of ROI depends on the X and Y offsets. For full-size set the offsets to 0. Note that the ROI_SIZE may or may not be same as the MSC frame size (set in MSC_FRAME_SIZE_j[28-16] HEIGHT and MSC_FRAME_SIZE_j[12-0] WIDTH) - depending on whether ROI is full or sub-frame sized.
- Output size - VPAC_MSC_CORE_OUT_SIZE_j[28-16] HEIGHT and VPAC_MSC_CORE_OUT_SIZE_j[12-0] WIDTH.
- Filter parameters - VPAC_MSC_CORE_FIRINC_j[14-0] HS, VPAC_MSC_CORE_FIRINC_j[30-16] VS, VPAC_MSC_CORE_ACC_INIT_j[27-16] VS, and VPAC_MSC_CORE_ACC_INIT_j[11-0] HS.

Filter thread mapping is done with output channel configuration parameters (MSC_BUF_CFG_j[7] THREAD_MAP). Specifying the mapping in one register for each channel prevents any resource conflict.

7.7.5.7.1.3 MSC_LSE Programming Details

Two processing threads can be activated independently.

7.7.5.7.1.3.1 Input Thread Configuration:

1. Pixel Data Format
   a. VPAC_MSC_LSE_SRC_CFG_j[1-0] PIX_FMT_PW, VPAC_MSC_LSE_SRC_CFG_j[3-2] PIX_FMT_CNTRSZ, and VPAC_MSC_LSE_SRC_CFG_j[4] PIX_FMT_ALIGN parameters define the pixel input data format for all input channels of the thread.
   b. Common data formats:
      i. Fully packed 12-bit: PIX_FMT_PW = 1, PIX_FMT_CNTRSZ = 1, PIX_FMT_ALIGN = 0
      ii. Fully packed 8-bit: PIX_FMT_PW = 0, PIX_FMT_CNTRSZ = 0, PIX_FMT_ALIGN = 0

2. Pixel Input Matrix Configuration

Copyright © 2026 Texas Instruments Incorporated

a. VPAC_MSC_LSE_SRC_CFG_j[15-12] KERN_LN_OFFSET defines the starting line offset number of the input pixel matrix (for 5-tap filter configuration, offset = 0, for 4-tap configuration, offset = 1)
b. VPAC_MSC_LSE_SRC_CFG_j[11-8] KERN_SZ_HEIGHT defines the height of the input pixel matrix (for example, MSC = 5, VISS = 1)
c. VPAC_MSC_LSE_SRC_CFG_j[21-19] KERN_TPAD_SZ, and VPAC_MSC_LSE_SRC_CFG_j[18-16] KERN_BPAD_SZ defines the number of padding lines required at top and bottom of the frame (for example, for 5-tap filter configuration for MSC, both of these parameters are set to 2. For 4-tap filter configuration for MSC, TPAD_SZ =1 while BPAD_SZ = 2).

3. Source Frame Size
   a. VPAC_MSC_LSE_SRC_FRAME_SIZE_j[28-16] HEIGHT and VPAC_MSC_LSE_SRC_FRAME_SIZE_j[12-0] WIDTH – define the width and height (pixels/lines) of the source video frame.
4. Source SL2 Circular Buffer Configuration
   a. VPAC_MSC_LSE_SRC_BUF_ATTR_j[15-6] BUF_STRIDE – defines the line stride size (must be 64 byte multiple)
   b. VPAC_MSC_LSE_SRC_BUF_ATTR_j[24-16] CBUF_SIZE – define the size of the circular buffer in the SL2 (number of lines)
   c. VPAC_MSC_LSE_SRC_BUF_ATTR_j[31-25] START_NIB_OFFSET – defines the line start offset in smaller resolution (start address within the first SL2 512-bit word in 4-bit resolution).
5. Source BA and Enable Configuration
   a. VPAC_MSC_LSE_SRC_i_BUF_BA_y[23-6] ADDR – defines the SL2 base address of the Circular buffer for the input channel. VPAC_MSC_LSE_SRC_i_BUF_BA_y[31] ENABLE – enables the input channel.

7.7.5.7.1.3.2 Output Channel Configuration

1. Pixel Data Format
   a. VPAC_MSC_LSE_DST_BUF_CFG_j[1-0] PIX_FMT_PW, VPAC_MSC_LSE_DST_BUF_CFG_j[3-2] PIX_FMT_CNTRSZ, and VPAC_MSC_LSE_DST_BUF_CFG_j[4] PIX_FMT_ALIGN parameters define the pixel output data format for this output channel.
   b. Common data formats:
      i. Fully packed 12-bit: PIX_FMT_PW = 1, PIX_FMT_CNTRSZ = 1, PIX_FMT_ALIGN = 0
      ii. Fully packed 8-bit: PIX_FMT_PW = 0, PIX_FMT_CNTRSZ = 0, PIX_FMT_ALIGN = 0
2. Output Thread Map
   a. VPAC_MSC_LSE_DST_BUF_CFG_j[7] THREAD_MAP defines the output channel thread mapping.
3. Output SL2 Circular Buffer Configuration
   a. VPAC_MSC_LSE_DST_BUF_ATTR0_j[15-6] BUF_STRIDE – define the line stride size (must be 64 byte multiple)
   b. VPAC_MSC_LSE_DST_BUF_ATTR0_j[24-16] CBUF_SIZE – define the size of the circular buffer in the SL2 (number of lines)
4. Base Address and Channel Enable:
   a. VPAC_MSC_LSE_DST_BUF_BA_j[23-6] ADDR – defines the SL2 base address of the Circular buffer for this output channel.
   b. VPAC_MSC_LSE_DST_BUF_BA_j[31] ENABLE – enables this output channel.

7.7.5.7.1.4 MSC HTS Programming Details

There is no configuration required for HTS interface in the MSC_HWA. All programming is done in the separate HTS module. It is expected that at least one input and output channel is enabled for each HTS thread. For more details on programming HTS, see *Hardware Accelerator (HWA) Thread Scheduler (HTS)*.

Copyright © 2026 Texas Instruments Incorporated

7.7.5.7.1.5 MSC Data Transfer Programming Details

Refer to *VPAC Subsystem Programmer's Guide*.

7.7.5.7.1.6 LSE Interrupt Programming

MSC does not have any interrupt MASK/SET/CLEAR registers . Interrupts are pulse output events that go to the VPAC top level interrupt aggregation logic. For more information about the aggregation logic, see Section 7.7.2, *VPAC Subsystem*.

MSC does, however, provide a set of statuses for VBUSM interface errors. These status are mapped to VPAC_MSC_STATUS_ERROR and they are cleared by writing all 1's to the status bit field.

7.7.5.7.2 Initialization Sequence

For details, see Section 7.7.2, *VPAC Subsystem Level*.

7.7.5.7.3 Real-Time Operating Requirements

For details, see Section 7.7.2, *VPAC Subsystem Level*.

7.7.5.7.4 Power Up/Down Sequence

For details, see Section 7.7.2, *VPAC Subsystem Level*.

Copyright © 2026 Texas Instruments Incorporated

7.8 Depth and Motion Perception Accelerator (DMPAC)

This chapter describes the Depth and Motion Perception Accelerator (DMPAC) in the device.

7.8.1 DMPAC Overview

The Depth and Motion Perception Accelerator (DMPAC) is a power efficient hardware accelerator that computes dense stereo depth maps (*depth*) and dense optical flow vectors (*motion*) from camera inputs.

The image/video sensor-based environmental perception (also known as scene understanding) is at the core of many emerging applications in automotive, industrial and consumer electronics. Typically, this involves detection of all objects in the scene along with their 3D position and motion with regards to the observer or the car by analyzing one or many related input video streams. Various computer vision algorithms are used to achieve these tasks.

A very robust method of obtaining the 3D depth from images is to use two cameras in a stereo setup - two cameras with known relative positions and camera parameters. The two images of the same scene, captured from two different camera poses/perspectives, are analyzed to find disparities among every pixel positions in the images. This is known as the Stereo Disparity map. The disparity values of every pixel can be used to obtain the 3D positions of the object/space they belong to via triangulation.

On the other hand, by analyzing two images from a single camera, captured at two different time instances (that is, two temporal frames in a video), one can determine where each pixel in a past frame moved to in the future frame. This is known as the Optical Flow vector. The flow vectors for each pixel position can be used to obtain 3D structure of the scene, identify moving objects and determine their relative speed and direction of motion.

The DMPAC is dedicated to the aforesaid image processing tasks. The stereo and optical flow processing is partitioned into two top level sub-blocks: the Dense Optical Flow (DOF) engine and the Stereo Disparity Engine (SDE). The DOF and SDE blocks share a common shared local memory, DMA, external messaging and control infrastructure.

7.8.1.1 DMPAC Features

The DMPAC supports the following main features:

- Input image resolution up to 2 MPix with maximum horizontal resolution of up to 2048 pixels and maximum vertical resolution of up to 1024 pixels
- Input pixel format of 12-bit fully packed luminance data, and other formats using the integrated Format Conversion (FOCO) module
- Pixel level output packed in 16bpp and 32bpp formats for stereo disparity estimates and optical flow estimates, respectively
- Simultaneous operation of SDE and DOF, each processing inputs of up to 1 MPix image resolution (1 MPix = 1280 × 720 pixels)
- Resolution vs. frame rate scalability (higher frame rate at lower resolution)
- A Dense Optical Flow (DOF) engine: The DOF engine implements Texas Instruments' proprietary algorithm for estimation/calculation of the optical flow between the input image pair. The algorithm uses hierarchical coarse-to-fine block search strategy leveraging image pyramids in which proprietary binary pixel descriptors are used for pixel correspondence determination. In the scheme accuracy and smoothness of the flow map is enforced using optical flow estimates of the causal neighbors of a pixel in the given and higher (lower resolution) pyramid level (pixels are processed in raster scan order in a pyramid level to refine existing optical flow estimates). The DOF engine supports:
  - Optical flow vector lengths up to +/- 191 pixels in horizontal direction and +/- 62 pixels in vertical direction for each input pixel
  - Dense flow vector map for each input pixel
  - Optical flow vector precision of $1/16^{th}$ of inter pixel distance
  - Post filtering of the optical flow estimates using 2D median filtering
  - 16 level confidence score for every output flow vector
- A Stereo Disparity Engine (SDE): The SDE implements Texas Instruments' proprietary algorithm in order to find the disparity map between the input image pair. The SDE supports:

Copyright © 2026 Texas Instruments Incorporated

  - Disparity search range (SR) of 64/128/192. It can support either "0 to SR-1" or "-3 to SR-4"
  - 1/16th pixel accurate sub-pixel level disparity estimate
  - Disparity estimate post processing using 2D median filtering
  - 8 level confidence score for each disparity output
- Common top level infrastructure:
  - Shared Level 2 (SL2) memory sub-system, which serves both DOF and SDE blocks, along with the FOCO modules when required as an intermediate storage exchange data across sub-modules (FOCO to DOF/SDE) and from DDR/System Memory
  - A Unified Transfer Controller (UTC), which serves as a DMA engine
  - Messaging and control mechanism via a Hardware Thread Scheduler (HTS) block
  - A Counter, Timer and System Event Trace (CTSET) module, which provides event tracing capability for the SDE and DOF hardware threads

The DMPAC does not support (but relies on other HWAs/processors on the SoC to perform these tasks):

- Epipolar rectification of the stereo input images
- Image pyramid generation for the optical flow input images

7.8.1.2 DMPAC Not Supported Features

The DMPAC does not support the following features:

- Arbitrary binary feature descriptor as input
- Flexible block matching on any image/feature vector plane
- Standalone use of any of the DMPAC internal modules by any SoC level resources, even when the DMPAC is not in use

The following processing steps, although essential parts of the stereo and optical flow algorithms supported inside DMPAC, are expected to be handled outside the DMPAC module:

- Epipolar Rectification on both the stereo input image streams
- 6 levels (including base) of Image Pyramid generation on the optical flow input image stream

7.9 Graphics Accelerator (GPU)

This chapter describes the integration of the Graphics Processing Unit (GPU) subsystem in the device.

7.9.1 GPU Overview

The Graphics Processing Unit (GPU) accelerates 3-dimensional (3D), 2-dimensional (2D) graphics, and compute applications.

The GPU is based on the PowerVR® BXS 4-64 MC1 core from Imagination Technologies.

7.9.1.1 Features Supported

- Base architecture, fully compliant with the following APIs:
  - OpenGL ES 3.2
  - OpenCL 1.2 EP
  - Vulkan 1.2
- Tile-based deferred rendering architecture for 3D graphics workloads, with concurrent processing of multiple tiles
- Support for DRM security
- Support for GPU virtualization
  - Up to 8 virtual GPUs
  - Separate IRQ for each OSID
- Multi-threaded Unified Shading Cluster (USC) engine incorporating pixel shader, vertex shader and GP-GPU (compute shader) functionality
- USC incorporates an ALU architecture with high SIMD efficiency
- Fully virtualized memory addressing (up to 64 GB address space), supporting unified memory architecture
- Fine-grained task switching, workload balancing and power management

Copyright © 2026 Texas Instruments Incorporated

- Advanced DMA driven operation for minimum host CPU interaction
- 256KB System Level Cache (SLC)
- Specialized Texture Cache Unit (TCU)
- Compressed Texture Decoding
- Lossless data compression (PVRGC) - The PowerVR's geometry compression, which is performed in the Geometry Processing phase of the 3D graphics workload.
- Dedicated RISC-V processor for firmware execution
- Separate power island for the firmware processor for low latency power domain transitions
- On-Chip Performance, Power and Statistics Registers.
- Resolution Support
  - Frame buffer max size = 8K × 8K
  - Texture max size = 8K × 8K
- Anti-aliasing
  - Maximum 4× multisampling
- Performance
  - Floating Point Operations (F32) - 64 operations per clock
  - Floating Point Operations (F16) - 128 operations per clock
  - Texture performance - 4 texels per clock (@32 BPP)
  - Pixel performance - 4 pixel(s) per clock (@32 BPP)
  - Geometry Performance - 0.25 triangles per clock
  - Max performance
    - ~50 GFLOPS, 3.2 GPix/s @ 800MHz

Note

Some features may not be available. See *Module Integration* for more information.

7.9.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Copyright © 2026 Texas Instruments Incorporated

Chapter 8

Inter-processor Communication Scheme (IPC)

Device has multiple hardware mechanisms to provide inter-processor level communication.

8.1 Mailbox 790
8.2 Spinlock 797
8.3 Secure Proxy (SEC_PROXY) 801

Copyright © 2026 Texas Instruments Incorporated

8.1 Mailbox

This chapter describes the Mailbox module of the device.

8.1.1 Mailbox Overview

Mailbox module serves to facilitate the communication between the various on-chip processors of the device by providing a queued mailbox-interrupt mechanism.

The queued mailbox-interrupt mechanism allows the software to establish a communication channel between two processors (users) through a set of registers and associated interrupt signals.

8.1.1.1 Mailbox Features

Each mailbox module supports the following features:

- Parameters configured at design time:
  - Number of mailbox clusters
    - Each mailbox cluster has the same configuration
    - Clusters are accessed via separate VBUS regions. Clusters can be treated as mailbox module instances.
  - Number of users
  - Number of mailbox message queues
  - Number of messages (FIFO depth) for each message queue
- 32-bit message width
- Partial writes do not advance the FIFO pointers
- Message reception and queue-not-full notification using interrupts
- Non-intrusive emulation

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

8.1.2 Mailbox Functional Description

The mailbox module provides a means of communication through message queues among the users. The individual mailbox modules, or FIFOs, can associate (or de-associate) with any of the processors using the MAILBOX_IRQ_ENABLE_SET_j (or MAILBOX_IRQ_ENABLE_CLR_j) register.

Each user has a dedicated interrupt signal from the corresponding mailbox module instance and dedicated interrupt enabling and status registers.

Each MAILBOX_IRQ_STATUS_RAW_j/MAILBOX_IRQ_STATUS_CLR_j interrupt status register corresponds to a particular user.

8.1.2.1 Mailbox Block Diagram

Figure 8-1 shows the mailbox internal block diagram.

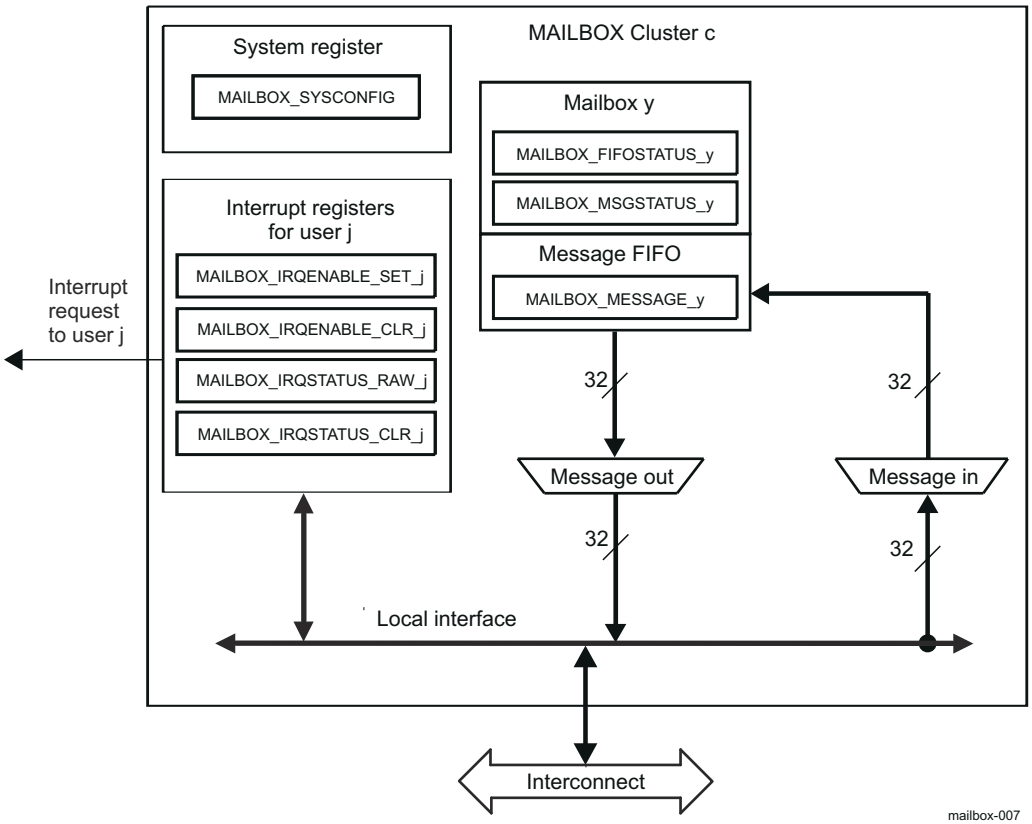


Figure 8-1. Mailbox Block Diagram

c = 0 to 11

j = 0 to 3

y = 0 to 15

8.1.2.2 Mailbox Software Reset

The mailbox module supports a software reset through the MAILBOX_SYSCONFIG[0] SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Reading the MAILBOX_SYSCONFIG[0] SOFTRESET bit gives the status of the software reset:

- Read 1: the software reset is on-going.
- Read 0: the software reset is complete.

The software must ensure that the software reset completes before doing mailbox operations.

Copyright © 2026 Texas Instruments Incorporated

8.1.2.3 Mailbox Power Management

The clkstop_idle will be asserted if all of the following are true:

- The VBUSP interface is inactive
- All mailboxes are empty
- There are no enabled events or outstanding interrupts

An enabled event means there is pending action required from one of the users, and it means one or more mailboxes still expect data.

8.1.2.4 Mailbox Interrupt Requests

An interrupt request allows the user of the mailbox to be notified when a message is received or when the message queue is not full. There is one interrupt per user.

Table 8-1 lists the event flags, and their mask, that can cause module interrupts.

Table 8-1. Interrupt Events

Non-Maskable Event Flag(1)	Maskable Event Flag	Event Mask Bit	Event Unmask Bit	Description
MAILBOX_IRQ_STATUS_RAW_j[0+y*2] NEWMSGSTATUSMBy	MAILBOX_IRQ_STATUS_CLR_j[0+y*2] NEWMSG_STATUSMBy	MAILBOX_IRQ_ENABLE_CLR_j[0+y*2] NEWMSG_STATUSMBy	MAILBOX_IRQ_ENABLE_SET_j[0+y*2] NEWMSG_STATUSMBy	Mailbox *y* receives a new message.
MAILBOX_IRQ_STATUS_RAW_j[1+y*2] NOTFULLSTATUSMBy	MAILBOX_IRQ_STATUS_CLR_j[1+y*2] NOTFULLSTATUSMBy	MAILBOX_IRQ_ENABLE_CLR_j[1+y*2] NOTFULLSTATUSMBy	MAILBOX_IRQ_ENABLE_SET_j[1+y*2] NOTFULLSTATUSMBy	Mailbox *y* message queue is not full.

(1) MAILBOX_IRQ_STATUS_RAW_j register is mostly used for debug purposes.

CAUTION

Once an event generating the interrupt request has been processed by the software, it must be cleared by writing a logical 1 in the corresponding bit of the MAILBOX_IRQ_STATUS_CLR_j register.

Writing a logical 1 in a bit of the MAILBOX_IRQ_STATUS_CLR_j register will also clear to 0 the corresponding bit in the appropriate MAILBOX_IRQ_STATUS_RAW_j register.

An event can generate an interrupt request when a logical 1 is written to the corresponding unmask bit in the MAILBOX_IRQ_ENABLE_SET_j register. Events are reported in the appropriate MAILBOX_IRQ_ENABLE_CLR_j and MAILBOX_IRQ_STATUS_RAW_j registers.

An event stops generating interrupt requests when a logical 1 is written to the corresponding mask bit in the MAILBOX_IRQ_ENABLE_CLR_j register. Events are only reported in the appropriate MAILBOX_IRQ_STATUS_RAW_j register.

In case of the MAILBOX_IRQ_STATUS_RAW_j register, the event is reported in the corresponding bit even if the interrupt request generation is disabled for this event.

8.1.2.5 Mailbox Assignment

8.1.2.5.1 Description

To assign a receiver to a mailbox, set the new message interrupt enable bit corresponding to the desired mailbox in the MAILBOX_IRQ_ENABLE_SET_j register. The receiver reads the MAILBOX_MESSAGE_y register to retrieve a message from the mailbox.

An alternate method for the receiver that does not use the interrupts is to poll the MAILBOX_FIFO_STATUS_y and/or MAILBOX_MSG_STATUS_y registers to know when to send or retrieve a message to or from the mailbox. This method does not require assigning a receiver to a mailbox. Because this method does not include the explicit assignment of the mailbox, the software must avoid having multiple receivers use the same mailbox, which can result in incoherency.

Copyright © 2026 Texas Instruments Incorporated

To assign a sender to a mailbox, set the queue-not-full interrupt enable bit of the desired mailbox in the MAILBOX_IRQ_ENABLE_SET_j register, where *u* is the number of the sending user. However, direct allocation of a mailbox to a sender is not recommended because it can cause the sending processor to be constantly interrupted.

It is recommended that register polling be used to:

- Check the status of either the MAILBOX_FIFO_STATUS_y or MAILBOX_MSG_STATUS_y registers
- Write the message to the corresponding MAILBOX_MESSAGE_y register, if space is available.

The sender might use the queue-not-full interrupt when the initial mailbox status check indicates the mailbox is full. In this case, the sender can enable the queue-not-full interrupt for its mailbox in the appropriate MAILBOX_IRQ_ENABLE_SET_j register. This allows the sender to be notified by interrupt only when a FIFO queue has at least one available entry.

Reading the MAILBOX_IRQ_STATUS_CLR_j register determines the status of the new message and the queue-not-full interrupts for a particular user. Writing 1 to the corresponding bit in the MAILBOX_IRQ_STATUS_CLR_j register acknowledges, and subsequently clears, an interrupt.

CAUTION

Assigning multiple senders or multiple receivers to the same mailbox is not recommended.

8.1.2.6 Sending and Receiving Messages

8.1.2.6.1 Description

When a 32-bit message is written to the MAILBOX_MESSAGE_y register, the message is appended into the FIFO queue. This queue holds four messages. If the queue is full, the message is discarded.

Queue overflow can be avoided by first reading the MAILBOX_FIFO_STATUS_y register to check that the mailbox message queue is not full before writing a new message to it.

Reading the MAILBOX_MESSAGE_y register returns the message at the beginning of the FIFO queue and removes it from the queue. If the FIFO queue is empty when the MAILBOX_MESSAGE_y register is read, the value 0 is returned.

The new message interrupt is asserted when at least one message is in the mailbox message FIFO queue. To determine the number of messages in the mailbox message FIFO queue, read the MAILBOX_MSG_STATUS_y register.

Copyright © 2026 Texas Instruments Incorporated

8.1.2.7 Example of Communication

Figure 8-2 shows an example of communication between two processors.

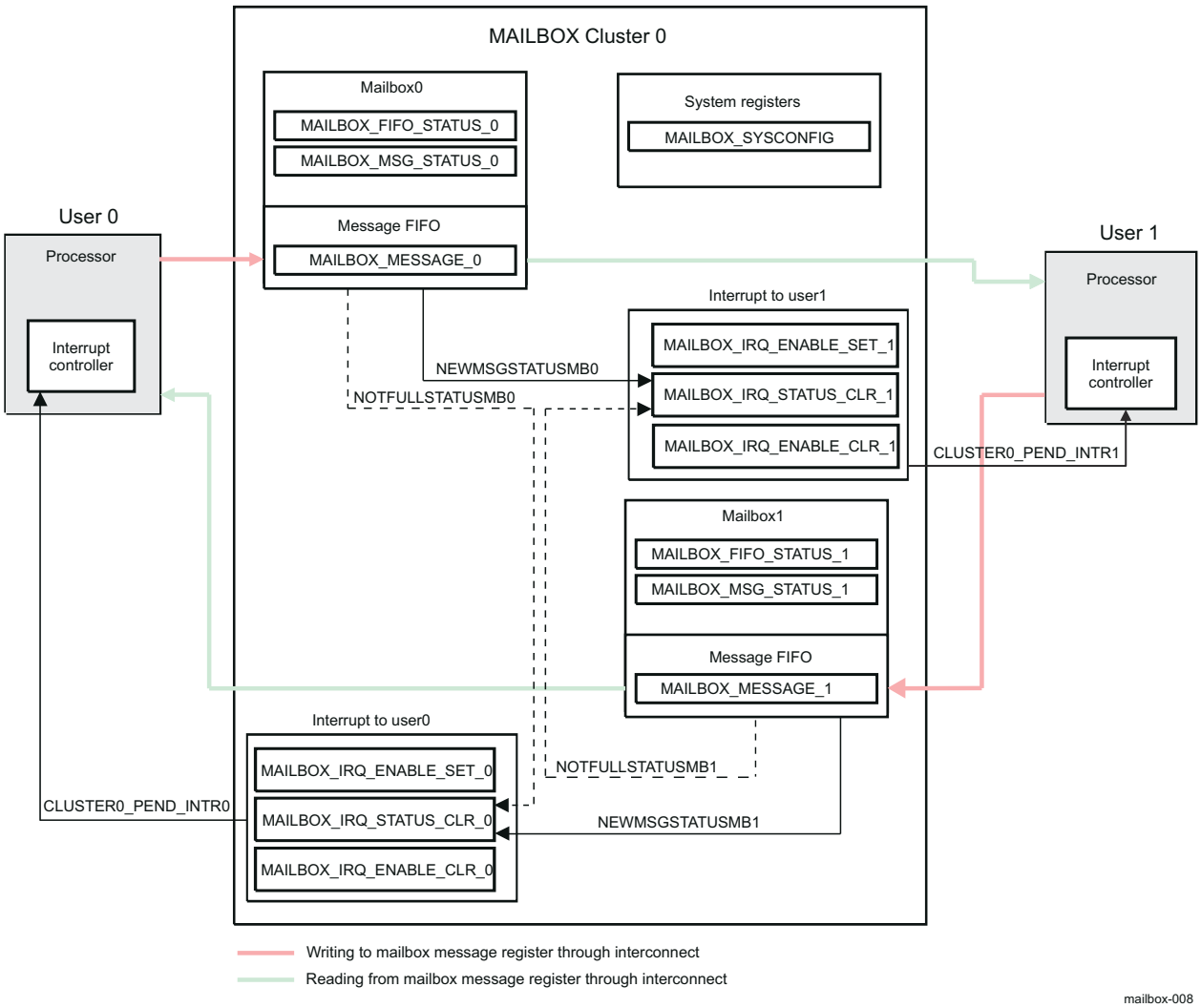


Figure 8-2. Example of Communication

Copyright © 2026 Texas Instruments Incorporated

8.1.3 Mailbox Programming Guide

8.1.3.1 Mailbox Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the mailbox module.

8.1.3.1.1 Global Initialization

8.1.3.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the mailbox module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the mailbox.

See *Mailbox Integration*, for further information.

Table 8-2. Global Initialization of Surrounding Modules for MAILBOX

Surrounding Modules	Comments
LPSC	The MAILBOX functional and interface clock must be enabled.
Interrupt Controllers	Processor's (user's) interrupt controller must be configured to enable the interrupt request generation.

8.1.3.1.2 Mailbox Global Initialization

8.1.3.1.2.1 Main Sequence - Mailbox Global Initialization

This procedure initializes the mailbox module after a power-on or software reset.

Table 8-3. Mailbox Global Initialization

Step	Register/Bit Field/Programming Model	Value
Perform a software reset	MAILBOX_SYSCONFIG[0] SOFT_RESET	0x1
Wait until reset is complete	MAILBOX_SYSCONFIG[0] SOFT_RESET	=0x0

8.1.3.1.2 Mailbox Operational Modes Configuration

8.1.3.1.2.1 Mailbox Processing modes

8.1.3.1.2.1.1 Main Sequence - Sending a Message (Polling Method)

Table 8-4. Sending a Message (Polling Method)

Step	Register/Bit Field/Programming Model	Value
IF: Is FIFO full?	MAILBOX_FIFO_STATUS_y[0] FULL	=0x1
Wait until at least one message slot is available	MAILBOX_FIFO_STATUS_y[0] FULL	=0x0
ELSE		
Write message	MAILBOX_MESSAGE_y[31:0] VALUE	0x-
ENDIF		

8.1.3.1.2.1.2 Main Sequence - Sending a Message (Interrupt Method)

Table 8-5. Sending a Message (Interrupt Method)

Step	Register/Bit Field/Programming Model	Value
IF: Is FIFO full?	MAILBOX_FIFO_STATUS_y**MAILBOX_FIFO_STATUS_y**[0] FULL	=0x1
Enable interrupt event	MAILBOX_IRQ_ENABLE_SET_j**MAILBOX_IRQ_ENABLE_SET_j**[1+ y*2]	0x1
User (processor) can perform another task until interrupt occurs See Section 8.1.3.1.3.1 for interrupt handling in sending mode		
ELSE		

Copyright © 2026 Texas Instruments Incorporated

Table 8-5. Sending a Message (Interrupt Method) (continued)

Step	Register/Bit Field/Programming Model	Value
Write message	MAILBOX_MESSAGE_y**MAILBOX_MESSAGE_y**[31:0] VALUE	0x-
ENDIF		

8.1.3.1.2.1.3 Main Sequence - Receiving a Message (Polling Method)

Table 8-6. Receiving a Message (Polling Method)

Step	Register/Bit Field/Programming Model	Value
IF: Number of messages is not equal to 0	MAILBOX_MSG_STATUS_y[2:0] NUM_MESSAGES	≠0x0
Read message	MAILBOX_MESSAGE_y[31:0] VALUE	0x-
ENDIF		

8.1.3.1.2.1.4 Main Sequence - Receiving a Message (Interrupt Method)

Table 8-7. Receiving a Message (Interrupt Method)

Step	Register/Bit Field/Programming Model	Value
Enable interrupt event	MAILBOX_IRQ_ENABLE_SET_j[0 + y*2]	0x1
User (processor) can perform another task until interrupt occurs See Section 8.1.3.1.3.2 for interrupt handling in receiving mode		

8.1.3.1.3 Mailbox Events Servicing

8.1.3.1.3.1 Events Servicing in Sending Mode

Table 8-8 describes the events servicing in sending mode.

Table 8-8. Events Servicing in Sending Mode

Step	Register/Bit Field/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQ_STATUS_CLR_j[1 + y*2]	0x1
Write message	MAILBOX_MESSAGE_y[31:0] VALUE	0x--
Write 1 to acknowledge interrupt	MAILBOX_IRQ_STATUS_CLR_j[1 + y*2]	0x1

8.1.3.1.3.2 Events Servicing in Receiving Mode

Table 8-9 describes the events servicing in receiving mode.

Table 8-9. Events Servicing in Receiving Mode

Step	Register/Bit Field/Programming Model	Value
Read interrupt status bit	MAILBOX_IRQ_STATUS_CLR_j[0 + y*2]	0x1
IF: Number of messages is not equal to 0?	MAILBOX_MSG_STATUS_y[2:0] NUM_MESSAGES	≠0x0
Read message	MAILBOX_MESSAGE_y[31:0] VALUE	0x--
ELSE		
Write 1 to acknowledge interrupt	MAILBOX_IRQ_STATUS_CLR_j[0 + y*2]	0x1
ENDIF		

Copyright © 2026 Texas Instruments Incorporated

8.2 Spinlock

This chapter describes the Spinlock module of the device.

8.2.1 Spinlock Overview

The Spinlock module provides hardware assistance for synchronizing the processes running on multiple processors in the device.

The Spinlock module implements 256 spinlocks (or hardware semaphores), which provide an efficient way to perform a lock operation of a device resource using a single read-access, avoiding the need of a read-modify-write bus transfer that the programmable cores are not capable of.

Figure 8-3 shows an overview of the Spinlock module.

Figure 8-3. Spinlock Overview

Note

Some features may not be available. See *Module Integration* for more information.

8.2.2 Spinlock Functional Description

8.2.2.1 Spinlock Software Reset

The Spinlock module can be reset by software through the SPINLOCK_SYSCONFIG[1] SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. The SPINLOCK_SYSTATUS[0] RESETDONE bit can be polled to check the reset status (reading 1 indicates that reset sequence is done; reading 0 indicates that reset sequence is in progress). The software must ensure that the software reset completes before doing Spinlock operations.

8.2.2.2 Spinlock Power Management

The Spinlock module supports the Power Idle interface. Software must arrange to only power off the Spinlock module when no locks would be lost. In general, the steps to powering down the Spinlock module are:

1. Software must check that all controllers that may be using the Spinlock module are either:
   a. Already powered off (all in-use flags are 0)
   b. Notified that Spinlock is not available and the notification is acknowledged
2. If desired, check that no locks are currently held in the Spinlock module. The status of each bank of 32 locks can be read from the SPINLOCK_SYSTATUS register. If any locks are held, they are orphaned because they are not held by any controller that is still active. Alternatively, you may decide to wait a timeout period to allow any active controller to clean up its locks before powering down.

The Spinlock module may now be powered off.

8.2.2.3 About Spinlocks

Spinlocks are present to solve the need for synchronization and mutual exclusion between heterogeneous processors and those not operating under a single, shared operating system. There is no alternative mechanism to accomplish these operations between processors in separate subsystems.

Spinlocks are not the best way to synchronize between tasks or threads on one CPU. Instead, spinlocks are for use in synchronization between different subsystems in the device that don't have any other means of hardware-based synchronization.

Spinlocks do not solve all system synchronization issues. They have limited applicability and should be used with care to implement higher level synchronization protocols.

A spinlock is appropriate for mutual exclusion for access to a shared data structure. It should be used only when:

1. The time to hold the lock is predictable and small (for example, a maximum hold time of less than 200 CPU cycles may be acceptable).

Copyright © 2026 Texas Instruments Incorporated

2. The locking task cannot be preempted, suspended, or interrupted while holding the lock (this would make the hold time large and unpredictable).
3. The lock is lightly contended, that is the chance of any other process (or processor) trying to acquire the lock while it is held is small.

If these conditions are met, then the locking code can retry a failed attempt to acquire the lock until success.

If the conditions are not met, then a spinlock is not a good candidate. One alternative is to use a spinlock for critical section control (engineered to meet the conditions) to implement a higher level semaphore that can support preemption, notification, timeout or other higher level properties.

8.2.2.4 Spinlock Functional Operation

The Spinlock module supports 256 spinlocks. It accepts only a single command at a time and processes the command fully before accepting the next command. A lock is requested by reading the SPINLOCK_LOCK_REG_i[0] TAKEN bit. There are two states: Taken (SPINLOCK_LOCK_REG_y[0] TAKEN = 1) or Not Taken (SPINLOCK_LOCK_REG_y[0] TAKEN = 0), where y = 0h to FFh.

When the status of lock *y* (where y = 0 to 255) is Not Taken (free), a read from the SPINLOCK_LOCK_REG_y register returns 0 and sets the lock to Taken (locked). When the status of lock *y* is Taken, a read returns 1 and does not change the state of the lock.

A write to the SPINLOCK_LOCK_REG_y register does not change the state of lock, unless when writing 0 when the lock is in Taken state. By doing this, the requester frees the lock.

Figure 8-4 shows the SPINLOCK_LOCK_REG_y register state diagram.

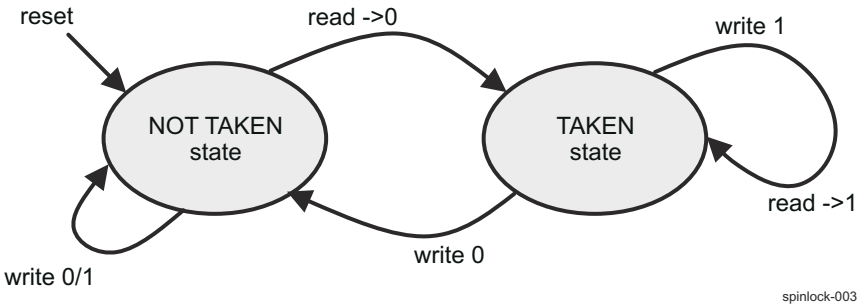


Figure 8-4. Lock Register State Diagram

Note

- There is no support to ensure that a lock register is locked and unlocked by the same process. This must be ensured in software.
- There is no support to check that the same initiator that acquired the lock is the one that is freeing the lock.

Copyright © 2026 Texas Instruments Incorporated

8.2.3 Spinlock Programming Guide

8.2.3.1 Spinlock Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the module.

8.2.3.1.1 Surrounding Modules Global Initialization

This procedure initializes the surrounding modules when the Spinlock module is used for the first time after a device reset.

Table 8-10. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
LPSC	Spinlock functional and interface clock must be enabled. For more information, see *Clocking*

8.2.3.1.2 Basic Spinlock Operations

The main Spinlock operations are:

- Clear all the Taken spinlocks (only after a system bug recovery)
- Take a spinlock
- Release spinlock

8.2.3.1.2.1 Spinlocks Clearing After a System Bug Recovery

Module initialization (after reset) is not needed, except after system bug recovery. The following table presents the Spinlock initialization after a system bug recovery. Software should store 0 into each of the SPINLOCK_LOCK_REG_y registers at system startup to insure that all locks are initialized to Not Taken.

Table 8-11. Spinlock System Bug Recovery

Step	Register	Value
IF: SPINLOCK_SYSTATUS[0] IU0 = 1?	SPINLOCK_SYSTATUS[0] IU0	=1
Free the 256 locks	SPINLOCK_LOCK_REG_y[0] TAKEN (y = 0 to 255)	0x0
END		

8.2.3.1.2.2 Take and Release Spinlock

This procedure configures the take and release (free) operations for the Spinlock module. A spinlock should only be held with interrupts disabled. So, before attempting to obtain the spinlock, software must disable interrupts. Then it must read the SPINLOCK_LOCK_REG_y[0] TAKEN bit to attempt to obtain the lock. If it succeeds, it must proceed directly through the critical section then unlock and re-enable interrupts. If the acquisition attempt fails, the acquisition must be reattempted. To prevent unknown interrupt disabled time, interrupts must be re-enabled and then disabled before reattempting to acquire the lock. Figure 8-5 shows the described above procedure.

Copyright © 2026 Texas Instruments Incorporated

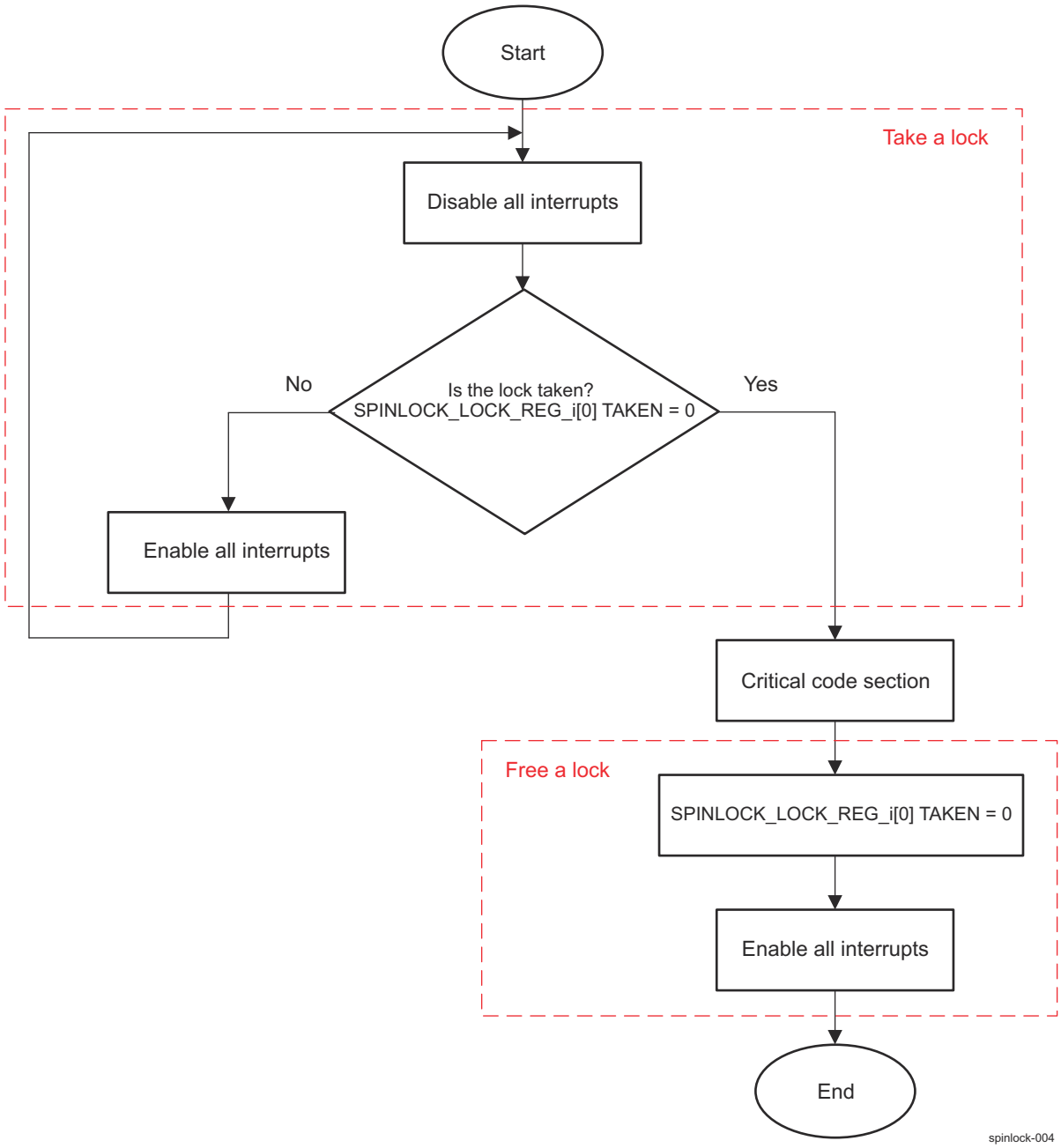


Figure 8-5. Take and Release Spinlock

Table 8-12. Register Call Summary

Register Name
SPINLOCK_LOCK_REG_y[0] TAKEN

Table 8-13. Subprocess Call Summary

Subprocess Name	Description
Disable (Mask) All Interrupts	For information about disabling/enabling all interrupts in an Arm® processor, refer to Arm *Technical Reference Manual*, available at infocenter.arm.com/help/index.jsp. For information about disabling/enabling all interrupts in other processors, refer to the corresponding processor chapter.
Enable (Unmask) All Interrupts	

Copyright © 2026 Texas Instruments Incorporated

8.3 Secure Proxy (SEC_PROXY)

The third IPC hardware mechanism is to use SEC_PROXY, which provides an IPC mechanism which carries a large message for complex communications among processors. It allows processors to read and write a message in small chunks.

There are two sets of SEC_PROXY in the device, one SEC_PROXY is dedicated for IPC for HSM M4F, and the second SEC_PROXY is for the rest. When the SEC_PROXY generates events, an IPC message is received or sent, and events are sent to interrupt aggregators before those events are converted into interrupts to the processors. The utilization of SEC_PROXY is user defined through memory map allocation. Each processor can be allocated with its own SEC_PROXY region to receive and send IPC messages.

Copyright © 2026 Texas Instruments Incorporated

Chapter 9

Memory Controllers

9.1 DDR Subsystem (DDRSS).....803

Copyright © 2026 Texas Instruments Incorporated

9.1 DDR Subsystem (DDRSS)

This section describes the DDR Subsystem (DDRSS) for the device.

9.1.1 DDRSS Overview

The DDR subsystem in this device comprises DDR controller, DDR PHY and wrapper logic to integrate these blocks in the device. The DDR subsystem is referred to as DDRSS0 and is used to provide an interface to external SDRAM devices which can be utilized for storing program or data. DDRSS0 is accessed via CBASS0 interconnect.

The DDRSS0 supports:

- Memory Types:
  - LPDDR4 up to @ 0.75V or 0.85V VDD_CORE
- Memory Bus Features:
  - Up to 32-bit width with in-line ECC
  - Up to 2 ranks
  - LPDDR4 densities up to 8GBytes (4GBytes each rank)
- Two System Bus Interfaces - one for real time and another for non-real time traffic:
  - 256-bit data width
  - Little endian only
  - Address aliasing prevention to block accesses to unpopulated SDRAM region
  - Clock asynchronous to DDR clock
- Configuration Bus Interface:
  - 32-bit data width
  - Linear incrementing addressing mode
  - 32-bit aligned accesses only
  - Little endian only
  - Clock asynchronous to DDR clock
- Key Features:
  - Full coherency across all commands
  - Bank interleaving
  - Priority based scheduling
  - Scheduling based on bank openness
  - Class of Service (CoS) - Three latency classes supported
  - Leaky bucket function to avoid blocking of Low Priority Thread
  - Drain function to expedite execution of reads in the controller
  - Read/write scheduling to avoid turn-around time
  - Prioritized refresh scheduling
  - Dynamic change of refresh rate via software for extended temperatures
  - Statistical counters for performance management
- SDRAM ECC Features:
  - In-line ECC
  - 64-byte ECC calculated over 512-byte data
  - Read-modify-write ECC for sub-word writes
  - Support ECC cache to improve in-line ECC performance
  - Support 64 cache-lines each 512-byte wide
  - In-line ECC performance impact shall be minimum of 12.5% for linear cached traffic
  - ECC address error logging
  - Statistical counters for counting ECC errors
- Low Power Features:
  - All power modes defined by JEDEC (clock stop for LPDDR4, self-refresh, power-down, etc.)
  - Self-refresh entry and exit via software or hardware clock stop request/acknowledge
  - System bus clock stop via hardware clock stop request/acknowledge when controller is idle

Copyright © 2026 Texas Instruments Incorporated

  - Automatic idle power saving mode when no or low activity is detected
  - DDR and system bus clock frequency change using self-refresh via software or hardware clock stop request/acknowledge
  - Turning off SoC power after DDR is put into self-refresh (DDR reset and CKE I/O retention)
  - Tri-stating of all DDR I/O cells via software while driving CKE and RESETn pins during self-refresh
  - LPDDR4 Frequency Set Point (FSP)
  - Support power switches on DDR subchip for low leakage power in low-power modesSupport power switches on DDR subchip for low leakage power in low-power modes
- Functional Safety Features:
  - VBUSM2AXI bridge AXI bus timeout
  - DDR4 command bus parity
- DDR PHY Features:
  - Automatic and software controllable initialization and calibration (ZQ) for the DDR PHY and I/O cells
  - Automatic and software controllable delay line calibrations with Voltage and Temperature (VT) compensation
  - Automatic and software controllable write levelling with VT compensation
  - Automatic read DQS gate training per rank with VT compensation
  - Automatic and software controllable DQ/DQS eye training per rank
  - Automatic and software controllable read and write data bit deskew
  - Automatic and software controllable Command/Address (CA) levelling with VT compensation for LPDDR4
  - Automatic and software controllable CA bit deskew for LPDDR4
  - Refreshes to SDRAM during leveling and training
  - No seeding requirement based on board topology for any of the leveling and training algorithms
  - Dynamic/automatic I/O Receiver disable when read transfer is not on going
  - Capability of disabling data macros and I/O cells when not in use

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

9.1.2 DDRSS Environment

Please refer to the AM62Ax DDR Board Design and Layout Guidelines application note for detailed information on DDR interface connections to LPDDR4 and DDR4 memory devices

Copyright © 2026 Texas Instruments Incorporated

Table 9-1 describes the DDRSS0 I/O signals used for connection to SDRAM devices.

Table 9-1. DDRSS0 I/O signals

Module Pin	Device Level Signal	I/O(1)	Description
RESETN	DDR0_RESET0_n	O	SDRAM reset
CK	DDR0_CK0	O	SDRAM differential clock pair
CKN	DDR0_CK0_n	O	
ALERTN	DDR0_ALERT_n	IO	SDRAM parity error output
A[13-0]	DDR0_A[13-0]	O	SDRAM address and command bus
WEN	DDR0_WE_n	O	SDRAM write enable
CASN/CSN[3]	DDR0_CAS_n/DDR0_CS3_n	O	SDRAM column address strobe (DDR4). SDRAM chip select (LPDDR4)
RASN/CSN[2]	DDR0_RAS_n/DDR0_CS2_n	O	SDRAM row address strobe(DDR4). SDRAM chip select(LPDDR4)
ACTN	DDR0_ACT_n	O	SDRAM activate
BA[1-0]	DDR0_BA[1-0]	O	SDRAM bank address
BG[1-0]	DDR0_BG[1-0]	O	SDRAM bank group
PAR	DDR0_PAR	O	SDRAM command parity
CSN[1-0]	DDR0_CS[1-0]_n	O	SDRAM chip select
ODT[1-0]	DDR0_ODT[1-0]	O	SDRAM on-die termination
CKE[1-0]	DDR0_CKE[1-0]	O	SDRAM CKE
DQ[31-0]	DDR0_DQ[31-0]	IO	SDRAM data bus
DM[3-0]	DDR0_DM[3-0]	IO	SDRAM data and mask/DBI
DQS[3-0]	DDR0_DQS[3-0]	IO	SDRAM data strobe
DQSN[3-0]	DDR0_DQS[3-0]_n	IO	SDRAM data strobe invert

(1) I = Input; O = Output

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

9.1.3 DDRSS Functional Description

9.1.3.1 Real Time and Non-Real Time Threads

The VBUSM2AXI supports two VBUSM interfaces:

- Real Time High Priority Thread (HPT)
- Non-Real Time Low Priority Thread (LPT)

HPT will have priority over LPT. In addition to the thread priority, the bridge will also reorder commands within a particular thread based on VBUSM priority. Therefore, the execution of commands from the command FIFO can be out-of-order. As a result, the read data can also be returned out-of-order.

The bridge will NOT maintain data coherency across threads. However, coherency within a particular thread will be maintained.

For low area implementation, the same FIFO resources are shared between HPT and LPT for write status and read data return interfaces. Therefore, system must ensure that there is no pushback on the write status and read data return interfaces by LPT to guarantee execution of HPT is not blocked.

9.1.3.2 Class of Service (CoS)

Commands arriving on LPT and HPT to the DDRSS0 carry the VBUSM priority whereas the DDR controller uses AXI priority. The VBUSM2AXI bridge has the following registers for flexible mapping of the VBUSM priority to DDR controllers's priority:

- Range match registers:
  - EMIF_SSCFG_V2A_R1_MAT_REG
  - EMIF_SSCFG_V2A_R2_MAT_REG
  - EMIF_SSCFG_V2A_R3_MAT_REG
- Priority map registers:
  - EMIF_SSCFG_V2A_LPT_DEF_PRI_MAP_REG
  - EMIF_SSCFG_V2A_LPT_R1_PRI_MAP_REG
  - EMIF_SSCFG_V2A_LPT_R2_PRI_MAP_REG
  - EMIF_SSCFG_V2A_LPT_R3_PRI_MAP_REG
  - EMIF_SSCFG_V2A_HPT_DEF_PRI_MAP_REG
  - EMIF_SSCFG_V2A_HPT_R1_PRI_MAP_REG
  - EMIF_SSCFG_V2A_HPT_R2_PRI_MAP_REG
  - EMIF_SSCFG_V2A_HPT_R3_PRI_MAP_REG

This allows the system to essentially create different classes of service based on the initiator (Route ID) and the priority of the commands.

Each thread has a set of corresponding Priority Map Registers to map the incoming priority on that thread to appropriate AXI priority. The default values of these registers are such that inherently the HPT traffic will have higher priority than the LPT traffic inside the controller. However, these settings can be changed via software per system needs.

The controller performs command arbitration based on AXI priority and the state of the DDR device (openness of banks, etc.). Commands with higher priority will be given execution preference over commands with lower priority. However, due to the state of the DDR device (openness of banks, etc.), a command with higher priority may not be executed before a lower priority command. Therefore, transactions within a thread can be reordered. In addition, the controller does not have any information on the thread type of the transactions. Therefore, commands can also be reordered between thread types. However, at all times the controller maintains coherency across all commands within a given thread.

Copyright © 2026 Texas Instruments Incorporated

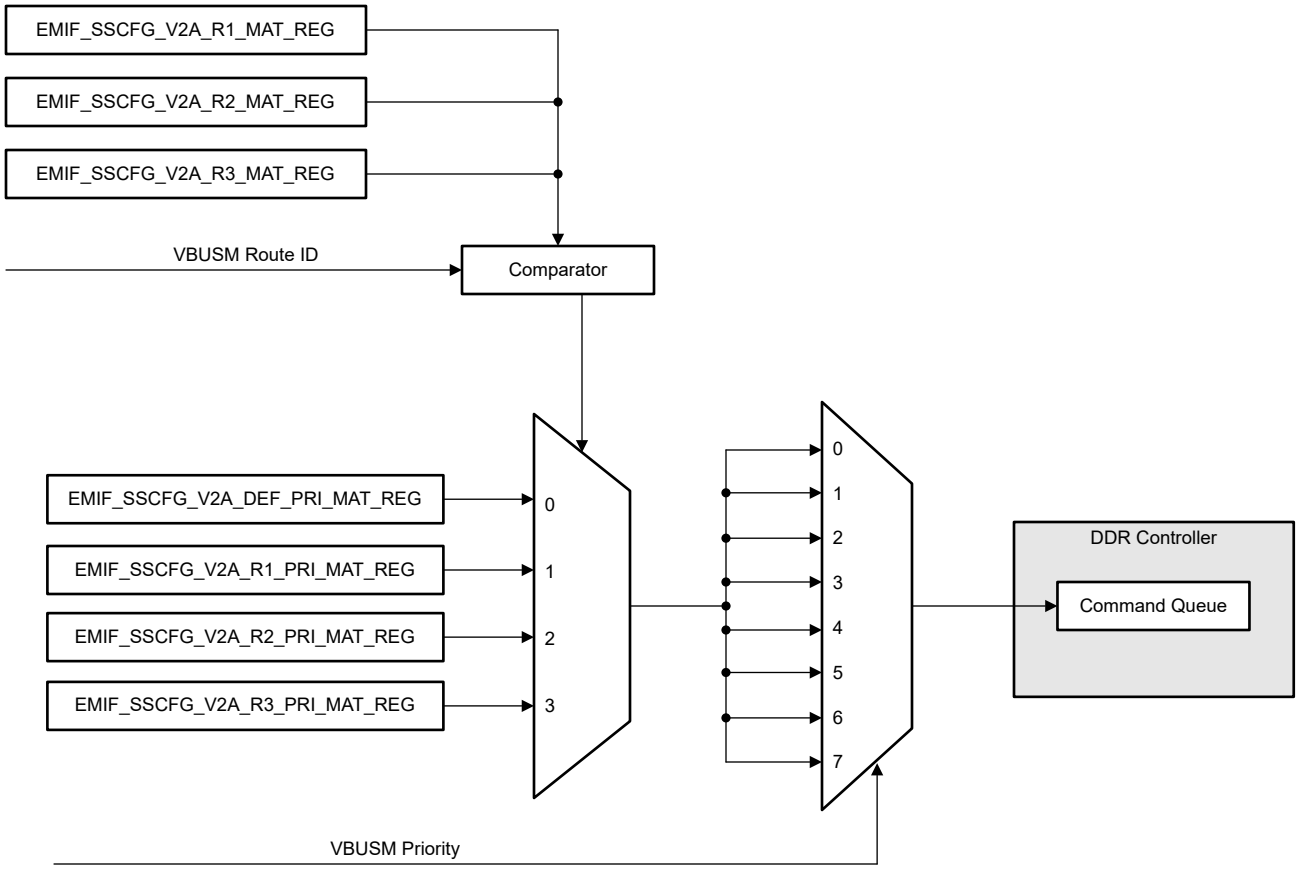


Figure 9-1. DDRSS CoS Mapping Diagram

9.1.3.3 AXI Write Data All-Strobes

The DDR controller data path is connected to the VBUSM2AXI bridge through AXI interface. To enable higher performance and lower latency, the bridge drives the AXI AWALLSTRB signal to a "1" when the AXI write data does not have any byte enable holes. This allows the DDR controller to accept and start processing the write command as soon as possible without waiting for all data. The AXI0_ALL_STROBES_USED_ENABLE bit in the DDR Controller must be set to a 0x1 to utilize this feature. The controller ignores the AWALLSTRB signal if AXI0_ALL_STROBES_USED_ENABLE is set to 0x0.

9.1.3.4 Inline ECC for SDRAM Data

For SDRAM data integrity, the VBUSM2AXI bridge supports inline ECC on the data written to or read from the SDRAM. ECC is enabled by programming the bits in the EMIF_SSCFG_ECC_CTRL_REG register. ECC is stored together with the data so that a dedicated SDRAM device for ECC is not required.

8-bit single error correction double error detection (SECDED) ECC is calculated over 64-bit data quanta. For every 512-byte data block 64 bytes of ECC is stored inline. Thus 1/9th of the total SDRAM space is used for ECC storage and the rest 8/9th is available for system use. From system point of view that 8/9th of the SDRAM data space are seen as consecutive byte addresses. Even if there are non-ECC protected regions the previously described 1/9th-8/9th rule still applies and consecutive byte addresses are seen from system point of view. Note that 8/9 of the total SDRAM space is available for system use regardless of the size of the ECC region(s). In other words, only 8/9 of the memory is available even in non-ECC regions when ECC is enabled.

The ECC is calculated for all accesses that are within the address ranges protected by ECC. The address ranges are specified through the following registers:

- EMIF_SSCFG_ECC_R0_STR_ADDR_REG
- EMIF_SSCFG_ECC_R0_END_ADDR_REG

Copyright © 2026 Texas Instruments Incorporated

- EMIF_SSCFG_ECC_R1_STR_ADDR_REG
- EMIF_SSCFG_ECC_R1_END_ADDR_REG
- EMIF_SSCFG_ECC_R2_STR_ADDR_REG
- EMIF_SSCFG_ECC_R2_END_ADDR_REG

Note that the addresses in these address range registers should be in terms of the DDRSS address space, not the SoC memory space. Even though the SoC memory space for DDR may be split into multiple regions, the DDRSS address space is continuous.

For example, if the beginning of DDR is in SoC memory space 0x8000_0000 to 0xFFFF_FFFF, this corresponds to DDRSS address space 0x00000000 to 0x7FFFFFFF. If the SoC has greater DDR address reach, even at a discontinuous address, the address range for these registers would be continuous (eg, if the next DDR address at the SoC level is 0x880000000, the corresponding address would be continuous at 0x80000000)

Note that the value programmed in the ECC range start/end registers does not include the lower 16 bits of the address. For example, to represent an address of 0x40000000, the register contents would be 0x4000.

The ECC is read and verified during reads if both the EMIF_SSCFG_ECC_CTRL_REG[0] ECC_EN and EMIF_SSCFG_ECC_CTRL_REG[2] ECC_CK bits are set to 0x1. For 1-bit ECC error, the bridge corrects the data and returns it to the requestor. Although the error is corrected on the returned data, the SDRAM is not corrected. It is responsibility of the system software to correct the ECC error at that location.

It is also responsibility of the system software to pre-load the ECC protected region with known data before functional reads and writes are performed. This can be done by writing to the SDRAM with ECC enabled (EMIF_SSCFG_ECC_CTRL_REG[0] ECC_EN = 0x1 and EMIF_SSCFG_ECC_CTRL_REG[1] RMW_EN = 0x1) and ECC check disabled (EMIF_SSCFG_ECC_CTRL_REG[2] ECC_CK = 0x0). Once the data is loaded in the SDRAM, ECC check must be enabled (EMIF_SSCFG_ECC_CTRL_REG[2] ECC_CK = 0x1) before using the DDR interface.

9.1.3.4.1 ECC Cache

The VBUSM2AXI bridge implements a 64-line deep ECC cache for improving inline ECC performance. Each cache line can be allocated to an initiator in the system based on its Route ID. The Route ID allocation to cache line can be done by writing to the EMIF_SSCFG_ECC_RID_INDX_REG and EMIF_SSCFG_ECC_RID_VAL_REG registers. Since the bridge uses unallocated cache lines for all read accesses without Route ID allocation, one or more locations in the cache should be kept unallocated for better performance. To ensure that at least one cache line remains unallocated, in the case all cache lines are allocated by software, the bridge automatically unallocates the 63rd cache line. Write accesses without Route ID allocation result in ECC and data writes to the SDRAM, when the EMIF_SSCFG_ECC_CTRL_REG[4] WR_ALLOC bit is set to 0x0. When EMIF_SSCFG_ECC_CTRL_REG[4] WR_ALLOC is set to 0x1, an unassigned cache-line is allocated to write accesses without Route ID allocation.

9.1.3.4.2 ECC Cache Flush

For low power modes, the bridge supports flushing the cache when DDRSS stop clock request is asserted. When a clock stop is requested, the bridge will start issuing writes to the DRAM until all dirty cache locations are written to the DRAM. Once the writes are complete, the bridge asserts an acknowledge signal to let the subsystem know that the operation is complete. The subsystem uses this acknowledge signal along with acknowledgments from other submodules to generate the final DDRSS stop clock acknowledge to the system.

9.1.3.4.3 ECC Statistics

For 1-bit ECC error, the bridge logs the address location for the error in an internal 2-level deep FIFO. It stores the first two 1-bit ECC errors. The EMIF_SSCFG_ECC_1B_ERR_ADR_LOG_REG register shows the address on top of the internal FIFO. Software should write 0x1 to the EMIF_SSCFG_ECC_1B_ERR_ADR_LOG_REG[29-0] ECC_1B_ERR_ADR field to pop the FIFO and display the next address stored. The FIFO is loaded with the address for the next 1-bit ECC error if it is not full. No address comparison will be performed, that is, if a single address is associated with more than one ECC error, that address is logged twice.

Copyright © 2026 Texas Instruments Incorporated

The number of 1-bit ECC errors can be counted using the EMIF_SSCFG_ECC_1B_ERR_CNT_REG register. The bridge also supports programming a threshold in the EMIF_SSCFG_ECC_1B_ERR_THRSH_REG register. When the 1-bit error count is equal to or greater than the programmed threshold, the bridge sets the EMIF_SSCFG_V2A_INT_RAW_REG[3] ECC1BERR bit and also triggers the DDR0_DDRSS_DRAM_ECC_CORR_ERR_LVL_0 interrupt. When servicing the interrupt, software needs to clear the error count otherwise further interrupts will not be triggered. For 2-bit ECC errors, the bridge sets the EMIF_SSCFG_V2A_INT_RAW_REG[4] ECC2BERR bit and also triggers the DDR0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_0 interrupt. The bridge does not correct the data for these uncorrectable errors. Along with generating the interrupt, the bridge also reports an error to the requesting initiatior and sends all zeros for the data. The bridge also logs the address location for the error in the EMIF_SSCFG_ECC_2B_ERR_ADR_LOG_REG register.

The bridge sets the EMIF_SSCFG_V2A_INT_RAW_REG[5] ECCM1BERR bit and triggers the DDR0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_0 interrupt whenever it receives multiple 1-bit errors in different data words of the same SDRAM burst. This is done because the probability of receiving multiple 1-bit errors in different data words of the same SDRAM burst is very low. If this occurs, the chances are that there were multi-bit errors in each data word. Therefore, for reliability, pessimistic approach is taken to report these as a fatal error. The threshold for the number of 1-bit errors that result in an uncorrected error, reporting can be set by writing to the EMIF_SSCFG_ECC_CTRL_REG[11-8] COR_ECC_THRESH field. For reliability, the threshold is always kept at 0/1 meaning 1-bit error in 2 or more data words is reported as a 2-bit error. For debug, the threshold can be changed to a higher value. Note that since these are multiple 1-bit errors, the statistic logging is done in the EMIF_SSCFG_ECC_1B_ERR_CNT_REG and EMIF_SSCFG_ECC_1B_ERR_ADR_LOG_REG registers.

9.1.3.5 Address Alias Prevention

The VBUSM2ZXI bridge checks the VBUSM address received against the valid SDRAM address space programmed in the EMIF_SSCFG_V2A_CTL_REG register. If the VBUSM address for an access falls outside the programmed range the bridge sets to 0x1 the EMIF_SSCFG_V2A_INT_RAW_REG[1] AERR bit and also triggers the DDR0_DDRSS_V2A_OTHER_ERR_LVL_0 interrupt. The address and the Route ID for the command that caused error are logged in the EMIF_SSCFG_V2A_AERR_LOG1_REG and EMIF_SSCFG_V2A_AERR_LOG2_REG registers.

The bridge fragments all incoming transactions into 64-byte accesses aligned at a 64-byte boundary. Therefore, for commands greater than 64 bytes with an address error, each fragment will report an error. This can cause multiple interrupts to be reported if software clears the error interrupt for the first fragment while other fragments are being executed. The address log register will always log the address for the fragment which caused the last interrupt. The valid address range can be programmed using the EMIF_SSCFG_V2A_CTL_REG[9-5] SDRAM_IDX and EMIF_SSCFG_V2A_CTL_REG[4-0] REGION_IDX fields.

Table 9-2 summarizes the scenarios for determining valid address range and interrupt generation.

Table 9-2. REGION_IDX and SDRAM_IDX Scenarios

Condition	Description
REGION_IDX = SDRAM_IDX	DDR region size is equal to the connected SDRAM size. It is the SoC responsibility to ensure that the addresses received by the DDR subsystem do not fall outside the region size. No address error is generated if the address received falls outside the region size.
REGION_IDX < SDRAM_IDX	DDR region size is less than the connected SDRAM size. It is SoC responsibility to ensure that the addresses received by the DDR subsystem do not fall outside the region size. No address error is generated if the address received falls outside the region size.
REGION_IDX > SDRAM_IDX	DDR region size is greater than the connected SDRAM size. Address error is generated if the address received falls outside the connected SDRAM size.

A write access outside the programmed range is discarded. A write error associated status is sent back to the VBUSM interface.

Copyright © 2026 Texas Instruments Incorporated

A read access outside the programmed range is executed on the SDRAM interface as a normal read, but data will contain all zeroes before forwarding it to the VBUSM interface. A read error associated status is sent back to the VBUSM interface along with the read data containing all zeroes.

When inline ECC is enabled, the available SDRAM size is reduced by 1/9th of the size programmed in the SDRAM_IDX field. Therefore, the bridge reports an error if an access falls outside the reduced SDRAM size when inline ECC is enabled. The reduced SDRAM size limit applies to all accesses, both to the protected and non-protected ECC regions.

9.1.3.6 AXI Bus Timeout

The VBUSM2AXI bridge has a timeout counter that expires when no AXI transaction can be sent to the DDR controller or no AXI response is received from the controller for a programmed time interval. The counter only counts when there are pending commands in the bridge and the AXI bus is idle. Therefore, the bridge will not time out during low-power modes. The time interval can be programmed by writing to the EMIF_SSCFG_V2A_BUS_TO[23-0] BUS_TIMER field.

Upon the expiry of the counter, the bridge terminates all pending commands in its internal FIFOs, returns error responses, sets the EMIF_SSCFG_V2A_INT_RAW_REG[2] TOERR bit and triggers the DDR0_DDRSS_V2A_OTHER_ERR_LVL_0 interrupt.

After a timeout occurs, the bridge terminates any new commands received and returns error response. Writing a value of 0x0 to the EMIF_SSCFG_V2A_BUS_TO[23-0] BUS_TIMER field exits the timeout mode. The other method to exit the timeout mode is to reset the DDRSS0, thus resetting the VBUSM2ZXI bridge, the DDR controller, and the DDR PHY.

9.1.3.7 Leaky Bucket Function

The VBUSM2AXI bridge implements a leaky bucket mechanism at each LPT-HPT arbitration point. This will prevent LPT commands from getting stuck in the pipe forever due to overload of HPT commands.

A programmable system threshold will be used to trigger leaky bucket. This can be programmed in the EMIF_SSCFG_V2A_LEAKY_THRESH_REG. The threshold determines how many HPT commands can be sent in a row before a single LPT command is forwarded.

9.1.3.8 Drain Function

The VBUSM2AXI bridge implements a mechanism to drain the DDR controller. The drain action will temporarily pause the submission of commands to the controller until the read command(s) that triggered the drain operation have been resolved.

Each read going to the DDR controller will have an associated internal counter that starts at zero. This counter will increment for each write or read command given to the controller after that particular read is sent. If any read command counter exceeds the programmed threshold the bridge will start a drain action. The threshold can be programmed in EMIF_SSCFG_V2A_DRAIN_THRESH_REG.

9.1.3.9 DDRSS Interrupts

The VBUSM2AXI bridge sets to 0x1 the EMIF_SSCFG_V2A_INT_RAW_REG[1] AERR bit, if the VBUSM address for an access that falls outside the DDR space programmed in the EMIF_SSCFG_V2A_CTL_REG register.

The VBUSM2AXI bridge sets to 0x1 the EMIF_SSCFG_V2A_INT_RAW_REG[2] TOERR bit, if it detects a hang on its interface to the DDR controller.

The VBUSM2AXI bridge sets to 0x1 the EMIF_SSCFG_V2A_INT_RAW_REG[3] ECC1BERR bit, if the threshold for 1-bit ECC errors is met.

The VBUSM2AXI bridge sets to 0x1 the EMIF_SSCFG_V2A_INT_RAW_REG[4] ECC2BERR bit, in case of 2-bit errors for a read access performed within the SDRAM address range protected by ECC.

Copyright © 2026 Texas Instruments Incorporated

The DDRSS0 asserts particular interrupt line only if the interrupts are enabled by writing 0x1 to the corresponding bit in the EMIF_SSCFG_V2A_INT_SET_REG register. The interrupts can be disabled by writing 0x1 to the corresponding bit in the EMIF_SSCFG_V2A_INT_CLR_REG register.

When interrupts are enabled, the corresponding bits in the EMIF_SSCFG_V2A_INT_STAT_REG register are also set if an interrupt condition occurs. The interrupts can be cleared once serviced by writing 0x1 to the corresponding bit in the EMIF_SSCFG_V2A_INT_STAT_REG register as well as writing to the EMIF_SSCFG_V2A_EOI_REG register. The subsystem will send an interrupt pulse, if there are any bits set in the Interrupt Status register (EMIF_SSCFG_V2A_INT_STAT_REG) when the End of Interrupt register (EMIF_SSCFG_V2A_EOI_REG) is written.

VBUSM2AXI Bridge Events shows the events that are generated by the VBUSM2AXI bridge.

Table 9-3. VBUSM2AXI Bridge Events

Event Flag	Event Mask	Description
EMIF_SSCFG_V2A_INT_RAW_REG[4] ECC2BERR EMIF_SSCFG_V2A_INT_STAT_REG[4] ECC2BERR	EMIF_SSCFG_V2A_INT_SET_REG[4] ECC2BERR_EN EMIF_SSCFG_V2A_INT_CLR_REG[4] ECC2BERR_EN	Generated in case of 2-bit errors for a read access within the ECC protected SDRAM address range. For more information, see Section 9.1.3.4.3.
EMIF_SSCFG_V2A_INT_RAW_REG[3] ECC1BERR EMIF_SSCFG_V2A_INT_STAT_REG[3] ECC1BERR	EMIF_SSCFG_V2A_INT_SET_REG[3] ECC1BERR_EN EMIF_SSCFG_V2A_INT_CLR_REG[3] ECC1BERR_EN	Generated in case of meeting the threshold for 1-bit ECC errors. For more information, see Section 9.1.3.4.3.
EMIF_SSCFG_V2A_INT_RAW_REG[2] TOERR EMIF_SSCFG_V2A_INT_STAT_REG[2] TOERR	EMIF_SSCFG_V2A_INT_SET_REG[2] TOERR_EN EMIF_SSCFG_V2A_INT_CLR_REG[2] TOERR_EN	Generated in case of a hang of the interface between the bridge and the DDR controller. For more information, see Section 9.1.3.6.
EMIF_SSCFG_V2A_INT_RAW_REG[1] AERR EMIF_SSCFG_V2A_INT_STAT_REG[1] AERR	EMIF_SSCFG_V2A_INT_SET_REG[1] AERR_EN EMIF_SSCFG_V2A_INT_CLR_REG[1] AERR_EN	Generated if the VBUSM address for an access falls outside the programmed range. For more information, see Section 9.1.3.5.

9.1.3.10 DDRSS Memory Regions

Table 9-4 shows all memory regions associated with the DDRSS0.

Table 9-4. DDRSS0 Memory Regions

Region	Start Address	End Address	Region Size
DDRSS0 wrapper logic registers	0x00F300000	0x00F3001FF	512 B
DDR controller registers	0x00F308000	0x00F309FFF	8 KB
DDR PHY independent module registers	0x00F30A000	0x00F30BFFF	8 KB
DDR PHY registers	0x00F30C000	0x00F30FFFF	16 KB
External SDRAM data space	0x080000000	0x0FFFFFFFF	2 GB
	0x880000000	0xFFFFFFFFF	30 GB

9.1.3.11 DDRSS Dynamic Frequency Change Interface

This interface is used for handshaking between DDRSS0 and CTRL_MMR0 to dynamically change DDR clock frequency to support LPDDR4 Frequency Set Point (FSP). The frequency change can be initiated either by a SoC processor or by the DDRSS0.

The associated CTRL_MMR0 registers for processor initiated frequency change are:

- CTRLMMR_CHNG_DDR4_FSP_REQ

Copyright © 2026 Texas Instruments Incorporated

- CTRLMMR_CHNG_DDR4_FSP_ACK

The associated CTRL_MMR0 registers for DDRSS0 initiated frequency change are:

- CTRLMMR_DDR4_FSP_CLKCHNG_REQ
- CTRLMMR_DDR4_FSP_CLKCHNG_ACK

The sequence for an SoC processor initiated frequency change is as follows:

1. The processor writes the desired frequency to the CTRLMMR_CHNG_DDR4_FSP_REQ[1-0] REQ_TYPE field.
2. The processor sets to 0x1 the CTRLMMR_CHNG_DDR4_FSP_REQ[8] REQ bit to initiate frequency change.
3. The processor programs PLL12 according to the CTRLMMR_CHNG_DDR4_FSP_REQ[1-0] REQ_TYPE value.
4. The processor polls the CTRLMMR_CHNG_DDR4_FSP_ACK[7] ACK and CTRLMMR_CHNG_DDR4_FSP_ACK[0] ERROR bits to check if the frequency change has been completed successfully.

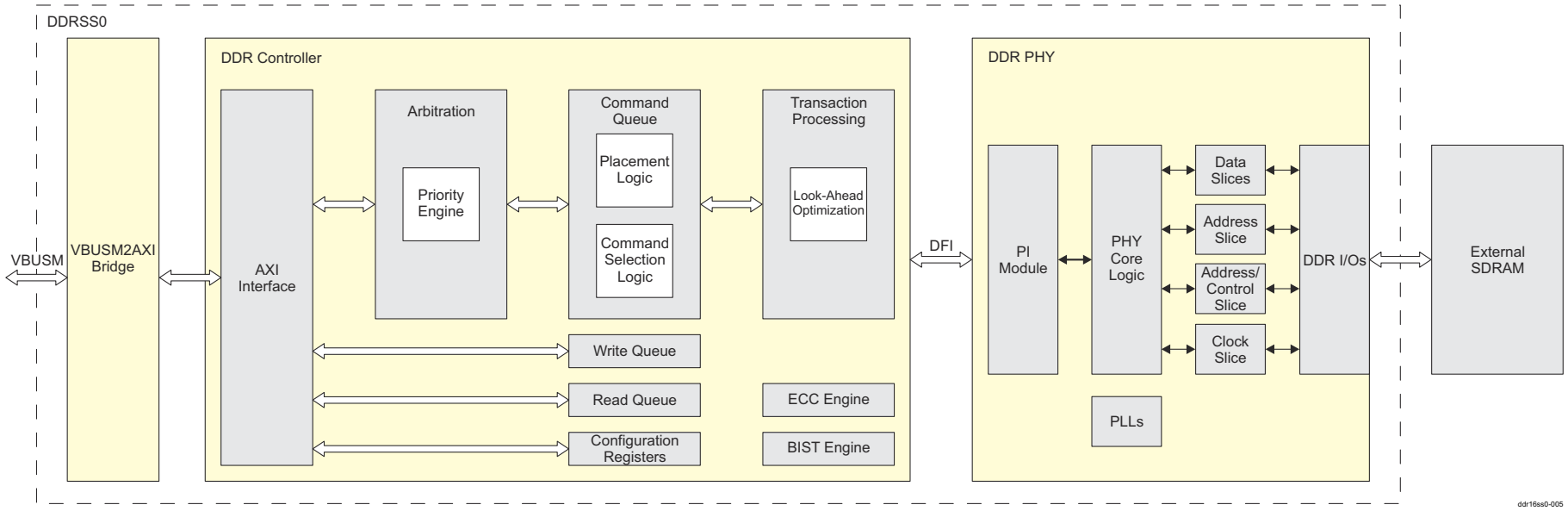
The sequence for DDRSS0 initiated frequency change is as follows:

1. DDRSS0 requests frequency change by asserting the CTRLMMR_DDR4_FSP_CLKCHNG_REQ[7] REQ bit and the CTRLMMR_DDR4_FSP_CLKCHNG_REQ[1-0] REQ_TYPE field. The REQ bit is also connected to the DDR0_DDRSS_PLL_FREQ_CHANGE_REQ_0 interrupt.
2. Software reads the CTRLMMR_DDR4_FSP_CLKCHNG_REQ[1-0] REQ_TYPE value and programs PLL12 accordingly.
3. After the DDRSS0_FCLK has been changed software should set to 0x1 the CTRLMMR_DDR4_FSP_CLKCHNG_ACK[0] ACK bit.

Copyright © 2026 Texas Instruments Incorporated

9.1.3.12 DDR Controller Functional Description

Figure 9-2 shows the DDR controller blocks along with the DDR PHY and VBUSM2AXI bridge.



The ECC Engine block of the DDR controller is not supported.

Figure 9-2. DDR Controller Functional Blocks

Copyright © 2026 Texas Instruments Incorporated

9.1.3.12.1 DDR PHY Interface (DFI)

The DDR controller and DDR PHY are connected via DDR PHY Interface (DFI) which is used for transferring control information and data between the PHY and the controller. For more information about DFI, see http://www.ddr-phy.org/.

Copyright © 2026 Texas Instruments Incorporated

9.1.3.12.2 Command Queue

The DDR controller contains a command queue which uses a placement algorithm to determine the order of commands to be placed into the command queue. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at a time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. The placement logic also attempts to maximize efficiency of the DDR controller through command grouping and bank splitting.

9.1.3.12.2.1 Placement Logic

The placement logic is a 2-stage queue which determines the order of commands that run in the DDR controller. The placement logic follows rules for determining placement of new commands into the queue, relative to the contents of the command queue at that time. Placement is determined by considering coherency, address collisions, source collisions, data collisions, user assigned priority, latency, age, and command type to offer low latency for critical controllers while optimizing bandwidth for all controllers. A second reordering stage allows ready-to-run commands to start even if the head-of-queue command is not yet ready to run. In addition, the queue can be disabled completely, resulting in an in-line queue that services requests in the order that they arrive.

The DDR controller also has a full look-ahead facility that reduces the effect of page misses by pre-conditioning rows for upcoming requests by using “spare” cycles in preceding transactions.

9.1.3.12.2.2 Command Selection Logic

After a command is in the command queue, command selection logic determines the method of pulling commands from the queue for running. On each clock cycle, the selection logic scans the entries of the command queue for determining the command to run. Commands for running are based on bank readiness, availability of at least 1 burst of data (writes), availability of storage for at least 1 burst of data (reads), bus turnaround timing, and conflicts. Similar to the placement rules, a command does not run before a command that was placed ahead of it in the command queue if it conflicts with address, source ID, or bank commands. Lower priority commands can run ahead of higher priority commands if the higher priority commands are not ready to run, as long as they do not conflict with commands that are ahead in the command queue.

9.1.3.12.3 Transaction Processing

The transaction command processing logic is used to process the commands in the command queue. The logic organizes the commands to the memories in such a way that data throughput is maximized. Bank opening and closing cycles are used for data transfers. The logic reviews the entire command queue for look-ahead of which banks have to be accessed in the future and ensures that the programed memory timing conditions are met. This flexibility allows the controller to be tuned to extract the maximum performance out of memories. During processing, the controller examines the commands and issues the appropriate set of signals to the memory.

9.1.3.12.4 Paging Policy

The DDR controller offers a flexible paging policy that allows open page operation, closed page operation, or an auto-precharge-per-command option that allows both modes simultaneously. Auto-precharge-per-command allows marking a particular controller or transaction (for example, a CPU cache) as a closed-page transaction. This type of transaction reduces power and improves latency for the next transaction to a different row in the same SDRAM bank. Other transactions can be marked as open page transactions. This type of transaction reduces power and improves latency and bandwidth to the next transaction to the same row in the same SDRAM bank.

9.1.3.12.5 DDR Controller Initialization

Once the power to the SDRAM and SoC is stable, the DDR controller must be initialized. It then automatically initializes the external memory. Please refer to the Processor SDK for the software drivers to initalize the DDR registers and memory

Copyright © 2026 Texas Instruments Incorporated

Note

To facilitate the programming of these registers, refer to the device-specific DDR Subsystem Register Configuration Tool available at http://dev.ti.com/sysconfig.

Copyright © 2026 Texas Instruments Incorporated

Chapter 10

Interrupts

10.1 Interrupt Architecture.....819
10.2 Interrupt Controllers.....828
10.3 Interrupt Router (INTROUTER).....833
10.4 Interrupt Sources.....833

Copyright © 2026 Texas Instruments Incorporated

10.1 Interrupt Architecture

J722S Interrupt Architecture includes information on how to utilize various interrupts and events in the system and relationships between interrupts and events.

An interrupt is defined as a physical signal which can be routed to the interrupt controller of various processors, which can cause the Interrupt Service Routine (ISR). This physical signal can be either a pulse or level and the polarity can be either negative or positive. And an interrupt is correspondent to an individual wire.

An event is defined as information transported by the event bus or PSI_L bus. Events are coded using an event index. The same event bus or PSI_L bus is used to transport multiple events. An event can also be routed and multiplexed to different locations. Events can't trigger a processor's ISR directly. They must go through the Interrupt Aggregator (IA) to convert the event into an interrupt line. For more information, see , *Interrupt Aggregator (INTAGGR)*

The DMA transfer using BCDMA and PktDMA in J722S can only be triggered using an event. The SoC level interrupt could be used as a BCDMA trigger using L2G logic. Interrupts in the SoC level can't be used to trigger a pktDMA transfer.

Each type of processor has its unique interrupt controller. All A53 cores share a single Generic Interrupt Controller (GIC). Each R5 micro controller has its own dedicated interrupt controller, called Vectored Interrupt Manager (VIM). Please refer to Section 10.2 for the detailed usage.

J722S also contains multiple SoC level interrupt routers. The main function of those SoC level interrupt routers is to provide flexibility to select interrupt sources from a large group of interrupt sources. Many places use this feature, such as GPIO interrupts and some time synchronization related interrupts.

J722S contains two ESM modules. ESM is used to consolidate/monitor the error events in the device. In this document, ESM may also be referred to as ESM0.

J722S also has some chip level glue logic to convert some miscellaneous signals used as interrupt sources. Those miscellaneous signals are normally self-clear signals and do not need to clear after the Interrupt Service Routine (ISR) like the normal interrupts generated by the peripherals.

Figure 10-1 shows the high level interrupt architecture in J722S.

Copyright © 2026 Texas Instruments Incorporated

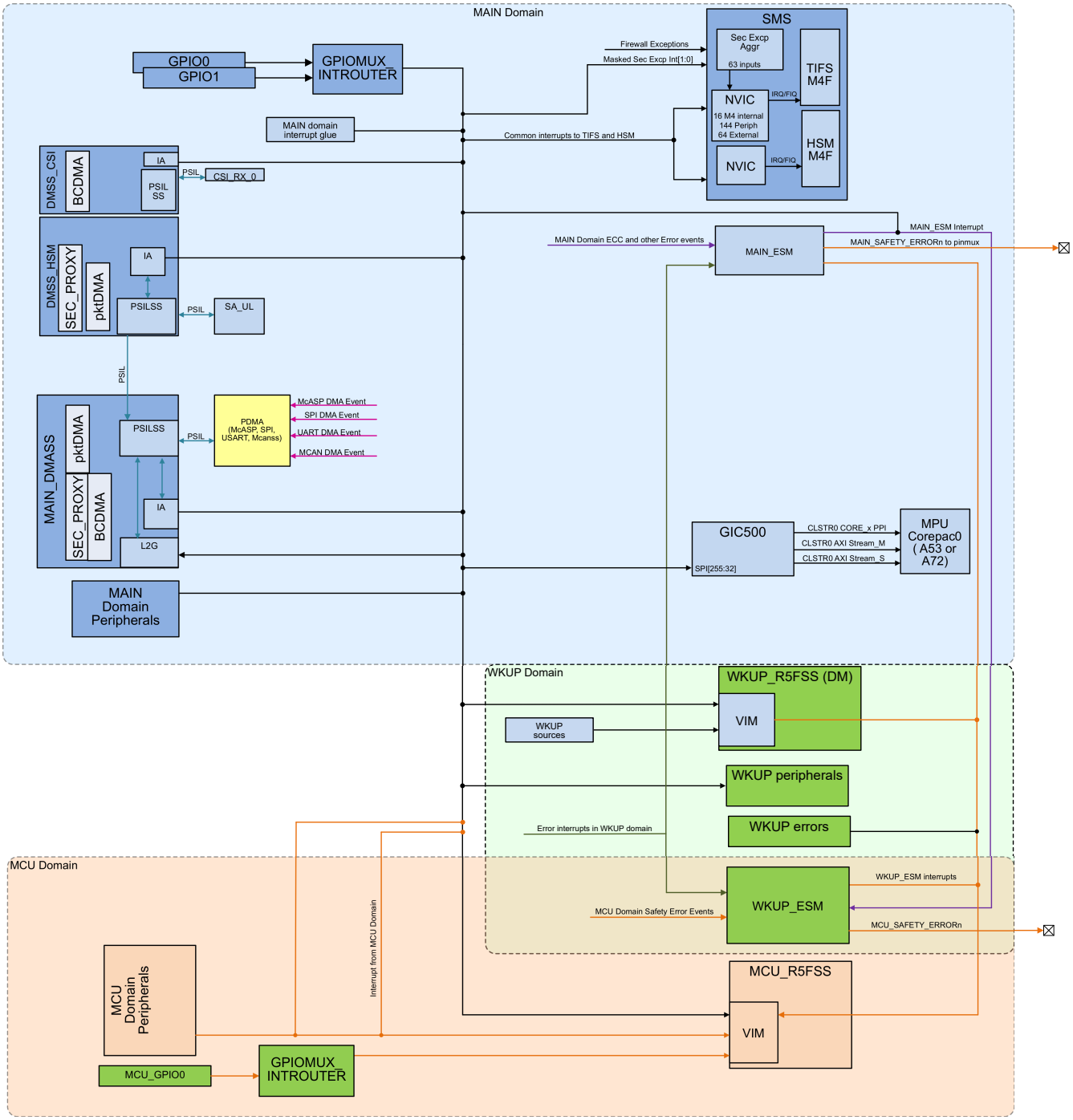


Figure 10-1. J722S Interrupt Architecture

10.1.1 ESM Connectivity

J722S contains two ESM modules to consolidate the error interrupts in SoC. One is in the MAIN domain and one is in the WKUP domain. Figure 10-2 shows how error interrupts are connected to these two ESM. See also *Interrupts (Inputs)* and *Interrupts (Outputs)*.

Copyright © 2026 Texas Instruments Incorporated

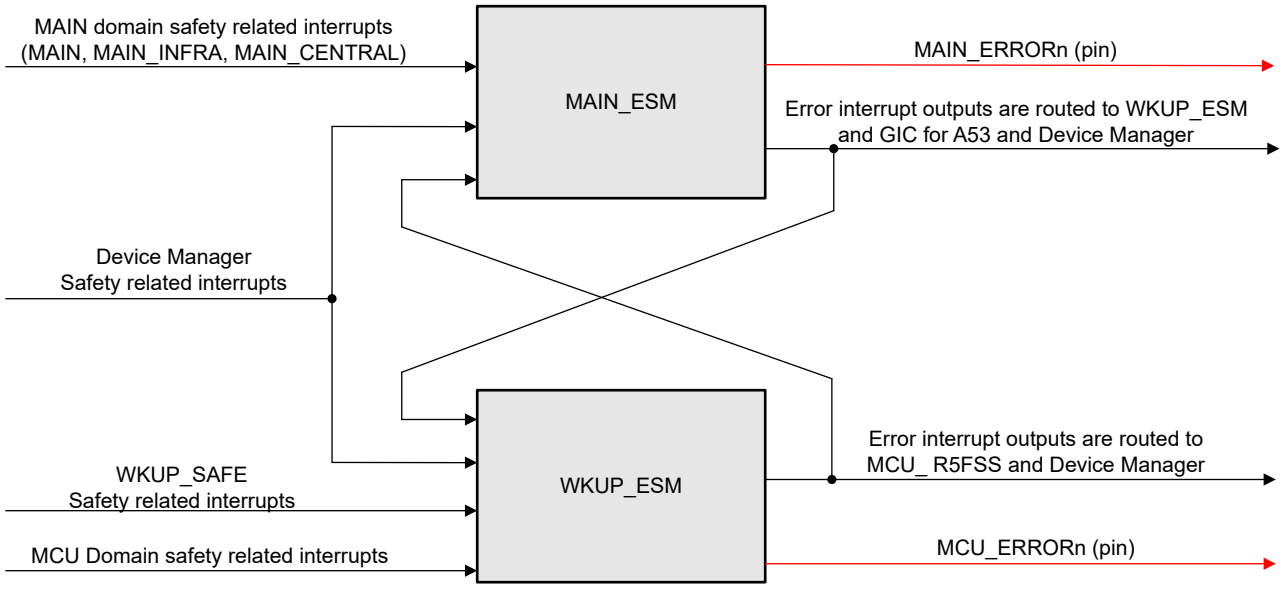


Figure 10-2. ESM Connections

The error interrupts from the WKUP_R5FSS domain are routed to both MAIN_ESM and WKUP_ESM. Users can configure ESM to cause ESM generating interrupt output from the interrupt inputs. The interrupt outputs of the MAIN_ESM are routed as interrupt inputs to the WKUP_ESM. The interrupt outputs of the WKUP_ESM are routed as the interrupt inputs to the MAIN_ESM. This means that user can configure the device to use one ESM to monitor the error interrupts for the whole device. J722S supports the following three use cases.

10.1.1.1 Using WKUP_ESM to Monitor All Error Events in SoC

In this use case, the MAIN_ESM is used to consolidate all the error interrupts in the MAIN domain and the WKUP_R5FSS domain as MAIN_ESM's interrupt outputs, which are routed as interrupt inputs to the WKUP_ESM.

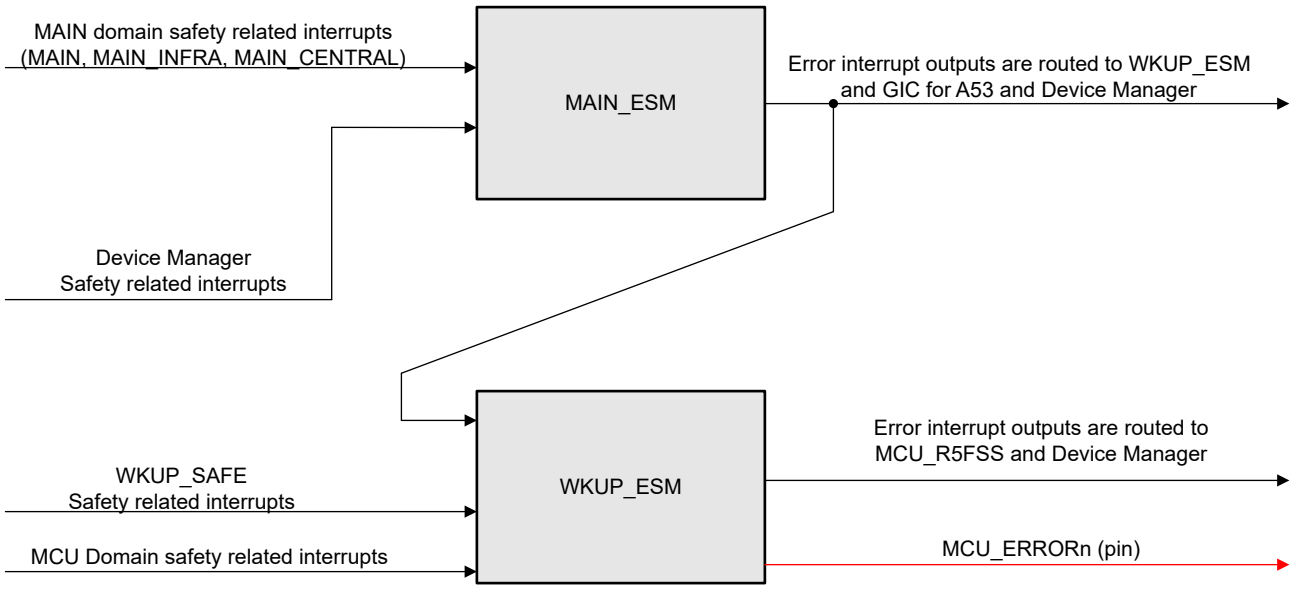


Figure 10-3. Using WKUP_ESM for All Error Interrupts in SoC

Copyright © 2026 Texas Instruments Incorporated

In this use case, the MAIN_ESM should be configured to disable the interrupt inputs coming from the WKUP_ESM interrupts. If those interrupt inputs are not disabled by the MAIN_ESM, the dependence loop will be formed as following:

1. Error interrupts in the MAIN domain are routed to MAIN_ESM trigger interrupt outputs generated by the MAIN_ESM
2. The error interrupt outputs from the MAIN_ESM are routed to the WKUP_ESM as interrupt inputs
3. Those error interrupt inputs trigger the WKUP_ESM generating its error interrupts outputs
4. Those interrupt outputs from the WKUP_ESM are routed back to the MAIN_ESM as interrupt inputs. If those interrupts are not disabled at the MAIN_ESM, they trigger MAIN_ESM generating interrupt outputs in the MAIN_ESM again. Going back to step 1 again.

In addition, since WKUP_R5FSS error events are already routed to the MAIN_ESM, so the input events in the WKUP_ESM connecting to those error events from the WKUP_R5FSS domain shall be disabled, otherwise those events can trigger the interrupt in both the MAIN_ESM as well as the WKUP_ESM

10.1.1.2 Using MAIN_ESM to Monitor All Error Interrupts in SoC

In this configuration, the WKUP_ESM is used to consolidate all the error interrupts from the WKUP domain and the WKUP_safe domain as interrupt outputs, and those interrupt outputs from the WKUP_ESM are routed as interrupt inputs to the MAIN_ESM. In order to avoid the interrupt dependence loop, the WKUP_ESM shall disable the interrupt inputs from ESM0.

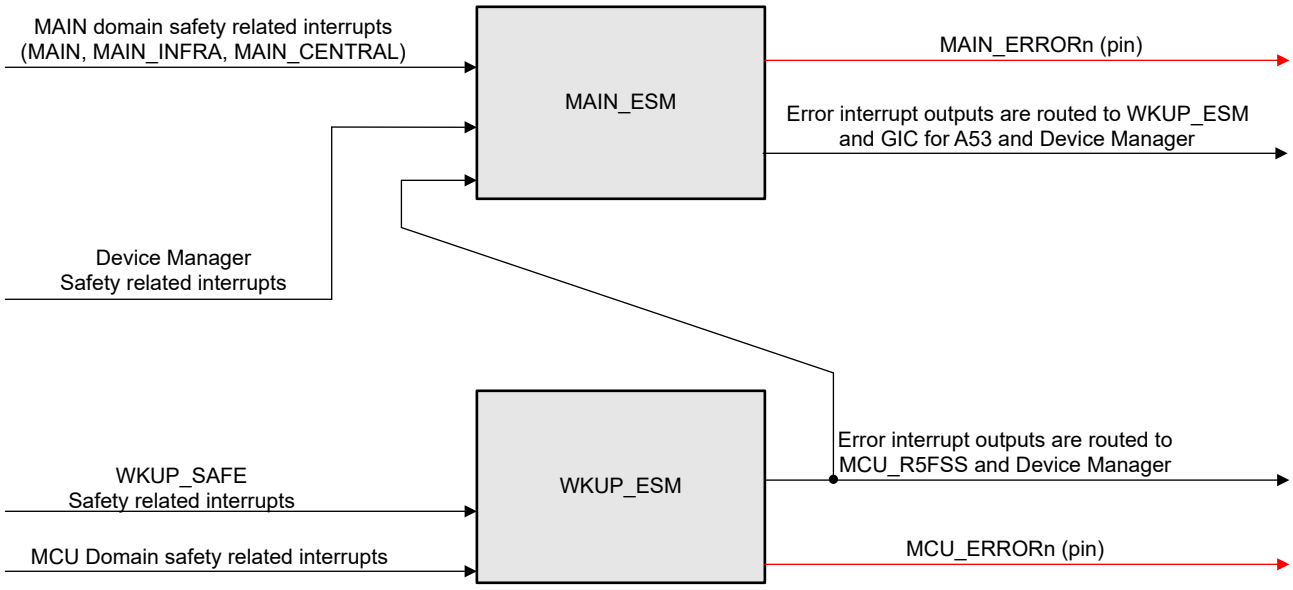


Figure 10-4. Using MAIN_ESM to Monitor All Error Interrupts in SoC

10.1.1.3 ESM Configuration During Deep Sleep Mode

When device is in deep sleep mode, only the components in the WKUP_R5FSS domain and the WKUP_SAFE are available. The MAIN domain is powered off, so the MAIN_ESM is no longer available. Since WKUP_ESM is located in the WKUP_SAFE domain, the WKUP_ESM is monitoring all of the error events during the deep sleep mode and informs the device manager.

Copyright © 2026 Texas Instruments Incorporated

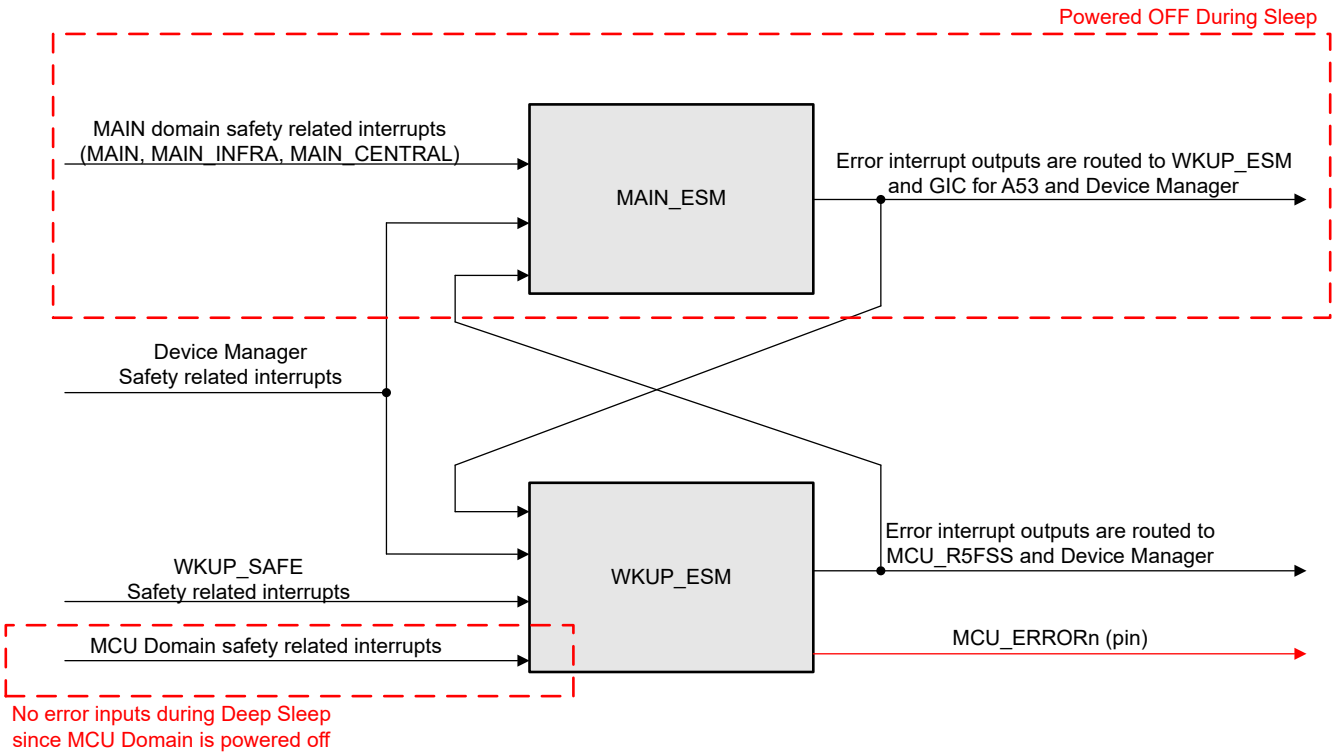


Figure 10-5. Deep Sleep Use Case

10.1.2 PSIL Events

PSIL Events are mainly generated by the BCDMA and the pktDMA. In addition, the SEC_PROXY IPC module also generates event outputs. PSIL Events can be directly utilized by the BCDMA and the pktDMA. If those events need to be processed by processor, the events need to be converted to interrupts through the Interrupt Aggregator (IA) Module. Refer to , *Interrupt Aggregator (INTAGGR)* for information on how IA is used. Figure 10-6 shows which modules generates events and how the events are converted to the interrupt.

BCDMA transfer can only be triggered through an event or by software. There are 32 SoC level interrupt signals that can be used to trigger a BCDMA transfer.

Copyright © 2026 Texas Instruments Incorporated

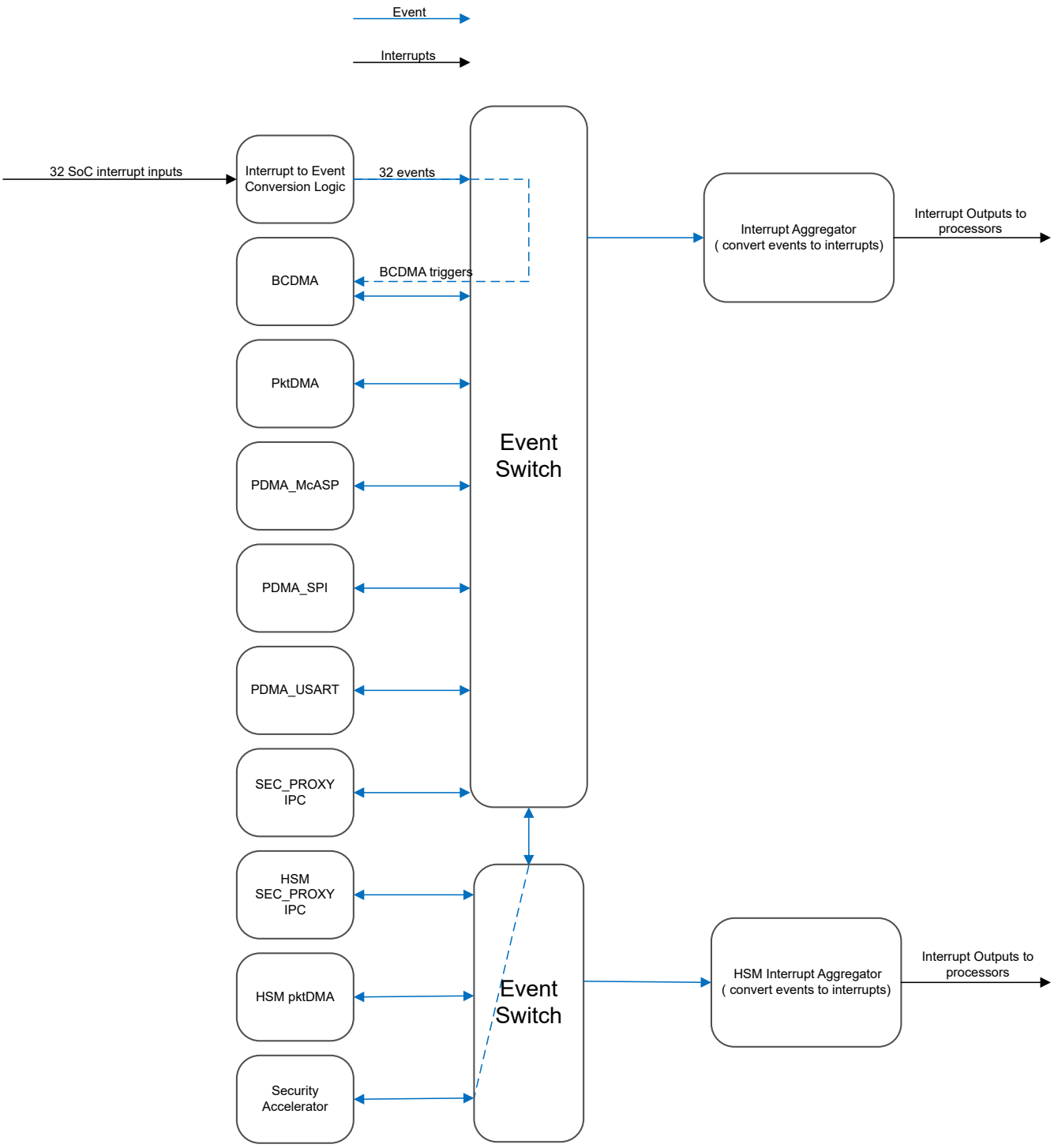


Figure 10-6. Event Handling

BCDMA Event Triggers shows all the possible interrupt sources which can be used as BCDMA triggers.

10.1.2.1 BCDMA Event Triggers

Table 10-1. BCDMA Event Triggers

Input Index	Interrupt Sources
0	CPSW0.CPTS_COMP.0

Copyright © 2026 Texas Instruments Incorporated

Table 10-1. BCDMA Event Triggers (continued)

Input Index	Interrupt Sources
1	PCIE0.PCIE_CPTS_COMP.0
2	DSS0.DISPC_INTR_REQ_0.0
3	DSS0.DISPC_INTR_REQ_1.0
4	DSS1.DISPC_INTR_REQ_0.0
5	DSS1.DISPC_INTR_REQ_1.0
7	MCRC64_0.INT_MCRC.0
8	TIMESYNC_EVENT_INTROUTER0.OUTL.0
9	TIMESYNC_EVENT_INTROUTER0.OUTL.1
10	TIMESYNC_EVENT_INTROUTER0.OUTL.2
11	TIMESYNC_EVENT_INTROUTER0.OUTL.3
12	TIMESYNC_EVENT_INTROUTER0.OUTL.4
13	TIMESYNC_EVENT_INTROUTER0.OUTL.5
14	TIMESYNC_EVENT_INTROUTER0.OUTL.6
15	TIMESYNC_EVENT_INTROUTER0.OUTL.7
16	MAIN_GPIOMUX_INTROUTER0.OUTP.24
17	MAIN_GPIOMUX_INTROUTER0.OUTP.25
18	MAIN_GPIOMUX_INTROUTER0.OUTP.26
19	MAIN_GPIOMUX_INTROUTER0.OUTP.27
20	MAIN_GPIOMUX_INTROUTER0.OUTP.28
21	MAIN_GPIOMUX_INTROUTER0.OUTP.29
22	MAIN_GPIOMUX_INTROUTER0.OUTP.30
23	MAIN_GPIOMUX_INTROUTER0.OUTP.31
24	MAIN_GPIOMUX_INTROUTER0.OUTP.22
25	MAIN_GPIOMUX_INTROUTER0.OUTP.23
26	GPMC0.GPMC_SDMAREQ.0
27	DEBUGSS0.DAVDMA_LEVEL.0
28	MCRC64_0.DMA_EVENT.0
29	MCRC64_0.DMA_EVENT.1
30	MCRC64_0.DMA_EVENT.2
31	MCRC64_0.DMA_EVENT.3

10.1.3 Object Missing

This object is not available in the repository.

10.1.4 Utilizing Miscellaneous Signals as Interrupt

Some of the signals are aggregated through glue logic and need to be handled separately.

10.1.4.1 Aggregated Interrupt from Timeout Gasket

There are multiple timeout gaskets in SoC, some of them are inserted at the initiator side and the others are inserted at the target interface side. Each timeout gasket module generates its own interrupt. Instead of routing the interrupt from each timeout gasket individually to the processor, the interrupts from all the timeout gasket inserted at the target sides are aggregated together (ORed), and all the interrupts from all the timeout gasket inserted at the initiator sides are aggregated together (ORed). The status of each interrupt status is captured by the TOG_STAT registers in both WKUP_CTRL_MMR and MCU_CTRL_MMR. Refer to Figure 10-7.

Copyright © 2026 Texas Instruments Incorporated

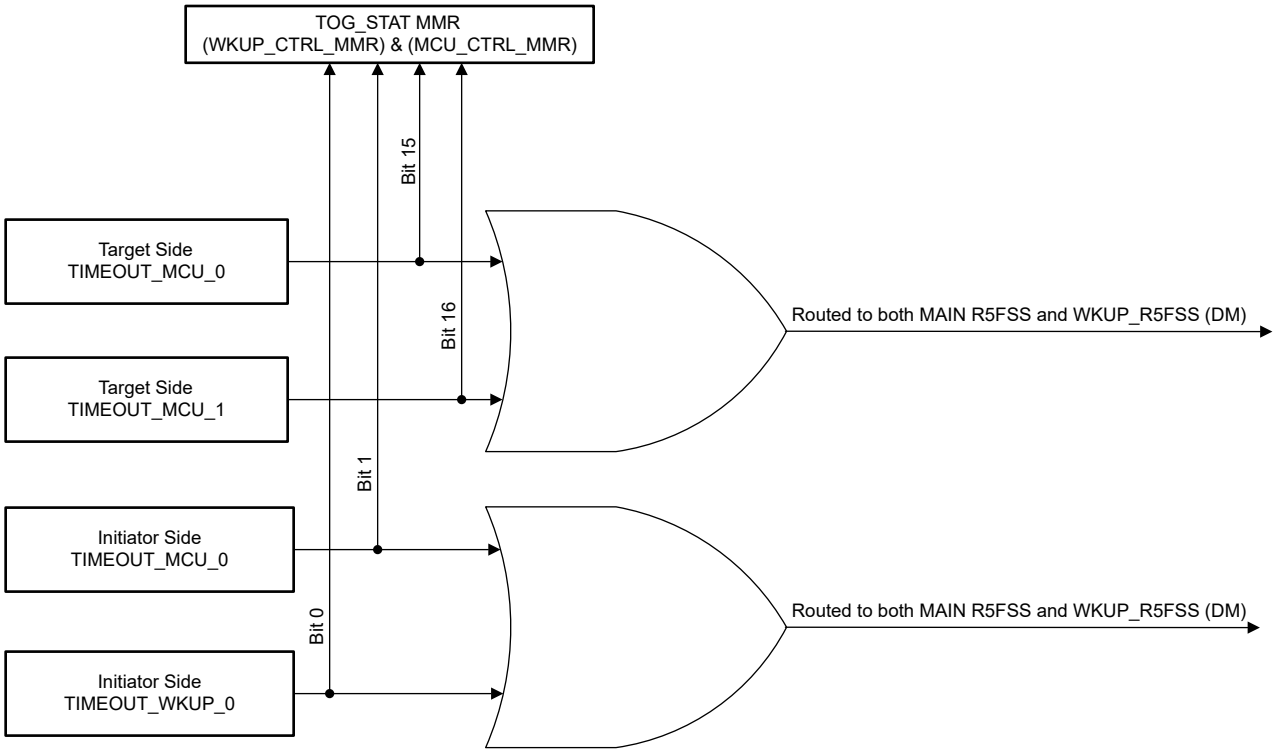


Figure 10-7. Interrupt from Timeout Gaskets

10.1.4.2 Aggregated DCC Interrupt

There are multiple DCC modules in the main domain and each DCC module generates its own DCC_done interrupt. The DCC_done interrupt from all the DCC modules in the main domain are aggregated (ORed) together before it is routed to all the processors, refer to Figure 10-8.

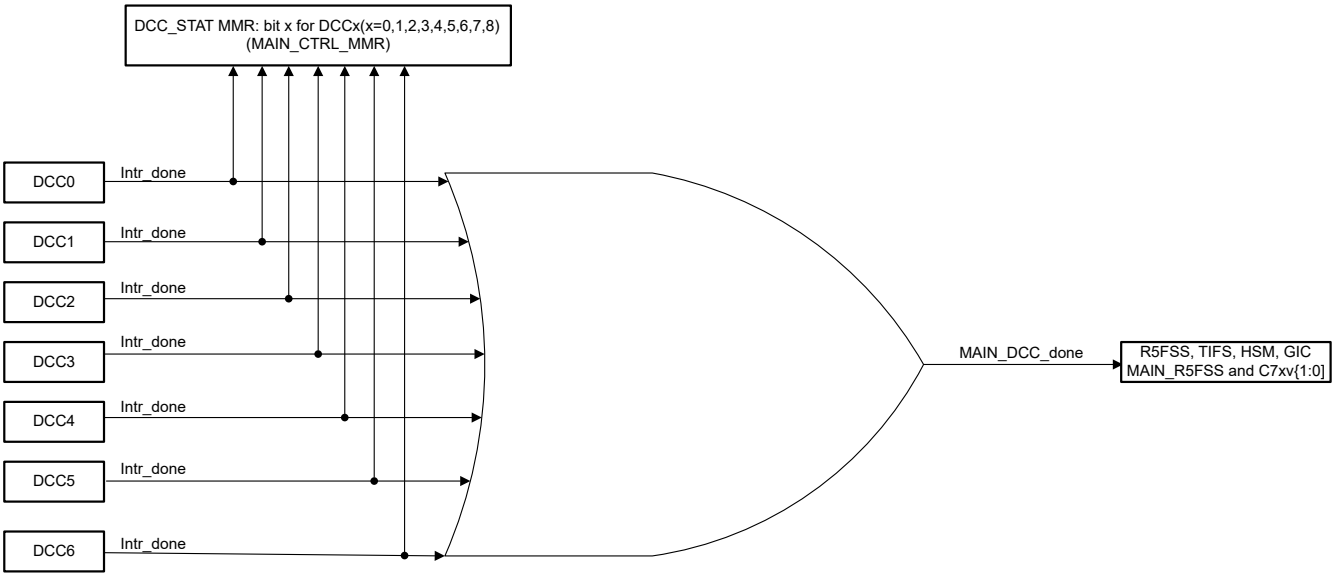


Figure 10-8. DCC Intr_done Interrupt Aggregation

Copyright © 2026 Texas Instruments Incorporated

10.1.4.3 Aggregated Access Error Interrupt from CBASS

CBASS modules are the interconnection blocks on the SoC level to provide access path between initiators and targets. When the transaction is not completed successfully, an interrupt is generated by the CBASS and the error transaction is logged by the CBASS module.

There are multiple CBASS modules on the SoC level to provide three layers of interconnect: the data plane, security configuration plane and debug configuration plane. The CBASS access error from the security configuration plane is only routed to TIFS and HSM, and the other CBASS access errors are aggregated before routing it to the processors, refer to Figure 10-9.

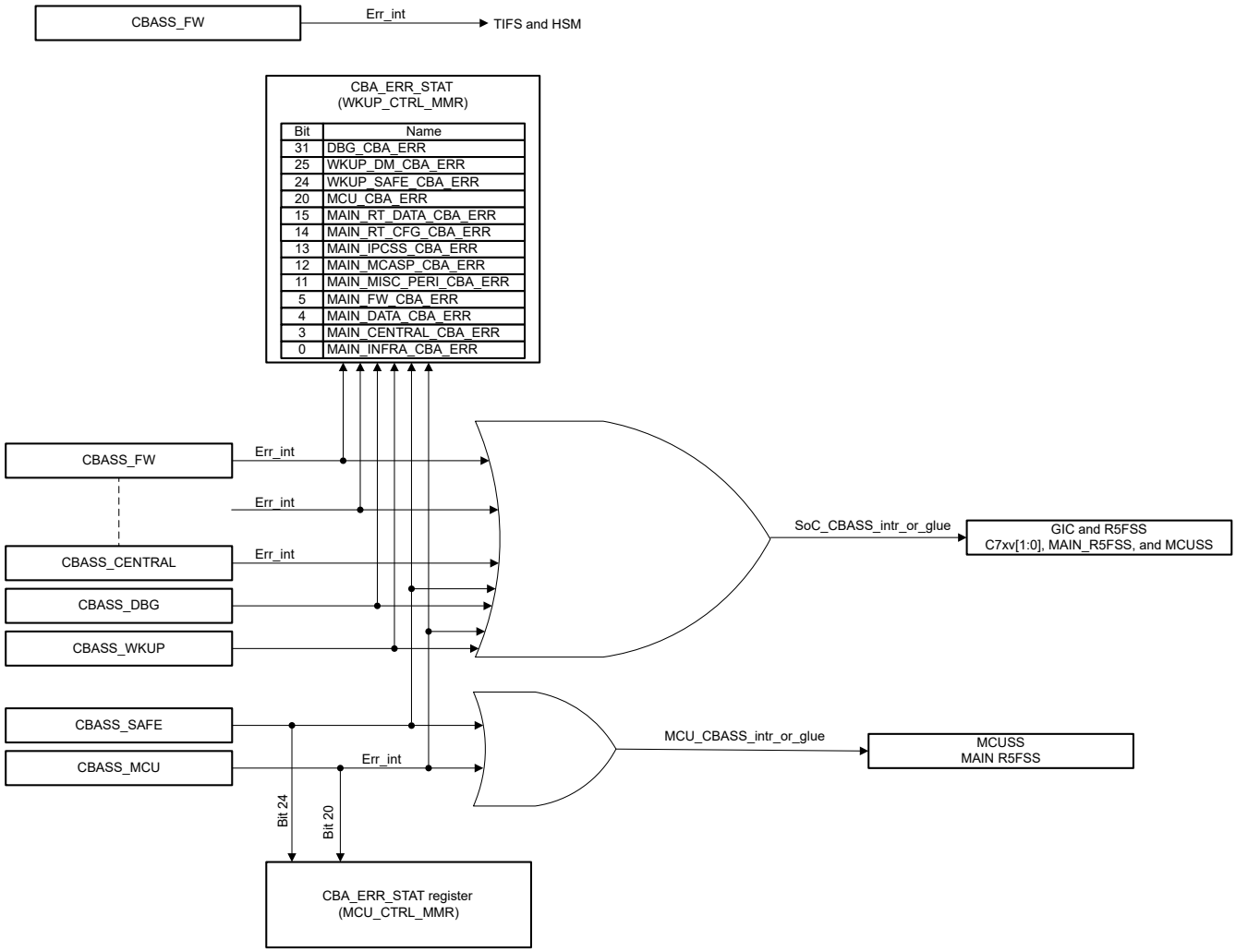


Figure 10-9. Aggregated CBASS Access Error

10.1.4.4 Access Error to Control Register Block Interrupt Aggregation

There are multiple register blocks on the SoC level, and each register block generates an error interrupt when there is an access error. Those access error interrupts are aggregated together before they are routed to processor, refer to Figure 10-10.

Copyright © 2026 Texas Instruments Incorporated

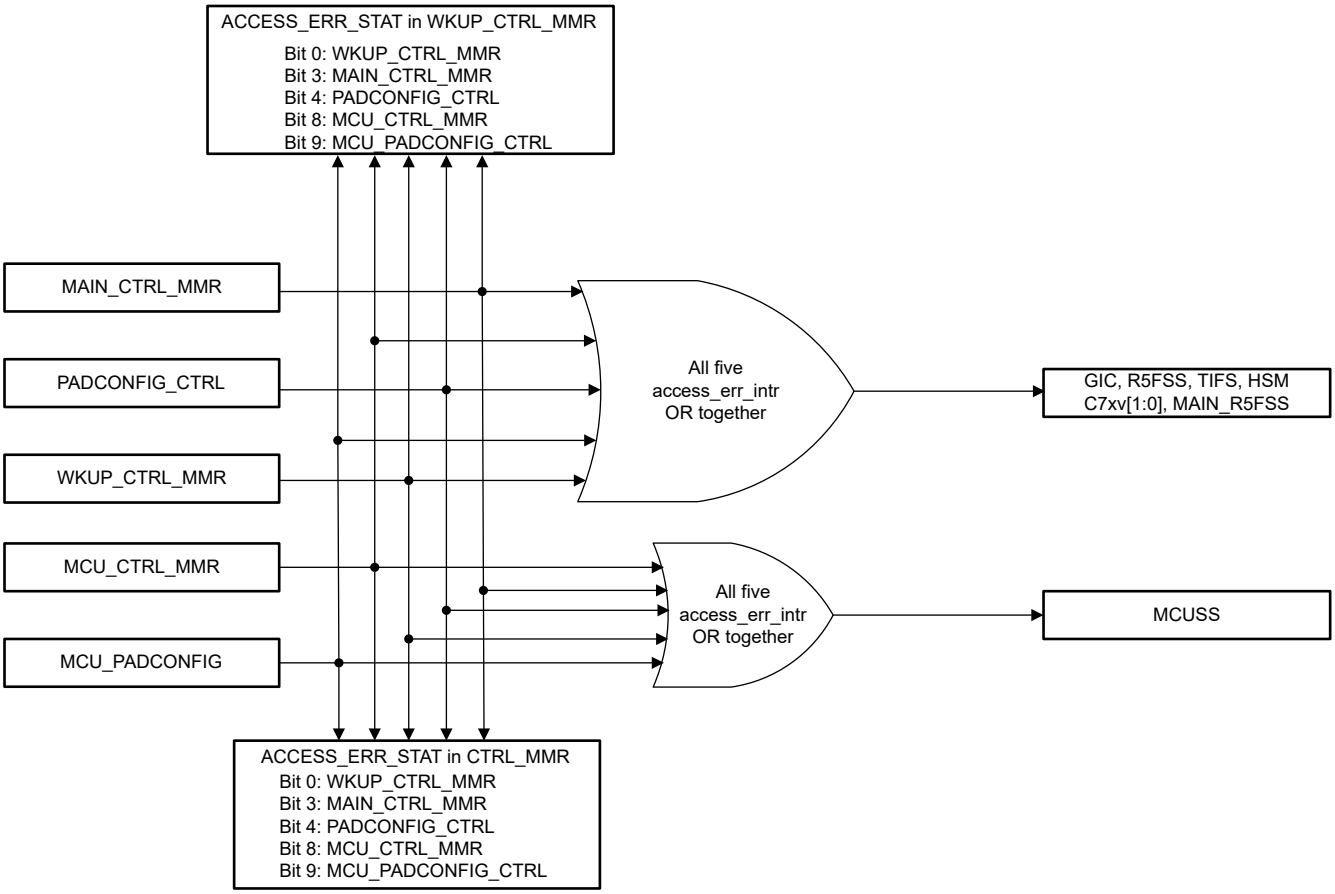


Figure 10-10. Access Error Aggregation

10.2 Interrupt Controllers

10.2.1 Generic Interrupt Controller (GICSS)

This section describes the Generic Interrupt Controller (GICSS) module in the device.

10.2.1.1 GICSS Overview

The device GICSS is a TI module that is based on the Arm GIC-500 interrupt controller. The Arm GIC-500 is a high-performance, area-optimized, built-time configurable interrupt controller which detects, manages, and distributes system interrupts to the Arm Cortex-A53 processors in the Compute Cluster.

The Arm GIC-500 is compliant to the Arm GICv3 standard. It only supports cores (such as A53) that implement the Armv8 architecture and the GIC CPU interface with the standard GIC Stream Protocol Interface. The GICSS includes additional logic (TI wrapper) to fully integrate the Arm GIC-500 into the SoC.

10.2.1.1.1 GICSS Features

Note

This chapter provides only a brief description of the GICSS functionality and features. For more details on this module, refer to the *Arm®CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

The Arm GIC-500 supports the following features in the device:

- Module revision: r1p1
- Supports both cores in the dual-A53 MPU cluster

Copyright © 2026 Texas Instruments Incorporated

- 16 software generated interrupts (SGIs) per core
- 16 private peripheral interrupts (PPIs) per core
- 256 shared peripheral interrupts (SPIs)
- Locality-specific peripheral interrupts (LPIs), used for message-based interrupts
- Interrupt translation service (ITS)
  - Device isolation
  - ID translation for message-based interrupts
  - This allows virtual machines to program devices directly
- Interrupt masking and prioritization
- Programmable, affinity-based interrupt routing
- 32 priorities for each interrupt
- Three different interrupt groups
  - Group 0
  - Non-secure group 1
  - Secure group 1

Additionally, the GICSS wrapper logic provides the following features:

- Synchronizes interrupt inputs to GICSS clock
- Implements AXI2VBUSM and VBUSM2AXI bridges
  - Used for bus protocol conversion (AXI-to-VBUSM and VBUSM-to-AXI, respectively)
- Implements an integrated ECC aggregator
  - Only used to inject errors (for testing purposes)

Note

Some features may not be available. See *Module Integration* for more information.

10.2.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Copyright © 2026 Texas Instruments Incorporated

10.2.1.2 GICSS Integration

See GICSS in the Module Integration section for information about clocks, resets, and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

10.2.1.3 GICSS Functional Description

Note

This chapter provides only a brief description of the GICSS functionality and features. For more details on this module, refer to the *Arm®CoreLink™ GIC-500 Generic Interrupt Controller Technical Reference Manual*.

10.2.1.3.1 GICSS Block Diagram

Figure 10-11 shows the GICSS block diagram.

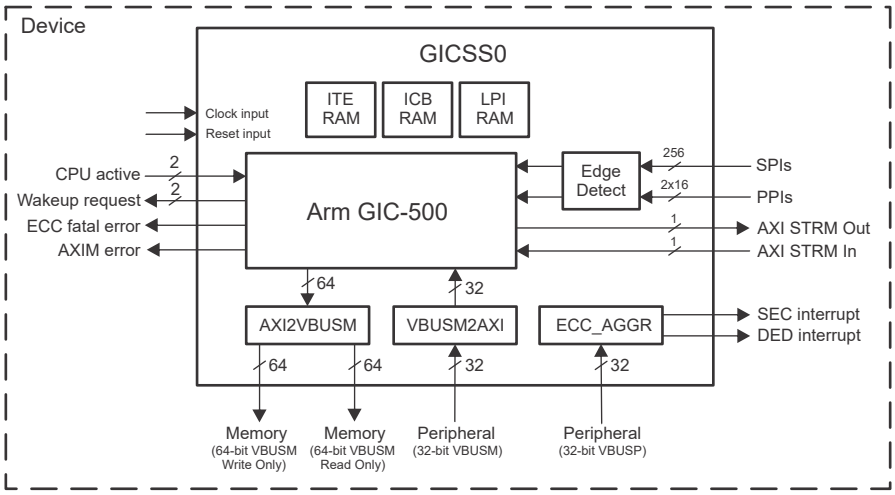


Figure 10-11. GICSS Block Diagram

10.2.1.3.2 Arm GIC-500

The Arm GIC-500 is responsible for detecting, managing and communicating system interrupts to the dual-core A53 cluster. The main difference between the Arm GIC-500 and previous Arm GIC versions is the direct communication with the CPU(s). Previously, an interrupt controller would set a pending bit to a CPU and that CPU would then have to query an interrupt controller MMR to find out what to do. Now, the Arm GIC-500 sends interrupts to a CPU via a dedicated message interface and the CPU communicates back about interrupts through the same interface. The CPU now uses writes and reads to system registers instead of over the memory interface. This leads to reduced latency and the ability to route interrupts to different CPUs based on a set of rules.

The Arm GIC-500 is divided into three main sections:

- **Distributor**: The Distributor receives interrupts from the wire interrupts and the programming interface. It is responsible for prioritizing these interrupts and sending them to the CPU interface via the GIC Stream Protocol Interface.
- **Redistributor**: There is one Redistributor per core. Each Redistributor holds the state that is individual to a particular core (such as the settings for PPIs and SGIs). It also stores the LPIs for that core after they have been generated using the ITS.
- **ITS**: The ITS is responsible for translating message-based interrupts from device peripherals into LPIs. The user can also use the ITS to manage existing LPIs. Note that the ITS is not used for other types of interrupt

10.2.1.3.3 GICSS Interrupt Types

The GICSS supports four types of interrupts:

- *Software Generated Interrupts (SGIs)*: There are 16 SGIs (ID0-ID15). These are inter-processor interrupts. SGIs can be sent using either Arm system registers or by writing to the Software Generated Interrupt Register (GICD_SGIR).

Copyright © 2026 Texas Instruments Incorporated

- *Private Peripheral Interrupts (PPIs)*: There are 16 PPIs (ID16-ID31). These are wired interrupts dedicated to a specific CPU. Many are reserved to specific functions via convention. These can be either active-low levels (by default), or rising edge triggered (software programmable).
- *Shared Peripheral Interrupts (SPIs)*: There are 256 SPIs (ID32-ID288). These are wired interrupts that can be routed to any core or cluster, based on the programming of that interrupt in the GICSS. These are active-high levels (by default), or rising edge triggered (software programmable).
- *Locality-Specific Peripheral Interrupts (LPIs)*: There are 57,344 of these. These interrupts are used for message-based interrupts from a peripheral. They are generated through writes to the GICSS peripheral interface. The information associated with these is located in memory. The GICSS keeps a small 64-entries cache on-board in order to reduce latency either for specific interrupts or for most-recently-used. LPIs can be routed to different CPUs based on programmed rules.

10.2.1.3.4 GICSS Interfaces

The GICSS has the following main interfaces:

- A pair of 64-bit VBUSM write-only and read-only controller interfaces
  - AXI2VBUSM bridge converts the standard 64-bit AXI4 controller interface into this pair of interfaces
  - The AXI4 controller interface allows the GICSS ITS and redistributors to access main memory
- 32-bit VBUSM peripheral interface
  - Provides access to Arm GIC-500 internal resources
  - Handles all message-based interrupts. Message-based interrupts can generate SPIs or LPIs, depending on the register that is written
  - VBUSM2AXI bridge converts this interface into an 32-bit AXI4 peripheral interface
- 32-bit VBUSP peripheral interface
  - Provides access to internal ECC aggregator
- Physical interrupt and power signals
  - Inputs: CPU active signals, SPIs and PPIs (can be programmed as either level-sensitive, or edge-triggered)
  - Outputs: Error signals and wake-up requests
- GIC Stream Protocol Interface
  - Consists of a pair of AXI4-Stream interfaces (one upstream and one downstream 16-bit AXI4-Stream interface) that the GICSS uses to send interrupts to the core and receive notifications when the core activates interrupts
  - There is a pair of physical interfaces, one in each direction, for each cluster

10.2.1.3.5 GICSS Interrupt Outputs

Table 10-2 lists the interrupts that are generated by the GICSS.

Table 10-2. GICSS Interrupt Outputs

Interrupt Name	Description
GICSS0_AXIM_ERR_0	GICSS bus error interrupt. This interrupt is triggered if the GICSS receives an error on a bus transaction, such as a decode or protection error. This is a pulse-high interrupt. If this interrupt occurs, the GICSS might lose interrupts and must be reset. If it is not reset, behavior becomes unpredictable.
GICSS0_ECC_FATAL_0	GICSS uncorrectable ECC error interrupt. Indicates an uncorrectable 2-bit error detected in one of the ITS cache memories. This is a pulse-high interrupt. If this interrupt occurs, the GICSS might lose interrupts and must be reset. If it is not reset, behavior becomes unpredictable.
GICSS0_ECC_AGGR_CORR_LEVEL_0	GICSS ECC aggregator correctable (SEC) error interrupt
GICSS0_ECC_AGGR_UNCORR_LEVEL_0	GICSS ECC aggregator uncorrectable (DED) error interrupt
GICSS0_GIC_PWR0_WAKE_REQUEST_0	GICSS wake requests for A53 cores. Asserted state indicates that an interrupt is pending for a processor that has set the PROCESSORSLEEP bit in the GICR_WAKER register. This bit indicates that the SoC should wake up the indicated CPU so that it can process the interrupt.
GICSS0_GIC_PWR0_WAKE_REQUEST_1	

Copyright © 2026 Texas Instruments Incorporated

10.2.1.3.6 GICSS ECC Support

The Arm GIC-500 has built-in SECDED ECC on its memories to protect against errors. The syndrome generation and checking is done internally.

Additionally, the GICSS wrapper integrates an ECC aggregator (GICSS0_ECC_AGGR) in order to allow errors to be injected for testing purposes. The generic ECC aggregator functionality is described in *ECC Aggregator*. Note that the GICSS0_ECC_AGGR supports only a subset of this functionality.

Table 10-3 shows the memory ID for each ECC endpoint. The corresponding memory ID needs to be written in the GICSS0_ECC_AGGR_VECTOR[10-0] ECC_VECTOR bit field for proper operation.

Table 10-3. GICSS0_ECC_AGGR Memory ID Mapping

ECC Aggregator	Memory ID	ECC Endpoint
GICSS0_ECC_AGGR	0	ICB RAM
	1	ITE RAM
	2	LPI RAM
	3	VBUSM2AXI bridge
	4	AXI2VBUSM read bridge
	5	AXI2VBUSM write bridge

10.2.1.3.7 GICSS AXI2VBUSM and VBUSM2AXI Bridges

The GICSS uses the AXI4 controller interface to allow the ITS and redistributors to access main memory. It is expected that the main memory (attached to a CBASS VBUSM port) holds the following:

- ITS translation tables
- ITS command queue, which is written by software in order to program the ITS
- LPI configuration table, which holds the priorities and enable bits for LPIs
- LPI pending tables, which can hold information on whether each LPI is pending on each core

The AXI2VBUSM bridge converts the standard 64-bit AXI4 controller interface into a pair of 64-bit VBUSM write-only and read-only controller interfaces. AXI4 has a separate command channel for read and write, and the AXI2VBUSM bridge keeps them separate, so that the system may prioritize or arbitrate the traffic between them as appropriate.

The GICSS uses the AXI4 peripheral interface to provide access to the programming interfaces of all its parts (distributor, redistributors, ITS). The VBUSM2AXI bridge converts the standard 32-bit VBUSM controller interface into an 32-bit AXI4 peripheral interface.

Note

The AXI2VBUSM and VBUSM2AXI bridges do not implement any MMRs.

10.3 Interrupt Router (INTROUTER)

10.3.1 INTROUTER Integration

See the Interrupts section of the Module Integration Chapter for details on clocks, resets and hardware requests.

10.4 Interrupt Sources

Copyright © 2026 Texas Instruments Incorporated

10.4.1 Interrupts (Inputs)

Table 10-4. Interrupts (Inputs)

Instance	Interrupt Input Line	Source Interrupt
PINFUNCTION_CP_GEMAC_CPTS0_TS_COMPout	PINFUNCTION_CP_GEMAC_CPTS0_TS_COMPOUT_CP_GEMAC_CPTS0_TS_COMP_IN_0	CPSW0_CPTS_COMP_OUT_0
PINFUNCTION_CP_GEMAC_CPTS0_TS_SYNCout	PINFUNCTION_CP_GEMAC_CPTS0_TS_SYNCOUT_CP_GEMAC_CPTS0_TS_SYNC_IN_0	CPSW0_CPTS_SYNC_OUT_0
PINFUNCTION_SYNC0_OUTout	PINFUNCTION_SYNC0_OUTOUT_SYNC0_OUT_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_20
PINFUNCTION_SYNC1_OUTout	PINFUNCTION_SYNC1_OUTOUT_SYNC1_OUT_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_21
PINFUNCTION_SYNC2_OUTout	PINFUNCTION_SYNC2_OUTOUT_SYNC2_OUT_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_22
PINFUNCTION_SYNC3_OUTout	PINFUNCTION_SYNC3_OUTOUT_SYNC3_OUT_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_23
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_0	GPIO0_GPIO_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_1	GPIO0_GPIO_OUT_1
	MAIN_GPIOMUX_INTROUTER0_IN_IN_2	GPIO0_GPIO_OUT_2
	MAIN_GPIOMUX_INTROUTER0_IN_IN_3	GPIO0_GPIO_OUT_3
	MAIN_GPIOMUX_INTROUTER0_IN_IN_4	GPIO0_GPIO_OUT_4
	MAIN_GPIOMUX_INTROUTER0_IN_IN_5	GPIO0_GPIO_OUT_5
	MAIN_GPIOMUX_INTROUTER0_IN_IN_6	GPIO0_GPIO_OUT_6
	MAIN_GPIOMUX_INTROUTER0_IN_IN_7	GPIO0_GPIO_OUT_7
	MAIN_GPIOMUX_INTROUTER0_IN_IN_8	GPIO0_GPIO_OUT_8
	MAIN_GPIOMUX_INTROUTER0_IN_IN_9	GPIO0_GPIO_OUT_9
	MAIN_GPIOMUX_INTROUTER0_IN_IN_10	GPIO0_GPIO_OUT_10
	MAIN_GPIOMUX_INTROUTER0_IN_IN_11	GPIO0_GPIO_OUT_11
	MAIN_GPIOMUX_INTROUTER0_IN_IN_12	GPIO0_GPIO_OUT_12
	MAIN_GPIOMUX_INTROUTER0_IN_IN_13	GPIO0_GPIO_OUT_13
	MAIN_GPIOMUX_INTROUTER0_IN_IN_14	GPIO0_GPIO_OUT_14
	MAIN_GPIOMUX_INTROUTER0_IN_IN_15	GPIO0_GPIO_OUT_15
	MAIN_GPIOMUX_INTROUTER0_IN_IN_16	GPIO0_GPIO_OUT_16
	MAIN_GPIOMUX_INTROUTER0_IN_IN_17	GPIO0_GPIO_OUT_17
	MAIN_GPIOMUX_INTROUTER0_IN_IN_18	GPIO0_GPIO_OUT_18
	MAIN_GPIOMUX_INTROUTER0_IN_IN_19	GPIO0_GPIO_OUT_19
	MAIN_GPIOMUX_INTROUTER0_IN_IN_20	GPIO0_GPIO_OUT_20
	MAIN_GPIOMUX_INTROUTER0_IN_IN_21	GPIO0_GPIO_OUT_21
	MAIN_GPIOMUX_INTROUTER0_IN_IN_22	GPIO0_GPIO_OUT_22
	MAIN_GPIOMUX_INTROUTER0_IN_IN_23	GPIO0_GPIO_OUT_23
	MAIN_GPIOMUX_INTROUTER0_IN_IN_24	GPIO0_GPIO_OUT_24
	MAIN_GPIOMUX_INTROUTER0_IN_IN_25	GPIO0_GPIO_OUT_25
	MAIN_GPIOMUX_INTROUTER0_IN_IN_26	GPIO0_GPIO_OUT_26

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_27	GPIO0_GPIO_OUT_27
	MAIN_GPIOMUX_INTROUTER0_IN_IN_28	GPIO0_GPIO_OUT_28
	MAIN_GPIOMUX_INTROUTER0_IN_IN_29	GPIO0_GPIO_OUT_29
	MAIN_GPIOMUX_INTROUTER0_IN_IN_30	GPIO0_GPIO_OUT_30
	MAIN_GPIOMUX_INTROUTER0_IN_IN_31	GPIO0_GPIO_OUT_31
	MAIN_GPIOMUX_INTROUTER0_IN_IN_32	GPIO0_GPIO_OUT_32
	MAIN_GPIOMUX_INTROUTER0_IN_IN_33	GPIO0_GPIO_OUT_33
	MAIN_GPIOMUX_INTROUTER0_IN_IN_34	GPIO0_GPIO_OUT_34
	MAIN_GPIOMUX_INTROUTER0_IN_IN_35	GPIO0_GPIO_OUT_35
	MAIN_GPIOMUX_INTROUTER0_IN_IN_36	GPIO0_GPIO_OUT_36
	MAIN_GPIOMUX_INTROUTER0_IN_IN_37	GPIO0_GPIO_OUT_37
	MAIN_GPIOMUX_INTROUTER0_IN_IN_38	GPIO0_GPIO_OUT_38
	MAIN_GPIOMUX_INTROUTER0_IN_IN_39	GPIO0_GPIO_OUT_39
	MAIN_GPIOMUX_INTROUTER0_IN_IN_40	GPIO0_GPIO_OUT_40
	MAIN_GPIOMUX_INTROUTER0_IN_IN_41	GPIO0_GPIO_OUT_41
	MAIN_GPIOMUX_INTROUTER0_IN_IN_42	GPIO0_GPIO_OUT_42
	MAIN_GPIOMUX_INTROUTER0_IN_IN_43	GPIO0_GPIO_OUT_43
	MAIN_GPIOMUX_INTROUTER0_IN_IN_44	GPIO0_GPIO_OUT_44
	MAIN_GPIOMUX_INTROUTER0_IN_IN_45	GPIO0_GPIO_OUT_45
	MAIN_GPIOMUX_INTROUTER0_IN_IN_46	GPIO0_GPIO_OUT_46
	MAIN_GPIOMUX_INTROUTER0_IN_IN_47	GPIO0_GPIO_OUT_47
	MAIN_GPIOMUX_INTROUTER0_IN_IN_48	GPIO0_GPIO_OUT_48
	MAIN_GPIOMUX_INTROUTER0_IN_IN_49	GPIO0_GPIO_OUT_49
	MAIN_GPIOMUX_INTROUTER0_IN_IN_50	GPIO0_GPIO_OUT_50
	MAIN_GPIOMUX_INTROUTER0_IN_IN_51	GPIO0_GPIO_OUT_51
	MAIN_GPIOMUX_INTROUTER0_IN_IN_52	GPIO0_GPIO_OUT_52
	MAIN_GPIOMUX_INTROUTER0_IN_IN_53	GPIO0_GPIO_OUT_53
	MAIN_GPIOMUX_INTROUTER0_IN_IN_54	GPIO0_GPIO_OUT_54
	MAIN_GPIOMUX_INTROUTER0_IN_IN_55	GPIO0_GPIO_OUT_55
	MAIN_GPIOMUX_INTROUTER0_IN_IN_56	GPIO0_GPIO_OUT_56
	MAIN_GPIOMUX_INTROUTER0_IN_IN_57	GPIO0_GPIO_OUT_57
	MAIN_GPIOMUX_INTROUTER0_IN_IN_58	GPIO0_GPIO_OUT_58
	MAIN_GPIOMUX_INTROUTER0_IN_IN_59	GPIO0_GPIO_OUT_59

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_60	GPIO0_GPIO_OUT_60
	MAIN_GPIOMUX_INTROUTER0_IN_IN_61	GPIO0_GPIO_OUT_61
	MAIN_GPIOMUX_INTROUTER0_IN_IN_62	GPIO0_GPIO_OUT_62
	MAIN_GPIOMUX_INTROUTER0_IN_IN_63	GPIO0_GPIO_OUT_63
	MAIN_GPIOMUX_INTROUTER0_IN_IN_64	GPIO0_GPIO_OUT_64
	MAIN_GPIOMUX_INTROUTER0_IN_IN_65	GPIO0_GPIO_OUT_65
	MAIN_GPIOMUX_INTROUTER0_IN_IN_66	GPIO0_GPIO_OUT_66
	MAIN_GPIOMUX_INTROUTER0_IN_IN_67	GPIO0_GPIO_OUT_67
	MAIN_GPIOMUX_INTROUTER0_IN_IN_68	GPIO0_GPIO_OUT_68
	MAIN_GPIOMUX_INTROUTER0_IN_IN_69	GPIO0_GPIO_OUT_69
	MAIN_GPIOMUX_INTROUTER0_IN_IN_70	GPIO0_GPIO_OUT_70
	MAIN_GPIOMUX_INTROUTER0_IN_IN_71	GPIO0_GPIO_OUT_71
	MAIN_GPIOMUX_INTROUTER0_IN_IN_72	GPIO0_GPIO_OUT_72
	MAIN_GPIOMUX_INTROUTER0_IN_IN_73	GPIO0_GPIO_OUT_73
	MAIN_GPIOMUX_INTROUTER0_IN_IN_74	GPIO0_GPIO_OUT_74
	MAIN_GPIOMUX_INTROUTER0_IN_IN_75	GPIO0_GPIO_OUT_75
	MAIN_GPIOMUX_INTROUTER0_IN_IN_76	GPIO0_GPIO_OUT_76
	MAIN_GPIOMUX_INTROUTER0_IN_IN_77	GPIO0_GPIO_OUT_77
	MAIN_GPIOMUX_INTROUTER0_IN_IN_78	GPIO0_GPIO_OUT_78
	MAIN_GPIOMUX_INTROUTER0_IN_IN_79	GPIO0_GPIO_OUT_79
	MAIN_GPIOMUX_INTROUTER0_IN_IN_80	GPIO0_GPIO_OUT_80
	MAIN_GPIOMUX_INTROUTER0_IN_IN_81	GPIO0_GPIO_OUT_81
	MAIN_GPIOMUX_INTROUTER0_IN_IN_82	GPIO0_GPIO_OUT_82
	MAIN_GPIOMUX_INTROUTER0_IN_IN_83	GPIO0_GPIO_OUT_83
	MAIN_GPIOMUX_INTROUTER0_IN_IN_84	GPIO0_GPIO_OUT_84
	MAIN_GPIOMUX_INTROUTER0_IN_IN_85	GPIO0_GPIO_OUT_85
	MAIN_GPIOMUX_INTROUTER0_IN_IN_86	GPIO0_GPIO_OUT_86
	MAIN_GPIOMUX_INTROUTER0_IN_IN_87	GPIO0_GPIO_OUT_87
	MAIN_GPIOMUX_INTROUTER0_IN_IN_88	GPIO0_GPIO_OUT_88
	MAIN_GPIOMUX_INTROUTER0_IN_IN_89	GPIO0_GPIO_OUT_89
	MAIN_GPIOMUX_INTROUTER0_IN_IN_90	GPIO1_GPIO_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_91	GPIO1_GPIO_OUT_1
	MAIN_GPIOMUX_INTROUTER0_IN_IN_92	GPIO1_GPIO_OUT_2

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_93	GPIO1_GPIO_OUT_3
	MAIN_GPIOMUX_INTROUTER0_IN_IN_94	GPIO1_GPIO_OUT_4
	MAIN_GPIOMUX_INTROUTER0_IN_IN_95	GPIO1_GPIO_OUT_5
	MAIN_GPIOMUX_INTROUTER0_IN_IN_96	GPIO1_GPIO_OUT_6
	MAIN_GPIOMUX_INTROUTER0_IN_IN_97	GPIO1_GPIO_OUT_7
	MAIN_GPIOMUX_INTROUTER0_IN_IN_98	GPIO1_GPIO_OUT_8
	MAIN_GPIOMUX_INTROUTER0_IN_IN_99	GPIO1_GPIO_OUT_9
	MAIN_GPIOMUX_INTROUTER0_IN_IN_100	GPIO1_GPIO_OUT_10
	MAIN_GPIOMUX_INTROUTER0_IN_IN_101	GPIO1_GPIO_OUT_11
	MAIN_GPIOMUX_INTROUTER0_IN_IN_102	GPIO1_GPIO_OUT_12
	MAIN_GPIOMUX_INTROUTER0_IN_IN_103	GPIO1_GPIO_OUT_13
	MAIN_GPIOMUX_INTROUTER0_IN_IN_104	GPIO1_GPIO_OUT_14
	MAIN_GPIOMUX_INTROUTER0_IN_IN_105	GPIO1_GPIO_OUT_15
	MAIN_GPIOMUX_INTROUTER0_IN_IN_106	GPIO1_GPIO_OUT_16
	MAIN_GPIOMUX_INTROUTER0_IN_IN_107	GPIO1_GPIO_OUT_17
	MAIN_GPIOMUX_INTROUTER0_IN_IN_108	GPIO1_GPIO_OUT_18
	MAIN_GPIOMUX_INTROUTER0_IN_IN_109	GPIO1_GPIO_OUT_19
	MAIN_GPIOMUX_INTROUTER0_IN_IN_110	GPIO1_GPIO_OUT_20
	MAIN_GPIOMUX_INTROUTER0_IN_IN_111	GPIO1_GPIO_OUT_21
	MAIN_GPIOMUX_INTROUTER0_IN_IN_112	GPIO1_GPIO_OUT_22
	MAIN_GPIOMUX_INTROUTER0_IN_IN_113	GPIO1_GPIO_OUT_23
	MAIN_GPIOMUX_INTROUTER0_IN_IN_114	GPIO1_GPIO_OUT_24
	MAIN_GPIOMUX_INTROUTER0_IN_IN_115	GPIO1_GPIO_OUT_25
	MAIN_GPIOMUX_INTROUTER0_IN_IN_116	GPIO1_GPIO_OUT_26
	MAIN_GPIOMUX_INTROUTER0_IN_IN_117	GPIO1_GPIO_OUT_27
	MAIN_GPIOMUX_INTROUTER0_IN_IN_118	GPIO1_GPIO_OUT_28
	MAIN_GPIOMUX_INTROUTER0_IN_IN_119	GPIO1_GPIO_OUT_29
	MAIN_GPIOMUX_INTROUTER0_IN_IN_120	GPIO1_GPIO_OUT_30
	MAIN_GPIOMUX_INTROUTER0_IN_IN_121	GPIO1_GPIO_OUT_31
	MAIN_GPIOMUX_INTROUTER0_IN_IN_122	GPIO1_GPIO_OUT_32
	MAIN_GPIOMUX_INTROUTER0_IN_IN_123	GPIO1_GPIO_OUT_33
	MAIN_GPIOMUX_INTROUTER0_IN_IN_124	GPIO1_GPIO_OUT_34
	MAIN_GPIOMUX_INTROUTER0_IN_IN_125	GPIO1_GPIO_OUT_35

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_126	GPIO1_GPIO_OUT_36
	MAIN_GPIOMUX_INTROUTER0_IN_IN_127	GPIO1_GPIO_OUT_37
	MAIN_GPIOMUX_INTROUTER0_IN_IN_128	GPIO1_GPIO_OUT_38
	MAIN_GPIOMUX_INTROUTER0_IN_IN_129	GPIO1_GPIO_OUT_39
	MAIN_GPIOMUX_INTROUTER0_IN_IN_130	GPIO1_GPIO_OUT_40
	MAIN_GPIOMUX_INTROUTER0_IN_IN_131	GPIO1_GPIO_OUT_41
	MAIN_GPIOMUX_INTROUTER0_IN_IN_132	GPIO1_GPIO_OUT_42
	MAIN_GPIOMUX_INTROUTER0_IN_IN_133	GPIO1_GPIO_OUT_43
	MAIN_GPIOMUX_INTROUTER0_IN_IN_134	GPIO1_GPIO_OUT_44
	MAIN_GPIOMUX_INTROUTER0_IN_IN_135	GPIO1_GPIO_OUT_45
	MAIN_GPIOMUX_INTROUTER0_IN_IN_136	GPIO1_GPIO_OUT_46
	MAIN_GPIOMUX_INTROUTER0_IN_IN_137	GPIO1_GPIO_OUT_47
	MAIN_GPIOMUX_INTROUTER0_IN_IN_138	GPIO1_GPIO_OUT_48
	MAIN_GPIOMUX_INTROUTER0_IN_IN_139	GPIO1_GPIO_OUT_49
	MAIN_GPIOMUX_INTROUTER0_IN_IN_140	GPIO1_GPIO_OUT_50
	MAIN_GPIOMUX_INTROUTER0_IN_IN_141	GPIO1_GPIO_OUT_51
	MAIN_GPIOMUX_INTROUTER0_IN_IN_142	GPIO1_GPIO_OUT_52
	MAIN_GPIOMUX_INTROUTER0_IN_IN_143	GPIO1_GPIO_OUT_53
	MAIN_GPIOMUX_INTROUTER0_IN_IN_144	GPIO1_GPIO_OUT_54
	MAIN_GPIOMUX_INTROUTER0_IN_IN_145	GPIO1_GPIO_OUT_55
	MAIN_GPIOMUX_INTROUTER0_IN_IN_146	GPIO1_GPIO_OUT_56
	MAIN_GPIOMUX_INTROUTER0_IN_IN_147	GPIO1_GPIO_OUT_57
	MAIN_GPIOMUX_INTROUTER0_IN_IN_148	GPIO1_GPIO_OUT_58
	MAIN_GPIOMUX_INTROUTER0_IN_IN_149	GPIO1_GPIO_OUT_59
	MAIN_GPIOMUX_INTROUTER0_IN_IN_150	GPIO1_GPIO_OUT_60
	MAIN_GPIOMUX_INTROUTER0_IN_IN_151	GPIO1_GPIO_OUT_61
	MAIN_GPIOMUX_INTROUTER0_IN_IN_152	GPIO1_GPIO_OUT_62
	MAIN_GPIOMUX_INTROUTER0_IN_IN_153	GPIO1_GPIO_OUT_63
	MAIN_GPIOMUX_INTROUTER0_IN_IN_154	GPIO1_GPIO_OUT_64
	MAIN_GPIOMUX_INTROUTER0_IN_IN_155	GPIO1_GPIO_OUT_65
	MAIN_GPIOMUX_INTROUTER0_IN_IN_156	GPIO1_GPIO_OUT_66
	MAIN_GPIOMUX_INTROUTER0_IN_IN_157	GPIO1_GPIO_OUT_67
	MAIN_GPIOMUX_INTROUTER0_IN_IN_158	GPIO1_GPIO_OUT_68

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_159	GPIO1_GPIO_OUT_69
	MAIN_GPIOMUX_INTROUTER0_IN_IN_160	GPIO1_GPIO_OUT_70
	MAIN_GPIOMUX_INTROUTER0_IN_IN_161	GPIO1_GPIO_OUT_71
	MAIN_GPIOMUX_INTROUTER0_IN_IN_162	TIMER0_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_163	TIMER1_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_164	TIMER2_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_165	TIMER3_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_166	TIMER4_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_167	TIMER5_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_168	TIMER6_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_169	TIMER7_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_170	MCU_TIMER0_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_171	MCU_TIMER1_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_172	MCU_TIMER2_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_173	MCU_TIMER3_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_174	WKUP_TIMER0_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_175	WKUP_TIMER1_TIMER_PWM_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_176	GPIO0_GPIO_OUT_90
	MAIN_GPIOMUX_INTROUTER0_IN_IN_177	GPIO0_GPIO_OUT_91
	MAIN_GPIOMUX_INTROUTER0_IN_IN_178	I2C4_POINTRPEND_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_180	GPIO1_GPIO_BANK_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_181	GPIO1_GPIO_BANK_OUT_1
	MAIN_GPIOMUX_INTROUTER0_IN_IN_182	GPIO1_GPIO_BANK_OUT_2
	MAIN_GPIOMUX_INTROUTER0_IN_IN_183	GPIO1_GPIO_BANK_OUT_3
	MAIN_GPIOMUX_INTROUTER0_IN_IN_184	GPIO1_GPIO_BANK_OUT_4
	MAIN_GPIOMUX_INTROUTER0_IN_IN_185	GPIO1_GPIO_BANK_OUT_5
	MAIN_GPIOMUX_INTROUTER0_IN_IN_186	MCASP3_REC_INTR_PEND_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_187	MCASP3_XMIT_INTR_PEND_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_188	MCASP4_REC_INTR_PEND_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_189	MCASP4_XMIT_INTR_PEND_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_190	GPIO0_GPIO_BANK_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_191	GPIO0_GPIO_BANK_OUT_1
	MAIN_GPIOMUX_INTROUTER0_IN_IN_192	GPIO0_GPIO_BANK_OUT_2

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_IN_IN_193	GPIO0_GPIO_BANK_OUT_3
	MAIN_GPIOMUX_INTROUTER0_IN_IN_194	GPIO0_GPIO_BANK_OUT_4
	MAIN_GPIOMUX_INTROUTER0_IN_IN_195	GPIO0_GPIO_BANK_OUT_5
	MAIN_GPIOMUX_INTROUTER0_IN_IN_196	DMPAC0_DMPAC_LEVEL_OUT_0
	MAIN_GPIOMUX_INTROUTER0_IN_IN_197	DMPAC0_DMPAC_LEVEL_OUT_1
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_0	MCU_GPIO0_GPIO_OUT_0
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_1	MCU_GPIO0_GPIO_OUT_1
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_2	MCU_GPIO0_GPIO_OUT_2
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_3	MCU_GPIO0_GPIO_OUT_3
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_4	MCU_GPIO0_GPIO_OUT_4
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_5	MCU_GPIO0_GPIO_OUT_5
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_6	MCU_GPIO0_GPIO_OUT_6
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_7	MCU_GPIO0_GPIO_OUT_7
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_8	MCU_GPIO0_GPIO_OUT_8
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_9	MCU_GPIO0_GPIO_OUT_9
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_10	MCU_GPIO0_GPIO_OUT_10
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_11	MCU_GPIO0_GPIO_OUT_11
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_12	MCU_GPIO0_GPIO_OUT_12
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_13	MCU_GPIO0_GPIO_OUT_13
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_14	MCU_GPIO0_GPIO_OUT_14
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_15	MCU_GPIO0_GPIO_OUT_15
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_16	MCU_GPIO0_GPIO_OUT_16
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_17	MCU_GPIO0_GPIO_OUT_17
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_18	MCU_GPIO0_GPIO_OUT_18
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_19	MCU_GPIO0_GPIO_OUT_19
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_20	MCU_GPIO0_GPIO_OUT_20
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_21	MCU_GPIO0_GPIO_OUT_21
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_22	MCU_GPIO0_GPIO_OUT_22
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_23	MCU_GPIO0_GPIO_OUT_23
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_30	MCU_GPIO0_GPIO_BANK_OUT_0
	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_31	MCU_GPIO0_GPIO_BANK_OUT_1
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_IN_IN_0	TIMER0_TIMER_PWM_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_1	TIMER1_TIMER_PWM_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_IN_IN_2	TIMER2_TIMER_PWM_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_3	TIMER3_TIMER_PWM_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_4	PCIE0_PCIE_CPTS_GENF0_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_5	PCIE0_PCIE_CPTS_HW1_PUSH_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_6	PCIE0_PCIE_CPTS_SYNC_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_7	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_8	EPWM0_EPWM_SYNCO_O_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_11	WKUP_GTC0_GTC_PUSH_EVENT_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_12	PINFUNCTION_CP_GEMAC_CPTS0_HW1TSPUSHIN_CP_GEMAC_CPTS0_HW1TSPUSH_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_13	PINFUNCTION_CP_GEMAC_CPTS0_HW2TSPUSHIN_CP_GEMAC_CPTS0_HW2TSPUSH_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_16	CPSW0_CPTS_GENF0_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_17	CPSW0_CPTS_GENF1_OUT_0
	TIMESYNC_EVENT_INTROUTER0_IN_IN_18	CPSW0_CPTS_SYNC_OUT_0
CPSW0	CPSW0_CPTS_HW1_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_10
	CPSW0_CPTS_HW2_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_11
	CPSW0_CPTS_HW3_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_12
	CPSW0_CPTS_HW4_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_13
	CPSW0_CPTS_HW5_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_14
	CPSW0_CPTS_HW6_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_15
	CPSW0_CPTS_HW7_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_16
	CPSW0_CPTS_HW8_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_17
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_0	CPSW0_CPTS_COMP_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_1	PCIE0_PCIE_CPTS_COMP_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_2	DSS0_DISPC_INTR_REQ_0_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_3	DSS0_DISPC_INTR_REQ_1_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_4	DSS1_DISPC_INTR_REQ_0_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_5	DSS1_DISPC_INTR_REQ_1_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_7	MCRC64_0_INT_MCRC_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_8	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_9	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_1
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_10	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_2
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_11	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_3
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_12	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_4

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_13	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_5
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_14	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_6
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_15	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_7
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_16	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_24
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_17	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_25
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_18	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_26
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_19	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_27
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_20	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_28
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_21	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_29
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_22	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_30
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_23	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_31
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_24	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_22
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_25	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_23
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_26	GPMC0_GPMC_SDMAREQ_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_27	DEBUGSS0_DAVDMA_LEVEL_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_28	MCRC64_0_DMA_EVENT_OUT_0
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_29	MCRC64_0_DMA_EVENT_OUT_1
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_30	MCRC64_0_DMA_EVENT_OUT_2
	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_31	MCRC64_0_DMA_EVENT_OUT_3
WKUP_ESM0	WKUP_ESM0_ESM_LVL_EVENT_IN_0	ESM0_ESM_INT_CFG_LVL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_1	ESM0_ESM_INT_HI_LVL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_2	ESM0_ESM_INT_LOW_LVL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_3	WKUP_CBASS0_DEFAULT_EXP_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_4	MCU_R5FSS0_CPU0_ECC_AGGR_0_CPU0_ECC_CORRECTED_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_5	MCU_R5FSS0_CPU0_ECC_AGGR_0_CPU0_ECC_UNCORRECTED_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_8	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_9	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_10	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_11	WKUP_VTM0_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_12	WKUP_VTM0_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_13	HFOSC0_CLKLOSS_DETECT_CIRCUIT_REF_CLK_LOSS_DETECT_OUT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_14	MCU_ECC_AGGR0_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_15	MCU_ECC_AGGR0_UNCORR_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_ESM0	WKUP_ESM0_ESM_LVL_EVENT_IN_16	MCU_MCAN0_MCANSS_ECC_CORR_LVL_INT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_17	MCU_MCAN0_MCANSS_ECC_UNCORR_LVL_INT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_18	MCU_MCAN1_MCANSS_ECC_CORR_LVL_INT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_19	MCU_MCAN1_MCANSS_ECC_UNCORR_LVL_INT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_20	WKUP_ECC_AGGR2_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_21	WKUP_ECC_AGGR2_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_23	WKUP_ECC_AGGR0_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_24	WKUP_ECC_AGGR0_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_26	WKUP_TIMEOUT1_TIMED_OUT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_26	WKUP_TIMEOUT0_TIMED_OUT_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_27	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_28	MCU_R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_29	MCU_R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_30	MCU_R5FSS0_CORE0_CPU0_EXP_INTR_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_32	MCU_MSRAM_256K0_ECC_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_33	MCU_MSRAM_256K0_ECC_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_34	MCU_MSRAM_256K1_ECC_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_35	MCU_MSRAM_256K1_ECC_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_36	MCU_DCC1_INTR_ERR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_37	MCU_DCC0_INTR_ERR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_38	WKUP_ECC_AGGR1_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_39	WKUP_ECC_AGGR1_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_40	MCU_ECC_AGGR1_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_41	MCU_ECC_AGGR1_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_42	WKUP_PSRAMECC_8K0_ECC_CORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_43	WKUP_PSRAMECC_8K0_ECC_UNCORR_LEVEL_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_51	PLLFRACF2_SSMOD_16FFT18_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_52	PLLFRACF2_SSMOD_16FFT16_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_53	PLLFRACF2_SSMOD_16FFT6_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_54	PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_55	PLLFRACF2_SSMOD_16FFT1_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_56	PLLFRACF2_SSMOD_16FFT2_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_57	PLLFRACF2_SSMOD_16FFT8_LOCKLOSS_IPCFG_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_ESM0	WKUP_ESM0_ESM_LVL_EVENT_IN_58	PLLFRACF2_SSMOD_16FFT12_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_59	PLLFRACF2_SSMOD_16FFT15_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_60	PLLFRACF2_SSMOD_16FFT5_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_61	PLLFRACF2_SSMOD_16FFT7_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_62	PLLFRACF2_SSMOD_16FFT17_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_LVL_EVENT_IN_63	MCU_PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_64	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_65	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT0_IN_66	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT0_IN_69	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT0_IN_70	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT0_IN_71	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_72	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT0_IN_73	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT0_IN_76	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT0_IN_77	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT0_IN_78	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_79	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT0_IN_80	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT0_IN_81	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT0_IN_82	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT0_IN_85	MCU_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_86	WKUP_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT0_IN_88	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8
	WKUP_ESM0_ESM_PLS_EVENT0_IN_89	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9
	WKUP_ESM0_ESM_PLS_EVENT0_IN_90	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10
	WKUP_ESM0_ESM_PLS_EVENT0_IN_91	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11
	WKUP_ESM0_ESM_PLS_EVENT1_IN_64	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT1_IN_65	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT1_IN_66	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT1_IN_69	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT1_IN_70	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT1_IN_71	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_ESM0	WKUP_ESM0_ESM_PLS_EVENT1_IN_72	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT1_IN_73	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT1_IN_76	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT1_IN_77	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT1_IN_78	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT1_IN_79	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT1_IN_80	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT1_IN_81	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT1_IN_82	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT1_IN_85	MCU_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT1_IN_86	WKUP_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT1_IN_88	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8
	WKUP_ESM0_ESM_PLS_EVENT1_IN_89	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9
	WKUP_ESM0_ESM_PLS_EVENT1_IN_90	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10
	WKUP_ESM0_ESM_PLS_EVENT1_IN_91	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11
	WKUP_ESM0_ESM_PLS_EVENT2_IN_64	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT2_IN_65	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT2_IN_66	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT2_IN_69	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT2_IN_70	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT2_IN_71	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT2_IN_72	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT2_IN_73	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT2_IN_76	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT2_IN_77	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT2_IN_78	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT2_IN_79	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1
	WKUP_ESM0_ESM_PLS_EVENT2_IN_80	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2
	WKUP_ESM0_ESM_PLS_EVENT2_IN_81	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3
	WKUP_ESM0_ESM_PLS_EVENT2_IN_82	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4
	WKUP_ESM0_ESM_PLS_EVENT2_IN_85	MCU_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT2_IN_86	WKUP_RTI0_INTR_WWD_OUT_0
	WKUP_ESM0_ESM_PLS_EVENT2_IN_88	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_ESM0	WKUP_ESM0_ESM_PLS_EVENT2_IN_89	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9
	WKUP_ESM0_ESM_PLS_EVENT2_IN_90	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10
	WKUP_ESM0_ESM_PLS_EVENT2_IN_91	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11
ESM0	ESM0_ESM_LVL_EVENT_IN_0	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_1	ECC_AGGR0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_2	ECC_AGGR0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_3	CPSW0_ECC_SEC_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_5	MSRAM8KX256E0_ECC_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_6	MSRAM8KX256E0_ECC_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_7	PLLFRACF2_SSMOD_16FFT17_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_8	WKUP_PSRAMECC_8K0_ECC_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_9	DMASS0_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_10	DMASS0_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_11	FSS0_OSPI_0_OSPI_ECC_CORR_LVL_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_12	GICSS0_ECC_AGGR_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_13	WKUP_PSRAMECC_8K0_ECC_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_15	PDMA0_ECC_SEC_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_16	MCAN0_MCANSS_ECC_CORR_LVL_INT_OUT_0
	ESM0_ESM_LVL_EVENT_IN_17	PLLFRACF2_SSMOD_16FFT5_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_18	PLLFRACF2_SSMOD_16FFT7_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_19	PLLFRACF2_SSMOD_16FFT6_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_20	WKUP_ECC_AGGR0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_21	WKUP_ECC_AGGR0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_22	PSC0_ECC_AGGR_0_FW_CH_BR_ECC_AGGR_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_23	PSC0_ECC_AGGR_0_FW_CH_BR_ECC_AGGR_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_24	A53SS0_ECC_ECCAGGR0_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_25	A53SS0_ECC_ECCAGGR1_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_26	A53SS0_ECC_ECCAGGR_COREPAC_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_27	PLLFRACF2_SSMOD_16FFT16_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_28	PDMA1_ECC_SEC_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_29	PSRAMECC0_ECC_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_30	WKUP_R5FSS0_CORE0_ECC_CORRECTED_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_31	CSI_RX_IF2_CSI_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_32	USB0_HOST_SYSTEM_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_33	USB1_HOST_SYSTEM_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_34	MMCSD2_EMMCSDSS_RXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_35	USB0_A_ECC_AGGR_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_36	MMCSD2_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_37	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_38	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_39	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_40	WKUP_R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0
	ESM0_ESM_LVL_EVENT_IN_41	CSI_RX_IF3_CSI_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_42	WKUP_R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0
	ESM0_ESM_LVL_EVENT_IN_43	PLLFRACF2_SSMOD_16FFT18_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_44	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_45	A53SS0_ECC_ECCAGGR2_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_46	A53SS0_ECC_ECCAGGR2_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_47	A53SS0_ECC_ECCAGGR3_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_48	A53SS0_ECC_ECCAGGR3_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_49	MMCSD2_EMMCSDSS_TXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_54	MMCSD0_EMMCSS_RXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_55	MMCSD0_EMMCSS_RXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_56	MMCSD0_EMMCSS_TXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_57	MMCSD0_EMMCSS_TXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_58	MMCSD1_EMMCSDSS_RXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_59	MMCSD1_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_60	MMCSD1_EMMCSDSS_TXMEM_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_61	MMCSD1_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_62	DMASS1_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_63	DMASS1_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_64	C7X256V1_CLEC_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_65	MMCSD2_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_66	CSI_RX_IF0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_67	CPSW0_ECC_DED_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_68	MCAN1_MCANSS_ECC_CORR_LVL_INT_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_69	MCAN1_MCANSS_ECC_UNCORR_LVL_INT_OUT_0
	ESM0_ESM_LVL_EVENT_IN_70	CSI_RX_IF0_CSI_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_71	CSI_RX_IF0_CSI_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_72	CSI_RX_IF0_CSI_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_73	DCC7_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_74	FSS0_OSPI_0_OSPI_ECC_UNCORR_LVL_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_75	GICSS0_ECC_AGGR_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_76	PSC0_FW_0_FW_CH_BR_DEFAULT_EXP_OUT_0
	ESM0_ESM_LVL_EVENT_IN_77	CSI_RX_IF0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_78	MCAN0_MCANSS_ECC_UNCORR_LVL_INT_OUT_0
	ESM0_ESM_LVL_EVENT_IN_79	DCC6_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_80	CBASS_CENTRAL2_DEFAULT_EXP_OUT_0
	ESM0_ESM_LVL_EVENT_IN_86	CBASS0_DEFAULT_EXP_OUT_0
	ESM0_ESM_LVL_EVENT_IN_88	PDMA0_ECC_DED_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_89	PDMA1_ECC_DED_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_90	PSRAMECC0_ECC_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_91	WKUP_R5FSS0_CORE0_ECC_UNCORRECTED_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_93	A53SS0_ECC_ECCAGGR1_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_94	A53SS0_ECC_ECCAGGR0_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_95	A53SS0_ECC_ECCAGGR_COREPAC_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_98	DFTSS0_DFT_SAFETY_123_OUT_0
	ESM0_ESM_LVL_EVENT_IN_99	DFTSS0_DFT_SAFETY_MULTI_OUT_0
	ESM0_ESM_LVL_EVENT_IN_100	DFTSS0_DFT_SAFETY_ONE_OUT_0
	ESM0_ESM_LVL_EVENT_IN_101	MCU_MCU_16FF0_VDD_CORE_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_102	MCU_MCU_16FF0_VDD_CORE_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_104	WKUP_ECC_AGGR1_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_105	WKUP_ECC_AGGR1_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_106	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_108	PSRAMECC1_ECC_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_109	PSRAMECC1_ECC_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_110	CBASS_IPCSS0_DEFAULT_EXP_OUT_0
	ESM0_ESM_LVL_EVENT_IN_111	USB0_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_112	DCC0_INTR_ERR_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_113	DCC1_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_114	DCC2_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_115	DCC3_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_116	DCC4_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_117	DCC5_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_118	SA3_SS0_DMSS_ECCAGGR_0_DMSS_ECC_DED_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_119	SA3_SS0_DMSS_ECCAGGR_0_DMSS_ECC_SEC_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_120	SA3_SS0_SA_UL_0_SA_UL_ECC_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_121	SA3_SS0_SA_UL_0_SA_UL_ECC_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_122	CSI_RX_IF1_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_123	CSI_TX_IF0_CDNS_RAM_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_124	WKUP_R5FSS0_CORE0_EXP_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_125	CSI_TX_IF0_CDNS_RAM_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_126	CSI_TX_IF0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_127	CSI_TX_IF0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_128	PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_129	PLLFRACF2_SSMOD_16FFT1_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_130	PLLFRACF2_SSMOD_16FFT2_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_131	PLLFRACF2_SSMOD_16FFT8_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_132	PLLFRACF2_SSMOD_16FFT12_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_133	PLLFRACF2_SSMOD_16FFT15_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_134	MCU_PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0
	ESM0_ESM_LVL_EVENT_IN_135	HFOSC0_CLKLOSS_DETECT_CIRCUIT_REF_CLK_LOSS_DETECT_OUT_OUT_0
	ESM0_ESM_LVL_EVENT_IN_136	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_137	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_138	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_139	WKUP_VTM0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_140	WKUP_VTM0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_141	FSS0_FSAS_0_ECC_INTR_ERR_PEND_OUT_0
	ESM0_ESM_LVL_EVENT_IN_142	DSS_DSI0_ECC_INTR_UNCORR_LEVEL_SYS_OUT_0
	ESM0_ESM_LVL_EVENT_IN_143	USB1_ASF_INT_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_144	A53SS0_EXTERRIRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_145	A53SS0_INTERRIRQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_146	USB1_A_ECC_AGGR_CORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_147	USB1_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_148	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_149	C7X256V0_CLEC_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_152	MCU_PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_155	USB1_ASF_INT_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_156	GPU0_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_157	PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_158	WKUP_PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_159	WKUP_PBIST1_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_160	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_161	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_1
	ESM0_ESM_LVL_EVENT_IN_162	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_2
	ESM0_ESM_LVL_EVENT_IN_163	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_3
	ESM0_ESM_LVL_EVENT_IN_164	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_4
	ESM0_ESM_LVL_EVENT_IN_165	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_5
	ESM0_ESM_LVL_EVENT_IN_166	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_6
	ESM0_ESM_LVL_EVENT_IN_167	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_7
	ESM0_ESM_LVL_EVENT_IN_168	VPAC0_ECC_INTR0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_169	VPAC0_ECC_INTR0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_170	VPAC0_ECC_INTR1_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_171	VPAC0_ECC_INTR1_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_172	VPAC0_ECC_INTR3_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_173	VPAC0_ECC_INTR3_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_174	DDR32SS0_DDRSS_DRAM_ECC_CORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_175	DDR32SS0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_176	DDR32SS0_DDRSS_V2A_OTHER_ERR_LVL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_177	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_178	DSS_DSI0_DSI_0_SAFETY_ERROR_FATAL_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_179	DSS_DSI0_DSI_0_SAFETY_ERROR_NONFATAL_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_180	PCIE0_PCIE_ECC0_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_181	PCIE0_PCIE_ECC0_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_182	PCIE0_PCIE_ECC1_UNCORR_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_183	WKUP_CBASS0_DEFAULT_EXP_OUT_0
	ESM0_ESM_LVL_EVENT_IN_184	GPU0_GPU_SAFETY_IRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_185	CSI_TX_IF0_CSI_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_186	CSI_TX_IF0_CSI_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_187	CSI_RX_IF1_CSI_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_188	CSI_RX_IF1_CSI_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_189	CSI_RX_IF1_CSI_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_190	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_191	CSI_RX_IF1_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_192	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_193	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_1
	ESM0_ESM_LVL_EVENT_IN_194	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_2
	ESM0_ESM_LVL_EVENT_IN_195	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_3
	ESM0_ESM_LVL_EVENT_IN_196	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_4
	ESM0_ESM_LVL_EVENT_IN_197	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_5
	ESM0_ESM_LVL_EVENT_IN_198	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_6
	ESM0_ESM_LVL_EVENT_IN_199	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_7
	ESM0_ESM_LVL_EVENT_IN_200	CSI_RX_IF2_CSI_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_201	CSI_RX_IF2_CSI_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_204	CSI_RX_IF3_CSI_FATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_205	CSI_RX_IF3_CSI_NONFATAL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_206	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_207	PBIST1_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_208	R5FSS0_CORE0_ECC_CORRECTED_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_209	R5FSS0_CORE0_ECC_UNCORRECTED_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_210	R5FSS0_CORE0_EXP_INTR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_211	R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0
	ESM0_ESM_LVL_EVENT_IN_212	R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0
	ESM0_ESM_LVL_EVENT_IN_213	PBIST2_DFT_PBIST_SAFETY_ERROR_OUT_0
	ESM0_ESM_LVL_EVENT_IN_214	CSI_RX_IF2_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_215	CSI_RX_IF2_UNCORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_216	CSI_RX_IF3_CORR_LEVEL_OUT_0
	ESM0_ESM_LVL_EVENT_IN_217	CSI_RX_IF3_UNCORR_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_LVL_EVENT_IN_222	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	ESM0_ESM_LVL_EVENT_IN_223	DCC8_INTR_ERR_LEVEL_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_224	RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_225	RTI1_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_226	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_227	WKUP_RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_228	PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_229	PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_230	GICSS0_AXIM_ERR_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_231	GICSS0_ECC_FATAL_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_232	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_233	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_234	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_235	MCU_PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_236	RTI4_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_237	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_238	GPU0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_239	RTI5_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_240	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_241	RTI2_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_242	RTI3_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_243	PBIST2_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_244	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_245	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_248	RTI15_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_249	RTI8_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_250	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0
	ESM0_ESM_PLS_EVENT0_IN_251	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_224	RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_225	RTI1_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_226	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_227	WKUP_RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_228	PBIST0_DFT_PBIST_CPU_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_PLS_EVENT1_IN_229	PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_230	GICSS0_AXIM_ERR_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_231	GICSS0_ECC_FATAL_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_232	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_233	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_234	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_235	MCU_PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_236	RTI4_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_237	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_238	GPU0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_239	RTI5_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_240	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_241	RTI2_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_242	RTI3_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_243	PBIST2_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_244	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_245	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_248	RTI15_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_249	RTI8_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_250	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0
	ESM0_ESM_PLS_EVENT1_IN_251	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_224	RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_225	RTI1_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_226	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_227	WKUP_RTI0_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_228	PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_229	PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_230	GICSS0_AXIM_ERR_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_231	GICSS0_ECC_FATAL_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_232	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_233	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_234	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_235	MCU_PBIST0_DFT_PBIST_CPU_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
ESM0	ESM0_ESM_PLS_EVENT2_IN_236	RTI4_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_237	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_238	GPU0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_239	RTI5_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_240	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_241	RTI2_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_242	RTI3_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_243	PBIST2_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_244	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_245	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_248	RTI15_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_249	RTI8_INTR_WWD_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_250	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0
	ESM0_ESM_PLS_EVENT2_IN_251	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0
GICSS0	GICSS0_PPI0_0_IN_18	A53SS0_CTIIRQ1_OUT_0
	GICSS0_PPI0_0_IN_19	A53SS0_CTIIRQ2_OUT_0
	GICSS0_PPI0_0_IN_20	A53SS0_CTIIRQ3_OUT_0
	GICSS0_PPI0_0_IN_22	A53SS0_COMMIRQ0_OUT_0
	GICSS0_PPI0_0_IN_23	A53SS0_PMUIRQ0_OUT_0
	GICSS0_PPI0_0_IN_24	A53SS0_CTIIRQ0_OUT_0
	GICSS0_PPI0_0_IN_25	A53SS0_VCPUMNTIRQ0_OUT_0
	GICSS0_PPI0_0_IN_26	A53SS0_CNTHPIRQ0_OUT_0
	GICSS0_PPI0_0_IN_27	A53SS0_CNTVIRQ0_OUT_0
	GICSS0_PPI0_0_IN_29	A53SS0_CNTPSIRQ0_OUT_0
	GICSS0_PPI0_0_IN_30	A53SS0_CNTPNSIRQ0_OUT_0
	GICSS0_PPI0_1_IN_17	A53SS0_CTIIRQ0_OUT_0
	GICSS0_PPI0_1_IN_19	A53SS0_CTIIRQ2_OUT_0
	GICSS0_PPI0_1_IN_20	A53SS0_CTIIRQ3_OUT_0
	GICSS0_PPI0_1_IN_22	A53SS0_COMMIRQ1_OUT_0
	GICSS0_PPI0_1_IN_23	A53SS0_PMUIRQ1_OUT_0
	GICSS0_PPI0_1_IN_24	A53SS0_CTIIRQ1_OUT_0
	GICSS0_PPI0_1_IN_25	A53SS0_VCPUMNTIRQ1_OUT_0
	GICSS0_PPI0_1_IN_26	A53SS0_CNTHPIRQ1_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_PPI0_1_IN_27	A53SS0_CNTVIRQ1_OUT_0
	GICSS0_PPI0_1_IN_29	A53SS0_CNTPSIRQ1_OUT_0
	GICSS0_PPI0_1_IN_30	A53SS0_CNTPNSIRQ1_OUT_0
	GICSS0_PPI0_2_IN_17	A53SS0_CTIIRQ0_OUT_0
	GICSS0_PPI0_2_IN_18	A53SS0_CTIIRQ1_OUT_0
	GICSS0_PPI0_2_IN_20	A53SS0_CTIIRQ3_OUT_0
	GICSS0_PPI0_2_IN_22	A53SS0_COMMIRQ2_OUT_0
	GICSS0_PPI0_2_IN_23	A53SS0_PMUIRQ2_OUT_0
	GICSS0_PPI0_2_IN_24	A53SS0_CTIIRQ2_OUT_0
	GICSS0_PPI0_2_IN_25	A53SS0_VCPUMNTIRQ2_OUT_0
	GICSS0_PPI0_2_IN_26	A53SS0_CNTHPIRQ2_OUT_0
	GICSS0_PPI0_2_IN_27	A53SS0_CNTVIRQ2_OUT_0
	GICSS0_PPI0_2_IN_29	A53SS0_CNTPSIRQ2_OUT_0
	GICSS0_PPI0_2_IN_30	A53SS0_CNTPNSIRQ2_OUT_0
	GICSS0_PPI0_3_IN_17	A53SS0_CTIIRQ0_OUT_0
	GICSS0_PPI0_3_IN_18	A53SS0_CTIIRQ1_OUT_0
	GICSS0_PPI0_3_IN_19	A53SS0_CTIIRQ2_OUT_0
	GICSS0_PPI0_3_IN_22	A53SS0_COMMIRQ3_OUT_0
	GICSS0_PPI0_3_IN_23	A53SS0_PMUIRQ3_OUT_0
	GICSS0_PPI0_3_IN_24	A53SS0_CTIIRQ3_OUT_0
	GICSS0_PPI0_3_IN_25	A53SS0_VCPUMNTIRQ3_OUT_0
	GICSS0_PPI0_3_IN_26	A53SS0_CNTHPIRQ3_OUT_0
	GICSS0_PPI0_3_IN_27	A53SS0_CNTVIRQ3_OUT_0
	GICSS0_PPI0_3_IN_29	A53SS0_CNTPSIRQ3_OUT_0
	GICSS0_PPI0_3_IN_30	A53SS0_CNTPNSIRQ3_OUT_0
	GICSS0_SPI_IN_32	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0
	GICSS0_SPI_IN_33	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1
	GICSS0_SPI_IN_34	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2
	GICSS0_SPI_IN_35	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3
	GICSS0_SPI_IN_36	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4
	GICSS0_SPI_IN_37	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5
	GICSS0_SPI_IN_38	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6
	GICSS0_SPI_IN_39	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_40	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8
	GICSS0_SPI_IN_41	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9
	GICSS0_SPI_IN_42	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10
	GICSS0_SPI_IN_43	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11
	GICSS0_SPI_IN_44	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12
	GICSS0_SPI_IN_45	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13
	GICSS0_SPI_IN_46	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14
	GICSS0_SPI_IN_47	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15
	GICSS0_SPI_IN_48	CPSW0_CPTS_COMP_OUT_0
	GICSS0_SPI_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	GICSS0_SPI_IN_50	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_0
	GICSS0_SPI_IN_51	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_1
	GICSS0_SPI_IN_52	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_2
	GICSS0_SPI_IN_53	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_3
	GICSS0_SPI_IN_54	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_4
	GICSS0_SPI_IN_55	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_5
	GICSS0_SPI_IN_56	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_6
	GICSS0_SPI_IN_57	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_7
	GICSS0_SPI_IN_58	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	GICSS0_SPI_IN_59	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	GICSS0_SPI_IN_60	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	GICSS0_SPI_IN_61	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	GICSS0_SPI_IN_62	DSS1_DISPC_INTR_REQ_0_OUT_0
	GICSS0_SPI_IN_63	DSS1_DISPC_INTR_REQ_1_OUT_0
	GICSS0_SPI_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	GICSS0_SPI_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1
	GICSS0_SPI_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	GICSS0_SPI_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	GICSS0_SPI_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	GICSS0_SPI_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	GICSS0_SPI_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	GICSS0_SPI_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	GICSS0_SPI_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9
	GICSS0_SPI_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10
	GICSS0_SPI_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11
	GICSS0_SPI_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12
	GICSS0_SPI_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13
	GICSS0_SPI_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14
	GICSS0_SPI_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15
	GICSS0_SPI_IN_80	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16
	GICSS0_SPI_IN_81	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17
	GICSS0_SPI_IN_82	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18
	GICSS0_SPI_IN_83	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19
	GICSS0_SPI_IN_84	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20
	GICSS0_SPI_IN_85	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21
	GICSS0_SPI_IN_86	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22
	GICSS0_SPI_IN_87	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23
	GICSS0_SPI_IN_88	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24
	GICSS0_SPI_IN_89	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25
	GICSS0_SPI_IN_90	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26
	GICSS0_SPI_IN_91	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27
	GICSS0_SPI_IN_92	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28
	GICSS0_SPI_IN_93	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29
	GICSS0_SPI_IN_94	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30
	GICSS0_SPI_IN_95	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31
	GICSS0_SPI_IN_96	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32
	GICSS0_SPI_IN_97	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33
	GICSS0_SPI_IN_98	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34
	GICSS0_SPI_IN_99	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35
	GICSS0_SPI_IN_100	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36
	GICSS0_SPI_IN_101	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37
	GICSS0_SPI_IN_102	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38
	GICSS0_SPI_IN_103	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39
	GICSS0_SPI_IN_104	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0
	GICSS0_SPI_IN_105	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_106	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2
	GICSS0_SPI_IN_107	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3
	GICSS0_SPI_IN_108	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0
	GICSS0_SPI_IN_109	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0
	GICSS0_SPI_IN_112	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4
	GICSS0_SPI_IN_113	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5
	GICSS0_SPI_IN_114	MMCSD2_EMMCSDSS_INTR_OUT_0
	GICSS0_SPI_IN_115	MMCSD1_EMMCSDSS_INTR_OUT_0
	GICSS0_SPI_IN_116	DSS0_DISPC_INTR_REQ_0_OUT_0
	GICSS0_SPI_IN_117	DSS0_DISPC_INTR_REQ_1_OUT_0
	GICSS0_SPI_IN_118	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	GICSS0_SPI_IN_119	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0
	GICSS0_SPI_IN_120	GPU0_GPU_PWRCTRL_REQ_OUT_0
	GICSS0_SPI_IN_121	PCIE0_PCIE_CPTS_PEND_OUT_0
	GICSS0_SPI_IN_122	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	GICSS0_SPI_IN_123	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	GICSS0_SPI_IN_124	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0
	GICSS0_SPI_IN_125	PCIE0_PCIE_ERROR_PULSE_OUT_0
	GICSS0_SPI_IN_126	PCIE0_PCIE_FLR_PULSE_OUT_0
	GICSS0_SPI_IN_127	PCIE0_PCIE_LEGACY_PULSE_OUT_0
	GICSS0_SPI_IN_128	DCC0_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC1_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC2_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC3_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC4_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC5_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC6_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC7_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_128	DCC8_INTR_DONE_LEVEL_OUT_0
	GICSS0_SPI_IN_129	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	GICSS0_SPI_IN_129	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	GICSS0_SPI_IN_129	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	GICSS0_SPI_IN_129	CTRL_MMR0_ACCESS_ERR_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_129	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	GICSS0_SPI_IN_130	JPGENC0_IRQ_OUT_0
	GICSS0_SPI_IN_131	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0
	GICSS0_SPI_IN_132	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	GICSS0_SPI_IN_133	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_133	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	GICSS0_SPI_IN_134	CPSW0_EVNT_PEND_OUT_0
	GICSS0_SPI_IN_135	CPSW0_MDIO_PEND_OUT_0
	GICSS0_SPI_IN_136	CPSW0_STAT_PEND_OUT_0
	GICSS0_SPI_IN_137	PCIE0_PCIE_LOCAL_LEVEL_OUT_0
	GICSS0_SPI_IN_138	GPMC0_GPMC_SINTERRUPT_OUT_0
	GICSS0_SPI_IN_139	MCU_I2C0_POINTRPEND_OUT_0
	GICSS0_SPI_IN_140	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0
	GICSS0_SPI_IN_141	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0
	GICSS0_SPI_IN_142	PCIE0_PCIE_PHY_LEVEL_OUT_0
	GICSS0_SPI_IN_145	ECAP0_ECAP_INT_OUT_0
	GICSS0_SPI_IN_146	ECAP1_ECAP_INT_OUT_0
	GICSS0_SPI_IN_147	ECAP2_ECAP_INT_OUT_0
	GICSS0_SPI_IN_148	EQEP0_EQEP_INT_OUT_0
	GICSS0_SPI_IN_149	EQEP1_EQEP_INT_OUT_0
	GICSS0_SPI_IN_150	EQEP2_EQEP_INT_OUT_0
	GICSS0_SPI_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_152	TIMER0_INTR_PEND_OUT_0
	GICSS0_SPI_IN_153	TIMER1_INTR_PEND_OUT_0
	GICSS0_SPI_IN_154	TIMER2_INTR_PEND_OUT_0
	GICSS0_SPI_IN_155	TIMER3_INTR_PEND_OUT_0
	GICSS0_SPI_IN_156	TIMER4_INTR_PEND_OUT_0
	GICSS0_SPI_IN_157	TIMER5_INTR_PEND_OUT_0
	GICSS0_SPI_IN_158	TIMER6_INTR_PEND_OUT_0
	GICSS0_SPI_IN_159	TIMER7_INTR_PEND_OUT_0
	GICSS0_SPI_IN_160	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0
	GICSS0_SPI_IN_161	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	GICSS0_SPI_IN_162	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0
	GICSS0_SPI_IN_163	PCIE0_PCIE_DPA_PULSE_OUT_0
	GICSS0_SPI_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	GICSS0_SPI_IN_165	MMCSD0_EMMCSS_INTR_OUT_0
	GICSS0_SPI_IN_166	MCRC64_0_INT_MCRC_OUT_0
	GICSS0_SPI_IN_167	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	GICSS0_SPI_IN_168	VPAC0_VPAC_LEVEL_OUT_0
	GICSS0_SPI_IN_169	VPAC0_VPAC_LEVEL_OUT_1
	GICSS0_SPI_IN_170	VPAC0_VPAC_LEVEL_OUT_2
	GICSS0_SPI_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	GICSS0_SPI_IN_172	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	GICSS0_SPI_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	GICSS0_SPI_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	GICSS0_SPI_IN_175	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	GICSS0_SPI_IN_177	DDPA0_DDPA_INTR_OUT_0
	GICSS0_SPI_IN_178	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	GICSS0_SPI_IN_179	CSI_RX_IF1_CSI_IRQ_OUT_0
	GICSS0_SPI_IN_180	ESM0_ESM_INT_CFG_LVL_OUT_0
	GICSS0_SPI_IN_181	ESM0_ESM_INT_HI_LVL_OUT_0
	GICSS0_SPI_IN_182	ESM0_ESM_INT_LOW_LVL_OUT_0
	GICSS0_SPI_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	GICSS0_SPI_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	GICSS0_SPI_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	GICSS0_SPI_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	GICSS0_SPI_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	GICSS0_SPI_IN_189	VPAC0_VPAC_LEVEL_OUT_3
	GICSS0_SPI_IN_190	VPAC0_VPAC_LEVEL_OUT_4
	GICSS0_SPI_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	GICSS0_SPI_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	GICSS0_SPI_IN_193	I2C0_POINTRPEND_OUT_0
	GICSS0_SPI_IN_194	I2C1_POINTRPEND_OUT_0
	GICSS0_SPI_IN_195	I2C2_POINTRPEND_OUT_0
	GICSS0_SPI_IN_196	I2C3_POINTRPEND_OUT_0
	GICSS0_SPI_IN_197	WKUP_I2C0_POINTRPEND_OUT_0
	GICSS0_SPI_IN_199	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	GICSS0_SPI_IN_200	CSI_RX_IF1_CSI_LEVEL_OUT_0
	GICSS0_SPI_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	GICSS0_SPI_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0
	GICSS0_SPI_IN_203	PSC0_PSC_ALLINT_OUT_0
	GICSS0_SPI_IN_204	MCSPI0_INTR_SPI_OUT_0
	GICSS0_SPI_IN_205	MCSPI1_INTR_SPI_OUT_0
	GICSS0_SPI_IN_206	MCSPI2_INTR_SPI_OUT_0
	GICSS0_SPI_IN_207	RTI15_INTR_WWD_OUT_0
	GICSS0_SPI_IN_208	MCU_MCSPI0_INTR_SPI_OUT_0
	GICSS0_SPI_IN_209	MCU_MCSPI1_INTR_SPI_OUT_0
	GICSS0_SPI_IN_210	UART0_USART_IRQ_OUT_0
	GICSS0_SPI_IN_211	UART1_USART_IRQ_OUT_0
	GICSS0_SPI_IN_212	UART2_USART_IRQ_OUT_0
	GICSS0_SPI_IN_213	UART3_USART_IRQ_OUT_0
	GICSS0_SPI_IN_214	UART4_USART_IRQ_OUT_0
	GICSS0_SPI_IN_215	UART5_USART_IRQ_OUT_0
	GICSS0_SPI_IN_216	UART6_USART_IRQ_OUT_0
	GICSS0_SPI_IN_217	MCU_UART0_USART_IRQ_OUT_0
	GICSS0_SPI_IN_218	WKUP_UART0_USART_IRQ_OUT_0
	GICSS0_SPI_IN_219	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_220	USB0_IRQ_OUT_0
	GICSS0_SPI_IN_221	USB0_IRQ_OUT_1
	GICSS0_SPI_IN_222	USB0_IRQ_OUT_2
	GICSS0_SPI_IN_223	USB0_IRQ_OUT_3
	GICSS0_SPI_IN_224	USB0_IRQ_OUT_4
	GICSS0_SPI_IN_225	USB0_IRQ_OUT_5
	GICSS0_SPI_IN_226	USB0_IRQ_OUT_6
	GICSS0_SPI_IN_227	USB0_IRQ_OUT_7
	GICSS0_SPI_IN_228	USB0_MISC_LEVEL_OUT_0
	GICSS0_SPI_IN_229	EPWM0_EPWM_ETINT_OUT_0
	GICSS0_SPI_IN_230	EPWM0_EPWM_TRIPZINT_OUT_0
	GICSS0_SPI_IN_231	EPWM1_EPWM_ETINT_OUT_0
	GICSS0_SPI_IN_232	CSI_RX_IF2_CSI_IRQ_OUT_0
	GICSS0_SPI_IN_233	EPWM1_EPWM_TRIPZINT_OUT_0
	GICSS0_SPI_IN_234	EPWM2_EPWM_ETINT_OUT_0
	GICSS0_SPI_IN_235	EPWM2_EPWM_TRIPZINT_OUT_0
	GICSS0_SPI_IN_236	CSI_RX_IF2_CSI_LEVEL_OUT_0
	GICSS0_SPI_IN_237	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	GICSS0_SPI_IN_238	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1
	GICSS0_SPI_IN_239	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	GICSS0_SPI_IN_240	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	GICSS0_SPI_IN_241	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	GICSS0_SPI_IN_242	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	GICSS0_SPI_IN_243	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	GICSS0_SPI_IN_244	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	GICSS0_SPI_IN_245	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	GICSS0_SPI_IN_246	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	GICSS0_SPI_IN_249	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	GICSS0_SPI_IN_250	CSI_RX_IF3_CSI_IRQ_OUT_0
	GICSS0_SPI_IN_251	CSI_RX_IF3_CSI_LEVEL_OUT_0
	GICSS0_SPI_IN_252	RTI0_INTR_WWD_OUT_0
	GICSS0_SPI_IN_253	RTI1_INTR_WWD_OUT_0
	GICSS0_SPI_IN_254	RTI2_INTR_WWD_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
GICSS0	GICSS0_SPI_IN_255	RTI3_INTR_WWD_OUT_0
	GICSS0_SPI_IN_256	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	GICSS0_SPI_IN_257	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	GICSS0_SPI_IN_258	USB1_IRQ_OUT_0
	GICSS0_SPI_IN_259	USB1_IRQ_OUT_1
	GICSS0_SPI_IN_260	USB1_IRQ_OUT_2
	GICSS0_SPI_IN_261	USB1_IRQ_OUT_3
	GICSS0_SPI_IN_262	USB1_IRQ_OUT_4
	GICSS0_SPI_IN_263	USB1_IRQ_OUT_5
	GICSS0_SPI_IN_264	USB1_IRQ_OUT_6
	GICSS0_SPI_IN_265	USB1_IRQ_OUT_7
	GICSS0_SPI_IN_266	USB1_HOST_SYSTEM_ERROR_OUT_0
	GICSS0_SPI_IN_267	MCASP0_REC_INTR_PEND_OUT_0
	GICSS0_SPI_IN_268	MCASP0_XMIT_INTR_PEND_OUT_0
	GICSS0_SPI_IN_269	MCASP1_REC_INTR_PEND_OUT_0
	GICSS0_SPI_IN_270	MCASP1_XMIT_INTR_PEND_OUT_0
	GICSS0_SPI_IN_271	MCASP2_REC_INTR_PEND_OUT_0
	GICSS0_SPI_IN_272	MCASP2_XMIT_INTR_PEND_OUT_0
	GICSS0_SPI_IN_273	GPU0_OS_IRQ_OUT_0
	GICSS0_SPI_IN_274	GPU0_OS_IRQ_OUT_1
	GICSS0_SPI_IN_275	GPU0_OS_IRQ_OUT_2
	GICSS0_SPI_IN_276	GPU0_OS_IRQ_OUT_3
	GICSS0_SPI_IN_277	USB1_OTGIRQ_OUT_0
	GICSS0_SPI_IN_278	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	GICSS0_SPI_IN_279	CSI_TX_IF0_CSI_LEVEL_OUT_0
	GICSS0_SPI_IN_280	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_0
	GICSS0_SPI_IN_281	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_1
	GICSS0_SPI_IN_282	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_2
	GICSS0_SPI_IN_283	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_3
	GICSS0_SPI_IN_284	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_4
	GICSS0_SPI_IN_285	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_5
	GICSS0_SPI_IN_286	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_6
	GICSS0_SPI_IN_287	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_7

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
EPWM0	EPWM0_EPWM_SYNCIN_IN_0	PINFUNCTION_EHRPWM0_SYNCIIN_EHRPWM0_SYNCI_OUT_0
	EPWM0_EPWM_SYNCIN_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_19
	EPWM0_EPWM_SYNCIN_IN_0	CPSW0_CPTS_COMP_OUT_0
	EPWM0_EPWM_SYNCIN_IN_0	PCIE0_PCIE_CPTS_COMP_OUT_0
EPWM1	EPWM1_EPWM_SYNCIN_IN_0	EPWM0_EPWM_SYNCOUT_OUT_0
EPWM2	EPWM2_EPWM_SYNCIN_IN_0	EPWM1_EPWM_SYNCOUT_OUT_0
PCIE0	PCIE0_PCIE_CPTS_HW2_PUSH_IN_0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_8
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_4	R5FSS0_CORE0_EXP_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_5	DMPAC0_DMPAC_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_6	DMPAC0_DMPAC_LEVEL_OUT_1
	R5FSS0_CORE0_INTR_IN_7	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_7
	R5FSS0_CORE0_INTR_IN_8	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32
	R5FSS0_CORE0_INTR_IN_9	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33
	R5FSS0_CORE0_INTR_IN_10	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34
	R5FSS0_CORE0_INTR_IN_11	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35
	R5FSS0_CORE0_INTR_IN_12	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36
	R5FSS0_CORE0_INTR_IN_13	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37
	R5FSS0_CORE0_INTR_IN_14	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38
	R5FSS0_CORE0_INTR_IN_15	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39
	R5FSS0_CORE0_INTR_IN_16	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0
	R5FSS0_CORE0_INTR_IN_17	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	R5FSS0_CORE0_INTR_IN_24	TIMER0_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_25	TIMER1_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_26	TIMER2_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_27	TIMER3_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_28	TIMER4_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_29	TIMER5_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_30	RTI8_INTR_WWD_OUT_0
	R5FSS0_CORE0_INTR_IN_31	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_31	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_32	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_12
	R5FSS0_CORE0_INTR_IN_33	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_13
	R5FSS0_CORE0_INTR_IN_34	TIMER6_INTR_PEND_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_35	TIMER7_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_36	EPWM0_EPWM_ETINT_OUT_0
	R5FSS0_CORE0_INTR_IN_37	EPWM1_EPWM_ETINT_OUT_0
	R5FSS0_CORE0_INTR_IN_38	EPWM2_EPWM_ETINT_OUT_0
	R5FSS0_CORE0_INTR_IN_39	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_39	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_39	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_39	CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_39	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_40	DSS0_DISPC_INTR_REQ_0_OUT_0
	R5FSS0_CORE0_INTR_IN_41	DSS0_DISPC_INTR_REQ_1_OUT_0
	R5FSS0_CORE0_INTR_IN_42	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_43	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_44	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	R5FSS0_CORE0_INTR_IN_45	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_46	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_47	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	R5FSS0_CORE0_INTR_IN_48	CPSW0_CPTS_COMP_OUT_0
	R5FSS0_CORE0_INTR_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	R5FSS0_CORE0_INTR_IN_50	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	R5FSS0_CORE0_INTR_IN_51	CSI_TX_IF0_CSI_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_52	USB1_OTGIRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_53	PCIE0_PCIE_CPTS_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_54	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	R5FSS0_CORE0_INTR_IN_55	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	R5FSS0_CORE0_INTR_IN_56	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_20
	R5FSS0_CORE0_INTR_IN_57	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_21
	R5FSS0_CORE0_INTR_IN_58	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_32
	R5FSS0_CORE0_INTR_IN_59	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_33
	R5FSS0_CORE0_INTR_IN_60	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_14
	R5FSS0_CORE0_INTR_IN_61	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_15
	R5FSS0_CORE0_INTR_IN_63	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	R5FSS0_CORE0_INTR_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_152

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_153
	R5FSS0_CORE0_INTR_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_154
	R5FSS0_CORE0_INTR_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_155
	R5FSS0_CORE0_INTR_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_156
	R5FSS0_CORE0_INTR_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_157
	R5FSS0_CORE0_INTR_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_158
	R5FSS0_CORE0_INTR_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_159
	R5FSS0_CORE0_INTR_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_160
	R5FSS0_CORE0_INTR_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_161
	R5FSS0_CORE0_INTR_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_162
	R5FSS0_CORE0_INTR_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_163
	R5FSS0_CORE0_INTR_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_164
	R5FSS0_CORE0_INTR_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_165
	R5FSS0_CORE0_INTR_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_166
	R5FSS0_CORE0_INTR_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_167
	R5FSS0_CORE0_INTR_IN_80	EPWM0_EPWM_TRIPZINT_OUT_0
	R5FSS0_CORE0_INTR_IN_81	EPWM1_EPWM_TRIPZINT_OUT_0
	R5FSS0_CORE0_INTR_IN_82	EPWM2_EPWM_TRIPZINT_OUT_0
	R5FSS0_CORE0_INTR_IN_83	ECAP0_ECAP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_84	ECAP1_ECAP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_85	ECAP2_ECAP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_86	EQEP0_EQEP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_87	EQEP1_EQEP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_88	EQEP2_EQEP_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_90	R5FSS0_COMMON0_COMMRX_LEVEL_0_OUT_0
	R5FSS0_CORE0_INTR_IN_91	R5FSS0_COMMON0_COMMTX_LEVEL_0_OUT_0
	R5FSS0_CORE0_INTR_IN_92	DSS1_DISPC_INTR_REQ_0_OUT_0
	R5FSS0_CORE0_INTR_IN_93	DSS1_DISPC_INTR_REQ_1_OUT_0
	R5FSS0_CORE0_INTR_IN_94	R5FSS0_CORE0_PMU_OUT_0
	R5FSS0_CORE0_INTR_IN_95	R5FSS0_CORE0_VALFIQ_OUT_0
	R5FSS0_CORE0_INTR_IN_96	R5FSS0_CORE0_VALIRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_97	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_99	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_100	JPGENC0_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_103	GPMC0_GPMC_SINTERRUPT_OUT_0
	R5FSS0_CORE0_INTR_IN_104	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_16
	R5FSS0_CORE0_INTR_IN_105	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_17
	R5FSS0_CORE0_INTR_IN_106	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_18
	R5FSS0_CORE0_INTR_IN_107	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_19
	R5FSS0_CORE0_INTR_IN_108	MCU_DCC0_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC0_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC1_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC2_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC3_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC4_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC5_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC6_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC7_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_109	DCC8_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_110	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_113	PBIST0_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	PBIST1_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	PBIST2_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	GPU0_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_113	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_114	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_114	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_115	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_116	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_117	MCASP3_XMIT_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_118	MCASP3_REC_INTR_PEND_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_119	MCRC64_0_INT_MCRC_OUT_0
	R5FSS0_CORE0_INTR_IN_120	MCASP0_REC_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_121	MCASP0_XMIT_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_122	MCASP1_REC_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_123	MCASP1_XMIT_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_124	MCASP2_REC_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_125	MCASP2_XMIT_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_126	PCIE0_PCIE_DPA_PULSE_OUT_0
	R5FSS0_CORE0_INTR_IN_128	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_128	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_128	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_128	CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_128	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	R5FSS0_CORE0_INTR_IN_129	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24
	R5FSS0_CORE0_INTR_IN_130	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25
	R5FSS0_CORE0_INTR_IN_131	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26
	R5FSS0_CORE0_INTR_IN_132	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27
	R5FSS0_CORE0_INTR_IN_133	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_134	CPSW0_EVNT_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_135	CPSW0_MDIO_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_136	CPSW0_STAT_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_137	MCU_DCC1_INTR_DONE_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_138	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28
	R5FSS0_CORE0_INTR_IN_139	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29
	R5FSS0_CORE0_INTR_IN_140	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_141	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_142	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_143	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30
	R5FSS0_CORE0_INTR_IN_144	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31
	R5FSS0_CORE0_INTR_IN_145	WKUP_PSC0_PSC_ALLINT_OUT_0
	R5FSS0_CORE0_INTR_IN_146	PSC0_PSC_ALLINT_OUT_0
	R5FSS0_CORE0_INTR_IN_147	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_147	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_147	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_149	MCU_PBIST0_DFT_PBIST_CPU_OUT_0
	R5FSS0_CORE0_INTR_IN_150	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23
	R5FSS0_CORE0_INTR_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0
	R5FSS0_CORE0_INTR_IN_152	WKUP_TIMEOUT1_TIMED_OUT_OUT_0
	R5FSS0_CORE0_INTR_IN_152	WKUP_TIMEOUT0_TIMED_OUT_OUT_0
	R5FSS0_CORE0_INTR_IN_153	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_154	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_155	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_156	CSI_RX_IF3_CSI_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_157	CSI_RX_IF3_CSI_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_158	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14
	R5FSS0_CORE0_INTR_IN_159	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15
	R5FSS0_CORE0_INTR_IN_160	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22
	R5FSS0_CORE0_INTR_IN_161	MMCSD0_EMMCSS_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_162	MMCSD1_EMMCSDSS_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_163	MMCSD2_EMMCSDSS_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_165	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0
	R5FSS0_CORE0_INTR_IN_166	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_167	ESM0_ESM_INT_CFG_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_168	ESM0_ESM_INT_HI_LVL_OUT_0
	R5FSS0_CORE0_INTR_IN_169	ESM0_ESM_INT_LOW_LVL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_170	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_172	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_175	R5FSS0_CORE0_CTI_OUT_0
	R5FSS0_CORE0_INTR_IN_176	CSI_RX_IF1_CSI_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_177	DDPA0_DDPA_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_178	VPAC0_VPAC_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_179	VPAC0_VPAC_LEVEL_OUT_1
	R5FSS0_CORE0_INTR_IN_180	VPAC0_VPAC_LEVEL_OUT_2
	R5FSS0_CORE0_INTR_IN_181	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	R5FSS0_CORE0_INTR_IN_182	VPAC0_VPAC_LEVEL_OUT_3
	R5FSS0_CORE0_INTR_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	R5FSS0_CORE0_INTR_IN_189	VPAC0_VPAC_LEVEL_OUT_4
	R5FSS0_CORE0_INTR_IN_190	WKUP_I2C0_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	R5FSS0_CORE0_INTR_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	R5FSS0_CORE0_INTR_IN_193	I2C0_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_194	I2C1_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_195	I2C2_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_196	I2C3_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_197	MCU_I2C0_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_198	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_199	CSI_RX_IF2_CSI_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_200	CSI_RX_IF2_CSI_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_203	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	R5FSS0_CORE0_INTR_IN_204	MCSPI0_INTR_SPI_OUT_0
	R5FSS0_CORE0_INTR_IN_205	MCSPI1_INTR_SPI_OUT_0
	R5FSS0_CORE0_INTR_IN_206	MCSPI2_INTR_SPI_OUT_0
	R5FSS0_CORE0_INTR_IN_207	MCU_MCSPI0_INTR_SPI_OUT_0
	R5FSS0_CORE0_INTR_IN_208	MCU_MCSPI1_INTR_SPI_OUT_0
	R5FSS0_CORE0_INTR_IN_209	CSI_RX_IF1_CSI_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_210	UART0_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_211	UART1_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_212	UART2_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_213	UART3_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_214	UART4_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_215	UART5_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_216	UART6_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_217	MCU_UART0_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_218	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	R5FSS0_CORE0_INTR_IN_219	WKUP_UART0_USART_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_220	USB0_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_221	USB0_IRQ_OUT_1
	R5FSS0_CORE0_INTR_IN_222	USB0_IRQ_OUT_2
	R5FSS0_CORE0_INTR_IN_223	USB0_IRQ_OUT_3
	R5FSS0_CORE0_INTR_IN_224	USB0_IRQ_OUT_4
	R5FSS0_CORE0_INTR_IN_225	USB0_IRQ_OUT_5
	R5FSS0_CORE0_INTR_IN_226	USB0_IRQ_OUT_6
	R5FSS0_CORE0_INTR_IN_227	USB0_IRQ_OUT_7
	R5FSS0_CORE0_INTR_IN_228	USB0_MISC_LEVEL_OUT_0
	R5FSS0_CORE0_INTR_IN_229	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	R5FSS0_CORE0_INTR_IN_230	USB1_IRQ_OUT_0
	R5FSS0_CORE0_INTR_IN_231	USB1_IRQ_OUT_1
	R5FSS0_CORE0_INTR_IN_232	USB1_IRQ_OUT_2
	R5FSS0_CORE0_INTR_IN_233	USB1_IRQ_OUT_3
	R5FSS0_CORE0_INTR_IN_234	USB1_IRQ_OUT_4
	R5FSS0_CORE0_INTR_IN_235	USB1_IRQ_OUT_5

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
R5FSS0_CORE0	R5FSS0_CORE0_INTR_IN_236	USB1_IRQ_OUT_6
	R5FSS0_CORE0_INTR_IN_237	USB1_IRQ_OUT_7
	R5FSS0_CORE0_INTR_IN_238	USB1_HOST_SYSTEM_ERROR_OUT_0
	R5FSS0_CORE0_INTR_IN_239	I2C4_POINTRPEND_OUT_0
	R5FSS0_CORE0_INTR_IN_240	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_241	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_242	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_243	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_3
	R5FSS0_CORE0_INTR_IN_244	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_24
	R5FSS0_CORE0_INTR_IN_245	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_25
	R5FSS0_CORE0_INTR_IN_246	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_26
	R5FSS0_CORE0_INTR_IN_247	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_27
	R5FSS0_CORE0_INTR_IN_248	GPU0_GPU_PWRCTRL_REQ_OUT_0
	R5FSS0_CORE0_INTR_IN_249	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_24
	R5FSS0_CORE0_INTR_IN_250	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_25
	R5FSS0_CORE0_INTR_IN_251	MCASP4_XMIT_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_252	MCASP4_REC_INTR_PEND_OUT_0
	R5FSS0_CORE0_INTR_IN_254	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_26
	R5FSS0_CORE0_INTR_IN_255	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_27
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_0	MCU_CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_3	DMPAC0_DFT_LBIST_BIST_DONE_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_4	WKUP_R5FSS0_CORE0_EXP_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_5	WKUP_R5FSS0_COMMON0_COMMRX_LEVEL_0_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_6	WKUP_R5FSS0_COMMON0_COMMTX_LEVEL_0_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_7	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_6
	WKUP_R5FSS0_CORE0_INTR_IN_8	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_72
	WKUP_R5FSS0_CORE0_INTR_IN_9	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_73
	WKUP_R5FSS0_CORE0_INTR_IN_10	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_74
	WKUP_R5FSS0_CORE0_INTR_IN_11	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_75
	WKUP_R5FSS0_CORE0_INTR_IN_12	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_76
	WKUP_R5FSS0_CORE0_INTR_IN_13	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_77
	WKUP_R5FSS0_CORE0_INTR_IN_14	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_78
	WKUP_R5FSS0_CORE0_INTR_IN_15	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_79

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_16	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_17	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_18	MCU_GPIO0_GPIO_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_23	WKUP_ICEMELTER0_PSC_FORCE_POWER_ON_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_24	DSS0_DISPC_INTR_REQ_0_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_25	DSS0_DISPC_INTR_REQ_1_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_27	WKUP_DEEPSLEEP_SOURCES0_WAKEUP_EVENT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_28	WKUP_TIMER0_TIMER_CLKSTOP_WAKEUP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_29	WKUP_TIMER1_TIMER_CLKSTOP_WAKEUP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_30	WKUP_RTI0_INTR_WWD_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_31	GPU0_GPU_PWRCTRL_REQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_32	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_33	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_34	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_35	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_36	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4
	WKUP_R5FSS0_CORE0_INTR_IN_37	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5
	WKUP_R5FSS0_CORE0_INTR_IN_38	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6
	WKUP_R5FSS0_CORE0_INTR_IN_39	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7
	WKUP_R5FSS0_CORE0_INTR_IN_40	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8
	WKUP_R5FSS0_CORE0_INTR_IN_41	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9
	WKUP_R5FSS0_CORE0_INTR_IN_42	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10
	WKUP_R5FSS0_CORE0_INTR_IN_43	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11
	WKUP_R5FSS0_CORE0_INTR_IN_44	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12
	WKUP_R5FSS0_CORE0_INTR_IN_45	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13
	WKUP_R5FSS0_CORE0_INTR_IN_46	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14
	WKUP_R5FSS0_CORE0_INTR_IN_47	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15
	WKUP_R5FSS0_CORE0_INTR_IN_48	CPSW0_CPTS_COMP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_50	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_51	CSI_TX_IF0_CSI_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_52	USB1_OTGIRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_53	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_54	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_55	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_56	GPIO0_GPIO_BANK_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_57	GPIO0_GPIO_BANK_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_58	WKUP_R5FSS0_CORE0_PMU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_59	WKUP_R5FSS0_CORE0_VALFIQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_60	WKUP_R5FSS0_CORE0_VALIRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_61	USB0_USB_WAKEUP_CLKSTOP_WAKEUP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_62	DMPAC0_DMPAC_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_63	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_40
	WKUP_R5FSS0_CORE0_INTR_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_41
	WKUP_R5FSS0_CORE0_INTR_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_42
	WKUP_R5FSS0_CORE0_INTR_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_43
	WKUP_R5FSS0_CORE0_INTR_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_44
	WKUP_R5FSS0_CORE0_INTR_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_45
	WKUP_R5FSS0_CORE0_INTR_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_46
	WKUP_R5FSS0_CORE0_INTR_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_47
	WKUP_R5FSS0_CORE0_INTR_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_48
	WKUP_R5FSS0_CORE0_INTR_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_49
	WKUP_R5FSS0_CORE0_INTR_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_50
	WKUP_R5FSS0_CORE0_INTR_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_51
	WKUP_R5FSS0_CORE0_INTR_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_52
	WKUP_R5FSS0_CORE0_INTR_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_53
	WKUP_R5FSS0_CORE0_INTR_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_54
	WKUP_R5FSS0_CORE0_INTR_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_55
	WKUP_R5FSS0_CORE0_INTR_IN_80	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_56
	WKUP_R5FSS0_CORE0_INTR_IN_81	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_57
	WKUP_R5FSS0_CORE0_INTR_IN_82	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_58
	WKUP_R5FSS0_CORE0_INTR_IN_83	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_59
	WKUP_R5FSS0_CORE0_INTR_IN_84	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_60
	WKUP_R5FSS0_CORE0_INTR_IN_85	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_61
	WKUP_R5FSS0_CORE0_INTR_IN_86	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_62

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_87	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_63
	WKUP_R5FSS0_CORE0_INTR_IN_88	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_64
	WKUP_R5FSS0_CORE0_INTR_IN_89	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_65
	WKUP_R5FSS0_CORE0_INTR_IN_90	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_66
	WKUP_R5FSS0_CORE0_INTR_IN_91	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_67
	WKUP_R5FSS0_CORE0_INTR_IN_92	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_68
	WKUP_R5FSS0_CORE0_INTR_IN_93	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_69
	WKUP_R5FSS0_CORE0_INTR_IN_94	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_70
	WKUP_R5FSS0_CORE0_INTR_IN_95	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_71
	WKUP_R5FSS0_CORE0_INTR_IN_96	DMPAC0_DMPAC_LEVEL_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_97	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_98	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_99	PCIE0_PCIE_DPA_PULSE_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_100	JPGENC0_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_103	GPMC0_GPMC_SINTERRUPT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_104	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_105	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_106	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_107	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_108	MCU_DCC0_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC0_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC1_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC2_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC3_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC4_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC5_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC6_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC7_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_109	DCC8_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_110	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	PBIST0_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	PBIST1_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	PBIST2_DFT_PBIST_CPU_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_113	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	GPU0_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_113	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_114	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_114	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_115	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_116	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_117	DSS1_DISPC_INTR_REQ_0_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_118	DSS1_DISPC_INTR_REQ_1_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_119	MCRC64_0_INT_MCRC_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_120	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_12
	WKUP_R5FSS0_CORE0_INTR_IN_121	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_13
	WKUP_R5FSS0_CORE0_INTR_IN_122	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_14
	WKUP_R5FSS0_CORE0_INTR_IN_123	EPWM0_EPWM_ETINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_124	EPWM0_EPWM_TRIPZINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_125	EPWM1_EPWM_ETINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_126	EPWM1_EPWM_TRIPZINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_127	EPWM2_EPWM_ETINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_128	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_128	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_128	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_128	CTRL_MMR0_ACCESS_ERR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_128	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_129	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8
	WKUP_R5FSS0_CORE0_INTR_IN_130	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9
	WKUP_R5FSS0_CORE0_INTR_IN_131	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10
	WKUP_R5FSS0_CORE0_INTR_IN_132	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11
	WKUP_R5FSS0_CORE0_INTR_IN_133	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_134	CPSW0_EVNT_PEND_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_135	CPSW0_MDIO_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_136	CPSW0_STAT_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_137	MCU_DCC1_INTR_DONE_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_138	WKUP_TIMER0_INTR_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_139	WKUP_TIMER1_INTR_PEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_140	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_141	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_142	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_143	WKUP_I2C0_CLKSTOP_WAKEUP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_144	WKUP_UART0_CLKSTOP_WAKEUP_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_145	WKUP_PSC0_PSC_ALLINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_146	PSC0_PSC_ALLINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_147	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_148	EPWM2_EPWM_TRIPZINT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_149	MCU_PBIST0_DFT_PBIST_CPU_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_150	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12
	WKUP_R5FSS0_CORE0_INTR_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_152	WKUP_TIMEOUT1_TIMED_OUT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_152	WKUP_TIMEOUT0_TIMED_OUT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_153	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_154	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_155	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_156	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_157	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_158	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13
	WKUP_R5FSS0_CORE0_INTR_IN_159	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14
	WKUP_R5FSS0_CORE0_INTR_IN_160	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15
	WKUP_R5FSS0_CORE0_INTR_IN_161	MMCSD0_EMMCSS_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_162	MMCSD1_EMMCSDSS_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_163	MMCSD2_EMMCSDSS_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_165	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_15
	WKUP_R5FSS0_CORE0_INTR_IN_166	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_167	ESM0_ESM_INT_CFG_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_168	ESM0_ESM_INT_HI_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_169	ESM0_ESM_INT_LOW_LVL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_170	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_172	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_175	WKUP_R5FSS0_CORE0_CTI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_176	CSI_RX_IF1_CSI_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_177	DDPA0_DDPA_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_178	VPAC0_VPAC_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_179	VPAC0_VPAC_LEVEL_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_180	VPAC0_VPAC_LEVEL_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_181	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_182	VPAC0_VPAC_LEVEL_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_189	VPAC0_VPAC_LEVEL_OUT_4
	WKUP_R5FSS0_CORE0_INTR_IN_190	WKUP_I2C0_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	WKUP_R5FSS0_CORE0_INTR_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_193	I2C0_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_194	I2C1_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_195	I2C2_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_196	I2C3_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_197	MCU_I2C0_POINTRPEND_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_198	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_199	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_200	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_203	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_204	MCSPI0_INTR_SPI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_205	MCSPI1_INTR_SPI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_206	MCSPI2_INTR_SPI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_207	MCU_MCSPI0_INTR_SPI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_208	MCU_MCSPI1_INTR_SPI_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_209	CSI_RX_IF1_CSI_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_210	UART0_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_211	UART1_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_212	UART2_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_213	UART3_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_214	UART4_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_215	UART5_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_216	UART6_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_217	MCU_UART0_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_218	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_219	WKUP_UART0_USART_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_220	USB0_IRQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_INTR_IN_221	USB0_IRQ_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_222	USB0_IRQ_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_223	USB0_IRQ_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_224	USB0_IRQ_OUT_4
	WKUP_R5FSS0_CORE0_INTR_IN_225	USB0_IRQ_OUT_5
	WKUP_R5FSS0_CORE0_INTR_IN_226	USB0_IRQ_OUT_6
	WKUP_R5FSS0_CORE0_INTR_IN_227	USB0_IRQ_OUT_7
	WKUP_R5FSS0_CORE0_INTR_IN_228	USB0_MISC_LEVEL_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_229	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_230	USB1_IRQ_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_231	USB1_IRQ_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_232	USB1_IRQ_OUT_2
	WKUP_R5FSS0_CORE0_INTR_IN_233	USB1_IRQ_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_234	USB1_IRQ_OUT_4
	WKUP_R5FSS0_CORE0_INTR_IN_235	USB1_IRQ_OUT_5
	WKUP_R5FSS0_CORE0_INTR_IN_236	USB1_IRQ_OUT_6
	WKUP_R5FSS0_CORE0_INTR_IN_237	USB1_IRQ_OUT_7
	WKUP_R5FSS0_CORE0_INTR_IN_238	USB1_HOST_SYSTEM_ERROR_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_239	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_240	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_241	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_242	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_243	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_3
	WKUP_R5FSS0_CORE0_INTR_IN_244	EQEP0_EQEP_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_245	EQEP1_EQEP_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_246	EQEP2_EQEP_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_247	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	WKUP_R5FSS0_CORE0_INTR_IN_248	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	WKUP_R5FSS0_CORE0_INTR_IN_249	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_12
	WKUP_R5FSS0_CORE0_INTR_IN_250	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_13
	WKUP_R5FSS0_CORE0_INTR_IN_254	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_14
	WKUP_R5FSS0_CORE0_INTR_IN_255	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_15
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_3	DMPAC0_DFT_LBIST_BIST_DONE_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_4	MCU_R5FSS0_CORE0_CPU0_EXP_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_5	DMPAC0_DMPAC_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_6	DMPAC0_DMPAC_LEVEL_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_7	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_7
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_16	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_17	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_28	MCU_TIMER0_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_29	MCU_TIMER1_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_30	MCU_RTI0_INTR_WWD_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_31	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_31	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_32	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_34
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_33	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_35
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_36	EPWM0_EPWM_ETINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_37	EPWM1_EPWM_ETINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_38	EPWM2_EPWM_ETINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39	CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_40	DSS0_DISPC_INTR_REQ_0_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_41	DSS0_DISPC_INTR_REQ_1_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_42	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_43	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_44	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_45	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_46	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_47	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_48	CPSW0_CPTS_COMP_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_50	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_51	CSI_TX_IF0_CSI_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_52	USB1_OTGIRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_53	PCIE0_PCIE_CPTS_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_54	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_55	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_63	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_168
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_169
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_170
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_171
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_172
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_173
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_174
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_175
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_176
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_177
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_178
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_179
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_180
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_181
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_182
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_183
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_80	EPWM0_EPWM_TRIPZINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_81	EPWM1_EPWM_TRIPZINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_82	EPWM2_EPWM_TRIPZINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_83	ECAP0_ECAP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_84	ECAP1_ECAP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_85	ECAP2_ECAP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_86	EQEP0_EQEP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_87	EQEP1_EQEP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_88	EQEP2_EQEP_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_90	MCU_R5FSS0_CORE0_COMMRX_LEVEL_0_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_91	MCU_R5FSS0_CORE0_COMMTX_LEVEL_0_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_92	DSS1_DISPC_INTR_REQ_0_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_93	DSS1_DISPC_INTR_REQ_1_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_94	MCU_R5FSS0_CORE0_CPU0_PMU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_95	MCU_R5FSS0_CORE0_CPU0_VALFIQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_96	MCU_R5FSS0_CORE0_CPU0_VALIRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_97	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_99	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_100	JPGENC0_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_103	GPMC0_GPMC_SINTERRUPT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_104	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_4
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_105	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_5
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_106	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_6
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_107	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_7
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_108	MCU_DCC0_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC0_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC1_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC2_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC3_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC4_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC5_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC6_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC7_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109	DCC8_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_110	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	PBIST0_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	PBIST1_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	PBIST2_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	GPU0_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_114	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_114	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_115	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_116	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_117	MCASP3_XMIT_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_118	MCASP3_REC_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_119	MCRC64_0_INT_MCRC_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_120	MCASP0_REC_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_121	MCASP0_XMIT_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_122	MCASP1_REC_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_123	MCASP1_XMIT_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_124	MCASP2_REC_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_125	MCASP2_XMIT_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_126	PCIE0_PCIE_DPA_PULSE_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128	CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_129	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_130	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_131	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_132	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_133	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_134	CPSW0_EVNT_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_135	CPSW0_MDIO_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_136	CPSW0_STAT_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_137	MCU_DCC1_INTR_DONE_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_138	MCU_TIMER2_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_139	MCU_TIMER3_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_140	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_141	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_142	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_145	WKUP_PSC0_PSC_ALLINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_146	PSC0_PSC_ALLINT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_149	MCU_PBIST0_DFT_PBIST_CPU_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_150	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_152	WKUP_TIMEOUT1_TIMED_OUT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_152	WKUP_TIMEOUT0_TIMED_OUT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_153	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_154	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_155	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_156	CSI_RX_IF3_CSI_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_157	CSI_RX_IF3_CSI_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_158	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_159	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_160	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_161	MMCSD0_EMMCSS_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_162	MMCSD1_EMMCSDSS_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_163	MMCSD2_EMMCSDSS_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_165	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_166	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_167	ESM0_ESM_INT_CFG_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_168	ESM0_ESM_INT_HI_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_169	ESM0_ESM_INT_LOW_LVL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_170	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_172	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_175	MCU_R5FSS0_CORE0_CPU0_CTI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_176	CSI_RX_IF1_CSI_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_177	DDPA0_DDPA_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_178	VPAC0_VPAC_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_179	VPAC0_VPAC_LEVEL_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_180	VPAC0_VPAC_LEVEL_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_181	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_182	VPAC0_VPAC_LEVEL_OUT_3
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_189	VPAC0_VPAC_LEVEL_OUT_4
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_190	WKUP_I2C0_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_193	I2C0_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_194	I2C1_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_195	I2C2_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_196	I2C3_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_197	MCU_I2C0_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_198	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_199	CSI_RX_IF2_CSI_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_200	CSI_RX_IF2_CSI_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_203	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_204	MCSPI0_INTR_SPI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_205	MCSPI1_INTR_SPI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_206	MCSPI2_INTR_SPI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_207	MCU_MCSPI0_INTR_SPI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_208	MCU_MCSPI1_INTR_SPI_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_209	CSI_RX_IF1_CSI_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_210	UART0_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_211	UART1_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_212	UART2_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_213	UART3_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_214	UART4_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_215	UART5_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_216	UART6_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_217	MCU_UART0_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_218	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_219	WKUP_UART0_USART_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_220	USB0_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_221	USB0_IRQ_OUT_1
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_222	USB0_IRQ_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_223	USB0_IRQ_OUT_3
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_224	USB0_IRQ_OUT_4
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_225	USB0_IRQ_OUT_5
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_226	USB0_IRQ_OUT_6
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_227	USB0_IRQ_OUT_7
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_228	USB0_MISC_LEVEL_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_229	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_230	USB1_IRQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_231	USB1_IRQ_OUT_1

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_232	USB1_IRQ_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_233	USB1_IRQ_OUT_3
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_234	USB1_IRQ_OUT_4
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_235	USB1_IRQ_OUT_5
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_236	USB1_IRQ_OUT_6
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_237	USB1_IRQ_OUT_7
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_238	USB1_HOST_SYSTEM_ERROR_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_239	I2C4_POINTRPEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_240	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_241	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_242	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_243	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_2
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_244	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_8
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_245	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_9
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_246	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_10
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_247	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_11
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_248	GPU0_GPU_PWRCTRL_REQ_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_249	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_8
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_250	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_9
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_251	MCASP4_XMIT_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_252	MCASP4_REC_INTR_PEND_OUT_0
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_254	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_10
	MCU_R5FSS0_CORE0_CPU0_INTR_IN_255	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_11
WKUP_DEEPSLEEP_SOURCES0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_0	WKUP_I2C0_CLKSTOP_WAKEUP_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_1	WKUP_UART0_CLKSTOP_WAKEUP_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_2	MCU_GPIO0_GPIO_LVL_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_3	WKUP_ICEMELTER0_PSC_FORCE_POWER_ON_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_5	WKUP_TIMER0_TIMER_CLKSTOP_WAKEUP_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_6	WKUP_TIMER1_TIMER_CLKSTOP_WAKEUP_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_7	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_9	USB0_USB_WAKEUP_CLKSTOP_WAKEUP_OUT_0
PDMA2	PDMA2_MCASP_MAIN_0_RX_IN_0	MCASP0_REC_DMA_EVENT_REQ_OUT_0
	PDMA2_MCASP_MAIN_0_TX_IN_0	MCASP0_XMIT_DMA_EVENT_REQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
PDMA2	PDMA2_MCASP_MAIN_1_RX_IN_0	MCASP1_REC_DMA_EVENT_REQ_OUT_0
	PDMA2_MCASP_MAIN_1_TX_IN_0	MCASP1_XMIT_DMA_EVENT_REQ_OUT_0
	PDMA2_MCASP_MAIN_2_RX_IN_0	MCASP2_REC_DMA_EVENT_REQ_OUT_0
	PDMA2_MCASP_MAIN_2_TX_IN_0	MCASP2_XMIT_DMA_EVENT_REQ_OUT_0
PDMA1	PDMA1_USART_MAIN_0_RX_IN_0	UART0_USART_DMA_OUT_1
	PDMA1_USART_MAIN_0_TX_IN_0	UART0_USART_DMA_OUT_0
	PDMA1_USART_MAIN_1_RX_IN_0	UART1_USART_DMA_OUT_1
	PDMA1_USART_MAIN_1_TX_IN_0	UART1_USART_DMA_OUT_0
	PDMA1_USART_MAIN_2_RX_IN_0	UART2_USART_DMA_OUT_1
	PDMA1_USART_MAIN_2_TX_IN_0	UART2_USART_DMA_OUT_0
	PDMA1_USART_MAIN_3_RX_IN_0	UART3_USART_DMA_OUT_1
	PDMA1_USART_MAIN_3_TX_IN_0	UART3_USART_DMA_OUT_0
	PDMA1_USART_MAIN_4_RX_IN_0	UART4_USART_DMA_OUT_1
	PDMA1_USART_MAIN_4_TX_IN_0	UART4_USART_DMA_OUT_0
	PDMA1_USART_MAIN_5_RX_IN_0	UART5_USART_DMA_OUT_1
	PDMA1_USART_MAIN_5_TX_IN_0	UART5_USART_DMA_OUT_0
	PDMA1_USART_MAIN_6_RX_IN_0	UART6_USART_DMA_OUT_1
	PDMA1_USART_MAIN_6_TX_IN_0	UART6_USART_DMA_OUT_0
COMPUTE_CLUSTER0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0
	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0	C7X256V0_CLEC_ARM_0_EVENTI_OUT_OUT_0
	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0	C7X256V1_CLEC_ARM_0_EVENTI_OUT_OUT_0
C7X256V0_CLEC	C7X256V0_CLEC_ARM_0_EVENTO_IN_IN_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_32	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_33	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_34	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_35	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_36	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_37	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_38	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6
	C7X256V0_CLEC_GIC_SPI_IN_39	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7
	C7X256V0_CLEC_GIC_SPI_IN_40	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8
	C7X256V0_CLEC_GIC_SPI_IN_41	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9
	C7X256V0_CLEC_GIC_SPI_IN_42	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_43	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11
	C7X256V0_CLEC_GIC_SPI_IN_44	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12
	C7X256V0_CLEC_GIC_SPI_IN_45	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13
	C7X256V0_CLEC_GIC_SPI_IN_46	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14
	C7X256V0_CLEC_GIC_SPI_IN_47	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15
	C7X256V0_CLEC_GIC_SPI_IN_48	CPSW0_CPTS_COMP_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_50	DMPAC0_DMPAC_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_51	DMPAC0_DMPAC_LEVEL_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_58	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_59	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_60	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_61	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_62	DSS1_DISPC_INTR_REQ_0_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_63	DSS1_DISPC_INTR_REQ_1_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	C7X256V0_CLEC_GIC_SPI_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	C7X256V0_CLEC_GIC_SPI_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8
	C7X256V0_CLEC_GIC_SPI_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9
	C7X256V0_CLEC_GIC_SPI_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10
	C7X256V0_CLEC_GIC_SPI_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11
	C7X256V0_CLEC_GIC_SPI_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12
	C7X256V0_CLEC_GIC_SPI_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13
	C7X256V0_CLEC_GIC_SPI_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14
	C7X256V0_CLEC_GIC_SPI_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15
	C7X256V0_CLEC_GIC_SPI_IN_80	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16
	C7X256V0_CLEC_GIC_SPI_IN_81	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_82	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18
	C7X256V0_CLEC_GIC_SPI_IN_83	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19
	C7X256V0_CLEC_GIC_SPI_IN_84	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20
	C7X256V0_CLEC_GIC_SPI_IN_85	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21
	C7X256V0_CLEC_GIC_SPI_IN_86	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22
	C7X256V0_CLEC_GIC_SPI_IN_87	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23
	C7X256V0_CLEC_GIC_SPI_IN_88	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24
	C7X256V0_CLEC_GIC_SPI_IN_89	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25
	C7X256V0_CLEC_GIC_SPI_IN_90	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26
	C7X256V0_CLEC_GIC_SPI_IN_91	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27
	C7X256V0_CLEC_GIC_SPI_IN_92	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28
	C7X256V0_CLEC_GIC_SPI_IN_93	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29
	C7X256V0_CLEC_GIC_SPI_IN_94	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30
	C7X256V0_CLEC_GIC_SPI_IN_95	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31
	C7X256V0_CLEC_GIC_SPI_IN_96	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32
	C7X256V0_CLEC_GIC_SPI_IN_97	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33
	C7X256V0_CLEC_GIC_SPI_IN_98	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34
	C7X256V0_CLEC_GIC_SPI_IN_99	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35
	C7X256V0_CLEC_GIC_SPI_IN_100	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36
	C7X256V0_CLEC_GIC_SPI_IN_101	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37
	C7X256V0_CLEC_GIC_SPI_IN_102	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38
	C7X256V0_CLEC_GIC_SPI_IN_103	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39
	C7X256V0_CLEC_GIC_SPI_IN_104	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_105	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_106	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_107	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_108	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_109	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_112	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_113	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_114	MMCSD2_EMMCSDSS_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_115	MMCSD1_EMMCSDSS_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_116	DSS0_DISPC_INTR_REQ_0_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_117	DSS0_DISPC_INTR_REQ_1_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_118	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_119	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_120	GPU0_GPU_PWRCTRL_REQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_121	PCIE0_PCIE_CPTS_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_122	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_123	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_124	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_125	PCIE0_PCIE_ERROR_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_126	PCIE0_PCIE_FLR_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_127	PCIE0_PCIE_LEGACY_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC0_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC1_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC2_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC3_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC4_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC5_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC6_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC7_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_128	DCC8_INTR_DONE_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_129	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_129	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_129	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_129	CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_129	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_130	JPGENC0_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_131	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_132	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_133	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_134	CPSW0_EVNT_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_135	CPSW0_MDIO_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_136	CPSW0_STAT_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_137	PCIE0_PCIE_LOCAL_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_138	GPMC0_GPMC_SINTERRUPT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_139	MCU_I2C0_POINTRPEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_140	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_141	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_142	PCIE0_PCIE_PHY_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_145	ECAP0_ECAP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_146	ECAP1_ECAP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_147	ECAP2_ECAP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_148	EQEP0_EQEP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_149	EQEP1_EQEP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_150	EQEP2_EQEP_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_152	TIMER0_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_153	TIMER1_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_154	TIMER2_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_155	TIMER3_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_156	TIMER4_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_157	TIMER5_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_158	TIMER6_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_159	TIMER7_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_160	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_161	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_162	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_163	PCIE0_PCIE_DPA_PULSE_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_165	MMCSD0_EMMCSS_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_166	MCRC64_0_INT_MCRC_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_167	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_168	VPAC0_VPAC_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_169	VPAC0_VPAC_LEVEL_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_170	VPAC0_VPAC_LEVEL_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_172	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_175	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_177	DDPA0_DDPA_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_178	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_179	CSI_RX_IF1_CSI_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_180	ESM0_ESM_INT_CFG_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_181	ESM0_ESM_INT_HI_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_182	ESM0_ESM_INT_LOW_LVL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_189	VPAC0_VPAC_LEVEL_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_190	VPAC0_VPAC_LEVEL_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_193	I2C0_POINTRPEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_194	I2C1_POINTRPEND_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_195	I2C2_POINTRPEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_196	I2C3_POINTRPEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_197	WKUP_I2C0_POINTRPEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_199	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_200	CSI_RX_IF1_CSI_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_203	PSC0_PSC_ALLINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_204	MCSPI0_INTR_SPI_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_205	MCSPI1_INTR_SPI_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_206	MCSPI2_INTR_SPI_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_207	RTI15_INTR_WWD_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_208	MCU_MCSPI0_INTR_SPI_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_209	MCU_MCSPI1_INTR_SPI_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_210	UART0_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_211	UART1_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_212	UART2_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_213	UART3_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_214	UART4_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_215	UART5_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_216	UART6_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_217	MCU_UART0_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_218	WKUP_UART0_USART_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_219	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_220	USB0_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_221	USB0_IRQ_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_222	USB0_IRQ_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_223	USB0_IRQ_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_224	USB0_IRQ_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_225	USB0_IRQ_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_226	USB0_IRQ_OUT_6
	C7X256V0_CLEC_GIC_SPI_IN_227	USB0_IRQ_OUT_7
	C7X256V0_CLEC_GIC_SPI_IN_228	USB0_MISC_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_229	EPWM0_EPWM_ETINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_230	EPWM0_EPWM_TRIPZINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_231	EPWM1_EPWM_ETINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_232	CSI_RX_IF2_CSI_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_233	EPWM1_EPWM_TRIPZINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_234	EPWM2_EPWM_ETINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_235	EPWM2_EPWM_TRIPZINT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_236	CSI_RX_IF2_CSI_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_237	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_238	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_239	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_240	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_241	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_242	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	C7X256V0_CLEC_GIC_SPI_IN_243	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	C7X256V0_CLEC_GIC_SPI_IN_244	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	C7X256V0_CLEC_GIC_SPI_IN_245	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_246	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_249	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_250	CSI_RX_IF3_CSI_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_251	CSI_RX_IF3_CSI_LEVEL_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_252	RTI0_INTR_WWD_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_253	RTI1_INTR_WWD_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_254	RTI2_INTR_WWD_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_255	RTI3_INTR_WWD_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_256	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_257	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_258	USB1_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_259	USB1_IRQ_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_260	USB1_IRQ_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_261	USB1_IRQ_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_262	USB1_IRQ_OUT_4
	C7X256V0_CLEC_GIC_SPI_IN_263	USB1_IRQ_OUT_5

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_GIC_SPI_IN_264	USB1_IRQ_OUT_6
	C7X256V0_CLEC_GIC_SPI_IN_265	USB1_IRQ_OUT_7
	C7X256V0_CLEC_GIC_SPI_IN_266	USB1_HOST_SYSTEM_ERROR_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_267	MCASP0_REC_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_268	MCASP0_XMIT_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_269	MCASP1_REC_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_270	MCASP1_XMIT_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_271	MCASP2_REC_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_272	MCASP2_XMIT_INTR_PEND_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_273	GPU0_OS_IRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_274	GPU0_OS_IRQ_OUT_1
	C7X256V0_CLEC_GIC_SPI_IN_275	GPU0_OS_IRQ_OUT_2
	C7X256V0_CLEC_GIC_SPI_IN_276	GPU0_OS_IRQ_OUT_3
	C7X256V0_CLEC_GIC_SPI_IN_277	USB1_OTGIRQ_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_278	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	C7X256V0_CLEC_GIC_SPI_IN_279	CSI_TX_IF0_CSI_LEVEL_OUT_0
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_4	RTI4_INTR_WWD_OUT_0
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_5	CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_OUT_0
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_6	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_7	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_8	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_9	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_10	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_11	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_1
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_12	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_20
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_13	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_21
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_14	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_22
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_15	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_23
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_16	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_84
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_17	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_85
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_18	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_86
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_19	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_87
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_20	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_88

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_SOC_EVENTS_IN_IN_21	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_89
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_22	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_90
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_23	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_91
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_24	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_92
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_25	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_93
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_26	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_94
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_27	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_95
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_28	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_96
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_29	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_97
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_30	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_98
	C7X256V0_CLEC_SOC_EVENTS_IN_IN_31	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_99
C7X256V1_CLEC	C7X256V1_CLEC_ARM_0_EVENTO_IN_IN_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_32	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_33	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_34	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_35	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_36	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_37	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_38	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6
	C7X256V1_CLEC_GIC_SPI_IN_39	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7
	C7X256V1_CLEC_GIC_SPI_IN_40	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8
	C7X256V1_CLEC_GIC_SPI_IN_41	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9
	C7X256V1_CLEC_GIC_SPI_IN_42	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10
	C7X256V1_CLEC_GIC_SPI_IN_43	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11
	C7X256V1_CLEC_GIC_SPI_IN_44	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12
	C7X256V1_CLEC_GIC_SPI_IN_45	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13
	C7X256V1_CLEC_GIC_SPI_IN_46	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14
	C7X256V1_CLEC_GIC_SPI_IN_47	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15
	C7X256V1_CLEC_GIC_SPI_IN_48	CPSW0_CPTS_COMP_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_49	PCIE0_PCIE_CPTS_COMP_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_50	DMPAC0_DMPAC_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_51	DMPAC0_DMPAC_LEVEL_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_58	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_59	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_60	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_61	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_62	DSS1_DISPC_INTR_REQ_0_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_63	DSS1_DISPC_INTR_REQ_1_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_64	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_65	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_66	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_67	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_68	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_69	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_70	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	C7X256V1_CLEC_GIC_SPI_IN_71	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	C7X256V1_CLEC_GIC_SPI_IN_72	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8
	C7X256V1_CLEC_GIC_SPI_IN_73	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9
	C7X256V1_CLEC_GIC_SPI_IN_74	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10
	C7X256V1_CLEC_GIC_SPI_IN_75	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11
	C7X256V1_CLEC_GIC_SPI_IN_76	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12
	C7X256V1_CLEC_GIC_SPI_IN_77	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13
	C7X256V1_CLEC_GIC_SPI_IN_78	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14
	C7X256V1_CLEC_GIC_SPI_IN_79	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15
	C7X256V1_CLEC_GIC_SPI_IN_80	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16
	C7X256V1_CLEC_GIC_SPI_IN_81	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17
	C7X256V1_CLEC_GIC_SPI_IN_82	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18
	C7X256V1_CLEC_GIC_SPI_IN_83	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19
	C7X256V1_CLEC_GIC_SPI_IN_84	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20
	C7X256V1_CLEC_GIC_SPI_IN_85	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21
	C7X256V1_CLEC_GIC_SPI_IN_86	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22
	C7X256V1_CLEC_GIC_SPI_IN_87	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23
	C7X256V1_CLEC_GIC_SPI_IN_88	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24
	C7X256V1_CLEC_GIC_SPI_IN_89	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25
	C7X256V1_CLEC_GIC_SPI_IN_90	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26
	C7X256V1_CLEC_GIC_SPI_IN_91	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_92	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28
	C7X256V1_CLEC_GIC_SPI_IN_93	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29
	C7X256V1_CLEC_GIC_SPI_IN_94	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30
	C7X256V1_CLEC_GIC_SPI_IN_95	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31
	C7X256V1_CLEC_GIC_SPI_IN_96	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32
	C7X256V1_CLEC_GIC_SPI_IN_97	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33
	C7X256V1_CLEC_GIC_SPI_IN_98	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34
	C7X256V1_CLEC_GIC_SPI_IN_99	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35
	C7X256V1_CLEC_GIC_SPI_IN_100	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36
	C7X256V1_CLEC_GIC_SPI_IN_101	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37
	C7X256V1_CLEC_GIC_SPI_IN_102	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38
	C7X256V1_CLEC_GIC_SPI_IN_103	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39
	C7X256V1_CLEC_GIC_SPI_IN_104	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_105	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_106	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_107	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_108	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_109	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_112	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_113	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_114	MMCSD2_EMMCSDSS_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_115	MMCSD1_EMMCSDSS_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_116	DSS0_DISPC_INTR_REQ_0_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_117	DSS0_DISPC_INTR_REQ_1_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_118	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_119	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_120	GPU0_GPU_PWRCTRL_REQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_121	PCIE0_PCIE_CPTS_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_122	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_123	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_124	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_125	PCIE0_PCIE_ERROR_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_126	PCIE0_PCIE_FLR_PULSE_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_127	PCIE0_PCIE_LEGACY_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC0_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC1_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC2_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC3_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC4_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC5_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC6_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC7_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_128	DCC8_INTR_DONE_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_129	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_129	PADCFG_CTRL0_ACCESS_ERR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_129	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_129	CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_129	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_130	JPGENC0_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_131	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_132	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_133	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_134	CPSW0_EVNT_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_135	CPSW0_MDIO_PEND_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_136	CPSW0_STAT_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_137	PCIE0_PCIE_LOCAL_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_138	GPMC0_GPMC_SINTERRUPT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_139	MCU_I2C0_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_140	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_141	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_142	PCIE0_PCIE_PHY_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_145	ECAP0_ECAP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_146	ECAP1_ECAP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_147	ECAP2_ECAP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_148	EQEP0_EQEP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_149	EQEP1_EQEP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_150	EQEP2_EQEP_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_151	DDR32SS0_DDRSS_CONTROLLER_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_152	TIMER0_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_153	TIMER1_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_154	TIMER2_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_155	TIMER3_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_156	TIMER4_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_157	TIMER5_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_158	TIMER6_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_159	TIMER7_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_160	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_161	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_162	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_163	PCIE0_PCIE_DPA_PULSE_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_164	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_165	MMCSD0_EMMCSS_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_166	MCRC64_0_INT_MCRC_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_167	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_168	VPAC0_VPAC_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_169	VPAC0_VPAC_LEVEL_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_170	VPAC0_VPAC_LEVEL_OUT_2

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_171	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_172	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_173	CSI_RX_IF0_CSI_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_174	CSI_RX_IF0_CSI_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_175	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_177	DDPA0_DDPA_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_178	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_179	CSI_RX_IF1_CSI_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_180	ESM0_ESM_INT_CFG_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_181	ESM0_ESM_INT_HI_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_182	ESM0_ESM_INT_LOW_LVL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_183	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_184	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_185	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_186	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_187	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_188	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_189	VPAC0_VPAC_LEVEL_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_190	VPAC0_VPAC_LEVEL_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_191	VPAC0_VPAC_LEVEL_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_192	MCU_MCRC64_0_INT_MCRC_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_193	I2C0_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_194	I2C1_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_195	I2C2_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_196	I2C3_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_197	WKUP_I2C0_POINTRPEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_199	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_200	CSI_RX_IF1_CSI_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_201	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_202	DEBUGSS0_CTM_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_203	PSC0_PSC_ALLINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_204	MCSPI0_INTR_SPI_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_205	MCSPI1_INTR_SPI_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_206	MCSPI2_INTR_SPI_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_207	RTI15_INTR_WWD_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_208	MCU_MCSPI0_INTR_SPI_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_209	MCU_MCSPI1_INTR_SPI_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_210	UART0_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_211	UART1_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_212	UART2_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_213	UART3_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_214	UART4_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_215	UART5_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_216	UART6_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_217	MCU_UART0_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_218	WKUP_UART0_USART_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_219	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_220	USB0_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_221	USB0_IRQ_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_222	USB0_IRQ_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_223	USB0_IRQ_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_224	USB0_IRQ_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_225	USB0_IRQ_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_226	USB0_IRQ_OUT_6
	C7X256V1_CLEC_GIC_SPI_IN_227	USB0_IRQ_OUT_7
	C7X256V1_CLEC_GIC_SPI_IN_228	USB0_MISC_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_229	EPWM0_EPWM_ETINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_230	EPWM0_EPWM_TRIPZINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_231	EPWM1_EPWM_ETINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_232	CSI_RX_IF2_CSI_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_233	EPWM1_EPWM_TRIPZINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_234	EPWM2_EPWM_ETINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_235	EPWM2_EPWM_TRIPZINT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_236	CSI_RX_IF2_CSI_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_237	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_238	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_239	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_240	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_241	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_242	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_243	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6
	C7X256V1_CLEC_GIC_SPI_IN_244	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7
	C7X256V1_CLEC_GIC_SPI_IN_245	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_246	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_249	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_250	CSI_RX_IF3_CSI_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_251	CSI_RX_IF3_CSI_LEVEL_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_252	RTI0_INTR_WWD_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_253	RTI1_INTR_WWD_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_254	RTI2_INTR_WWD_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_255	RTI3_INTR_WWD_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_256	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_257	CODEC0_VPU_WAVE521CL_INTR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_258	USB1_IRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_259	USB1_IRQ_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_260	USB1_IRQ_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_261	USB1_IRQ_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_262	USB1_IRQ_OUT_4
	C7X256V1_CLEC_GIC_SPI_IN_263	USB1_IRQ_OUT_5
	C7X256V1_CLEC_GIC_SPI_IN_264	USB1_IRQ_OUT_6
	C7X256V1_CLEC_GIC_SPI_IN_265	USB1_IRQ_OUT_7
	C7X256V1_CLEC_GIC_SPI_IN_266	USB1_HOST_SYSTEM_ERROR_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_267	MCASP0_REC_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_268	MCASP0_XMIT_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_269	MCASP1_REC_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_270	MCASP1_XMIT_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_271	MCASP2_REC_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_272	MCASP2_XMIT_INTR_PEND_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_273	GPU0_OS_IRQ_OUT_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_GIC_SPI_IN_274	GPU0_OS_IRQ_OUT_1
	C7X256V1_CLEC_GIC_SPI_IN_275	GPU0_OS_IRQ_OUT_2
	C7X256V1_CLEC_GIC_SPI_IN_276	GPU0_OS_IRQ_OUT_3
	C7X256V1_CLEC_GIC_SPI_IN_277	USB1_OTGIRQ_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_278	CSI_TX_IF0_CSI_INTERRUPT_OUT_0
	C7X256V1_CLEC_GIC_SPI_IN_279	CSI_TX_IF0_CSI_LEVEL_OUT_0
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_4	RTI5_INTR_WWD_OUT_0
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_5	CTRL_MMR0_IPC_SET1_IPC_SET_IPCFG_OUT_0
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_6	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_7	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_8	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_9	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_10	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_11	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_2
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_12	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_20
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_13	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_21
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_14	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_22
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_15	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_23
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_16	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_100
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_17	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_101
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_18	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_102
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_19	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_103
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_20	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_104
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_21	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_105
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_22	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_106
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_23	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_107
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_24	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_108
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_25	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_109
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_26	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_110
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_27	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_111
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_28	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_112
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_29	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_113
	C7X256V1_CLEC_SOC_EVENTS_IN_IN_30	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_114

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_SOC_EVENTS_IN_IN_31	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_115
PDMA3	PDMA3_MCASP_MAIN_3_RX_IN_0	MCASP3_REC_DMA_EVENT_REQ_OUT_0
	PDMA3_MCASP_MAIN_3_TX_IN_0	MCASP3_XMIT_DMA_EVENT_REQ_OUT_0
	PDMA3_MCASP_MAIN_4_RX_IN_0	MCASP4_REC_DMA_EVENT_REQ_OUT_0
	PDMA3_MCASP_MAIN_4_TX_IN_0	MCASP4_XMIT_DMA_EVENT_REQ_OUT_0
PDMA0	PDMA0_MCANSS_MAIN_0_FE_IN_0	MCAN0_MCANSS_FE_OUT_0
	PDMA0_MCANSS_MAIN_0_FE_IN_1	MCAN0_MCANSS_FE_OUT_1
	PDMA0_MCANSS_MAIN_0_FE_IN_2	MCAN0_MCANSS_FE_OUT_2
	PDMA0_MCANSS_MAIN_0_TX_IN_0	MCAN0_MCANSS_TX_DMA_OUT_0
	PDMA0_MCANSS_MAIN_0_TX_IN_1	MCAN0_MCANSS_TX_DMA_OUT_1
	PDMA0_MCANSS_MAIN_0_TX_IN_2	MCAN0_MCANSS_TX_DMA_OUT_2
	PDMA0_MCANSS_MAIN_1_FE_IN_0	MCAN1_MCANSS_FE_OUT_0
	PDMA0_MCANSS_MAIN_1_FE_IN_1	MCAN1_MCANSS_FE_OUT_1
	PDMA0_MCANSS_MAIN_1_FE_IN_2	MCAN1_MCANSS_FE_OUT_2
	PDMA0_MCANSS_MAIN_1_TX_IN_0	MCAN1_MCANSS_TX_DMA_OUT_0
	PDMA0_MCANSS_MAIN_1_TX_IN_1	MCAN1_MCANSS_TX_DMA_OUT_1
	PDMA0_MCANSS_MAIN_1_TX_IN_2	MCAN1_MCANSS_TX_DMA_OUT_2
	PDMA0_SPI_MAIN_0_RX_IN_0	MCSPI0_DMA_READ_EVENT_OUT_0
	PDMA0_SPI_MAIN_0_RX_IN_1	MCSPI0_DMA_READ_EVENT_OUT_1
	PDMA0_SPI_MAIN_0_RX_IN_2	MCSPI0_DMA_READ_EVENT_OUT_2
	PDMA0_SPI_MAIN_0_RX_IN_3	MCSPI0_DMA_READ_EVENT_OUT_3
	PDMA0_SPI_MAIN_0_TX_IN_0	MCSPI0_DMA_WRITE_EVENT_OUT_0
	PDMA0_SPI_MAIN_0_TX_IN_1	MCSPI0_DMA_WRITE_EVENT_OUT_1
	PDMA0_SPI_MAIN_0_TX_IN_2	MCSPI0_DMA_WRITE_EVENT_OUT_2
	PDMA0_SPI_MAIN_0_TX_IN_3	MCSPI0_DMA_WRITE_EVENT_OUT_3
	PDMA0_SPI_MAIN_1_RX_IN_0	MCSPI1_DMA_READ_EVENT_OUT_0
	PDMA0_SPI_MAIN_1_RX_IN_1	MCSPI1_DMA_READ_EVENT_OUT_1
	PDMA0_SPI_MAIN_1_RX_IN_2	MCSPI1_DMA_READ_EVENT_OUT_2
	PDMA0_SPI_MAIN_1_RX_IN_3	MCSPI1_DMA_READ_EVENT_OUT_3
	PDMA0_SPI_MAIN_1_TX_IN_0	MCSPI1_DMA_WRITE_EVENT_OUT_0
	PDMA0_SPI_MAIN_1_TX_IN_1	MCSPI1_DMA_WRITE_EVENT_OUT_1
	PDMA0_SPI_MAIN_1_TX_IN_2	MCSPI1_DMA_WRITE_EVENT_OUT_2
	PDMA0_SPI_MAIN_1_TX_IN_3	MCSPI1_DMA_WRITE_EVENT_OUT_3

Copyright © 2026 Texas Instruments Incorporated

Table 10-4. Interrupts (Inputs) (continued)

Instance	Interrupt Input Line	Source Interrupt
PDMA0	PDMA0_SPI_MAIN_2_RX_IN_0	MCSPI2_DMA_READ_EVENT_OUT_0
	PDMA0_SPI_MAIN_2_RX_IN_1	MCSPI2_DMA_READ_EVENT_OUT_1
	PDMA0_SPI_MAIN_2_RX_IN_2	MCSPI2_DMA_READ_EVENT_OUT_2
	PDMA0_SPI_MAIN_2_RX_IN_3	MCSPI2_DMA_READ_EVENT_OUT_3
	PDMA0_SPI_MAIN_2_TX_IN_0	MCSPI2_DMA_WRITE_EVENT_OUT_0
	PDMA0_SPI_MAIN_2_TX_IN_1	MCSPI2_DMA_WRITE_EVENT_OUT_1
	PDMA0_SPI_MAIN_2_TX_IN_2	MCSPI2_DMA_WRITE_EVENT_OUT_2
	PDMA0_SPI_MAIN_2_TX_IN_3	MCSPI2_DMA_WRITE_EVENT_OUT_3
USB0	USB0_TRACE_INEP_PKT_BUFF_AVAIL_IN_0	DEBUGSS0_DAVDMA_LEVEL_OUT_0

Copyright © 2026 Texas Instruments Incorporated

10.4.2 Interrupts (Outputs)

Table 10-5. Interrupts (Outputs)

Instance	Interrupt Output Line	Destination Interrupt
hfosc0_clkloss_detect_circuit	HFOSC0_CLKLOSS_DETECT_CIRCUIT_REF_CLK_LOSS_DETECT_OUT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_13
	HFOSC0_CLKLOSS_DETECT_CIRCUIT_REF_CLK_LOSS_DETECT_OUT_OUT_0	ESM0_ESM_LVL_EVENT_IN_135
J7AEN_main_dbg_cbass0	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	J7AEN_MAIN_DBG_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
PINFUNCTION_CP_GEMAC_CPTS0_HW1TSPUSHin	PINFUNCTION_CP_GEMAC_CPTS0_HW1TSPUSHIN_CP_GEMAC_CPTS0_HW1TSPUSH_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_12
PINFUNCTION_CP_GEMAC_CPTS0_HW2TSPUSHin	PINFUNCTION_CP_GEMAC_CPTS0_HW2TSPUSHIN_CP_GEMAC_CPTS0_HW2TSPUSH_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_13
PINFUNCTION_EHRPWM0_SYNCIin	PINFUNCTION_EHRPWM0_SYNCIIN_EHRPWM0_SYNCI_OUT_0	EPWM0_EPWM_SYNCIN_IN_0
PINFUNCTION_EXTINTnin	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	GICSS0_SPI_IN_256
	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	R5FSS0_CORE0_INTR_IN_203
	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_203
	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_203
	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_256
	PINFUNCTION_EXTINTNIN_EXTINTN_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_256
CBASS_IPCSS0	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_IPCSS0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	CBASS_IPCSS0_DEFAULT_EXP_OUT_0	ESM0_ESM_LVL_EVENT_IN_110
CBASS_RT_CFG0	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_RT_CFG0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
CBASS_RT_DATA0	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
CBASS_RT_DATA0	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_RT_DATA0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
CBASS_AUDIO0	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_AUDIO0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
CBASS_CENTRAL2	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_CENTRAL2_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	CBASS_CENTRAL2_DEFAULT_EXP_OUT_0	ESM0_ESM_LVL_EVENT_IN_80
CBASS0	CBASS0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	CBASS0_DEFAULT_EXP_OUT_0	ESM0_ESM_LVL_EVENT_IN_86
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0	GICSS0_SPI_IN_32
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_32
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_32
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_32
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1	GICSS0_SPI_IN_33
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_33
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_33
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_33
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2	GICSS0_SPI_IN_34
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_34

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_34
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_34
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3	GICSS0_SPI_IN_35
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_35
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_35
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_35
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4	GICSS0_SPI_IN_36
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4	WKUP_R5FSS0_CORE0_INTR_IN_36
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_36
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_36
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5	GICSS0_SPI_IN_37
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5	WKUP_R5FSS0_CORE0_INTR_IN_37
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_37
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_37
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6	GICSS0_SPI_IN_38
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6	WKUP_R5FSS0_CORE0_INTR_IN_38
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6	C7X256V0_CLEC_GIC_SPI_IN_38
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_6	C7X256V1_CLEC_GIC_SPI_IN_38
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7	GICSS0_SPI_IN_39
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7	WKUP_R5FSS0_CORE0_INTR_IN_39
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7	C7X256V0_CLEC_GIC_SPI_IN_39
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_7	C7X256V1_CLEC_GIC_SPI_IN_39
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8	GICSS0_SPI_IN_40
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8	WKUP_R5FSS0_CORE0_INTR_IN_40
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8	C7X256V0_CLEC_GIC_SPI_IN_40
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_8	C7X256V1_CLEC_GIC_SPI_IN_40
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9	GICSS0_SPI_IN_41
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9	WKUP_R5FSS0_CORE0_INTR_IN_41
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9	C7X256V0_CLEC_GIC_SPI_IN_41
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_9	C7X256V1_CLEC_GIC_SPI_IN_41
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10	GICSS0_SPI_IN_42
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10	WKUP_R5FSS0_CORE0_INTR_IN_42
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10	C7X256V0_CLEC_GIC_SPI_IN_42

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_10	C7X256V1_CLEC_GIC_SPI_IN_42
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11	GICSS0_SPI_IN_43
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11	WKUP_R5FSS0_CORE0_INTR_IN_43
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11	C7X256V0_CLEC_GIC_SPI_IN_43
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_11	C7X256V1_CLEC_GIC_SPI_IN_43
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12	GICSS0_SPI_IN_44
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12	WKUP_R5FSS0_CORE0_INTR_IN_44
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12	C7X256V0_CLEC_GIC_SPI_IN_44
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_12	C7X256V1_CLEC_GIC_SPI_IN_44
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13	GICSS0_SPI_IN_45
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13	WKUP_R5FSS0_CORE0_INTR_IN_45
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13	C7X256V0_CLEC_GIC_SPI_IN_45
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_13	C7X256V1_CLEC_GIC_SPI_IN_45
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14	GICSS0_SPI_IN_46
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14	WKUP_R5FSS0_CORE0_INTR_IN_46
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14	C7X256V0_CLEC_GIC_SPI_IN_46
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_14	C7X256V1_CLEC_GIC_SPI_IN_46
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15	GICSS0_SPI_IN_47
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15	WKUP_R5FSS0_CORE0_INTR_IN_47
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15	C7X256V0_CLEC_GIC_SPI_IN_47
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_15	C7X256V1_CLEC_GIC_SPI_IN_47
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_16	R5FSS0_CORE0_INTR_IN_104
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_17	R5FSS0_CORE0_INTR_IN_105
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_18	R5FSS0_CORE0_INTR_IN_106
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_19	R5FSS0_CORE0_INTR_IN_107
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_20	R5FSS0_CORE0_INTR_IN_56
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_21	R5FSS0_CORE0_INTR_IN_57
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_22	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_24
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_23	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_25
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_24	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_16
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_25	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_17
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_26	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_18
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_27	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_19

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MAIN_GPIOMUX_INTROUTER0	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_28	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_20
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_29	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_21
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_30	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_22
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_31	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_23
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_32	R5FSS0_CORE0_INTR_IN_58
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_33	R5FSS0_CORE0_INTR_IN_59
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_34	MCU_R5FSS0_CORE0_CPU0_INTR_IN_32
	MAIN_GPIOMUX_INTROUTER0_OUTP_OUT_35	MCU_R5FSS0_CORE0_CPU0_INTR_IN_33
CBASS_INFRA1	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_INFRA1_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
CBASS_MCASP0	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_MCASP0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
CBASS_MISC_PERI0	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	CBASS_MISC_PERI0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
MCU_CBASS0	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_31

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_CBASS0	MCU_CBASS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_31
MCU_CTRL_MMR0	MCU_CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_0
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_39
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	GICSS0_SPI_IN_129
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_128
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_128
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_129
	MCU_CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_129
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0	GICSS0_SPI_IN_104
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_104
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_104
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_104
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1	GICSS0_SPI_IN_105
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_105
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_105
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_105
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2	GICSS0_SPI_IN_106
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_106
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_106
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_106
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3	GICSS0_SPI_IN_107
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_107
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_107
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_107
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_4	MCU_R5FSS0_CORE0_CPU0_INTR_IN_104
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_5	MCU_R5FSS0_CORE0_CPU0_INTR_IN_105
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_6	MCU_R5FSS0_CORE0_CPU0_INTR_IN_106
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_7	MCU_R5FSS0_CORE0_CPU0_INTR_IN_107
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8	WKUP_ESM0_ESM_PLS_EVENT0_IN_88
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8	WKUP_ESM0_ESM_PLS_EVENT1_IN_88
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_8	WKUP_ESM0_ESM_PLS_EVENT2_IN_88

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_MCU_GPIOMUX_INTROUTER0	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9	WKUP_ESM0_ESM_PLS_EVENT0_IN_89
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9	WKUP_ESM0_ESM_PLS_EVENT1_IN_89
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_9	WKUP_ESM0_ESM_PLS_EVENT2_IN_89
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10	WKUP_ESM0_ESM_PLS_EVENT0_IN_90
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10	WKUP_ESM0_ESM_PLS_EVENT1_IN_90
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_10	WKUP_ESM0_ESM_PLS_EVENT2_IN_90
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11	WKUP_ESM0_ESM_PLS_EVENT0_IN_91
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11	WKUP_ESM0_ESM_PLS_EVENT1_IN_91
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_11	WKUP_ESM0_ESM_PLS_EVENT2_IN_91
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_12	R5FSS0_CORE0_INTR_IN_32
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_13	R5FSS0_CORE0_INTR_IN_33
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_14	R5FSS0_CORE0_INTR_IN_60
	WKUP_MCU_GPIOMUX_INTROUTER0_OUTP_OUT_15	R5FSS0_CORE0_INTR_IN_61
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_8
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_1	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_9
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_2	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_10
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_3	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_11
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_4	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_12
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_5	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_13
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_6	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_14
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_7	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_15
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_8	PCIE0_PCIE_CPTS_HW2_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_10	CPSW0_CPTS_HW1_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_11	CPSW0_CPTS_HW2_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_12	CPSW0_CPTS_HW3_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_13	CPSW0_CPTS_HW4_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_14	CPSW0_CPTS_HW5_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_15	CPSW0_CPTS_HW6_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_16	CPSW0_CPTS_HW7_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_17	CPSW0_CPTS_HW8_PUSH_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_19	EPWM0_EPWM_SYNCIN_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_20	PINFUNCTION_SYNC0_OUTOUT_SYNC0_OUT_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_21	PINFUNCTION_SYNC1_OUTOUT_SYNC1_OUT_IN_0

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
TIMESYNC_EVENT_INTROUTER0	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_22	PINFUNCTION_SYNC2_OUTOUT_SYNC2_OUT_IN_0
	TIMESYNC_EVENT_INTROUTER0_OUTL_OUT_23	PINFUNCTION_SYNC3_OUTOUT_SYNC3_OUT_IN_0
WKUP_CBASS0	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	WKUP_CBASS0_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	WKUP_CBASS0_DEFAULT_EXP_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_3
	WKUP_CBASS0_DEFAULT_EXP_OUT_0	ESM0_ESM_LVL_EVENT_IN_183
WKUP_CBASS_SAFE1	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	GICSS0_SPI_IN_133
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_147
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_147
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_147
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_133
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_133
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_31
	WKUP_CBASS_SAFE1_DEFAULT_ERR_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_31
PADCFG_CTRL0	PADCFG_CTRL0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_39
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	GICSS0_SPI_IN_129
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_128
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_128
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_129
	PADCFG_CTRL0_ACCESS_ERR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_129
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_39
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	GICSS0_SPI_IN_129
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_128
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_128
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128
	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_129

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_PADCFG_CTRL0	MCU_PADCFG_CTRL0_ACCESS_ERR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_129
CPSW0	CPSW0_CPTS_COMP_OUT_0	EPWM0_EPWM_SYNCIN_IN_0
	CPSW0_CPTS_COMP_OUT_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_COMPOUT_CP_GEMAC_CPTS0_TS_COMP_IN_0
	CPSW0_CPTS_COMP_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_0
	CPSW0_CPTS_COMP_OUT_0	GICSS0_SPI_IN_48
	CPSW0_CPTS_COMP_OUT_0	R5FSS0_CORE0_INTR_IN_48
	CPSW0_CPTS_COMP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_48
	CPSW0_CPTS_COMP_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_48
	CPSW0_CPTS_COMP_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_48
	CPSW0_CPTS_COMP_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_48
	CPSW0_CPTS_GENF0_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_16
	CPSW0_CPTS_GENF1_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_17
	CPSW0_CPTS_SYNC_OUT_0	PINFUNCTION_CP_GEMAC_CPTS0_TS_SYNCOUT_CP_GEMAC_CPTS0_TS_SYNC_IN_0
	CPSW0_CPTS_SYNC_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_18
	CPSW0_ECC_DED_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_67
	CPSW0_ECC_SEC_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_3
	CPSW0_EVNT_PEND_OUT_0	GICSS0_SPI_IN_134
	CPSW0_EVNT_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_134
	CPSW0_EVNT_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_134
	CPSW0_EVNT_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_134
	CPSW0_EVNT_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_134
	CPSW0_EVNT_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_134
	CPSW0_MDIO_PEND_OUT_0	GICSS0_SPI_IN_135
	CPSW0_MDIO_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_135
	CPSW0_MDIO_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_135
	CPSW0_MDIO_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_135
	CPSW0_MDIO_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_135
	CPSW0_MDIO_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_135
	CPSW0_STAT_PEND_OUT_0	GICSS0_SPI_IN_136
	CPSW0_STAT_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_136
	CPSW0_STAT_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_136
	CPSW0_STAT_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_136
	CPSW0_STAT_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_136

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
CPSW0	CPSW0_STAT_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_136
csi_rx_if0	CSI_RX_IF0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_66
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_0
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	GICSS0_SPI_IN_175
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_170
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_170
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_170
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_175
	CSI_RX_IF0_CSI_ERR_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_175
	CSI_RX_IF0_CSI_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_70
	CSI_RX_IF0_CSI_IRQ_OUT_0	GICSS0_SPI_IN_173
	CSI_RX_IF0_CSI_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_173
	CSI_RX_IF0_CSI_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_173
	CSI_RX_IF0_CSI_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_173
	CSI_RX_IF0_CSI_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_173
	CSI_RX_IF0_CSI_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_173
	CSI_RX_IF0_CSI_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_72
	CSI_RX_IF0_CSI_LEVEL_OUT_0	GICSS0_SPI_IN_174
	CSI_RX_IF0_CSI_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_174
	CSI_RX_IF0_CSI_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_174
	CSI_RX_IF0_CSI_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_174
	CSI_RX_IF0_CSI_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_174
	CSI_RX_IF0_CSI_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_174
	CSI_RX_IF0_CSI_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_71
	CSI_RX_IF0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_77
csi_rx_if1	CSI_RX_IF1_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_191
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_190
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	GICSS0_SPI_IN_178
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_172
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_172
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_172
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_178
	CSI_RX_IF1_CSI_ERR_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_178

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
csi_rx_if1	CSI_RX_IF1_CSI_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_189
	CSI_RX_IF1_CSI_IRQ_OUT_0	GICSS0_SPI_IN_179
	CSI_RX_IF1_CSI_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_176
	CSI_RX_IF1_CSI_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_176
	CSI_RX_IF1_CSI_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_176
	CSI_RX_IF1_CSI_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_179
	CSI_RX_IF1_CSI_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_179
	CSI_RX_IF1_CSI_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_188
	CSI_RX_IF1_CSI_LEVEL_OUT_0	GICSS0_SPI_IN_200
	CSI_RX_IF1_CSI_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_209
	CSI_RX_IF1_CSI_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_209
	CSI_RX_IF1_CSI_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_209
	CSI_RX_IF1_CSI_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_200
	CSI_RX_IF1_CSI_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_200
	CSI_RX_IF1_CSI_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_187
	CSI_RX_IF1_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_122
csi_rx_if2	CSI_RX_IF2_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_214
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_206
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	GICSS0_SPI_IN_219
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_198
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_198
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_219
	CSI_RX_IF2_CSI_ERR_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_219
	CSI_RX_IF2_CSI_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_200
	CSI_RX_IF2_CSI_IRQ_OUT_0	GICSS0_SPI_IN_232
	CSI_RX_IF2_CSI_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_199
	CSI_RX_IF2_CSI_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_199
	CSI_RX_IF2_CSI_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_232
	CSI_RX_IF2_CSI_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_232
	CSI_RX_IF2_CSI_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_31
	CSI_RX_IF2_CSI_LEVEL_OUT_0	GICSS0_SPI_IN_236
	CSI_RX_IF2_CSI_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_200
	CSI_RX_IF2_CSI_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_200

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
csi_rx_if2	CSI_RX_IF2_CSI_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_236
	CSI_RX_IF2_CSI_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_236
	CSI_RX_IF2_CSI_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_201
	CSI_RX_IF2_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_215
csi_rx_if3	CSI_RX_IF3_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_216
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_222
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	GICSS0_SPI_IN_249
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_155
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_155
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_249
	CSI_RX_IF3_CSI_ERR_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_249
	CSI_RX_IF3_CSI_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_204
	CSI_RX_IF3_CSI_IRQ_OUT_0	GICSS0_SPI_IN_250
	CSI_RX_IF3_CSI_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_156
	CSI_RX_IF3_CSI_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_156
	CSI_RX_IF3_CSI_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_250
	CSI_RX_IF3_CSI_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_250
	CSI_RX_IF3_CSI_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_41
	CSI_RX_IF3_CSI_LEVEL_OUT_0	GICSS0_SPI_IN_251
	CSI_RX_IF3_CSI_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_157
	CSI_RX_IF3_CSI_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_157
	CSI_RX_IF3_CSI_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_251
	CSI_RX_IF3_CSI_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_251
	CSI_RX_IF3_CSI_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_205
	CSI_RX_IF3_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_217
CSI_TX_IF0	CSI_TX_IF0_CDNS_RAM_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_123
	CSI_TX_IF0_CDNS_RAM_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_125
	CSI_TX_IF0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_126
	CSI_TX_IF0_CSI_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_186
	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	GICSS0_SPI_IN_278
	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	R5FSS0_CORE0_INTR_IN_50
	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_50
	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_50

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
CSI_TX_IF0	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_278
	CSI_TX_IF0_CSI_INTERRUPT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_278
	CSI_TX_IF0_CSI_LEVEL_OUT_0	GICSS0_SPI_IN_279
	CSI_TX_IF0_CSI_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_51
	CSI_TX_IF0_CSI_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_51
	CSI_TX_IF0_CSI_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_51
	CSI_TX_IF0_CSI_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_279
	CSI_TX_IF0_CSI_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_279
	CSI_TX_IF0_CSI_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_185
	CSI_TX_IF0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_127
DCC0	DCC0_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC0_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC0_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC0_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC0_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC0_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC0_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_112
DCC1	DCC1_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC1_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC1_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC1_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC1_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC1_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC1_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_113
DCC2	DCC2_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC2_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC2_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC2_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC2_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC2_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC2_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_114
DCC3	DCC3_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC3_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DCC3	DCC3_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC3_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC3_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC3_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC3_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_115
DCC4	DCC4_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC4_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC4_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC4_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC4_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC4_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC4_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_116
DCC5	DCC5_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC5_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC5_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC5_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC5_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC5_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC5_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_117
DCC6	DCC6_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC6_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC6_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC6_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC6_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC6_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC6_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_79
DCC7	DCC7_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC7_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC7_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC7_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC7_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC7_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC7_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_73

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DCC8	DCC8_INTR_DONE_LEVEL_OUT_0	GICSS0_SPI_IN_128
	DCC8_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_109
	DCC8_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_109
	DCC8_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_109
	DCC8_INTR_DONE_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_128
	DCC8_INTR_DONE_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_128
	DCC8_INTR_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_223
MCU_DCC0	MCU_DCC0_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_108
	MCU_DCC0_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_108
	MCU_DCC0_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_108
	MCU_DCC0_INTR_ERR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_37
MCU_DCC1	MCU_DCC1_INTR_DONE_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_137
	MCU_DCC1_INTR_DONE_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_137
	MCU_DCC1_INTR_DONE_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_137
	MCU_DCC1_INTR_ERR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_36
DMASS0_ECC_AGGR_0	DMASS0_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_9
	DMASS0_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_10
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	GICSS0_SPI_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	GICSS0_SPI_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	GICSS0_SPI_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	GICSS0_SPI_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	GICSS0_SPI_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	GICSS0_SPI_IN_69

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_69
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_69
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	GICSS0_SPI_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	C7X256V0_CLEC_GIC_SPI_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	C7X256V1_CLEC_GIC_SPI_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	GICSS0_SPI_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	C7X256V0_CLEC_GIC_SPI_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	C7X256V1_CLEC_GIC_SPI_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8	GICSS0_SPI_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8	C7X256V0_CLEC_GIC_SPI_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8	C7X256V1_CLEC_GIC_SPI_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9	GICSS0_SPI_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9	C7X256V0_CLEC_GIC_SPI_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9	C7X256V1_CLEC_GIC_SPI_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10	GICSS0_SPI_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10	C7X256V0_CLEC_GIC_SPI_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10	C7X256V1_CLEC_GIC_SPI_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11	GICSS0_SPI_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11	C7X256V0_CLEC_GIC_SPI_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11	C7X256V1_CLEC_GIC_SPI_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12	GICSS0_SPI_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12	C7X256V0_CLEC_GIC_SPI_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12	C7X256V1_CLEC_GIC_SPI_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13	GICSS0_SPI_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13	C7X256V0_CLEC_GIC_SPI_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13	C7X256V1_CLEC_GIC_SPI_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14	GICSS0_SPI_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14	C7X256V0_CLEC_GIC_SPI_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14	C7X256V1_CLEC_GIC_SPI_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15	GICSS0_SPI_IN_79
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15	C7X256V0_CLEC_GIC_SPI_IN_79
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15	C7X256V1_CLEC_GIC_SPI_IN_79
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16	GICSS0_SPI_IN_80

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16	C7X256V0_CLEC_GIC_SPI_IN_80
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16	C7X256V1_CLEC_GIC_SPI_IN_80
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17	GICSS0_SPI_IN_81
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17	C7X256V0_CLEC_GIC_SPI_IN_81
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17	C7X256V1_CLEC_GIC_SPI_IN_81
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18	GICSS0_SPI_IN_82
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18	C7X256V0_CLEC_GIC_SPI_IN_82
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18	C7X256V1_CLEC_GIC_SPI_IN_82
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19	GICSS0_SPI_IN_83
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19	C7X256V0_CLEC_GIC_SPI_IN_83
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19	C7X256V1_CLEC_GIC_SPI_IN_83
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20	GICSS0_SPI_IN_84
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20	C7X256V0_CLEC_GIC_SPI_IN_84
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20	C7X256V1_CLEC_GIC_SPI_IN_84
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21	GICSS0_SPI_IN_85
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21	C7X256V0_CLEC_GIC_SPI_IN_85
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21	C7X256V1_CLEC_GIC_SPI_IN_85
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22	GICSS0_SPI_IN_86
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22	C7X256V0_CLEC_GIC_SPI_IN_86
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22	C7X256V1_CLEC_GIC_SPI_IN_86
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23	GICSS0_SPI_IN_87
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23	C7X256V0_CLEC_GIC_SPI_IN_87
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23	C7X256V1_CLEC_GIC_SPI_IN_87
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24	GICSS0_SPI_IN_88
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24	C7X256V0_CLEC_GIC_SPI_IN_88
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24	C7X256V1_CLEC_GIC_SPI_IN_88
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25	GICSS0_SPI_IN_89
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25	C7X256V0_CLEC_GIC_SPI_IN_89
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25	C7X256V1_CLEC_GIC_SPI_IN_89
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26	GICSS0_SPI_IN_90
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26	C7X256V0_CLEC_GIC_SPI_IN_90
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26	C7X256V1_CLEC_GIC_SPI_IN_90
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27	GICSS0_SPI_IN_91

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27	C7X256V0_CLEC_GIC_SPI_IN_91
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27	C7X256V1_CLEC_GIC_SPI_IN_91
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28	GICSS0_SPI_IN_92
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28	C7X256V0_CLEC_GIC_SPI_IN_92
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28	C7X256V1_CLEC_GIC_SPI_IN_92
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29	GICSS0_SPI_IN_93
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29	C7X256V0_CLEC_GIC_SPI_IN_93
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29	C7X256V1_CLEC_GIC_SPI_IN_93
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30	GICSS0_SPI_IN_94
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30	C7X256V0_CLEC_GIC_SPI_IN_94
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30	C7X256V1_CLEC_GIC_SPI_IN_94
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31	GICSS0_SPI_IN_95
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31	C7X256V0_CLEC_GIC_SPI_IN_95
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31	C7X256V1_CLEC_GIC_SPI_IN_95
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32	GICSS0_SPI_IN_96
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32	C7X256V0_CLEC_GIC_SPI_IN_96
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32	C7X256V1_CLEC_GIC_SPI_IN_96
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33	GICSS0_SPI_IN_97
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33	C7X256V0_CLEC_GIC_SPI_IN_97
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33	C7X256V1_CLEC_GIC_SPI_IN_97
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34	GICSS0_SPI_IN_98
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34	C7X256V0_CLEC_GIC_SPI_IN_98
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34	C7X256V1_CLEC_GIC_SPI_IN_98
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35	GICSS0_SPI_IN_99
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35	C7X256V0_CLEC_GIC_SPI_IN_99
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35	C7X256V1_CLEC_GIC_SPI_IN_99
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36	GICSS0_SPI_IN_100
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36	C7X256V0_CLEC_GIC_SPI_IN_100
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36	C7X256V1_CLEC_GIC_SPI_IN_100
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37	GICSS0_SPI_IN_101
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37	C7X256V0_CLEC_GIC_SPI_IN_101
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37	C7X256V1_CLEC_GIC_SPI_IN_101
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38	GICSS0_SPI_IN_102

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38	C7X256V0_CLEC_GIC_SPI_IN_102
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38	C7X256V1_CLEC_GIC_SPI_IN_102
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39	GICSS0_SPI_IN_103
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39	C7X256V0_CLEC_GIC_SPI_IN_103
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39	C7X256V1_CLEC_GIC_SPI_IN_103
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_40	WKUP_R5FSS0_CORE0_INTR_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_41	WKUP_R5FSS0_CORE0_INTR_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_42	WKUP_R5FSS0_CORE0_INTR_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_43	WKUP_R5FSS0_CORE0_INTR_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_44	WKUP_R5FSS0_CORE0_INTR_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_45	WKUP_R5FSS0_CORE0_INTR_IN_69
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_46	WKUP_R5FSS0_CORE0_INTR_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_47	WKUP_R5FSS0_CORE0_INTR_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_48	WKUP_R5FSS0_CORE0_INTR_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_49	WKUP_R5FSS0_CORE0_INTR_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_50	WKUP_R5FSS0_CORE0_INTR_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_51	WKUP_R5FSS0_CORE0_INTR_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_52	WKUP_R5FSS0_CORE0_INTR_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_53	WKUP_R5FSS0_CORE0_INTR_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_54	WKUP_R5FSS0_CORE0_INTR_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_55	WKUP_R5FSS0_CORE0_INTR_IN_79
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_56	WKUP_R5FSS0_CORE0_INTR_IN_80
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_57	WKUP_R5FSS0_CORE0_INTR_IN_81
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_58	WKUP_R5FSS0_CORE0_INTR_IN_82
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_59	WKUP_R5FSS0_CORE0_INTR_IN_83
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_60	WKUP_R5FSS0_CORE0_INTR_IN_84
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_61	WKUP_R5FSS0_CORE0_INTR_IN_85
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_62	WKUP_R5FSS0_CORE0_INTR_IN_86
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_63	WKUP_R5FSS0_CORE0_INTR_IN_87
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_64	WKUP_R5FSS0_CORE0_INTR_IN_88
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_65	WKUP_R5FSS0_CORE0_INTR_IN_89
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_66	WKUP_R5FSS0_CORE0_INTR_IN_90
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_67	WKUP_R5FSS0_CORE0_INTR_IN_91

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_68	WKUP_R5FSS0_CORE0_INTR_IN_92
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_69	WKUP_R5FSS0_CORE0_INTR_IN_93
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_70	WKUP_R5FSS0_CORE0_INTR_IN_94
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_71	WKUP_R5FSS0_CORE0_INTR_IN_95
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_72	WKUP_R5FSS0_CORE0_INTR_IN_8
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_73	WKUP_R5FSS0_CORE0_INTR_IN_9
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_74	WKUP_R5FSS0_CORE0_INTR_IN_10
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_75	WKUP_R5FSS0_CORE0_INTR_IN_11
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_76	WKUP_R5FSS0_CORE0_INTR_IN_12
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_77	WKUP_R5FSS0_CORE0_INTR_IN_13
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_78	WKUP_R5FSS0_CORE0_INTR_IN_14
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_79	WKUP_R5FSS0_CORE0_INTR_IN_15
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_84	C7X256V0_CLEC_SOC_EVENTS_IN_IN_16
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_85	C7X256V0_CLEC_SOC_EVENTS_IN_IN_17
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_86	C7X256V0_CLEC_SOC_EVENTS_IN_IN_18
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_87	C7X256V0_CLEC_SOC_EVENTS_IN_IN_19
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_88	C7X256V0_CLEC_SOC_EVENTS_IN_IN_20
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_89	C7X256V0_CLEC_SOC_EVENTS_IN_IN_21
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_90	C7X256V0_CLEC_SOC_EVENTS_IN_IN_22
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_91	C7X256V0_CLEC_SOC_EVENTS_IN_IN_23
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_92	C7X256V0_CLEC_SOC_EVENTS_IN_IN_24
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_93	C7X256V0_CLEC_SOC_EVENTS_IN_IN_25
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_94	C7X256V0_CLEC_SOC_EVENTS_IN_IN_26
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_95	C7X256V0_CLEC_SOC_EVENTS_IN_IN_27
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_96	C7X256V0_CLEC_SOC_EVENTS_IN_IN_28
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_97	C7X256V0_CLEC_SOC_EVENTS_IN_IN_29
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_98	C7X256V0_CLEC_SOC_EVENTS_IN_IN_30
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_99	C7X256V0_CLEC_SOC_EVENTS_IN_IN_31
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_100	C7X256V1_CLEC_SOC_EVENTS_IN_IN_16
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_101	C7X256V1_CLEC_SOC_EVENTS_IN_IN_17
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_102	C7X256V1_CLEC_SOC_EVENTS_IN_IN_18
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_103	C7X256V1_CLEC_SOC_EVENTS_IN_IN_19
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_104	C7X256V1_CLEC_SOC_EVENTS_IN_IN_20

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_105	C7X256V1_CLEC_SOC_EVENTS_IN_IN_21
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_106	C7X256V1_CLEC_SOC_EVENTS_IN_IN_22
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_107	C7X256V1_CLEC_SOC_EVENTS_IN_IN_23
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_108	C7X256V1_CLEC_SOC_EVENTS_IN_IN_24
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_109	C7X256V1_CLEC_SOC_EVENTS_IN_IN_25
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_110	C7X256V1_CLEC_SOC_EVENTS_IN_IN_26
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_111	C7X256V1_CLEC_SOC_EVENTS_IN_IN_27
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_112	C7X256V1_CLEC_SOC_EVENTS_IN_IN_28
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_113	C7X256V1_CLEC_SOC_EVENTS_IN_IN_29
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_114	C7X256V1_CLEC_SOC_EVENTS_IN_IN_30
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_115	C7X256V1_CLEC_SOC_EVENTS_IN_IN_31
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_152	R5FSS0_CORE0_INTR_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_153	R5FSS0_CORE0_INTR_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_154	R5FSS0_CORE0_INTR_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_155	R5FSS0_CORE0_INTR_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_156	R5FSS0_CORE0_INTR_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_157	R5FSS0_CORE0_INTR_IN_69
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_158	R5FSS0_CORE0_INTR_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_159	R5FSS0_CORE0_INTR_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_160	R5FSS0_CORE0_INTR_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_161	R5FSS0_CORE0_INTR_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_162	R5FSS0_CORE0_INTR_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_163	R5FSS0_CORE0_INTR_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_164	R5FSS0_CORE0_INTR_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_165	R5FSS0_CORE0_INTR_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_166	R5FSS0_CORE0_INTR_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_167	R5FSS0_CORE0_INTR_IN_79
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_168	MCU_R5FSS0_CORE0_CPU0_INTR_IN_64
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_169	MCU_R5FSS0_CORE0_CPU0_INTR_IN_65
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_170	MCU_R5FSS0_CORE0_CPU0_INTR_IN_66
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_171	MCU_R5FSS0_CORE0_CPU0_INTR_IN_67
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_172	MCU_R5FSS0_CORE0_CPU0_INTR_IN_68
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_173	MCU_R5FSS0_CORE0_CPU0_INTR_IN_69

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS0_INTAGGR_0	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_174	MCU_R5FSS0_CORE0_CPU0_INTR_IN_70
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_175	MCU_R5FSS0_CORE0_CPU0_INTR_IN_71
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_176	MCU_R5FSS0_CORE0_CPU0_INTR_IN_72
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_177	MCU_R5FSS0_CORE0_CPU0_INTR_IN_73
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_178	MCU_R5FSS0_CORE0_CPU0_INTR_IN_74
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_179	MCU_R5FSS0_CORE0_CPU0_INTR_IN_75
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_180	MCU_R5FSS0_CORE0_CPU0_INTR_IN_76
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_181	MCU_R5FSS0_CORE0_CPU0_INTR_IN_77
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_182	MCU_R5FSS0_CORE0_CPU0_INTR_IN_78
	DMASS0_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_183	MCU_R5FSS0_CORE0_CPU0_INTR_IN_79
DMASS1_ECC_AGGR_0	DMASS1_ECC_AGGR_0_ECC_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_63
	DMASS1_ECC_AGGR_0_ECC_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_62
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	GICSS0_SPI_IN_237
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_237
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_237
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	GICSS0_SPI_IN_238
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_238
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_238
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	GICSS0_SPI_IN_239
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_239
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_239
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	GICSS0_SPI_IN_240
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_240
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_240
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	GICSS0_SPI_IN_241
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_241
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_241
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	GICSS0_SPI_IN_242
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_242
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_242
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	GICSS0_SPI_IN_243
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	C7X256V0_CLEC_GIC_SPI_IN_243
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_6	C7X256V1_CLEC_GIC_SPI_IN_243

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	GICSS0_SPI_IN_244
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	C7X256V0_CLEC_GIC_SPI_IN_244
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_7	C7X256V1_CLEC_GIC_SPI_IN_244
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_8	WKUP_R5FSS0_CORE0_INTR_IN_129
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_9	WKUP_R5FSS0_CORE0_INTR_IN_130
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_10	WKUP_R5FSS0_CORE0_INTR_IN_131
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_11	WKUP_R5FSS0_CORE0_INTR_IN_132
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_12	WKUP_R5FSS0_CORE0_INTR_IN_150
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_13	WKUP_R5FSS0_CORE0_INTR_IN_158
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14	R5FSS0_CORE0_INTR_IN_158
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_14	WKUP_R5FSS0_CORE0_INTR_IN_159
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15	R5FSS0_CORE0_INTR_IN_159
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_15	WKUP_R5FSS0_CORE0_INTR_IN_160
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_16	MCU_R5FSS0_CORE0_CPU0_INTR_IN_129
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_17	MCU_R5FSS0_CORE0_CPU0_INTR_IN_130
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_18	MCU_R5FSS0_CORE0_CPU0_INTR_IN_131
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_19	MCU_R5FSS0_CORE0_CPU0_INTR_IN_132
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_20	MCU_R5FSS0_CORE0_CPU0_INTR_IN_150
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_21	MCU_R5FSS0_CORE0_CPU0_INTR_IN_158
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22	R5FSS0_CORE0_INTR_IN_160
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_22	MCU_R5FSS0_CORE0_CPU0_INTR_IN_159
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23	R5FSS0_CORE0_INTR_IN_150
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_23	MCU_R5FSS0_CORE0_CPU0_INTR_IN_160
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_24	R5FSS0_CORE0_INTR_IN_129
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_25	R5FSS0_CORE0_INTR_IN_130
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_26	R5FSS0_CORE0_INTR_IN_131
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_27	R5FSS0_CORE0_INTR_IN_132
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_28	R5FSS0_CORE0_INTR_IN_138
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_29	R5FSS0_CORE0_INTR_IN_139
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_30	R5FSS0_CORE0_INTR_IN_143
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_31	R5FSS0_CORE0_INTR_IN_144
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_32	R5FSS0_CORE0_INTR_IN_8
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_33	R5FSS0_CORE0_INTR_IN_9

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMASS1_INTAGGR_0	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_34	R5FSS0_CORE0_INTR_IN_10
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_35	R5FSS0_CORE0_INTR_IN_11
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_36	R5FSS0_CORE0_INTR_IN_12
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_37	R5FSS0_CORE0_INTR_IN_13
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_38	R5FSS0_CORE0_INTR_IN_14
	DMASS1_INTAGGR_0_INTAGGR_VINTR_PEND_OUT_39	R5FSS0_CORE0_INTR_IN_15
TIMER0	TIMER0_INTR_PEND_OUT_0	GICSS0_SPI_IN_152
	TIMER0_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_24
	TIMER0_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_152
	TIMER0_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_152
	TIMER0_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_162
	TIMER0_TIMER_PWM_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_0
TIMER1	TIMER1_INTR_PEND_OUT_0	GICSS0_SPI_IN_153
	TIMER1_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_25
	TIMER1_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_153
	TIMER1_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_153
	TIMER1_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_163
	TIMER1_TIMER_PWM_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_1
TIMER2	TIMER2_INTR_PEND_OUT_0	GICSS0_SPI_IN_154
	TIMER2_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_26
	TIMER2_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_154
	TIMER2_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_154
	TIMER2_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_164
	TIMER2_TIMER_PWM_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_2
TIMER3	TIMER3_INTR_PEND_OUT_0	GICSS0_SPI_IN_155
	TIMER3_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_27
	TIMER3_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_155
	TIMER3_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_155
	TIMER3_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_165
	TIMER3_TIMER_PWM_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_3
TIMER4	TIMER4_INTR_PEND_OUT_0	GICSS0_SPI_IN_156
	TIMER4_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_28
	TIMER4_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_156

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
TIMER4	TIMER4_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_156
	TIMER4_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_166
TIMER5	TIMER5_INTR_PEND_OUT_0	GICSS0_SPI_IN_157
	TIMER5_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_29
	TIMER5_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_157
	TIMER5_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_157
	TIMER5_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_167
TIMER6	TIMER6_INTR_PEND_OUT_0	GICSS0_SPI_IN_158
	TIMER6_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_34
	TIMER6_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_158
	TIMER6_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_158
	TIMER6_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_168
TIMER7	TIMER7_INTR_PEND_OUT_0	GICSS0_SPI_IN_159
	TIMER7_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_35
	TIMER7_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_159
	TIMER7_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_159
	TIMER7_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_169
MCU_TIMER0	MCU_TIMER0_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_28
	MCU_TIMER0_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_170
MCU_TIMER1	MCU_TIMER1_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_29
	MCU_TIMER1_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_171
MCU_TIMER2	MCU_TIMER2_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_138
	MCU_TIMER2_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_172
MCU_TIMER3	MCU_TIMER3_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_139
	MCU_TIMER3_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_173
WKUP_TIMER0	WKUP_TIMER0_INTR_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_138
	WKUP_TIMER0_TIMER_CLKSTOP_WAKEUP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_28
	WKUP_TIMER0_TIMER_CLKSTOP_WAKEUP_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_5
	WKUP_TIMER0_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_174
WKUP_TIMER1	WKUP_TIMER1_INTR_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_139
	WKUP_TIMER1_TIMER_CLKSTOP_WAKEUP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_29
	WKUP_TIMER1_TIMER_CLKSTOP_WAKEUP_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_6
	WKUP_TIMER1_TIMER_PWM_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_175

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
ECAP0	ECAP0_ECAP_INT_OUT_0	GICSS0_SPI_IN_145
	ECAP0_ECAP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_83
	ECAP0_ECAP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_83
	ECAP0_ECAP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_145
	ECAP0_ECAP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_145
ECAP1	ECAP1_ECAP_INT_OUT_0	GICSS0_SPI_IN_146
	ECAP1_ECAP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_84
	ECAP1_ECAP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_84
	ECAP1_ECAP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_146
	ECAP1_ECAP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_146
ECAP2	ECAP2_ECAP_INT_OUT_0	GICSS0_SPI_IN_147
	ECAP2_ECAP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_85
	ECAP2_ECAP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_85
	ECAP2_ECAP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_147
	ECAP2_ECAP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_147
ELM0	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	GICSS0_SPI_IN_164
	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_164
	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_164
	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_164
	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_164
	ELM0_ELM_POROCPSINTERRUPT_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_164
MMCSD0	MMCSD0_EMMCSS_INTR_OUT_0	GICSS0_SPI_IN_165
	MMCSD0_EMMCSS_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_161
	MMCSD0_EMMCSS_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_161
	MMCSD0_EMMCSS_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_161
	MMCSD0_EMMCSS_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_165
	MMCSD0_EMMCSS_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_165
	MMCSD0_EMMCSS_RXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_54
	MMCSD0_EMMCSS_RXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_55
	MMCSD0_EMMCSS_TXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_56
	MMCSD0_EMMCSS_TXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_57
MMCSD1	MMCSD1_EMMCSDSS_INTR_OUT_0	GICSS0_SPI_IN_115
	MMCSD1_EMMCSDSS_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_162

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MMCSD1	MMCSD1_EMMCSDSS_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_162
	MMCSD1_EMMCSDSS_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_162
	MMCSD1_EMMCSDSS_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_115
	MMCSD1_EMMCSDSS_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_115
	MMCSD1_EMMCSDSS_RXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_58
	MMCSD1_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_59
	MMCSD1_EMMCSDSS_TXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_60
	MMCSD1_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_61
MMCSD2	MMCSD2_EMMCSDSS_INTR_OUT_0	GICSS0_SPI_IN_114
	MMCSD2_EMMCSDSS_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_163
	MMCSD2_EMMCSDSS_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_163
	MMCSD2_EMMCSDSS_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_163
	MMCSD2_EMMCSDSS_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_114
	MMCSD2_EMMCSDSS_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_114
	MMCSD2_EMMCSDSS_RXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_34
	MMCSD2_EMMCSDSS_RXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_36
	MMCSD2_EMMCSDSS_TXMEM_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_49
	MMCSD2_EMMCSDSS_TXMEM_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_65
EQEP0	EQEP0_EQEP_INT_OUT_0	GICSS0_SPI_IN_148
	EQEP0_EQEP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_86
	EQEP0_EQEP_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_244
	EQEP0_EQEP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_86
	EQEP0_EQEP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_148
	EQEP0_EQEP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_148
EQEP1	EQEP1_EQEP_INT_OUT_0	GICSS0_SPI_IN_149
	EQEP1_EQEP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_87
	EQEP1_EQEP_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_245
	EQEP1_EQEP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_87
	EQEP1_EQEP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_149
	EQEP1_EQEP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_149
EQEP2	EQEP2_EQEP_INT_OUT_0	GICSS0_SPI_IN_150
	EQEP2_EQEP_INT_OUT_0	R5FSS0_CORE0_INTR_IN_88
	EQEP2_EQEP_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_246

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
EQEP2	EQEP2_EQEP_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_88
	EQEP2_EQEP_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_150
	EQEP2_EQEP_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_150
WKUP_ESM0	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_37
	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_140
	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_140
	WKUP_ESM0_ESM_INT_CFG_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_140
	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_38
	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_141
	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_141
	WKUP_ESM0_ESM_INT_HI_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_141
	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_39
	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_142
	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_142
	WKUP_ESM0_ESM_INT_LOW_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_142
ESM0	ESM0_ESM_INT_CFG_LVL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_0
	ESM0_ESM_INT_CFG_LVL_OUT_0	GICSS0_SPI_IN_180
	ESM0_ESM_INT_CFG_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_167
	ESM0_ESM_INT_CFG_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_167
	ESM0_ESM_INT_CFG_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_167
	ESM0_ESM_INT_CFG_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_180
	ESM0_ESM_INT_CFG_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_180
	ESM0_ESM_INT_HI_LVL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_1
	ESM0_ESM_INT_HI_LVL_OUT_0	GICSS0_SPI_IN_181
	ESM0_ESM_INT_HI_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_168
	ESM0_ESM_INT_HI_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_168
	ESM0_ESM_INT_HI_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_168
	ESM0_ESM_INT_HI_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_181
	ESM0_ESM_INT_HI_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_181
	ESM0_ESM_INT_LOW_LVL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_2
	ESM0_ESM_INT_LOW_LVL_OUT_0	GICSS0_SPI_IN_182
	ESM0_ESM_INT_LOW_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_169
	ESM0_ESM_INT_LOW_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_169

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
ESM0	ESM0_ESM_INT_LOW_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_169
	ESM0_ESM_INT_LOW_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_182
	ESM0_ESM_INT_LOW_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_182
FSS0_FSAS_0	FSS0_FSAS_0_ECC_INTR_ERR_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_141
FSS0_OSPI_0	FSS0_OSPI_0_OSPI_ECC_CORR_LVL_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_11
	FSS0_OSPI_0_OSPI_ECC_UNCORR_LVL_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_74
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	GICSS0_SPI_IN_171
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_171
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_171
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_171
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_171
	FSS0_OSPI_0_OSPI_LVL_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_171
GICSS0	GICSS0_AXIM_ERR_OUT_0	ESM0_ESM_PLS_EVENT0_IN_230
	GICSS0_AXIM_ERR_OUT_0	ESM0_ESM_PLS_EVENT1_IN_230
	GICSS0_AXIM_ERR_OUT_0	ESM0_ESM_PLS_EVENT2_IN_230
	GICSS0_ECC_AGGR_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_12
	GICSS0_ECC_AGGR_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_75
	GICSS0_ECC_FATAL_OUT_0	ESM0_ESM_PLS_EVENT0_IN_231
	GICSS0_ECC_FATAL_OUT_0	ESM0_ESM_PLS_EVENT1_IN_231
	GICSS0_ECC_FATAL_OUT_0	ESM0_ESM_PLS_EVENT2_IN_231
	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_154
	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_155
	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_156
	GICSS0_GIC_PWR0_WAKE_REQUEST_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_157
GPIO0	GPIO0_GPIO_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_0
	GPIO0_GPIO_OUT_1	MAIN_GPIOMUX_INTROUTER0_IN_IN_1
	GPIO0_GPIO_OUT_2	MAIN_GPIOMUX_INTROUTER0_IN_IN_2
	GPIO0_GPIO_OUT_3	MAIN_GPIOMUX_INTROUTER0_IN_IN_3
	GPIO0_GPIO_OUT_4	MAIN_GPIOMUX_INTROUTER0_IN_IN_4
	GPIO0_GPIO_OUT_5	MAIN_GPIOMUX_INTROUTER0_IN_IN_5
	GPIO0_GPIO_OUT_6	MAIN_GPIOMUX_INTROUTER0_IN_IN_6
	GPIO0_GPIO_OUT_7	MAIN_GPIOMUX_INTROUTER0_IN_IN_7
	GPIO0_GPIO_OUT_8	MAIN_GPIOMUX_INTROUTER0_IN_IN_8

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO0	GPIO0_GPIO_OUT_9	MAIN_GPIOMUX_INTROUTER0_IN_IN_9
	GPIO0_GPIO_OUT_10	MAIN_GPIOMUX_INTROUTER0_IN_IN_10
	GPIO0_GPIO_OUT_11	MAIN_GPIOMUX_INTROUTER0_IN_IN_11
	GPIO0_GPIO_OUT_12	MAIN_GPIOMUX_INTROUTER0_IN_IN_12
	GPIO0_GPIO_OUT_13	MAIN_GPIOMUX_INTROUTER0_IN_IN_13
	GPIO0_GPIO_OUT_14	MAIN_GPIOMUX_INTROUTER0_IN_IN_14
	GPIO0_GPIO_OUT_15	MAIN_GPIOMUX_INTROUTER0_IN_IN_15
	GPIO0_GPIO_OUT_16	MAIN_GPIOMUX_INTROUTER0_IN_IN_16
	GPIO0_GPIO_OUT_17	MAIN_GPIOMUX_INTROUTER0_IN_IN_17
	GPIO0_GPIO_OUT_18	MAIN_GPIOMUX_INTROUTER0_IN_IN_18
	GPIO0_GPIO_OUT_19	MAIN_GPIOMUX_INTROUTER0_IN_IN_19
	GPIO0_GPIO_OUT_20	MAIN_GPIOMUX_INTROUTER0_IN_IN_20
	GPIO0_GPIO_OUT_21	MAIN_GPIOMUX_INTROUTER0_IN_IN_21
	GPIO0_GPIO_OUT_22	MAIN_GPIOMUX_INTROUTER0_IN_IN_22
	GPIO0_GPIO_OUT_23	MAIN_GPIOMUX_INTROUTER0_IN_IN_23
	GPIO0_GPIO_OUT_24	MAIN_GPIOMUX_INTROUTER0_IN_IN_24
	GPIO0_GPIO_OUT_25	MAIN_GPIOMUX_INTROUTER0_IN_IN_25
	GPIO0_GPIO_OUT_26	MAIN_GPIOMUX_INTROUTER0_IN_IN_26
	GPIO0_GPIO_OUT_27	MAIN_GPIOMUX_INTROUTER0_IN_IN_27
	GPIO0_GPIO_OUT_28	MAIN_GPIOMUX_INTROUTER0_IN_IN_28
	GPIO0_GPIO_OUT_29	MAIN_GPIOMUX_INTROUTER0_IN_IN_29
	GPIO0_GPIO_OUT_30	MAIN_GPIOMUX_INTROUTER0_IN_IN_30
	GPIO0_GPIO_OUT_31	MAIN_GPIOMUX_INTROUTER0_IN_IN_31
	GPIO0_GPIO_OUT_32	MAIN_GPIOMUX_INTROUTER0_IN_IN_32
	GPIO0_GPIO_OUT_33	MAIN_GPIOMUX_INTROUTER0_IN_IN_33
	GPIO0_GPIO_OUT_34	MAIN_GPIOMUX_INTROUTER0_IN_IN_34
	GPIO0_GPIO_OUT_35	MAIN_GPIOMUX_INTROUTER0_IN_IN_35
	GPIO0_GPIO_OUT_36	MAIN_GPIOMUX_INTROUTER0_IN_IN_36
	GPIO0_GPIO_OUT_37	MAIN_GPIOMUX_INTROUTER0_IN_IN_37
	GPIO0_GPIO_OUT_38	MAIN_GPIOMUX_INTROUTER0_IN_IN_38
	GPIO0_GPIO_OUT_39	MAIN_GPIOMUX_INTROUTER0_IN_IN_39
	GPIO0_GPIO_OUT_40	MAIN_GPIOMUX_INTROUTER0_IN_IN_40
	GPIO0_GPIO_OUT_41	MAIN_GPIOMUX_INTROUTER0_IN_IN_41

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO0	GPIO0_GPIO_OUT_42	MAIN_GPIOMUX_INTROUTER0_IN_IN_42
	GPIO0_GPIO_OUT_43	MAIN_GPIOMUX_INTROUTER0_IN_IN_43
	GPIO0_GPIO_OUT_44	MAIN_GPIOMUX_INTROUTER0_IN_IN_44
	GPIO0_GPIO_OUT_45	MAIN_GPIOMUX_INTROUTER0_IN_IN_45
	GPIO0_GPIO_OUT_46	MAIN_GPIOMUX_INTROUTER0_IN_IN_46
	GPIO0_GPIO_OUT_47	MAIN_GPIOMUX_INTROUTER0_IN_IN_47
	GPIO0_GPIO_OUT_48	MAIN_GPIOMUX_INTROUTER0_IN_IN_48
	GPIO0_GPIO_OUT_49	MAIN_GPIOMUX_INTROUTER0_IN_IN_49
	GPIO0_GPIO_OUT_50	MAIN_GPIOMUX_INTROUTER0_IN_IN_50
	GPIO0_GPIO_OUT_51	MAIN_GPIOMUX_INTROUTER0_IN_IN_51
	GPIO0_GPIO_OUT_52	MAIN_GPIOMUX_INTROUTER0_IN_IN_52
	GPIO0_GPIO_OUT_53	MAIN_GPIOMUX_INTROUTER0_IN_IN_53
	GPIO0_GPIO_OUT_54	MAIN_GPIOMUX_INTROUTER0_IN_IN_54
	GPIO0_GPIO_OUT_55	MAIN_GPIOMUX_INTROUTER0_IN_IN_55
	GPIO0_GPIO_OUT_56	MAIN_GPIOMUX_INTROUTER0_IN_IN_56
	GPIO0_GPIO_OUT_57	MAIN_GPIOMUX_INTROUTER0_IN_IN_57
	GPIO0_GPIO_OUT_58	MAIN_GPIOMUX_INTROUTER0_IN_IN_58
	GPIO0_GPIO_OUT_59	MAIN_GPIOMUX_INTROUTER0_IN_IN_59
	GPIO0_GPIO_OUT_60	MAIN_GPIOMUX_INTROUTER0_IN_IN_60
	GPIO0_GPIO_OUT_61	MAIN_GPIOMUX_INTROUTER0_IN_IN_61
	GPIO0_GPIO_OUT_62	MAIN_GPIOMUX_INTROUTER0_IN_IN_62
	GPIO0_GPIO_OUT_63	MAIN_GPIOMUX_INTROUTER0_IN_IN_63
	GPIO0_GPIO_OUT_64	MAIN_GPIOMUX_INTROUTER0_IN_IN_64
	GPIO0_GPIO_OUT_65	MAIN_GPIOMUX_INTROUTER0_IN_IN_65
	GPIO0_GPIO_OUT_66	MAIN_GPIOMUX_INTROUTER0_IN_IN_66
	GPIO0_GPIO_OUT_67	MAIN_GPIOMUX_INTROUTER0_IN_IN_67
	GPIO0_GPIO_OUT_68	MAIN_GPIOMUX_INTROUTER0_IN_IN_68
	GPIO0_GPIO_OUT_69	MAIN_GPIOMUX_INTROUTER0_IN_IN_69
	GPIO0_GPIO_OUT_70	MAIN_GPIOMUX_INTROUTER0_IN_IN_70
	GPIO0_GPIO_OUT_71	MAIN_GPIOMUX_INTROUTER0_IN_IN_71
	GPIO0_GPIO_OUT_72	MAIN_GPIOMUX_INTROUTER0_IN_IN_72
	GPIO0_GPIO_OUT_73	MAIN_GPIOMUX_INTROUTER0_IN_IN_73
	GPIO0_GPIO_OUT_74	MAIN_GPIOMUX_INTROUTER0_IN_IN_74

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO0	GPIO0_GPIO_OUT_75	MAIN_GPIOMUX_INTROUTER0_IN_IN_75
	GPIO0_GPIO_OUT_76	MAIN_GPIOMUX_INTROUTER0_IN_IN_76
	GPIO0_GPIO_OUT_77	MAIN_GPIOMUX_INTROUTER0_IN_IN_77
	GPIO0_GPIO_OUT_78	MAIN_GPIOMUX_INTROUTER0_IN_IN_78
	GPIO0_GPIO_OUT_79	MAIN_GPIOMUX_INTROUTER0_IN_IN_79
	GPIO0_GPIO_OUT_80	MAIN_GPIOMUX_INTROUTER0_IN_IN_80
	GPIO0_GPIO_OUT_81	MAIN_GPIOMUX_INTROUTER0_IN_IN_81
	GPIO0_GPIO_OUT_82	MAIN_GPIOMUX_INTROUTER0_IN_IN_82
	GPIO0_GPIO_OUT_83	MAIN_GPIOMUX_INTROUTER0_IN_IN_83
	GPIO0_GPIO_OUT_84	MAIN_GPIOMUX_INTROUTER0_IN_IN_84
	GPIO0_GPIO_OUT_85	MAIN_GPIOMUX_INTROUTER0_IN_IN_85
	GPIO0_GPIO_OUT_86	MAIN_GPIOMUX_INTROUTER0_IN_IN_86
	GPIO0_GPIO_OUT_87	MAIN_GPIOMUX_INTROUTER0_IN_IN_87
	GPIO0_GPIO_OUT_88	MAIN_GPIOMUX_INTROUTER0_IN_IN_88
	GPIO0_GPIO_OUT_89	MAIN_GPIOMUX_INTROUTER0_IN_IN_89
	GPIO0_GPIO_OUT_90	MAIN_GPIOMUX_INTROUTER0_IN_IN_176
	GPIO0_GPIO_OUT_91	MAIN_GPIOMUX_INTROUTER0_IN_IN_177
	GPIO0_GPIO_BANK_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_190
	GPIO0_GPIO_BANK_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_56
	GPIO0_GPIO_BANK_OUT_1	MAIN_GPIOMUX_INTROUTER0_IN_IN_191
	GPIO0_GPIO_BANK_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_57
	GPIO0_GPIO_BANK_OUT_2	MAIN_GPIOMUX_INTROUTER0_IN_IN_192
	GPIO0_GPIO_BANK_OUT_3	MAIN_GPIOMUX_INTROUTER0_IN_IN_193
	GPIO0_GPIO_BANK_OUT_4	MAIN_GPIOMUX_INTROUTER0_IN_IN_194
	GPIO0_GPIO_BANK_OUT_5	MAIN_GPIOMUX_INTROUTER0_IN_IN_195
GPIO1	GPIO1_GPIO_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_90
	GPIO1_GPIO_OUT_1	MAIN_GPIOMUX_INTROUTER0_IN_IN_91
	GPIO1_GPIO_OUT_2	MAIN_GPIOMUX_INTROUTER0_IN_IN_92
	GPIO1_GPIO_OUT_3	MAIN_GPIOMUX_INTROUTER0_IN_IN_93
	GPIO1_GPIO_OUT_4	MAIN_GPIOMUX_INTROUTER0_IN_IN_94
	GPIO1_GPIO_OUT_5	MAIN_GPIOMUX_INTROUTER0_IN_IN_95
	GPIO1_GPIO_OUT_6	MAIN_GPIOMUX_INTROUTER0_IN_IN_96
	GPIO1_GPIO_OUT_7	MAIN_GPIOMUX_INTROUTER0_IN_IN_97

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO1	GPIO1_GPIO_OUT_8	MAIN_GPIOMUX_INTROUTER0_IN_IN_98
	GPIO1_GPIO_OUT_9	MAIN_GPIOMUX_INTROUTER0_IN_IN_99
	GPIO1_GPIO_OUT_10	MAIN_GPIOMUX_INTROUTER0_IN_IN_100
	GPIO1_GPIO_OUT_11	MAIN_GPIOMUX_INTROUTER0_IN_IN_101
	GPIO1_GPIO_OUT_12	MAIN_GPIOMUX_INTROUTER0_IN_IN_102
	GPIO1_GPIO_OUT_13	MAIN_GPIOMUX_INTROUTER0_IN_IN_103
	GPIO1_GPIO_OUT_14	MAIN_GPIOMUX_INTROUTER0_IN_IN_104
	GPIO1_GPIO_OUT_15	MAIN_GPIOMUX_INTROUTER0_IN_IN_105
	GPIO1_GPIO_OUT_16	MAIN_GPIOMUX_INTROUTER0_IN_IN_106
	GPIO1_GPIO_OUT_17	MAIN_GPIOMUX_INTROUTER0_IN_IN_107
	GPIO1_GPIO_OUT_18	MAIN_GPIOMUX_INTROUTER0_IN_IN_108
	GPIO1_GPIO_OUT_19	MAIN_GPIOMUX_INTROUTER0_IN_IN_109
	GPIO1_GPIO_OUT_20	MAIN_GPIOMUX_INTROUTER0_IN_IN_110
	GPIO1_GPIO_OUT_21	MAIN_GPIOMUX_INTROUTER0_IN_IN_111
	GPIO1_GPIO_OUT_22	MAIN_GPIOMUX_INTROUTER0_IN_IN_112
	GPIO1_GPIO_OUT_23	MAIN_GPIOMUX_INTROUTER0_IN_IN_113
	GPIO1_GPIO_OUT_24	MAIN_GPIOMUX_INTROUTER0_IN_IN_114
	GPIO1_GPIO_OUT_25	MAIN_GPIOMUX_INTROUTER0_IN_IN_115
	GPIO1_GPIO_OUT_26	MAIN_GPIOMUX_INTROUTER0_IN_IN_116
	GPIO1_GPIO_OUT_27	MAIN_GPIOMUX_INTROUTER0_IN_IN_117
	GPIO1_GPIO_OUT_28	MAIN_GPIOMUX_INTROUTER0_IN_IN_118
	GPIO1_GPIO_OUT_29	MAIN_GPIOMUX_INTROUTER0_IN_IN_119
	GPIO1_GPIO_OUT_30	MAIN_GPIOMUX_INTROUTER0_IN_IN_120
	GPIO1_GPIO_OUT_31	MAIN_GPIOMUX_INTROUTER0_IN_IN_121
	GPIO1_GPIO_OUT_32	MAIN_GPIOMUX_INTROUTER0_IN_IN_122
	GPIO1_GPIO_OUT_33	MAIN_GPIOMUX_INTROUTER0_IN_IN_123
	GPIO1_GPIO_OUT_34	MAIN_GPIOMUX_INTROUTER0_IN_IN_124
	GPIO1_GPIO_OUT_35	MAIN_GPIOMUX_INTROUTER0_IN_IN_125
	GPIO1_GPIO_OUT_36	MAIN_GPIOMUX_INTROUTER0_IN_IN_126
	GPIO1_GPIO_OUT_37	MAIN_GPIOMUX_INTROUTER0_IN_IN_127
	GPIO1_GPIO_OUT_38	MAIN_GPIOMUX_INTROUTER0_IN_IN_128
	GPIO1_GPIO_OUT_39	MAIN_GPIOMUX_INTROUTER0_IN_IN_129
	GPIO1_GPIO_OUT_40	MAIN_GPIOMUX_INTROUTER0_IN_IN_130

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO1	GPIO1_GPIO_OUT_41	MAIN_GPIOMUX_INTROUTER0_IN_IN_131
	GPIO1_GPIO_OUT_42	MAIN_GPIOMUX_INTROUTER0_IN_IN_132
	GPIO1_GPIO_OUT_43	MAIN_GPIOMUX_INTROUTER0_IN_IN_133
	GPIO1_GPIO_OUT_44	MAIN_GPIOMUX_INTROUTER0_IN_IN_134
	GPIO1_GPIO_OUT_45	MAIN_GPIOMUX_INTROUTER0_IN_IN_135
	GPIO1_GPIO_OUT_46	MAIN_GPIOMUX_INTROUTER0_IN_IN_136
	GPIO1_GPIO_OUT_47	MAIN_GPIOMUX_INTROUTER0_IN_IN_137
	GPIO1_GPIO_OUT_48	MAIN_GPIOMUX_INTROUTER0_IN_IN_138
	GPIO1_GPIO_OUT_49	MAIN_GPIOMUX_INTROUTER0_IN_IN_139
	GPIO1_GPIO_OUT_50	MAIN_GPIOMUX_INTROUTER0_IN_IN_140
	GPIO1_GPIO_OUT_51	MAIN_GPIOMUX_INTROUTER0_IN_IN_141
	GPIO1_GPIO_OUT_52	MAIN_GPIOMUX_INTROUTER0_IN_IN_142
	GPIO1_GPIO_OUT_53	MAIN_GPIOMUX_INTROUTER0_IN_IN_143
	GPIO1_GPIO_OUT_54	MAIN_GPIOMUX_INTROUTER0_IN_IN_144
	GPIO1_GPIO_OUT_55	MAIN_GPIOMUX_INTROUTER0_IN_IN_145
	GPIO1_GPIO_OUT_56	MAIN_GPIOMUX_INTROUTER0_IN_IN_146
	GPIO1_GPIO_OUT_57	MAIN_GPIOMUX_INTROUTER0_IN_IN_147
	GPIO1_GPIO_OUT_58	MAIN_GPIOMUX_INTROUTER0_IN_IN_148
	GPIO1_GPIO_OUT_59	MAIN_GPIOMUX_INTROUTER0_IN_IN_149
	GPIO1_GPIO_OUT_60	MAIN_GPIOMUX_INTROUTER0_IN_IN_150
	GPIO1_GPIO_OUT_61	MAIN_GPIOMUX_INTROUTER0_IN_IN_151
	GPIO1_GPIO_OUT_62	MAIN_GPIOMUX_INTROUTER0_IN_IN_152
	GPIO1_GPIO_OUT_63	MAIN_GPIOMUX_INTROUTER0_IN_IN_153
	GPIO1_GPIO_OUT_64	MAIN_GPIOMUX_INTROUTER0_IN_IN_154
	GPIO1_GPIO_OUT_65	MAIN_GPIOMUX_INTROUTER0_IN_IN_155
	GPIO1_GPIO_OUT_66	MAIN_GPIOMUX_INTROUTER0_IN_IN_156
	GPIO1_GPIO_OUT_67	MAIN_GPIOMUX_INTROUTER0_IN_IN_157
	GPIO1_GPIO_OUT_68	MAIN_GPIOMUX_INTROUTER0_IN_IN_158
	GPIO1_GPIO_OUT_69	MAIN_GPIOMUX_INTROUTER0_IN_IN_159
	GPIO1_GPIO_OUT_70	MAIN_GPIOMUX_INTROUTER0_IN_IN_160
	GPIO1_GPIO_OUT_71	MAIN_GPIOMUX_INTROUTER0_IN_IN_161
	GPIO1_GPIO_BANK_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_180
	GPIO1_GPIO_BANK_OUT_1	MAIN_GPIOMUX_INTROUTER0_IN_IN_181

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPIO1	GPIO1_GPIO_BANK_OUT_2	MAIN_GPIOMUX_INTROUTER0_IN_IN_182
	GPIO1_GPIO_BANK_OUT_3	MAIN_GPIOMUX_INTROUTER0_IN_IN_183
	GPIO1_GPIO_BANK_OUT_4	MAIN_GPIOMUX_INTROUTER0_IN_IN_184
	GPIO1_GPIO_BANK_OUT_5	MAIN_GPIOMUX_INTROUTER0_IN_IN_185
MCU_GPIO0	MCU_GPIO0_GPIO_OUT_0	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_0
	MCU_GPIO0_GPIO_OUT_1	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_1
	MCU_GPIO0_GPIO_OUT_2	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_2
	MCU_GPIO0_GPIO_OUT_3	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_3
	MCU_GPIO0_GPIO_OUT_4	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_4
	MCU_GPIO0_GPIO_OUT_5	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_5
	MCU_GPIO0_GPIO_OUT_6	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_6
	MCU_GPIO0_GPIO_OUT_7	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_7
	MCU_GPIO0_GPIO_OUT_8	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_8
	MCU_GPIO0_GPIO_OUT_9	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_9
	MCU_GPIO0_GPIO_OUT_10	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_10
	MCU_GPIO0_GPIO_OUT_11	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_11
	MCU_GPIO0_GPIO_OUT_12	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_12
	MCU_GPIO0_GPIO_OUT_13	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_13
	MCU_GPIO0_GPIO_OUT_14	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_14
	MCU_GPIO0_GPIO_OUT_15	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_15
	MCU_GPIO0_GPIO_OUT_16	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_16
	MCU_GPIO0_GPIO_OUT_17	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_17
	MCU_GPIO0_GPIO_OUT_18	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_18
	MCU_GPIO0_GPIO_OUT_19	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_19
	MCU_GPIO0_GPIO_OUT_20	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_20
	MCU_GPIO0_GPIO_OUT_21	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_21
	MCU_GPIO0_GPIO_OUT_22	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_22
	MCU_GPIO0_GPIO_OUT_23	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_23
	MCU_GPIO0_GPIO_BANK_OUT_0	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_30
	MCU_GPIO0_GPIO_BANK_OUT_1	WKUP_MCU_GPIOMUX_INTROUTER0_IN_IN_31
	MCU_GPIO0_GPIO_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_18
	MCU_GPIO0_GPIO_LVL_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_2
GPMC0	GPMC0_GPMC_SDMAREQ_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_26

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPMC0	GPMC0_GPMC_SINTERRUPT_OUT_0	GICSS0_SPI_IN_138
	GPMC0_GPMC_SINTERRUPT_OUT_0	R5FSS0_CORE0_INTR_IN_103
	GPMC0_GPMC_SINTERRUPT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_103
	GPMC0_GPMC_SINTERRUPT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_103
	GPMC0_GPMC_SINTERRUPT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_138
	GPMC0_GPMC_SINTERRUPT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_138
WKUP_GTC0	WKUP_GTC0_GTC_PUSH_EVENT_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_11
WKUP_icemelter0	WKUP_ICEMELTER0_PSC_FORCE_POWER_ON_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_23
	WKUP_ICEMELTER0_PSC_FORCE_POWER_ON_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_3
CTRL_MMR0	CTRL_MMR0_IPC_SET0_IPC_SET_IPCFG_OUT_0	C7X256V0_CLEC_SOC_EVENTS_IN_IN_5
	CTRL_MMR0_IPC_SET1_IPC_SET_IPCFG_OUT_0	C7X256V1_CLEC_SOC_EVENTS_IN_IN_5
	CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_39
	CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39
	CTRL_MMR0_ACCESS_ERR_OUT_0	GICSS0_SPI_IN_129
	CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_128
	CTRL_MMR0_ACCESS_ERR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_128
	CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128
	CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_129
	CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_129
WKUP_CTRL_MMR0	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_39
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_39
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	GICSS0_SPI_IN_129
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	R5FSS0_CORE0_INTR_IN_128
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_128
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_128
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_129
	WKUP_CTRL_MMR0_ACCESS_ERR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_129
DDPA0	DDPA0_DDPA_INTR_OUT_0	GICSS0_SPI_IN_177
	DDPA0_DDPA_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_177
	DDPA0_DDPA_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_177
	DDPA0_DDPA_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_177
	DDPA0_DDPA_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_177
	DDPA0_DDPA_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_177

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DSS_DSI0	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	GICSS0_SPI_IN_118
	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_218
	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_218
	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_218
	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_118
	DSS_DSI0_DSI_0_FUNC_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_118
	DSS_DSI0_DSI_0_SAFETY_ERROR_FATAL_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_178
	DSS_DSI0_DSI_0_SAFETY_ERROR_NONFATAL_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_179
	DSS_DSI0_ECC_INTR_UNCORR_LEVEL_SYS_OUT_0	ESM0_ESM_LVL_EVENT_IN_142
DSS0	DSS0_DISPC_INTR_REQ_0_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_2
	DSS0_DISPC_INTR_REQ_0_OUT_0	GICSS0_SPI_IN_116
	DSS0_DISPC_INTR_REQ_0_OUT_0	R5FSS0_CORE0_INTR_IN_40
	DSS0_DISPC_INTR_REQ_0_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_24
	DSS0_DISPC_INTR_REQ_0_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_40
	DSS0_DISPC_INTR_REQ_0_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_116
	DSS0_DISPC_INTR_REQ_0_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_116
	DSS0_DISPC_INTR_REQ_1_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_3
	DSS0_DISPC_INTR_REQ_1_OUT_0	GICSS0_SPI_IN_117
	DSS0_DISPC_INTR_REQ_1_OUT_0	R5FSS0_CORE0_INTR_IN_41
	DSS0_DISPC_INTR_REQ_1_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_25
	DSS0_DISPC_INTR_REQ_1_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_41
	DSS0_DISPC_INTR_REQ_1_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_117
	DSS0_DISPC_INTR_REQ_1_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_117
DSS1	DSS1_DISPC_INTR_REQ_0_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_4
	DSS1_DISPC_INTR_REQ_0_OUT_0	GICSS0_SPI_IN_62
	DSS1_DISPC_INTR_REQ_0_OUT_0	R5FSS0_CORE0_INTR_IN_92
	DSS1_DISPC_INTR_REQ_0_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_117
	DSS1_DISPC_INTR_REQ_0_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_92
	DSS1_DISPC_INTR_REQ_0_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_62
	DSS1_DISPC_INTR_REQ_0_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_62
	DSS1_DISPC_INTR_REQ_1_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_5
	DSS1_DISPC_INTR_REQ_1_OUT_0	GICSS0_SPI_IN_63
	DSS1_DISPC_INTR_REQ_1_OUT_0	R5FSS0_CORE0_INTR_IN_93

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DSS1	DSS1_DISPC_INTR_REQ_1_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_118
	DSS1_DISPC_INTR_REQ_1_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_93
	DSS1_DISPC_INTR_REQ_1_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_63
	DSS1_DISPC_INTR_REQ_1_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_63
EPWM0	EPWM0_EPWM_ETINT_OUT_0	GICSS0_SPI_IN_229
	EPWM0_EPWM_ETINT_OUT_0	R5FSS0_CORE0_INTR_IN_36
	EPWM0_EPWM_ETINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_123
	EPWM0_EPWM_ETINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_36
	EPWM0_EPWM_ETINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_229
	EPWM0_EPWM_ETINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_229
	EPWM0_EPWM_SYNCO_O_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_8
	EPWM0_EPWM_SYNCOUT_OUT_0	EPWM1_EPWM_SYNCIN_IN_0
	EPWM0_EPWM_TRIPZINT_OUT_0	GICSS0_SPI_IN_230
	EPWM0_EPWM_TRIPZINT_OUT_0	R5FSS0_CORE0_INTR_IN_80
	EPWM0_EPWM_TRIPZINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_124
	EPWM0_EPWM_TRIPZINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_80
	EPWM0_EPWM_TRIPZINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_230
	EPWM0_EPWM_TRIPZINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_230
EPWM1	EPWM1_EPWM_ETINT_OUT_0	GICSS0_SPI_IN_231
	EPWM1_EPWM_ETINT_OUT_0	R5FSS0_CORE0_INTR_IN_37
	EPWM1_EPWM_ETINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_125
	EPWM1_EPWM_ETINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_37
	EPWM1_EPWM_ETINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_231
	EPWM1_EPWM_ETINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_231
	EPWM1_EPWM_SYNCOUT_OUT_0	EPWM2_EPWM_SYNCIN_IN_0
	EPWM1_EPWM_TRIPZINT_OUT_0	GICSS0_SPI_IN_233
	EPWM1_EPWM_TRIPZINT_OUT_0	R5FSS0_CORE0_INTR_IN_81
	EPWM1_EPWM_TRIPZINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_126
	EPWM1_EPWM_TRIPZINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_81
	EPWM1_EPWM_TRIPZINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_233
	EPWM1_EPWM_TRIPZINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_233
EPWM2	EPWM2_EPWM_ETINT_OUT_0	GICSS0_SPI_IN_234
	EPWM2_EPWM_ETINT_OUT_0	R5FSS0_CORE0_INTR_IN_38

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
EPWM2	EPWM2_EPWM_ETINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_127
	EPWM2_EPWM_ETINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_38
	EPWM2_EPWM_ETINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_234
	EPWM2_EPWM_ETINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_234
	EPWM2_EPWM_TRIPZINT_OUT_0	GICSS0_SPI_IN_235
	EPWM2_EPWM_TRIPZINT_OUT_0	R5FSS0_CORE0_INTR_IN_82
	EPWM2_EPWM_TRIPZINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_148
	EPWM2_EPWM_TRIPZINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_82
	EPWM2_EPWM_TRIPZINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_235
	EPWM2_EPWM_TRIPZINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_235
JPGENC0	JPGENC0_IRQ_OUT_0	GICSS0_SPI_IN_130
	JPGENC0_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_100
	JPGENC0_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_100
	JPGENC0_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_100
	JPGENC0_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_130
	JPGENC0_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_130
PBIST0	PBIST0_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	PBIST0_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	PBIST0_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_228
	PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_228
	PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_228
	PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_157
PBIST1	PBIST1_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	PBIST1_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	PBIST1_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_229
	PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_229
	PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_229
	PBIST1_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_207
WKUP_PBIST0	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_114
	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_114
	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_114

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_PBIST0	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_234
	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_234
	WKUP_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_234
	WKUP_PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_158
WKUP_PBIST1	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_114
	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_114
	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_114
	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_233
	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_233
	WKUP_PBIST1_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_233
	WKUP_PBIST1_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_159
PBIST2	PBIST2_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	PBIST2_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	PBIST2_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	PBIST2_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_243
	PBIST2_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_243
	PBIST2_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_243
	PBIST2_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_213
MCU_PBIST0	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_235
	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_235
	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_235
	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_149
	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_149
	MCU_PBIST0_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_149
	MCU_PBIST0_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_152
CODEC0	CODEC0_VPU_WAVE521CL_INTR_OUT_0	GICSS0_SPI_IN_257
	CODEC0_VPU_WAVE521CL_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_133
	CODEC0_VPU_WAVE521CL_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_133
	CODEC0_VPU_WAVE521CL_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_133
	CODEC0_VPU_WAVE521CL_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_257
	CODEC0_VPU_WAVE521CL_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_257
WKUP_VTM0	WKUP_VTM0_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_11
	WKUP_VTM0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_139

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_VTM0	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_8
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_137
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	GICSS0_SPI_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH1_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_183
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_10
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_138
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	GICSS0_SPI_IN_184
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_184
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_184
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_184
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_184
	WKUP_VTM0_THERM_LVL_GT_TH2_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_184
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_9
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_136
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	GICSS0_SPI_IN_185
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_185
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_185
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_185
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_185
	WKUP_VTM0_THERM_LVL_LT_TH0_INTR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_185
	WKUP_VTM0_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_12
	WKUP_VTM0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_140
MAILBOX0_MAILBOX_CLUSTER_0	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0	GICSS0_SPI_IN_108
	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_108
	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_108
	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_240
	MAILBOX0_MAILBOX_CLUSTER_0_MAILBOX_CLUSTER_PEND_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_240
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0	GICSS0_SPI_IN_109
	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_109

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MAILBOX0_MAILBOX_CLUSTER_1	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_109
	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_241
	MAILBOX0_MAILBOX_CLUSTER_1_MAILBOX_CLUSTER_PEND_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_241
MAILBOX0_MAILBOX_CLUSTER_2	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0	GICSS0_SPI_IN_140
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_140
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_140
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_6
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_242
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_6
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_115
	MAILBOX0_MAILBOX_CLUSTER_2_MAILBOX_CLUSTER_PEND_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_242
MAILBOX0_MAILBOX_CLUSTER_3	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0	GICSS0_SPI_IN_141
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_141
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_141
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_7
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_243
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_7
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_116
	MAILBOX0_MAILBOX_CLUSTER_3_MAILBOX_CLUSTER_PEND_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_243
MAILBOX0_MAILBOX_CLUSTER_4	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_115
	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_8
	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_8
	MAILBOX0_MAILBOX_CLUSTER_4_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_240
MAILBOX0_MAILBOX_CLUSTER_5	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_116
	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_9
	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_9
	MAILBOX0_MAILBOX_CLUSTER_5_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_241
MAILBOX0_MAILBOX_CLUSTER_6	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_115
	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_10
	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_10
	MAILBOX0_MAILBOX_CLUSTER_6_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_242
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_116
	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_1	C7X256V0_CLEC_SOC_EVENTS_IN_IN_11

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MAILBOX0_MAILBOX_CLUSTER_7	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_2	C7X256V1_CLEC_SOC_EVENTS_IN_IN_11
	MAILBOX0_MAILBOX_CLUSTER_7_MAILBOX_CLUSTER_PEND_OUT_3	R5FSS0_CORE0_INTR_IN_243
MCAN0	MCAN0_MCANSS_ECC_CORR_LVL_INT_OUT_0	ESM0_ESM_LVL_EVENT_IN_16
	MCAN0_MCANSS_ECC_UNCORR_LVL_INT_OUT_0	ESM0_ESM_LVL_EVENT_IN_78
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	GICSS0_SPI_IN_186
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_186
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_186
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_186
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_186
	MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_186
	MCAN0_MCANSS_FE_OUT_0	PDMA0_MCANSS_MAIN_0_FE_IN_0
	MCAN0_MCANSS_FE_OUT_1	PDMA0_MCANSS_MAIN_0_FE_IN_1
	MCAN0_MCANSS_FE_OUT_2	PDMA0_MCANSS_MAIN_0_FE_IN_2
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	GICSS0_SPI_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_187
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	GICSS0_SPI_IN_188
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	R5FSS0_CORE0_INTR_IN_188
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_188
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_188
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_188
	MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_188
	MCAN0_MCANSS_TX_DMA_OUT_0	PDMA0_MCANSS_MAIN_0_TX_IN_0
	MCAN0_MCANSS_TX_DMA_OUT_1	PDMA0_MCANSS_MAIN_0_TX_IN_1
	MCAN0_MCANSS_TX_DMA_OUT_2	PDMA0_MCANSS_MAIN_0_TX_IN_2
MCAN1	MCAN1_MCANSS_ECC_CORR_LVL_INT_OUT_0	ESM0_ESM_LVL_EVENT_IN_68
	MCAN1_MCANSS_ECC_UNCORR_LVL_INT_OUT_0	ESM0_ESM_LVL_EVENT_IN_69
	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	GICSS0_SPI_IN_199
	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_229
	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_229

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCAN1	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_229
	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_199
	MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_199
	MCAN1_MCANSS_FE_OUT_0	PDMA0_MCANSS_MAIN_1_FE_IN_0
	MCAN1_MCANSS_FE_OUT_1	PDMA0_MCANSS_MAIN_1_FE_IN_1
	MCAN1_MCANSS_FE_OUT_2	PDMA0_MCANSS_MAIN_1_FE_IN_2
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	GICSS0_SPI_IN_245
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_110
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_110
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_110
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_245
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_245
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	GICSS0_SPI_IN_246
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	R5FSS0_CORE0_INTR_IN_63
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_63
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_63
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_246
	MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_246
	MCAN1_MCANSS_TX_DMA_OUT_0	PDMA0_MCANSS_MAIN_1_TX_IN_0
	MCAN1_MCANSS_TX_DMA_OUT_1	PDMA0_MCANSS_MAIN_1_TX_IN_1
	MCAN1_MCANSS_TX_DMA_OUT_2	PDMA0_MCANSS_MAIN_1_TX_IN_2
MCU_MCAN0	MCU_MCAN0_MCANSS_ECC_CORR_LVL_INT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_16
	MCU_MCAN0_MCANSS_ECC_UNCORR_LVL_INT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_17
	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_42
	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_198
	MCU_MCAN0_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_42
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	GICSS0_SPI_IN_58
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_43
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_199
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_43
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_58
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_58
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	GICSS0_SPI_IN_59

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_MCAN0	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	R5FSS0_CORE0_INTR_IN_44
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_200
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_44
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_59
	MCU_MCAN0_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_59
MCU_MCAN1	MCU_MCAN1_MCANSS_ECC_CORR_LVL_INT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_18
	MCU_MCAN1_MCANSS_ECC_UNCORR_LVL_INT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_19
	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_45
	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_239
	MCU_MCAN1_MCANSS_EXT_TS_ROLLOVER_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_45
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	GICSS0_SPI_IN_60
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	R5FSS0_CORE0_INTR_IN_46
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_247
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_46
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_60
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_60
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	GICSS0_SPI_IN_61
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	R5FSS0_CORE0_INTR_IN_47
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_248
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_47
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_61
	MCU_MCAN1_MCANSS_MCAN_LVL_INT_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_61
MCASP0	MCASP0_REC_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_0_RX_IN_0
	MCASP0_REC_INTR_PEND_OUT_0	GICSS0_SPI_IN_267
	MCASP0_REC_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_120
	MCASP0_REC_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_120
	MCASP0_REC_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_267
	MCASP0_REC_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_267
	MCASP0_XMIT_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_0_TX_IN_0
	MCASP0_XMIT_INTR_PEND_OUT_0	GICSS0_SPI_IN_268
	MCASP0_XMIT_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_121
	MCASP0_XMIT_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_121
	MCASP0_XMIT_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_268

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCASP0	MCASP0_XMIT_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_268
MCASP1	MCASP1_REC_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_1_RX_IN_0
	MCASP1_REC_INTR_PEND_OUT_0	GICSS0_SPI_IN_269
	MCASP1_REC_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_122
	MCASP1_REC_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_122
	MCASP1_REC_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_269
	MCASP1_REC_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_269
	MCASP1_XMIT_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_1_TX_IN_0
	MCASP1_XMIT_INTR_PEND_OUT_0	GICSS0_SPI_IN_270
	MCASP1_XMIT_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_123
	MCASP1_XMIT_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_123
	MCASP1_XMIT_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_270
	MCASP1_XMIT_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_270
MCASP2	MCASP2_REC_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_2_RX_IN_0
	MCASP2_REC_INTR_PEND_OUT_0	GICSS0_SPI_IN_271
	MCASP2_REC_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_124
	MCASP2_REC_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_124
	MCASP2_REC_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_271
	MCASP2_REC_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_271
	MCASP2_XMIT_DMA_EVENT_REQ_OUT_0	PDMA2_MCASP_MAIN_2_TX_IN_0
	MCASP2_XMIT_INTR_PEND_OUT_0	GICSS0_SPI_IN_272
	MCASP2_XMIT_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_125
	MCASP2_XMIT_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_125
	MCASP2_XMIT_INTR_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_272
	MCASP2_XMIT_INTR_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_272
MCASP3	MCASP3_REC_DMA_EVENT_REQ_OUT_0	PDMA3_MCASP_MAIN_3_RX_IN_0
	MCASP3_REC_INTR_PEND_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_186
	MCASP3_REC_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_118
	MCASP3_REC_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_118
	MCASP3_XMIT_DMA_EVENT_REQ_OUT_0	PDMA3_MCASP_MAIN_3_TX_IN_0
	MCASP3_XMIT_INTR_PEND_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_187
	MCASP3_XMIT_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_117
	MCASP3_XMIT_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_117

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCASP4	MCASP4_REC_DMA_EVENT_REQ_OUT_0	PDMA3_MCASP_MAIN_4_RX_IN_0
	MCASP4_REC_INTR_PEND_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_188
	MCASP4_REC_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_252
	MCASP4_REC_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_252
	MCASP4_XMIT_DMA_EVENT_REQ_OUT_0	PDMA3_MCASP_MAIN_4_TX_IN_0
	MCASP4_XMIT_INTR_PEND_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_189
	MCASP4_XMIT_INTR_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_251
	MCASP4_XMIT_INTR_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_251
MCRC64_0	MCRC64_0_DMA_EVENT_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_28
	MCRC64_0_DMA_EVENT_OUT_1	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_29
	MCRC64_0_DMA_EVENT_OUT_2	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_30
	MCRC64_0_DMA_EVENT_OUT_3	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_31
	MCRC64_0_INT_MCRC_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_7
	MCRC64_0_INT_MCRC_OUT_0	GICSS0_SPI_IN_166
	MCRC64_0_INT_MCRC_OUT_0	R5FSS0_CORE0_INTR_IN_119
	MCRC64_0_INT_MCRC_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_119
	MCRC64_0_INT_MCRC_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_119
	MCRC64_0_INT_MCRC_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_166
	MCRC64_0_INT_MCRC_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_166
MCU_MCRC64_0	MCU_MCRC64_0_INT_MCRC_OUT_0	GICSS0_SPI_IN_192
	MCU_MCRC64_0_INT_MCRC_OUT_0	R5FSS0_CORE0_INTR_IN_192
	MCU_MCRC64_0_INT_MCRC_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_192
	MCU_MCRC64_0_INT_MCRC_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_192
	MCU_MCRC64_0_INT_MCRC_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_192
	MCU_MCRC64_0_INT_MCRC_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_192
I2C0	I2C0_POINTRPEND_OUT_0	GICSS0_SPI_IN_193
	I2C0_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_193
	I2C0_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_193
	I2C0_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_193
	I2C0_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_193
	I2C0_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_193
I2C1	I2C1_POINTRPEND_OUT_0	GICSS0_SPI_IN_194
	I2C1_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_194

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
I2C1	I2C1_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_194
	I2C1_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_194
	I2C1_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_194
	I2C1_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_194
I2C2	I2C2_POINTRPEND_OUT_0	GICSS0_SPI_IN_195
	I2C2_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_195
	I2C2_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_195
	I2C2_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_195
	I2C2_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_195
	I2C2_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_195
I2C3	I2C3_POINTRPEND_OUT_0	GICSS0_SPI_IN_196
	I2C3_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_196
	I2C3_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_196
	I2C3_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_196
	I2C3_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_196
	I2C3_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_196
I2C4	I2C4_POINTRPEND_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_178
	I2C4_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_239
	I2C4_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_239
MCU_I2C0	MCU_I2C0_POINTRPEND_OUT_0	GICSS0_SPI_IN_139
	MCU_I2C0_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_197
	MCU_I2C0_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_197
	MCU_I2C0_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_197
	MCU_I2C0_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_139
	MCU_I2C0_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_139
WKUP_I2C0	WKUP_I2C0_CLKSTOP_WAKEUP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_143
	WKUP_I2C0_CLKSTOP_WAKEUP_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_0
	WKUP_I2C0_POINTRPEND_OUT_0	GICSS0_SPI_IN_197
	WKUP_I2C0_POINTRPEND_OUT_0	R5FSS0_CORE0_INTR_IN_190
	WKUP_I2C0_POINTRPEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_190
	WKUP_I2C0_POINTRPEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_190
	WKUP_I2C0_POINTRPEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_197
	WKUP_I2C0_POINTRPEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_197

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_MSRAM_256K0	MCU_MSRAM_256K0_ECC_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_32
	MCU_MSRAM_256K0_ECC_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_33
MCU_MSRAM_256K1	MCU_MSRAM_256K1_ECC_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_34
	MCU_MSRAM_256K1_ECC_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_35
msram8kx256e0	MSRAM8KX256E0_ECC_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_5
	MSRAM8KX256E0_ECC_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_6
PCIE0	PCIE0_PCIE_CPTS_COMP_OUT_0	EPWM0_EPWM_SYNCIN_IN_0
	PCIE0_PCIE_CPTS_COMP_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_1
	PCIE0_PCIE_CPTS_COMP_OUT_0	GICSS0_SPI_IN_49
	PCIE0_PCIE_CPTS_COMP_OUT_0	R5FSS0_CORE0_INTR_IN_49
	PCIE0_PCIE_CPTS_COMP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_49
	PCIE0_PCIE_CPTS_COMP_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_49
	PCIE0_PCIE_CPTS_COMP_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_49
	PCIE0_PCIE_CPTS_COMP_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_49
	PCIE0_PCIE_CPTS_GENF0_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_4
	PCIE0_PCIE_CPTS_HW1_PUSH_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_5
	PCIE0_PCIE_CPTS_PEND_OUT_0	GICSS0_SPI_IN_121
	PCIE0_PCIE_CPTS_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_53
	PCIE0_PCIE_CPTS_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_53
	PCIE0_PCIE_CPTS_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_121
	PCIE0_PCIE_CPTS_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_121
	PCIE0_PCIE_CPTS_SYNC_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_6
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	GICSS0_SPI_IN_162
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	R5FSS0_CORE0_INTR_IN_99
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_98
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_99
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_162
	PCIE0_PCIE_DOWNSTREAM_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_162
	PCIE0_PCIE_DPA_PULSE_OUT_0	GICSS0_SPI_IN_163
	PCIE0_PCIE_DPA_PULSE_OUT_0	R5FSS0_CORE0_INTR_IN_126
	PCIE0_PCIE_DPA_PULSE_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_99
	PCIE0_PCIE_DPA_PULSE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_126
	PCIE0_PCIE_DPA_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_163

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
PCIE0	PCIE0_PCIE_DPA_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_163
	PCIE0_PCIE_ECC0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_180
	PCIE0_PCIE_ECC0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_181
	PCIE0_PCIE_ECC1_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_182
	PCIE0_PCIE_ERROR_PULSE_OUT_0	GICSS0_SPI_IN_125
	PCIE0_PCIE_ERROR_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_125
	PCIE0_PCIE_ERROR_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_125
	PCIE0_PCIE_FLR_PULSE_OUT_0	GICSS0_SPI_IN_126
	PCIE0_PCIE_FLR_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_126
	PCIE0_PCIE_FLR_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_126
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	GICSS0_SPI_IN_123
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	R5FSS0_CORE0_INTR_IN_55
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_55
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_55
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_123
	PCIE0_PCIE_HOT_RESET_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_123
	PCIE0_PCIE_LEGACY_PULSE_OUT_0	GICSS0_SPI_IN_127
	PCIE0_PCIE_LEGACY_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_127
	PCIE0_PCIE_LEGACY_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_127
	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0	GICSS0_SPI_IN_131
	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_131
	PCIE0_PCIE_LINK_STATE_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_131
	PCIE0_PCIE_LOCAL_LEVEL_OUT_0	GICSS0_SPI_IN_137
	PCIE0_PCIE_LOCAL_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_137
	PCIE0_PCIE_LOCAL_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_137
	PCIE0_PCIE_PHY_LEVEL_OUT_0	GICSS0_SPI_IN_142
	PCIE0_PCIE_PHY_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_142
	PCIE0_PCIE_PHY_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_142
	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	TIMESYNC_EVENT_INTROUTER0_IN_IN_7
	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	GICSS0_SPI_IN_124
	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	R5FSS0_CORE0_INTR_IN_165
	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_165
	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_124

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
PCIE0	PCIE0_PCIE_PTM_VALID_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_124
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	GICSS0_SPI_IN_122
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	R5FSS0_CORE0_INTR_IN_54
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_54
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_54
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_122
	PCIE0_PCIE_PWR_STATE_PULSE_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_122
pllfracf2_ssmod_16fft0	PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_54
	PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_128
pllfracf2_ssmod_16fft1	PLLFRACF2_SSMOD_16FFT1_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_55
	PLLFRACF2_SSMOD_16FFT1_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_129
pllfracf2_ssmod_16fft12	PLLFRACF2_SSMOD_16FFT12_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_58
	PLLFRACF2_SSMOD_16FFT12_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_132
pllfracf2_ssmod_16fft15	PLLFRACF2_SSMOD_16FFT15_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_59
	PLLFRACF2_SSMOD_16FFT15_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_133
pllfracf2_ssmod_16fft16	PLLFRACF2_SSMOD_16FFT16_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_52
	PLLFRACF2_SSMOD_16FFT16_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_27
pllfracf2_ssmod_16fft17	PLLFRACF2_SSMOD_16FFT17_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_62
	PLLFRACF2_SSMOD_16FFT17_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_7
pllfracf2_ssmod_16fft18	PLLFRACF2_SSMOD_16FFT18_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_51
	PLLFRACF2_SSMOD_16FFT18_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_43
pllfracf2_ssmod_16fft2	PLLFRACF2_SSMOD_16FFT2_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_56
	PLLFRACF2_SSMOD_16FFT2_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_130
pllfracf2_ssmod_16fft5	PLLFRACF2_SSMOD_16FFT5_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_60
	PLLFRACF2_SSMOD_16FFT5_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_17
pllfracf2_ssmod_16fft6	PLLFRACF2_SSMOD_16FFT6_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_53
	PLLFRACF2_SSMOD_16FFT6_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_19
pllfracf2_ssmod_16fft7	PLLFRACF2_SSMOD_16FFT7_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_61
	PLLFRACF2_SSMOD_16FFT7_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_18
pllfracf2_ssmod_16fft8	PLLFRACF2_SSMOD_16FFT8_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_57
	PLLFRACF2_SSMOD_16FFT8_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_131
MCU_pllfracf2_ssmod_16fft0	MCU_PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_63
	MCU_PLLFRACF2_SSMOD_16FFT0_LOCKLOSS_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_134

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
PSRAMECC0	PSRAMECC0_ECC_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_29
	PSRAMECC0_ECC_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_90
PSRAMECC1	PSRAMECC1_ECC_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_108
	PSRAMECC1_ECC_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_109
WKUP_PSRAMECC_8K0	WKUP_PSRAMECC_8K0_ECC_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_42
	WKUP_PSRAMECC_8K0_ECC_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_8
	WKUP_PSRAMECC_8K0_ECC_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_43
	WKUP_PSRAMECC_8K0_ECC_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_13
R5FSS0_CORE0	R5FSS0_CORE0_CTI_OUT_0	R5FSS0_CORE0_INTR_IN_175
	R5FSS0_CORE0_ECC_CORRECTED_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_208
	R5FSS0_CORE0_ECC_UNCORRECTED_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_209
	R5FSS0_CORE0_EXP_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_210
	R5FSS0_CORE0_EXP_INTR_OUT_0	R5FSS0_CORE0_INTR_IN_4
	R5FSS0_CORE0_PMU_OUT_0	R5FSS0_CORE0_INTR_IN_94
	R5FSS0_CORE0_VALFIQ_OUT_0	R5FSS0_CORE0_INTR_IN_95
	R5FSS0_CORE0_VALIRQ_OUT_0	R5FSS0_CORE0_INTR_IN_96
R5FSS0_COMMON0	R5FSS0_COMMON0_COMMRX_LEVEL_0_OUT_0	R5FSS0_CORE0_INTR_IN_90
	R5FSS0_COMMON0_COMMTX_LEVEL_0_OUT_0	R5FSS0_CORE0_INTR_IN_91
	R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0	ESM0_ESM_LVL_EVENT_IN_211
	R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0	ESM0_ESM_LVL_EVENT_IN_212
WKUP_R5FSS0_CORE0	WKUP_R5FSS0_CORE0_CTI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_175
	WKUP_R5FSS0_CORE0_ECC_CORRECTED_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_30
	WKUP_R5FSS0_CORE0_ECC_UNCORRECTED_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_91
	WKUP_R5FSS0_CORE0_EXP_INTR_OUT_0	ESM0_ESM_LVL_EVENT_IN_124
	WKUP_R5FSS0_CORE0_EXP_INTR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_4
	WKUP_R5FSS0_CORE0_PMU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_58
	WKUP_R5FSS0_CORE0_VALFIQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_59
	WKUP_R5FSS0_CORE0_VALIRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_60
WKUP_R5FSS0_COMMON0	WKUP_R5FSS0_COMMON0_COMMRX_LEVEL_0_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_5
	WKUP_R5FSS0_COMMON0_COMMTX_LEVEL_0_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_6
	WKUP_R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0	ESM0_ESM_LVL_EVENT_IN_40
	WKUP_R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0	ESM0_ESM_LVL_EVENT_IN_42
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_COMMRX_LEVEL_0_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_90

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_R5FSS0_CORE0	MCU_R5FSS0_CORE0_COMMTX_LEVEL_0_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_91
	MCU_R5FSS0_CORE0_CPU0_CTI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_175
	MCU_R5FSS0_CORE0_CPU0_EXP_INTR_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_30
	MCU_R5FSS0_CORE0_CPU0_EXP_INTR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_4
	MCU_R5FSS0_CORE0_CPU0_PMU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_94
	MCU_R5FSS0_CORE0_CPU0_VALFIQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_95
	MCU_R5FSS0_CORE0_CPU0_VALIRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_96
MCU_R5FSS0_COMMON0	MCU_R5FSS0_COMMON0_ECC_DE_TO_ESM_0_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_28
	MCU_R5FSS0_COMMON0_ECC_SE_TO_ESM_0_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_29
MCU_R5FSS0_CPU0_ECC_AGGR_0	MCU_R5FSS0_CPU0_ECC_AGGR_0_CPU0_ECC_CORRECTED_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_4
	MCU_R5FSS0_CPU0_ECC_AGGR_0_CPU0_ECC_UNCORRECTED_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_5
WKUP_rtcss0	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	GICSS0_SPI_IN_132
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	R5FSS0_CORE0_INTR_IN_97
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_97
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_97
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_7
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_132
	WKUP_RTCSS0_RTC_EVENT_PEND_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_132
RTI4	RTI4_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_236
	RTI4_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_236
	RTI4_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_236
	RTI4_INTR_WWD_OUT_0	C7X256V0_CLEC_SOC_EVENTS_IN_IN_4
RTI5	RTI5_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_239
	RTI5_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_239
	RTI5_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_239
	RTI5_INTR_WWD_OUT_0	C7X256V1_CLEC_SOC_EVENTS_IN_IN_4
RTI15	RTI15_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_248
	RTI15_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_248
	RTI15_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_248
	RTI15_INTR_WWD_OUT_0	GICSS0_SPI_IN_207
	RTI15_INTR_WWD_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_207
	RTI15_INTR_WWD_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_207
RTI0	RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_224

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
RTI0	RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_224
	RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_224
	RTI0_INTR_WWD_OUT_0	GICSS0_SPI_IN_252
	RTI0_INTR_WWD_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_252
	RTI0_INTR_WWD_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_252
RTI1	RTI1_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_225
	RTI1_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_225
	RTI1_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_225
	RTI1_INTR_WWD_OUT_0	GICSS0_SPI_IN_253
	RTI1_INTR_WWD_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_253
	RTI1_INTR_WWD_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_253
RTI2	RTI2_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_241
	RTI2_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_241
	RTI2_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_241
	RTI2_INTR_WWD_OUT_0	GICSS0_SPI_IN_254
	RTI2_INTR_WWD_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_254
	RTI2_INTR_WWD_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_254
RTI3	RTI3_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_242
	RTI3_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_242
	RTI3_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_242
	RTI3_INTR_WWD_OUT_0	GICSS0_SPI_IN_255
	RTI3_INTR_WWD_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_255
	RTI3_INTR_WWD_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_255
RTI8	RTI8_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_249
	RTI8_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_249
	RTI8_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_249
	RTI8_INTR_WWD_OUT_0	R5FSS0_CORE0_INTR_IN_30
MCU_RTI0	MCU_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT0_IN_85
	MCU_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT1_IN_85
	MCU_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT2_IN_85
	MCU_RTI0_INTR_WWD_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_30
WKUP_RTI0	WKUP_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT0_IN_86
	WKUP_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT1_IN_86

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_RTI0	WKUP_RTI0_INTR_WWD_OUT_0	WKUP_ESM0_ESM_PLS_EVENT2_IN_86
	WKUP_RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT0_IN_227
	WKUP_RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT1_IN_227
	WKUP_RTI0_INTR_WWD_OUT_0	ESM0_ESM_PLS_EVENT2_IN_227
	WKUP_RTI0_INTR_WWD_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_30
SA3_SS0_DMSS_ECCAGGR_0	SA3_SS0_DMSS_ECCAGGR_0_DMSS_ECC_DED_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_118
	SA3_SS0_DMSS_ECCAGGR_0_DMSS_ECC_SEC_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_119
SA3_SS0_INTAGGR_0	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4	GICSS0_SPI_IN_112
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_112
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_112
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5	GICSS0_SPI_IN_113
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_113
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_113
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_6	WKUP_R5FSS0_CORE0_INTR_IN_7
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_7	R5FSS0_CORE0_INTR_IN_7
	SA3_SS0_INTAGGR_0_INTAGGR_VINTR_OUT_7	MCU_R5FSS0_CORE0_CPU0_INTR_IN_7
SA3_SS0_SA_UL_0	SA3_SS0_SA_UL_0_SA_UL_ECC_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_120
	SA3_SS0_SA_UL_0_SA_UL_ECC_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_121
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	GICSS0_SPI_IN_160
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	R5FSS0_CORE0_INTR_IN_16
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_16
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_16
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_160
	SA3_SS0_SA_UL_0_SA_UL_PKA_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_160
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	GICSS0_SPI_IN_161
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	R5FSS0_CORE0_INTR_IN_17
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_17
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_17
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_161
	SA3_SS0_SA_UL_0_SA_UL_TRNG_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_161
WKUP_DEEPSLEEP_SOURCES0	WKUP_DEEPSLEEP_SOURCES0_WAKEUP_EVENT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_27
PDMA1	PDMA1_ECC_DED_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_89
	PDMA1_ECC_SEC_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_28

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
COMPUTE_CLUSTER0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0
	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0	C7X256V0_CLEC_ARM_0_EVENTO_IN_IN_0
	COMPUTE_CLUSTER0_CPU_EVNT_EVENTO_OUT_0	C7X256V1_CLEC_ARM_0_EVENTO_IN_IN_0
A53SS0	A53SS0_CNTHPIRQ0_OUT_0	GICSS0_PPIO_0_IN_26
	A53SS0_CNTHPIRQ1_OUT_0	GICSS0_PPIO_1_IN_26
	A53SS0_CNTHPIRQ2_OUT_0	GICSS0_PPIO_2_IN_26
	A53SS0_CNTHPIRQ3_OUT_0	GICSS0_PPIO_3_IN_26
	A53SS0_CNTPNSIRQ0_OUT_0	GICSS0_PPIO_0_IN_30
	A53SS0_CNTPNSIRQ1_OUT_0	GICSS0_PPIO_1_IN_30
	A53SS0_CNTPNSIRQ2_OUT_0	GICSS0_PPIO_2_IN_30
	A53SS0_CNTPNSIRQ3_OUT_0	GICSS0_PPIO_3_IN_30
	A53SS0_CNTPSIRQ0_OUT_0	GICSS0_PPIO_0_IN_29
	A53SS0_CNTPSIRQ1_OUT_0	GICSS0_PPIO_1_IN_29
	A53SS0_CNTPSIRQ2_OUT_0	GICSS0_PPIO_2_IN_29
	A53SS0_CNTPSIRQ3_OUT_0	GICSS0_PPIO_3_IN_29
	A53SS0_CNTVIRQ0_OUT_0	GICSS0_PPIO_0_IN_27
	A53SS0_CNTVIRQ1_OUT_0	GICSS0_PPIO_1_IN_27
	A53SS0_CNTVIRQ2_OUT_0	GICSS0_PPIO_2_IN_27
	A53SS0_CNTVIRQ3_OUT_0	GICSS0_PPIO_3_IN_27
	A53SS0_COMMIRQ0_OUT_0	GICSS0_PPIO_0_IN_22
	A53SS0_COMMIRQ1_OUT_0	GICSS0_PPIO_1_IN_22
	A53SS0_COMMIRQ2_OUT_0	GICSS0_PPIO_2_IN_22
	A53SS0_COMMIRQ3_OUT_0	GICSS0_PPIO_3_IN_22
	A53SS0_CTIIRQ0_OUT_0	GICSS0_PPIO_0_IN_24
	A53SS0_CTIIRQ0_OUT_0	GICSS0_PPIO_1_IN_17
	A53SS0_CTIIRQ0_OUT_0	GICSS0_PPIO_2_IN_17
	A53SS0_CTIIRQ0_OUT_0	GICSS0_PPIO_3_IN_17
	A53SS0_CTIIRQ1_OUT_0	GICSS0_PPIO_0_IN_18
	A53SS0_CTIIRQ1_OUT_0	GICSS0_PPIO_1_IN_24
	A53SS0_CTIIRQ1_OUT_0	GICSS0_PPIO_2_IN_18
	A53SS0_CTIIRQ1_OUT_0	GICSS0_PPIO_3_IN_18
	A53SS0_CTIIRQ2_OUT_0	GICSS0_PPIO_0_IN_19
	A53SS0_CTIIRQ2_OUT_0	GICSS0_PPIO_1_IN_19

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
A53SS0	A53SS0_CTIIRQ2_OUT_0	GICSS0_PPI0_2_IN_24
	A53SS0_CTIIRQ2_OUT_0	GICSS0_PPI0_3_IN_19
	A53SS0_CTIIRQ3_OUT_0	GICSS0_PPI0_0_IN_20
	A53SS0_CTIIRQ3_OUT_0	GICSS0_PPI0_1_IN_20
	A53SS0_CTIIRQ3_OUT_0	GICSS0_PPI0_2_IN_20
	A53SS0_CTIIRQ3_OUT_0	GICSS0_PPI0_3_IN_24
	A53SS0_ECC_ECCAGGR0_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_24
	A53SS0_ECC_ECCAGGR0_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_94
	A53SS0_ECC_ECCAGGR1_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_25
	A53SS0_ECC_ECCAGGR1_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_93
	A53SS0_ECC_ECCAGGR2_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_45
	A53SS0_ECC_ECCAGGR2_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_46
	A53SS0_ECC_ECCAGGR3_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_47
	A53SS0_ECC_ECCAGGR3_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_48
	A53SS0_ECC_ECCAGGR_COREPAC_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_26
	A53SS0_ECC_ECCAGGR_COREPAC_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_95
	A53SS0_EXTERRIRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_144
	A53SS0_INTERRIRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_145
	A53SS0_PMUIRQ0_OUT_0	GICSS0_PPI0_0_IN_23
	A53SS0_PMUIRQ1_OUT_0	GICSS0_PPI0_1_IN_23
	A53SS0_PMUIRQ2_OUT_0	GICSS0_PPI0_2_IN_23
	A53SS0_PMUIRQ3_OUT_0	GICSS0_PPI0_3_IN_23
	A53SS0_VCPUMNTIRQ0_OUT_0	GICSS0_PPI0_0_IN_25
	A53SS0_VCPUMNTIRQ1_OUT_0	GICSS0_PPI0_1_IN_25
	A53SS0_VCPUMNTIRQ2_OUT_0	GICSS0_PPI0_2_IN_25
	A53SS0_VCPUMNTIRQ3_OUT_0	GICSS0_PPI0_3_IN_25
COMPUTE_CLUSTER0_PBIST_0	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_240
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_240
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_240
	COMPUTE_CLUSTER0_PBIST_0_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_44

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DEBUGSS0	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	GICSS0_SPI_IN_201
	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_201
	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_201
	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_201
	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_201
	DEBUGSS0_AQCMPINTR_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_201
	DEBUGSS0_CTM_LEVEL_OUT_0	GICSS0_SPI_IN_202
	DEBUGSS0_CTM_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_202
	DEBUGSS0_CTM_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_202
	DEBUGSS0_CTM_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_202
	DEBUGSS0_CTM_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_202
	DEBUGSS0_CTM_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_202
	DEBUGSS0_DAVDMA_LEVEL_OUT_0	DMASS0_INTAGGR_0_INTAGGR_LEVI_PEND_IN_27
	DEBUGSS0_DAVDMA_LEVEL_OUT_0	USB0_TRACE_INEP_PKT_BUFF_AVAIL_IN_0
MCU_MCU_16FF0	MCU_MCU_16FF0_VDD_CORE_GLDTC_STAT_THRESH_HI_FLAG_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_101
	MCU_MCU_16FF0_VDD_CORE_GLDTC_STAT_THRESH_LOW_FLAG_IPCFG_OUT_0	ESM0_ESM_LVL_EVENT_IN_102
WKUP_PSC0	WKUP_PSC0_PSC_ALLINT_OUT_0	R5FSS0_CORE0_INTR_IN_145
	WKUP_PSC0_PSC_ALLINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_145
	WKUP_PSC0_PSC_ALLINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_145
MCU_ECC_AGGR1	MCU_ECC_AGGR1_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_40
	MCU_ECC_AGGR1_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_41
MCU_PRG_MCU_5POKS0	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT0_IN_64
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT1_IN_64
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT2_IN_64
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT0_IN_65
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT1_IN_65
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT2_IN_65
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT0_IN_66
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT1_IN_66
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT2_IN_66
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT0_IN_69
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT1_IN_69
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT2_IN_69

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_PRG_MCU_5POKS0	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT0_IN_70
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT1_IN_70
	MCU_PRG_MCU_5POKS0_POK_PGOOD_OV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT2_IN_70
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT0_IN_71
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT1_IN_71
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT2_IN_71
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT0_IN_72
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT1_IN_72
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT2_IN_72
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT0_IN_73
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT1_IN_73
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT2_IN_73
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT0_IN_76
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT1_IN_76
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT2_IN_76
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT0_IN_77
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT1_IN_77
	MCU_PRG_MCU_5POKS0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT2_IN_77
MCU_PRG_MCU0	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT0_IN_78
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT1_IN_78
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_0	WKUP_ESM0_ESM_PLS_EVENT2_IN_78
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT0_IN_79
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT1_IN_79
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_1	WKUP_ESM0_ESM_PLS_EVENT2_IN_79
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT0_IN_80
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT1_IN_80
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_2	WKUP_ESM0_ESM_PLS_EVENT2_IN_80
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT0_IN_81
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT1_IN_81
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_3	WKUP_ESM0_ESM_PLS_EVENT2_IN_81
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT0_IN_82
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT1_IN_82
	MCU_PRG_MCU0_POK_PGOOD_UV_OUT_N_TO_ESM_OUT_4	WKUP_ESM0_ESM_PLS_EVENT2_IN_82

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_ARM_0_EVENTI_OUT_OUT_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_226
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_226
	C7X256V0_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_226
	C7X256V0_CLEC_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_149
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_160
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_1	ESM0_ESM_LVL_EVENT_IN_161
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_2	ESM0_ESM_LVL_EVENT_IN_162
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_3	ESM0_ESM_LVL_EVENT_IN_163
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_4	ESM0_ESM_LVL_EVENT_IN_164
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_5	ESM0_ESM_LVL_EVENT_IN_165
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_6	ESM0_ESM_LVL_EVENT_IN_166
	C7X256V0_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_7	ESM0_ESM_LVL_EVENT_IN_167
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_0	GICSS0_SPI_IN_280
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_1	GICSS0_SPI_IN_281
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_2	GICSS0_SPI_IN_282
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_3	GICSS0_SPI_IN_283
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_4	GICSS0_SPI_IN_284
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_5	GICSS0_SPI_IN_285
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_6	GICSS0_SPI_IN_286
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_7	GICSS0_SPI_IN_287
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_8	MCU_R5FSS0_CORE0_CPU0_INTR_IN_249
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_9	MCU_R5FSS0_CORE0_CPU0_INTR_IN_250
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_10	MCU_R5FSS0_CORE0_CPU0_INTR_IN_254
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_11	MCU_R5FSS0_CORE0_CPU0_INTR_IN_255
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_12	WKUP_R5FSS0_CORE0_INTR_IN_249
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_13	WKUP_R5FSS0_CORE0_INTR_IN_250
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_14	WKUP_R5FSS0_CORE0_INTR_IN_254
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_15	WKUP_R5FSS0_CORE0_INTR_IN_255
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_20	C7X256V1_CLEC_SOC_EVENTS_IN_IN_12

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
C7X256V0_CLEC	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_21	C7X256V1_CLEC_SOC_EVENTS_IN_IN_13
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_22	C7X256V1_CLEC_SOC_EVENTS_IN_IN_14
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_23	C7X256V1_CLEC_SOC_EVENTS_IN_IN_15
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_24	R5FSS0_CORE0_INTR_IN_249
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_25	R5FSS0_CORE0_INTR_IN_250
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_26	R5FSS0_CORE0_INTR_IN_254
	C7X256V0_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_27	R5FSS0_CORE0_INTR_IN_255
C7X256V1_CLEC	C7X256V1_CLEC_ARM_0_EVENTI_OUT_OUT_0	COMPUTE_CLUSTER0_CPU_EVNT_EVENTI_IN_0
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_245
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_245
	C7X256V1_CLEC_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_245
	C7X256V1_CLEC_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_64
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_192
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_1	ESM0_ESM_LVL_EVENT_IN_193
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_2	ESM0_ESM_LVL_EVENT_IN_194
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_3	ESM0_ESM_LVL_EVENT_IN_195
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_4	ESM0_ESM_LVL_EVENT_IN_196
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_5	ESM0_ESM_LVL_EVENT_IN_197
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_6	ESM0_ESM_LVL_EVENT_IN_198
	C7X256V1_CLEC_ESM_EVENTS_OUT_LEVEL_OUT_7	ESM0_ESM_LVL_EVENT_IN_199
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_0	GICSS0_SPI_IN_50
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_1	GICSS0_SPI_IN_51
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_2	GICSS0_SPI_IN_52
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_3	GICSS0_SPI_IN_53
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_4	GICSS0_SPI_IN_54
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_5	GICSS0_SPI_IN_55
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_6	GICSS0_SPI_IN_56
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_7	GICSS0_SPI_IN_57
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_8	MCU_R5FSS0_CORE0_CPU0_INTR_IN_244
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_9	MCU_R5FSS0_CORE0_CPU0_INTR_IN_245

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
C7X256V1_CLEC	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_10	MCU_R5FSS0_CORE0_CPU0_INTR_IN_246
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_11	MCU_R5FSS0_CORE0_CPU0_INTR_IN_247
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_12	WKUP_R5FSS0_CORE0_INTR_IN_120
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_13	WKUP_R5FSS0_CORE0_INTR_IN_121
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_14	WKUP_R5FSS0_CORE0_INTR_IN_122
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_15	WKUP_R5FSS0_CORE0_INTR_IN_165
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_20	C7X256V0_CLEC_SOC_EVENTS_IN_IN_12
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_21	C7X256V0_CLEC_SOC_EVENTS_IN_IN_13
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_22	C7X256V0_CLEC_SOC_EVENTS_IN_IN_14
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_23	C7X256V0_CLEC_SOC_EVENTS_IN_IN_15
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_24	R5FSS0_CORE0_INTR_IN_244
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_25	R5FSS0_CORE0_INTR_IN_245
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_26	R5FSS0_CORE0_INTR_IN_246
	C7X256V1_CLEC_SOC_EVENTS_OUT_LEVEL_OUT_27	R5FSS0_CORE0_INTR_IN_247
DDR32SS0	DDR32SS0_DDRSS_CONTROLLER_OUT_0	GICSS0_SPI_IN_151
	DDR32SS0_DDRSS_CONTROLLER_OUT_0	R5FSS0_CORE0_INTR_IN_151
	DDR32SS0_DDRSS_CONTROLLER_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_151
	DDR32SS0_DDRSS_CONTROLLER_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_151
	DDR32SS0_DDRSS_CONTROLLER_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_151
	DDR32SS0_DDRSS_CONTROLLER_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_151
	DDR32SS0_DDRSS_DRAM_ECC_CORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_174
	DDR32SS0_DDRSS_DRAM_ECC_UNCORR_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_175
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	GICSS0_SPI_IN_172
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	R5FSS0_CORE0_INTR_IN_181
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_181
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_181
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_172
	DDR32SS0_DDRSS_PLL_FREQ_CHANGE_REQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_172
	DDR32SS0_DDRSS_V2A_OTHER_ERR_LVL_OUT_0	ESM0_ESM_LVL_EVENT_IN_176
DFTSS0	DFTSS0_DFT_SAFETY_123_OUT_0	ESM0_ESM_LVL_EVENT_IN_98
	DFTSS0_DFT_SAFETY_MULTI_OUT_0	ESM0_ESM_LVL_EVENT_IN_99
	DFTSS0_DFT_SAFETY_ONE_OUT_0	ESM0_ESM_LVL_EVENT_IN_100
WKUP_TIMEOUT1	WKUP_TIMEOUT1_TIMED_OUT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_26

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_TIMEOUT1	WKUP_TIMEOUT1_TIMED_OUT_OUT_0	R5FSS0_CORE0_INTR_IN_152
	WKUP_TIMEOUT1_TIMED_OUT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_152
	WKUP_TIMEOUT1_TIMED_OUT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_152
WKUP_TIMEOUT0	WKUP_TIMEOUT0_TIMED_OUT_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_26
	WKUP_TIMEOUT0_TIMED_OUT_OUT_0	R5FSS0_CORE0_INTR_IN_152
	WKUP_TIMEOUT0_TIMED_OUT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_152
	WKUP_TIMEOUT0_TIMED_OUT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_152
WKUP_ECC_AGGR0	WKUP_ECC_AGGR0_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_23
	WKUP_ECC_AGGR0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_20
	WKUP_ECC_AGGR0_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_24
	WKUP_ECC_AGGR0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_21
WKUP_ECC_AGGR1	WKUP_ECC_AGGR1_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_38
	WKUP_ECC_AGGR1_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_104
	WKUP_ECC_AGGR1_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_39
	WKUP_ECC_AGGR1_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_105
DMPAC0	DMPAC0_DFT_LBIST_BIST_DONE_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_3
	DMPAC0_DFT_LBIST_BIST_DONE_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_3
	DMPAC0_DMPAC_LEVEL_OUT_0	MAIN_GPIOMUX_INTROUTER0_IN_IN_196
	DMPAC0_DMPAC_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_5
	DMPAC0_DMPAC_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_62
	DMPAC0_DMPAC_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_5
	DMPAC0_DMPAC_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_50
	DMPAC0_DMPAC_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_50
	DMPAC0_DMPAC_LEVEL_OUT_1	MAIN_GPIOMUX_INTROUTER0_IN_IN_197
	DMPAC0_DMPAC_LEVEL_OUT_1	R5FSS0_CORE0_INTR_IN_6
	DMPAC0_DMPAC_LEVEL_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_96
	DMPAC0_DMPAC_LEVEL_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_6
	DMPAC0_DMPAC_LEVEL_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_51
	DMPAC0_DMPAC_LEVEL_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_51
	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT0_IN_250
	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT1_IN_250
	DMPAC0_ECC_CORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT2_IN_250
	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT0_IN_251

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
DMPAC0	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT1_IN_251
	DMPAC0_ECC_UNCORRECTED_ERR_PULSE_OUT_0	ESM0_ESM_PLS_EVENT2_IN_251
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_244
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_244
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_244
	DMPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_106
GPU0	GPU0_DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	GPU0_DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	GPU0_DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	GPU0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_238
	GPU0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_238
	GPU0_DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_238
	GPU0_DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_156
	GPU0_GPU_PWRCTRL_REQ_OUT_0	GICSS0_SPI_IN_120
	GPU0_GPU_PWRCTRL_REQ_OUT_0	R5FSS0_CORE0_INTR_IN_248
	GPU0_GPU_PWRCTRL_REQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_31
	GPU0_GPU_PWRCTRL_REQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_248
	GPU0_GPU_PWRCTRL_REQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_120
	GPU0_GPU_PWRCTRL_REQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_120
	GPU0_GPU_SAFETY_IRQ_OUT_0	ESM0_ESM_LVL_EVENT_IN_184
	GPU0_OS_IRQ_OUT_0	GICSS0_SPI_IN_273
	GPU0_OS_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_273
	GPU0_OS_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_273
	GPU0_OS_IRQ_OUT_1	GICSS0_SPI_IN_274
	GPU0_OS_IRQ_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_274
	GPU0_OS_IRQ_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_274
	GPU0_OS_IRQ_OUT_2	GICSS0_SPI_IN_275
	GPU0_OS_IRQ_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_275
	GPU0_OS_IRQ_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_275
	GPU0_OS_IRQ_OUT_3	GICSS0_SPI_IN_276

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
GPU0	GPU0_OS_IRQ_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_276
	GPU0_OS_IRQ_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_276
PSC0_ECC_AGGR_0	PSC0_ECC_AGGR_0_FW_CH_BR_ECC_AGGR_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_22
	PSC0_ECC_AGGR_0_FW_CH_BR_ECC_AGGR_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_23
PSC0_FW_0	PSC0_FW_0_FW_CH_BR_DEFAULT_EXP_OUT_0	ESM0_ESM_LVL_EVENT_IN_76
PSC0	PSC0_PSC_ALLINT_OUT_0	GICSS0_SPI_IN_203
	PSC0_PSC_ALLINT_OUT_0	R5FSS0_CORE0_INTR_IN_146
	PSC0_PSC_ALLINT_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_146
	PSC0_PSC_ALLINT_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_146
	PSC0_PSC_ALLINT_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_203
	PSC0_PSC_ALLINT_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_203
MCU_TIMEOUT0	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_27
	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_153
	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_153
	MCU_TIMEOUT0_TRANS_ERR_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_153
MCU_ECC_AGGR0	MCU_ECC_AGGR0_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_14
	MCU_ECC_AGGR0_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_15
PDMA0	PDMA0_ECC_DED_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_88
	PDMA0_ECC_SEC_PEND_OUT_0	ESM0_ESM_LVL_EVENT_IN_15
ECC_AGGR0	ECC_AGGR0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_2
	ECC_AGGR0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_1
VPAC0	VPAC0_ECC_INTR0_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_168
	VPAC0_ECC_INTR0_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_169
	VPAC0_ECC_INTR1_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_170
	VPAC0_ECC_INTR1_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_171
	VPAC0_ECC_INTR3_CORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_172
	VPAC0_ECC_INTR3_UNCORR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_173
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_237
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_237
	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_237

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
VPAC0	VPAC0_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_177
	VPAC0_VPAC_LEVEL_OUT_0	GICSS0_SPI_IN_168
	VPAC0_VPAC_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_178
	VPAC0_VPAC_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_178
	VPAC0_VPAC_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_178
	VPAC0_VPAC_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_168
	VPAC0_VPAC_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_168
	VPAC0_VPAC_LEVEL_OUT_1	GICSS0_SPI_IN_169
	VPAC0_VPAC_LEVEL_OUT_1	R5FSS0_CORE0_INTR_IN_179
	VPAC0_VPAC_LEVEL_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_179
	VPAC0_VPAC_LEVEL_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_179
	VPAC0_VPAC_LEVEL_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_169
	VPAC0_VPAC_LEVEL_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_169
	VPAC0_VPAC_LEVEL_OUT_2	GICSS0_SPI_IN_170
	VPAC0_VPAC_LEVEL_OUT_2	R5FSS0_CORE0_INTR_IN_180
	VPAC0_VPAC_LEVEL_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_180
	VPAC0_VPAC_LEVEL_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_180
	VPAC0_VPAC_LEVEL_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_170
	VPAC0_VPAC_LEVEL_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_170
	VPAC0_VPAC_LEVEL_OUT_3	GICSS0_SPI_IN_189
	VPAC0_VPAC_LEVEL_OUT_3	R5FSS0_CORE0_INTR_IN_182
	VPAC0_VPAC_LEVEL_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_182
	VPAC0_VPAC_LEVEL_OUT_3	MCU_R5FSS0_CORE0_CPU0_INTR_IN_182
	VPAC0_VPAC_LEVEL_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_189
	VPAC0_VPAC_LEVEL_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_189
	VPAC0_VPAC_LEVEL_OUT_4	GICSS0_SPI_IN_190
	VPAC0_VPAC_LEVEL_OUT_4	R5FSS0_CORE0_INTR_IN_189
	VPAC0_VPAC_LEVEL_OUT_4	WKUP_R5FSS0_CORE0_INTR_IN_189
	VPAC0_VPAC_LEVEL_OUT_4	MCU_R5FSS0_CORE0_CPU0_INTR_IN_189
	VPAC0_VPAC_LEVEL_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_190
	VPAC0_VPAC_LEVEL_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_190
	VPAC0_VPAC_LEVEL_OUT_5	GICSS0_SPI_IN_191
	VPAC0_VPAC_LEVEL_OUT_5	R5FSS0_CORE0_INTR_IN_191

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
VPAC0	VPAC0_VPAC_LEVEL_OUT_5	WKUP_R5FSS0_CORE0_INTR_IN_191
	VPAC0_VPAC_LEVEL_OUT_5	MCU_R5FSS0_CORE0_CPU0_INTR_IN_191
	VPAC0_VPAC_LEVEL_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_191
	VPAC0_VPAC_LEVEL_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_191
PBIST3	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	R5FSS0_CORE0_INTR_IN_113
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_113
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_113
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT0_IN_232
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT1_IN_232
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_CPU_OUT_0	ESM0_ESM_PLS_EVENT2_IN_232
	PBIST3_K3_PBIST_8C28P_4BIT_WRAP__DFT_PBIST_SAFETY_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_148
WKUP_ECC_AGGR2	WKUP_ECC_AGGR2_CORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_20
	WKUP_ECC_AGGR2_UNCORR_LEVEL_OUT_0	WKUP_ESM0_ESM_LVL_EVENT_IN_21
MCSPI0	MCSPI0_DMA_READ_EVENT_OUT_0	PDMA0_SPI_MAIN_0_RX_IN_0
	MCSPI0_DMA_READ_EVENT_OUT_1	PDMA0_SPI_MAIN_0_RX_IN_1
	MCSPI0_DMA_READ_EVENT_OUT_2	PDMA0_SPI_MAIN_0_RX_IN_2
	MCSPI0_DMA_READ_EVENT_OUT_3	PDMA0_SPI_MAIN_0_RX_IN_3
	MCSPI0_DMA_WRITE_EVENT_OUT_0	PDMA0_SPI_MAIN_0_TX_IN_0
	MCSPI0_DMA_WRITE_EVENT_OUT_1	PDMA0_SPI_MAIN_0_TX_IN_1
	MCSPI0_DMA_WRITE_EVENT_OUT_2	PDMA0_SPI_MAIN_0_TX_IN_2
	MCSPI0_DMA_WRITE_EVENT_OUT_3	PDMA0_SPI_MAIN_0_TX_IN_3
	MCSPI0_INTR_SPI_OUT_0	GICSS0_SPI_IN_204
	MCSPI0_INTR_SPI_OUT_0	R5FSS0_CORE0_INTR_IN_204
	MCSPI0_INTR_SPI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_204
	MCSPI0_INTR_SPI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_204
	MCSPI0_INTR_SPI_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_204
	MCSPI0_INTR_SPI_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_204
MCSPI1	MCSPI1_DMA_READ_EVENT_OUT_0	PDMA0_SPI_MAIN_1_RX_IN_0
	MCSPI1_DMA_READ_EVENT_OUT_1	PDMA0_SPI_MAIN_1_RX_IN_1
	MCSPI1_DMA_READ_EVENT_OUT_2	PDMA0_SPI_MAIN_1_RX_IN_2
	MCSPI1_DMA_READ_EVENT_OUT_3	PDMA0_SPI_MAIN_1_RX_IN_3
	MCSPI1_DMA_WRITE_EVENT_OUT_0	PDMA0_SPI_MAIN_1_TX_IN_0
	MCSPI1_DMA_WRITE_EVENT_OUT_1	PDMA0_SPI_MAIN_1_TX_IN_1

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCSPI1	MCSPI1_DMA_WRITE_EVENT_OUT_2	PDMA0_SPI_MAIN_1_TX_IN_2
	MCSPI1_DMA_WRITE_EVENT_OUT_3	PDMA0_SPI_MAIN_1_TX_IN_3
	MCSPI1_INTR_SPI_OUT_0	GICSS0_SPI_IN_205
	MCSPI1_INTR_SPI_OUT_0	R5FSS0_CORE0_INTR_IN_205
	MCSPI1_INTR_SPI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_205
	MCSPI1_INTR_SPI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_205
	MCSPI1_INTR_SPI_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_205
	MCSPI1_INTR_SPI_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_205
MCSPI2	MCSPI2_DMA_READ_EVENT_OUT_0	PDMA0_SPI_MAIN_2_RX_IN_0
	MCSPI2_DMA_READ_EVENT_OUT_1	PDMA0_SPI_MAIN_2_RX_IN_1
	MCSPI2_DMA_READ_EVENT_OUT_2	PDMA0_SPI_MAIN_2_RX_IN_2
	MCSPI2_DMA_READ_EVENT_OUT_3	PDMA0_SPI_MAIN_2_RX_IN_3
	MCSPI2_DMA_WRITE_EVENT_OUT_0	PDMA0_SPI_MAIN_2_TX_IN_0
	MCSPI2_DMA_WRITE_EVENT_OUT_1	PDMA0_SPI_MAIN_2_TX_IN_1
	MCSPI2_DMA_WRITE_EVENT_OUT_2	PDMA0_SPI_MAIN_2_TX_IN_2
	MCSPI2_DMA_WRITE_EVENT_OUT_3	PDMA0_SPI_MAIN_2_TX_IN_3
	MCSPI2_INTR_SPI_OUT_0	GICSS0_SPI_IN_206
	MCSPI2_INTR_SPI_OUT_0	R5FSS0_CORE0_INTR_IN_206
	MCSPI2_INTR_SPI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_206
	MCSPI2_INTR_SPI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_206
	MCSPI2_INTR_SPI_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_206
	MCSPI2_INTR_SPI_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_206
MCU_MCSPI0	MCU_MCSPI0_INTR_SPI_OUT_0	GICSS0_SPI_IN_208
	MCU_MCSPI0_INTR_SPI_OUT_0	R5FSS0_CORE0_INTR_IN_207
	MCU_MCSPI0_INTR_SPI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_207
	MCU_MCSPI0_INTR_SPI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_207
	MCU_MCSPI0_INTR_SPI_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_208
	MCU_MCSPI0_INTR_SPI_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_208
MCU_MCSPI1	MCU_MCSPI1_INTR_SPI_OUT_0	GICSS0_SPI_IN_209
	MCU_MCSPI1_INTR_SPI_OUT_0	R5FSS0_CORE0_INTR_IN_208
	MCU_MCSPI1_INTR_SPI_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_208
	MCU_MCSPI1_INTR_SPI_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_208
	MCU_MCSPI1_INTR_SPI_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_209

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
MCU_MCSPI1	MCU_MCSPI1_INTR_SPI_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_209
UART0	UART0_USART_DMA_OUT_0	PDMA1_USART_MAIN_0_TX_IN_0
	UART0_USART_DMA_OUT_1	PDMA1_USART_MAIN_0_RX_IN_0
	UART0_USART_IRQ_OUT_0	GICSS0_SPI_IN_210
	UART0_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_210
	UART0_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_210
	UART0_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_210
	UART0_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_210
	UART0_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_210
UART1	UART1_USART_DMA_OUT_0	PDMA1_USART_MAIN_1_TX_IN_0
	UART1_USART_DMA_OUT_1	PDMA1_USART_MAIN_1_RX_IN_0
	UART1_USART_IRQ_OUT_0	GICSS0_SPI_IN_211
	UART1_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_211
	UART1_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_211
	UART1_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_211
	UART1_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_211
	UART1_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_211
UART2	UART2_USART_DMA_OUT_0	PDMA1_USART_MAIN_2_TX_IN_0
	UART2_USART_DMA_OUT_1	PDMA1_USART_MAIN_2_RX_IN_0
	UART2_USART_IRQ_OUT_0	GICSS0_SPI_IN_212
	UART2_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_212
	UART2_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_212
	UART2_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_212
	UART2_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_212
	UART2_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_212
UART3	UART3_USART_DMA_OUT_0	PDMA1_USART_MAIN_3_TX_IN_0
	UART3_USART_DMA_OUT_1	PDMA1_USART_MAIN_3_RX_IN_0
	UART3_USART_IRQ_OUT_0	GICSS0_SPI_IN_213
	UART3_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_213
	UART3_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_213
	UART3_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_213
	UART3_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_213
	UART3_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_213

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
UART4	UART4_USART_DMA_OUT_0	PDMA1_USART_MAIN_4_TX_IN_0
	UART4_USART_DMA_OUT_1	PDMA1_USART_MAIN_4_RX_IN_0
	UART4_USART_IRQ_OUT_0	GICSS0_SPI_IN_214
	UART4_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_214
	UART4_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_214
	UART4_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_214
	UART4_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_214
	UART4_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_214
UART5	UART5_USART_DMA_OUT_0	PDMA1_USART_MAIN_5_TX_IN_0
	UART5_USART_DMA_OUT_1	PDMA1_USART_MAIN_5_RX_IN_0
	UART5_USART_IRQ_OUT_0	GICSS0_SPI_IN_215
	UART5_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_215
	UART5_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_215
	UART5_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_215
	UART5_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_215
	UART5_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_215
UART6	UART6_USART_DMA_OUT_0	PDMA1_USART_MAIN_6_TX_IN_0
	UART6_USART_DMA_OUT_1	PDMA1_USART_MAIN_6_RX_IN_0
	UART6_USART_IRQ_OUT_0	GICSS0_SPI_IN_216
	UART6_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_216
	UART6_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_216
	UART6_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_216
	UART6_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_216
	UART6_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_216
MCU_UART0	MCU_UART0_USART_IRQ_OUT_0	GICSS0_SPI_IN_217
	MCU_UART0_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_217
	MCU_UART0_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_217
	MCU_UART0_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_217
	MCU_UART0_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_217
	MCU_UART0_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_217
WKUP_UART0	WKUP_UART0_CLKSTOP_WAKEUP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_144
	WKUP_UART0_CLKSTOP_WAKEUP_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_1
	WKUP_UART0_USART_IRQ_OUT_0	GICSS0_SPI_IN_218

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
WKUP_UART0	WKUP_UART0_USART_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_219
	WKUP_UART0_USART_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_219
	WKUP_UART0_USART_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_219
	WKUP_UART0_USART_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_218
	WKUP_UART0_USART_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_218
USB0	USB0_A_ECC_AGGR_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_35
	USB0_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_111
	USB0_HOST_SYSTEM_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_32
	USB0_IRQ_OUT_0	GICSS0_SPI_IN_220
	USB0_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_220
	USB0_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_220
	USB0_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_220
	USB0_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_220
	USB0_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_220
	USB0_IRQ_OUT_1	GICSS0_SPI_IN_221
	USB0_IRQ_OUT_1	R5FSS0_CORE0_INTR_IN_221
	USB0_IRQ_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_221
	USB0_IRQ_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_221
	USB0_IRQ_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_221
	USB0_IRQ_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_221
	USB0_IRQ_OUT_2	GICSS0_SPI_IN_222
	USB0_IRQ_OUT_2	R5FSS0_CORE0_INTR_IN_222
	USB0_IRQ_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_222
	USB0_IRQ_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_222
	USB0_IRQ_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_222
	USB0_IRQ_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_222
	USB0_IRQ_OUT_3	GICSS0_SPI_IN_223
	USB0_IRQ_OUT_3	R5FSS0_CORE0_INTR_IN_223
	USB0_IRQ_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_223
	USB0_IRQ_OUT_3	MCU_R5FSS0_CORE0_CPU0_INTR_IN_223
	USB0_IRQ_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_223
	USB0_IRQ_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_223
	USB0_IRQ_OUT_4	GICSS0_SPI_IN_224

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
USB0	USB0_IRQ_OUT_4	R5FSS0_CORE0_INTR_IN_224
	USB0_IRQ_OUT_4	WKUP_R5FSS0_CORE0_INTR_IN_224
	USB0_IRQ_OUT_4	MCU_R5FSS0_CORE0_CPU0_INTR_IN_224
	USB0_IRQ_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_224
	USB0_IRQ_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_224
	USB0_IRQ_OUT_5	GICSS0_SPI_IN_225
	USB0_IRQ_OUT_5	R5FSS0_CORE0_INTR_IN_225
	USB0_IRQ_OUT_5	WKUP_R5FSS0_CORE0_INTR_IN_225
	USB0_IRQ_OUT_5	MCU_R5FSS0_CORE0_CPU0_INTR_IN_225
	USB0_IRQ_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_225
	USB0_IRQ_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_225
	USB0_IRQ_OUT_6	GICSS0_SPI_IN_226
	USB0_IRQ_OUT_6	R5FSS0_CORE0_INTR_IN_226
	USB0_IRQ_OUT_6	WKUP_R5FSS0_CORE0_INTR_IN_226
	USB0_IRQ_OUT_6	MCU_R5FSS0_CORE0_CPU0_INTR_IN_226
	USB0_IRQ_OUT_6	C7X256V0_CLEC_GIC_SPI_IN_226
	USB0_IRQ_OUT_6	C7X256V1_CLEC_GIC_SPI_IN_226
	USB0_IRQ_OUT_7	GICSS0_SPI_IN_227
	USB0_IRQ_OUT_7	R5FSS0_CORE0_INTR_IN_227
	USB0_IRQ_OUT_7	WKUP_R5FSS0_CORE0_INTR_IN_227
	USB0_IRQ_OUT_7	MCU_R5FSS0_CORE0_CPU0_INTR_IN_227
	USB0_IRQ_OUT_7	C7X256V0_CLEC_GIC_SPI_IN_227
	USB0_IRQ_OUT_7	C7X256V1_CLEC_GIC_SPI_IN_227
	USB0_MISC_LEVEL_OUT_0	GICSS0_SPI_IN_228
	USB0_MISC_LEVEL_OUT_0	R5FSS0_CORE0_INTR_IN_228
	USB0_MISC_LEVEL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_228
	USB0_MISC_LEVEL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_228
	USB0_MISC_LEVEL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_228
	USB0_MISC_LEVEL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_228
	USB0_USB_WAKEUP_CLKSTOP_WAKEUP_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_61
	USB0_USB_WAKEUP_CLKSTOP_WAKEUP_OUT_0	WKUP_DEEPSLEEP_SOURCES0_ISAM62_DM_WAKEUP_DEEPSLEEP_SOURCES_IN_9
USB1	USB1_A_ECC_AGGR_CORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_146
	USB1_A_ECC_AGGR_UNCORRECTED_ERR_LEVEL_OUT_0	ESM0_ESM_LVL_EVENT_IN_147

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
USB1	USB1_ASF_INT_FATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_143
	USB1_ASF_INT_NONFATAL_OUT_0	ESM0_ESM_LVL_EVENT_IN_155
	USB1_HOST_SYSTEM_ERROR_OUT_0	ESM0_ESM_LVL_EVENT_IN_33
	USB1_HOST_SYSTEM_ERROR_OUT_0	GICSS0_SPI_IN_266
	USB1_HOST_SYSTEM_ERROR_OUT_0	R5FSS0_CORE0_INTR_IN_238
	USB1_HOST_SYSTEM_ERROR_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_238
	USB1_HOST_SYSTEM_ERROR_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_238
	USB1_HOST_SYSTEM_ERROR_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_266
	USB1_HOST_SYSTEM_ERROR_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_266
	USB1_IRQ_OUT_0	GICSS0_SPI_IN_258
	USB1_IRQ_OUT_0	R5FSS0_CORE0_INTR_IN_230
	USB1_IRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_230
	USB1_IRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_230
	USB1_IRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_258
	USB1_IRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_258
	USB1_IRQ_OUT_1	GICSS0_SPI_IN_259
	USB1_IRQ_OUT_1	R5FSS0_CORE0_INTR_IN_231
	USB1_IRQ_OUT_1	WKUP_R5FSS0_CORE0_INTR_IN_231
	USB1_IRQ_OUT_1	MCU_R5FSS0_CORE0_CPU0_INTR_IN_231
	USB1_IRQ_OUT_1	C7X256V0_CLEC_GIC_SPI_IN_259
	USB1_IRQ_OUT_1	C7X256V1_CLEC_GIC_SPI_IN_259
	USB1_IRQ_OUT_2	GICSS0_SPI_IN_260
	USB1_IRQ_OUT_2	R5FSS0_CORE0_INTR_IN_232
	USB1_IRQ_OUT_2	WKUP_R5FSS0_CORE0_INTR_IN_232
	USB1_IRQ_OUT_2	MCU_R5FSS0_CORE0_CPU0_INTR_IN_232
	USB1_IRQ_OUT_2	C7X256V0_CLEC_GIC_SPI_IN_260
	USB1_IRQ_OUT_2	C7X256V1_CLEC_GIC_SPI_IN_260
	USB1_IRQ_OUT_3	GICSS0_SPI_IN_261
	USB1_IRQ_OUT_3	R5FSS0_CORE0_INTR_IN_233
	USB1_IRQ_OUT_3	WKUP_R5FSS0_CORE0_INTR_IN_233
	USB1_IRQ_OUT_3	MCU_R5FSS0_CORE0_CPU0_INTR_IN_233
	USB1_IRQ_OUT_3	C7X256V0_CLEC_GIC_SPI_IN_261
	USB1_IRQ_OUT_3	C7X256V1_CLEC_GIC_SPI_IN_261

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
USB1	USB1_IRQ_OUT_4	GICSS0_SPI_IN_262
	USB1_IRQ_OUT_4	R5FSS0_CORE0_INTR_IN_234
	USB1_IRQ_OUT_4	WKUP_R5FSS0_CORE0_INTR_IN_234
	USB1_IRQ_OUT_4	MCU_R5FSS0_CORE0_CPU0_INTR_IN_234
	USB1_IRQ_OUT_4	C7X256V0_CLEC_GIC_SPI_IN_262
	USB1_IRQ_OUT_4	C7X256V1_CLEC_GIC_SPI_IN_262
	USB1_IRQ_OUT_5	GICSS0_SPI_IN_263
	USB1_IRQ_OUT_5	R5FSS0_CORE0_INTR_IN_235
	USB1_IRQ_OUT_5	WKUP_R5FSS0_CORE0_INTR_IN_235
	USB1_IRQ_OUT_5	MCU_R5FSS0_CORE0_CPU0_INTR_IN_235
	USB1_IRQ_OUT_5	C7X256V0_CLEC_GIC_SPI_IN_263
	USB1_IRQ_OUT_5	C7X256V1_CLEC_GIC_SPI_IN_263
	USB1_IRQ_OUT_6	GICSS0_SPI_IN_264
	USB1_IRQ_OUT_6	R5FSS0_CORE0_INTR_IN_236
	USB1_IRQ_OUT_6	WKUP_R5FSS0_CORE0_INTR_IN_236
	USB1_IRQ_OUT_6	MCU_R5FSS0_CORE0_CPU0_INTR_IN_236
	USB1_IRQ_OUT_6	C7X256V0_CLEC_GIC_SPI_IN_264
	USB1_IRQ_OUT_6	C7X256V1_CLEC_GIC_SPI_IN_264
	USB1_IRQ_OUT_7	GICSS0_SPI_IN_265
	USB1_IRQ_OUT_7	R5FSS0_CORE0_INTR_IN_237
	USB1_IRQ_OUT_7	WKUP_R5FSS0_CORE0_INTR_IN_237
	USB1_IRQ_OUT_7	MCU_R5FSS0_CORE0_CPU0_INTR_IN_237
	USB1_IRQ_OUT_7	C7X256V0_CLEC_GIC_SPI_IN_265
	USB1_IRQ_OUT_7	C7X256V1_CLEC_GIC_SPI_IN_265
	USB1_OTGIRQ_OUT_0	GICSS0_SPI_IN_277
	USB1_OTGIRQ_OUT_0	R5FSS0_CORE0_INTR_IN_52
	USB1_OTGIRQ_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_52
	USB1_OTGIRQ_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_52
	USB1_OTGIRQ_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_277
	USB1_OTGIRQ_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_277
SERDES_10G0	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	GICSS0_SPI_IN_167
	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_166
	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_166

Copyright © 2026 Texas Instruments Incorporated

Table 10-5. Interrupts (Outputs) (continued)

Instance	Interrupt Output Line	Destination Interrupt
SERDES_10G0	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_166
	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_167
	SERDES_10G0_PHY_PWR_TIMEOUT_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_167
SERDES_10G1	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	GICSS0_SPI_IN_119
	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	R5FSS0_CORE0_INTR_IN_154
	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	WKUP_R5FSS0_CORE0_INTR_IN_53
	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	MCU_R5FSS0_CORE0_CPU0_INTR_IN_154
	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	C7X256V0_CLEC_GIC_SPI_IN_119
	SERDES_10G1_PHY_PWR_TIMEOUT_LVL_OUT_0	C7X256V1_CLEC_GIC_SPI_IN_119

Copyright © 2026 Texas Instruments Incorporated

Chapter 11

Data Movement Architecture

This chapter describes the data movement architecture of the device.

11.1 Data Movement Architecture Overview 985
11.2 Data Movement Subsystem (DMSS) 1047
11.3 Peripheral DMA (PDMA) 1079

Copyright © 2026 Texas Instruments Incorporated

11.1 Data Movement Architecture Overview

This chapter is a high-level summary of the data movement architecture implemented in the device.

11.1.1 Overview

The primary goal of the device Data Movement Architecture and related Subsystems is to ensure that data can be efficiently transferred from a producer to a consumer while meeting the real time requirements of the system can be met.

The Data Movement Subsystem (DMSS) aims to facilitate direct memory access (DMA) and provides a consistent Application Programming Interface (API) to the host software.

Data movement tasks are commonly offloaded from the host processor to peripheral hardware to increase system performance. Significant performance gains may result from careful design of the interface between the host software and the underlying acceleration hardware. In networking applications, packet transmission and reception are critical tasks. In general purpose compute, ping pong buffer prefetch and store are critical tasks as well as general mis-aligned block copy operations.

The design goals for the device Data Movement Architecture and are as follows:

- Minimize cost
- Minimize host overhead
- Maximize memory use efficiency
- Maximize bus burst efficiency
- Maximize symmetry between transmit/receive operations
- Maximize scalability for number of connections / buffer sizes / queue sizes / protocols supported
- Minimize protocol specific features
- Minimize complexity

11.1.1.1 Ring Accelerator (RINGACC)

The ring accelerator (RINGACC or RA) is a hardware module that is responsible for accelerating management of various types of queues in the system. The RINGACC accelerates passing of packets between a producer and consumer in normal system memory (including cached memory). The RINGACC is capable of accelerating the read/write pointer maintenance and occupancy tracking operations.

The ring accelerator operates like a bus infrastructure bridge in that it passes transactions through it between a source and destination interface. As transactions are passed, the RINGACC modifies the address, byte count, and transaction identifiers and internally updates the occupancies/pointers.

Each queue that the RINGACC provides can operate in either exposed-ring-mode or private-queue-mode.

- The exposed ring mode allows software to directly access the underlying ring structure to add or remove items from the tail or head of the ring respectively. Whenever items are added or removed from a ring, the host software is required to write to a corresponding doorbell register (RINGRT[a]_RT_DB) to increment or decrement the ring occupancy. When using the exposed ring mode, no proxy is required between the software and the RINGACC but all queue add operations require a memory fence to be performed to ensure that the data has landed in a snoopable cache before the doorbell register is written.
- The private queue mode provides an abstract view of the ring so that the host software does not need to know the actual physical address location or current read or write indexes of the ring. The private queue mode provides a memory window for each ring which when written redirects the write to the address pointed to by the write pointer and when read redirects the read to the address pointed to by the read pointer. The same address range is always used to push or pop elements from a given queue.

Rings are an implementation of a logical queue with the following limitations:

- Each ring has a finite size. When rings are used in exposed ring mode the following conditions apply:
  - A separate operation is required to write/read the contents of a ring and to update the occupancy of the ring

Copyright © 2026 Texas Instruments Incorporated

- It is not straightforward to allow multiple producers to write to a ring or multiple consumers to read from a ring. Additional synchronization and pointer passing is required since rings require software to manage one of the pointers for the queue.
- An exposed ring cannot be used when software or hardware needs to both read and write the same queue (such as for a DMA RX free queue where errors can have the DMA write the element back onto the same RX free queue). Since software owns one side of the pointers and hardware owns the other, they cannot be kept coherent if either side needs to update either at any time.

The RINGACC allows each ring to be configured to a different primary element size. Element sized chunks of data are placed onto and retrieved from rings when queuing or de-queueing occurs. Element sizes can be as small as 4 bytes or as large as 256 bytes.

11.1.1.2 Secure Proxy (SEC_PROXY)

A proxy provides a mechanism for software to access the RINGACC coherently when the processor does not support large bursts. The RINGACC requires a single burst for each operation so that it maintains atomicity and coherence by relying on the atomicity of the bursts on the bus interconnect fabric, since it only delivers a single burst to the RINGACC before the next. Since processors are usually limited in the size of a burst they can deliver natively, such as 32 to 128 bits, they cannot be used directly with the data access region of the RINGACC for larger element sizes. The proxy solves this gap by providing a temporary space for the software to form a larger burst of data before it is sent to the RINGACC. The proxy allows smaller processor accesses to the data and only forwards to the RINGACC when the entire data burst is complete, as directed by the software.

Each proxy hardware can support multiple threads of software (whether on the same or different processors) operating on bursts at the same time, and they do not interfere with each other, as long as each thread of software only operates on its thread in the proxy. Each proxy thread also supports access to any RINGACC queue via different offsets similar to how the RINGACC provides access to the queues via different offsets.

A special version of a proxy is the Secure Proxy that provides secured access to RINGACC queues for communication to the security initiator of the system. It provides basically the same function as a normal proxy. The data burst size is fixed for message sizes to the security initiator, so there is no support for all the element sizes of the RINGACC. The security initiator locks down the RINGACC queue to access and direction for each proxy thread so that the software cannot access anything they shouldn't, so the software only sees a single queue (unlike the normal proxy), and any attempts to access another queue or access it in violation of how the security initiator defined and an error will be flagged. This allows for a much more controlled setup for passing messages to the security initiator.

11.1.1.3 Interrupt Aggregator (INTAGGR)

The interrupt aggregator (INTAGGR or INTA) is a hardware module which provides a persistent view of the real time status of various DMA components within the overall SoC DMA system. The main function of the module is to convert between 'global events' (assertion and de-assertion events transmitted on the event-transport-lane bus) and 'local events' (level sensitive signals on output, and level or edge sensitive signals on input).

The INTA block provides the following functions:

- Conversion of local event signal lines into global events
- Performing an 'Y-split' operation of global events
- Counting global events
- Steering and aggregation of global events into specified bit positions in one of N interrupt cause registers
- SoC interrupt-architecture-compliant set/clear and mask set/clear functionality to cover interrupt cause registers

11.1.1.4 Packet DMA (PKTDMA)

The PKTDMA module supports the transmission and reception of various packet types. The PKTDMA is architected to facilitate the segmentation and reassembly of DMA data structure compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each connected peripheral. Multiple Tx and Rx channels are provided within the DMA which allow multiple segmentation or reassembly operations to be ongoing. The DMA controller maintains state information for each of the channels which allows

Copyright © 2026 Texas Instruments Incorporated

packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. An internal DMA scheduler is used to control the ordering and rate at which this multiplexing occurs for Transmit operations. The ordering and rate of Receive operations is indirectly controlled by the order in which blocks are pushed into the DMA on the Rx PSI-L interface.

A block diagram of the PKTDMA Controller is shown below:

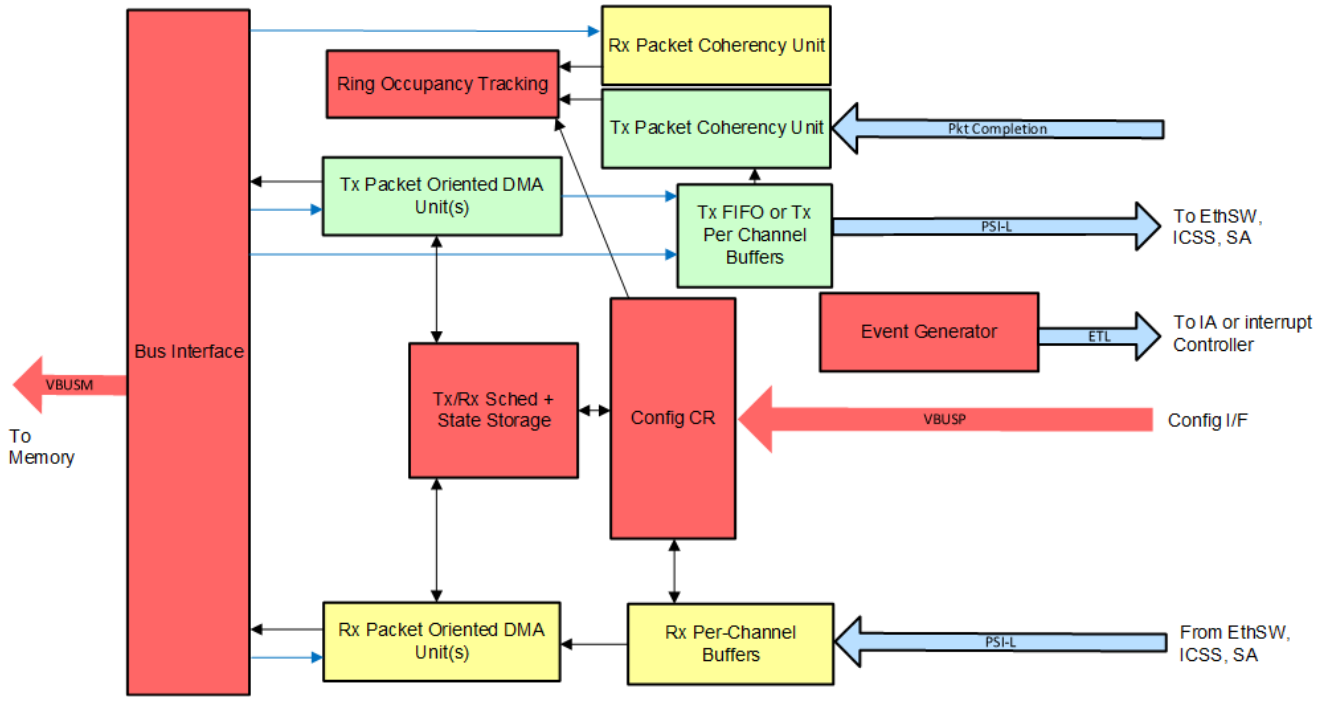


Figure 11-1. PKTDMA Block Diagram

11.1.1.4.1 PKTDMA Submodule Descriptions

11.1.1.4.1.1 Bus Interface Unit

The Bus Interface Unit is responsible for merging and buffering all of the transactions that originate from the various initiator blocks inside the DMA controller into the 4 separate VBUSM initiator interfaces. Arbitration between the blocks to a given VBUSM interface is round robin within a single priority level. A 2 word deep retiming buffer is provided on each sub interface of each provided VBUSM bus.

11.1.1.4.1.2 Config CR

The function of the Config CR is to switch transactions which arrive on the configuration interface to the various configuration targets within the DMA. The Config CR also provides a retiming buffer on the configuration bus primary interface ports.

11.1.1.4.1.3 Configuration Registers

The Configuration Registers block is responsible for monitoring the fullness level of the Tx Per Channel FIFOs, monitoring data transfer work which is pending, maintaining data movement thread state information, arbitrating which channel will be allowed to perform work next, issuing scheduler commands to the Tx PKTDMA core blocks, and writing back the updated state returned from those same DMA core blocks.

The Configuration Registers block is also responsible for providing memory mapped registers for configuration of the Rx DMA functions including the default settings for the free descriptor and destination queues. For modularity and high speed pipelining reasons, the Rx traffic is looped through the Configuration Registers block where the original stream information is merged with information from the configuration registers on its way to the Rx DMA unit module. This prevents the Rx DMA Core from having to spend cycles accessing the channel configuration information for the channel.

Copyright © 2026 Texas Instruments Incorporated

11.1.1.4.1.3.1 RX State Mapping

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sop_descriptor_ptr[35:4]																															
0x80																															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
sop_descriptor_ptr[35:4]												sop_descriptor_asel				curr_descriptor_ptr															
0x84																															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
curr_descriptor_ptr																												curr_descriptor_asel			
0x88																															
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
curr_buffer_ptr																															
0x8C																															
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
curr_buffer_ptr																curr_buffer_asel				curr_buffer_length											
0x90																															
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
curr_buffer_length										curr_buffer_offset																					
0x94																															
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
curr_buffer_eop	pkt_length																					flush	ps_wordcnt						flowid_is_default	drop_flush	
0x98																															
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
ps offset							ps_flags				pktid_data													orderid				priority			
0x9C																															

Copyright © 2026 Texas Instruments Incorporated

Table 11-1. RX State Mapping Table

Total Bits	Bits	Register Offset	Field	Description
31 - 0	31 - 0	0x80	sop_descriptor_ptr[35:4]	The descriptor pointer at start of packet
43 - 32	11 - 0	0x84	sop_descriptor_ptr[35:4]	The descriptor pointer at start of packet
47 - 44	15 - 12	0x84	sop_descriptor_asel	The asel for descriptor pointer at start of packet
63 - 48	31 - 16	0x84	curr_descriptor_ptr	The pointer for the current descriptor aligned on 16 byte boundary
91 -64	27 - 0	0x88	curr_descriptor_ptr	The pointer for the current descriptor aligned on 16 byte boundary
95 - 92	31 - 28	0x88	curr_descriptor_asel	The asel of the current descriptor
127 - 96	31 - 0	0x8C	curr_buffer_ptr	The pointer for the current buffer
143 - 128	15 - 0	0x90	curr_buffer_ptr	The pointer for the current buffer
147 - 144	19 - 16	0x90	curr_buffer_asel	The asel for the current buffer
159 - 148	9 - 0	0x90	curr_buffer_length	The length of the current buffer
169 - 160	31 - 20	0x90	curr_buffer_length	The length of the current buffer
191 - 170	31 - 10	0x94	curr_buffer_offset	The offset inside the current buffer
192	0	0x98	curr_buffer_eop	The end of packet flag for the current buffer
213 - 193	21 - 1	0x98	pkt_length	The size of the packet length
214	22	0x98	flush	The packet is being flushed
220 - 215	28 - 23	0x98	ps_wordcnt	protocol specific word count
221	29	0x98	flowid_is_default	Use default flow id
223 - 222	31 - 30	0x98	drop_flush	The flush is a drop condition
230 - 224	6 - 0	0x9C	ps offset	protocol specific offset
234 - 231	10 - 7	0x9C	ps_flags	protocol specific flags
247 - 235	23 - 11	0x9C	pktid_data	The pktid number for scoreboard
251 - 248	27 - 24	0x9C	orderid	The orderid of the packet
254 - 252	30 - 28	0x9C	priority	The priority of the packet

Copyright © 2026 Texas Instruments Incorporated

11.1.1.4.1.3.2 TX State Mapping

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sop_descriptor_ptr[35:4]																															
0x80																															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
sop_descriptor_ptr[35:4]												sop_descriptor_asel				curr_descriptor_ptr															
0x84																															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
curr_descriptor_ptr																												curr_descriptor_asel			
0x88																															
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
curr_buffer_ptr																															
0x8C																															
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
curr_buffer_ptr																curr_buffer_asel				curr_buffer_length											
0x90																															
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
curr_buffer_length										curr_buffer_offset																					
0x94																															
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
curr_buffer_eop	rem_length																						info_present	truncated_non_eop	psbytes_rem						
0x98																															
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
psbytes_rem	psbytes_off								wptr_data													pktid_data	orderid					priority			
0x9C																															

Copyright © 2026 Texas Instruments Incorporated

Table 11-2. TX State Mapping Table

Total Bits	Bits	Register Offset	Field	Description
31 - 0	31 - 0	0x80	sop_descriptor_ptr[35:4]	The descriptor pointer at start of packet
43 - 32	11 - 0	0x84	sop_descriptor_ptr[35:4]	The descriptor pointer at start of packet
47 - 44	15 - 12	0x84	sop_descriptor_asel	The asel for descriptor pointer at start of packet
63 - 48	31 - 16	0x84	curr_descriptor_ptr	The pointer for the current descriptor aligned on 16 byte boundary
91 -64	27 - 0	0x88	curr_descriptor_ptr	The pointer for the current descriptor aligned on 16 byte boundary
95 - 92	31 - 28	0x88	curr_descriptor_asel	The asel of the current descriptor
127 - 96	31 - 0	0x8C	curr_buffer_ptr	The pointer for the current buffer
143 - 128	15 - 0	0x90	curr_buffer_ptr	The pointer for the current buffer
147 - 144	19 - 16	0x90	curr_buffer_asel	The asel for the current buffer
159 - 148	9 - 0	0x90	curr_buffer_length	The length of the current buffer
169 - 160	31 - 20	0x90	curr_buffer_length	The length of the current buffer
191 - 170	31 - 10	0x94	curr_buffer_offset	The offset inside the current buffer
192	0	0x98	curr_buffer_eop	The end of packet flag for the current buffer
214 - 193	22 - 1	0x98	rem_length	The remaining length in the buffer
215	23	0x98	info_present	The packet has info present
216	24	0x98	truncated_non_eop	Eop if packet length is greater than the total buffer length
223 - 217	31 - 25	0x98	psbytes_rem	Number of protocol specific bytes remaining
224	0	0x9C	psbytes_rem	Number of protocol specific bytes remaining
232 - 225	8 - 1	0x9C	psbytes_off	The offset of the protocol specific bytes
245 - 233	21 - 9	0x9C	wptr_data	The current write pointer in the data fifo
247 - 246	23 - 22	0x9C	pktid_data	The pktid number for scoreboard
251 - 248	27 - 24	0x9C	orderid	The orderid of the packet
254 - 252	30 - 28	0x9C	priority	The priority of the packet

11.1.1.4.1.4 Tx Packet DMA Unit

The Tx Packet DMA Unit block implements all of the state machine functionality necessary to implement the KS3 DMA Host Tx protocol. The Tx Packet DMA unit initiates VBUSM transactions in order to read descriptor pointers from the memory mapped Tx ring, read descriptors from memory, and read data from buffers in memory.

11.1.1.4.1.5 Tx Packet Coherency Unit

The Tx Packet Coherency Unit is responsible for ensuring that all control structures and data have been read (and updated if applicable) by the Tx DMA unit(s) and that all previous packets on that channel/flow have also completed prior to incrementing the reverse ring occupancy for the channel/flow.

11.1.1.4.1.6 Tx Per Channel Buffers

The Tx Per Channel Buffers implement a FIFO for each Tx DMA channel that is used for buffering packet control and payload data that has been fetched by the Tx Packet DMA units. The buffers are byte oriented on write so that the data from the DMA units which may not be full words can be packed properly. The buffers are block

Copyright © 2026 Texas Instruments Incorporated

oriented on read and each Tx (source) channel in the PKTDMA controller maps directly onto a thread in the Tx PSI interface. The Tx Per Channel Buffer block outputs queue fullness information to the Tx Scheduler block which it then uses to determine when it should initiate DMA opportunities to backfill the buffers. The Tx Per Channel Buffer will initiate transfers to the target whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block will simultaneously monitor the status of all of the threads and will perform a round robin arbitration between the different threads for the use of the Transmit PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer will be included in the arbitration.

11.1.1.4.1.7 Rx Per Channel Buffers

The Rx Per Channel Buffers implement a FIFO for each Rx DMA channel that is used for buffering packet control and payload data that has been pushed into the DMA from the Rx PSI-L interface. The Rx Per Channel Buffers also includes an arbitration unit which determines which Rx DMA channel should be serviced next.

11.1.1.4.1.8 Rx Packet DMA Unit

The Rx Packet DMA Unit block implements all of the state machine functionality necessary to implement the KS3 DMA Rx protocol for Host descriptor types. The Rx Packet DMA Unit initiates VBUSM transactions in order to read descriptor pointers from memory mapped rings, read buffer descriptor information, write descriptors to memory, and write data to buffers in memory.

11.1.1.4.1.9 Rx Packet Coherency Unit

The Rx Packet Coherency Unit is responsible for ensuring that all control structures and data have been written by the Rx DMA unit(s) and that all previous packets on that channel/flow have completed prior to incrementing the reverse ring occupancy for the channel/flow. This unit ensures that the ordering of the Packet Descriptor pointer writes to the return queue directly matches the ordering in which those packets were fetched from the Rx free queues. Writes are not considered complete by this unit until the entire write status has been returned for all outstanding transactions for a given packet ID.

11.1.1.4.1.10 Event Handler

The Event Handler block is responsible for generating channel completion and error events.

11.1.1.4.2 Channel Classes

Not all channels within the PKTDMA require the same performance capabilities. Raw bandwidth and latency tolerance requirements will vary greatly between data transfers at various points in the system. Due to area concerns, several different classes of DMA channels are provided in the PKTDMA and are described as follows:

Class Name	Description
Normal Capacity	Provides baseline amount of descriptor prefetch and Tx/Rx control and data buffering. Suitable for most peripheral transfers which are communicating with on-chip memories and DDR
High Capacity	Provides an elevated amount of descriptor prefetch and custom Tx/Rx control and data buffering. Suitable for applications which require moderate per-channel bandwidth with significantly increased latency (ex. Transferring data to/from PCIe)
Ultra-High Capacity	Provides identical descriptor and TR prefetch as High Capacity but with increased Tx data buffering to allow for more read data in flight. Suitable for applications requiring high per-channel bandwidth with significantly increased latency (ex. 16+Gbps transfers to/from PCIe).

PKTDMA instances may provide support for any or all of the above channel classes through design time configuration parameters.

Copyright © 2026 Texas Instruments Incorporated

11.1.1.5 Block Copy DMA (BCDMA)

The Block Copy DMA is intended to perform similar functions as an Embedded DMA (EDMA) and the UDMA-P. The BCDMA module moves data from a memory mapped source address set to a corresponding memory mapped address. The BCDMA maintains state information for each of the channels which allows data copy operations to be time division multiplexed between channels in order to share the underlying DMA hardware. An internal DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

A block diagram of the Block Copy DMA Controller is shown below:

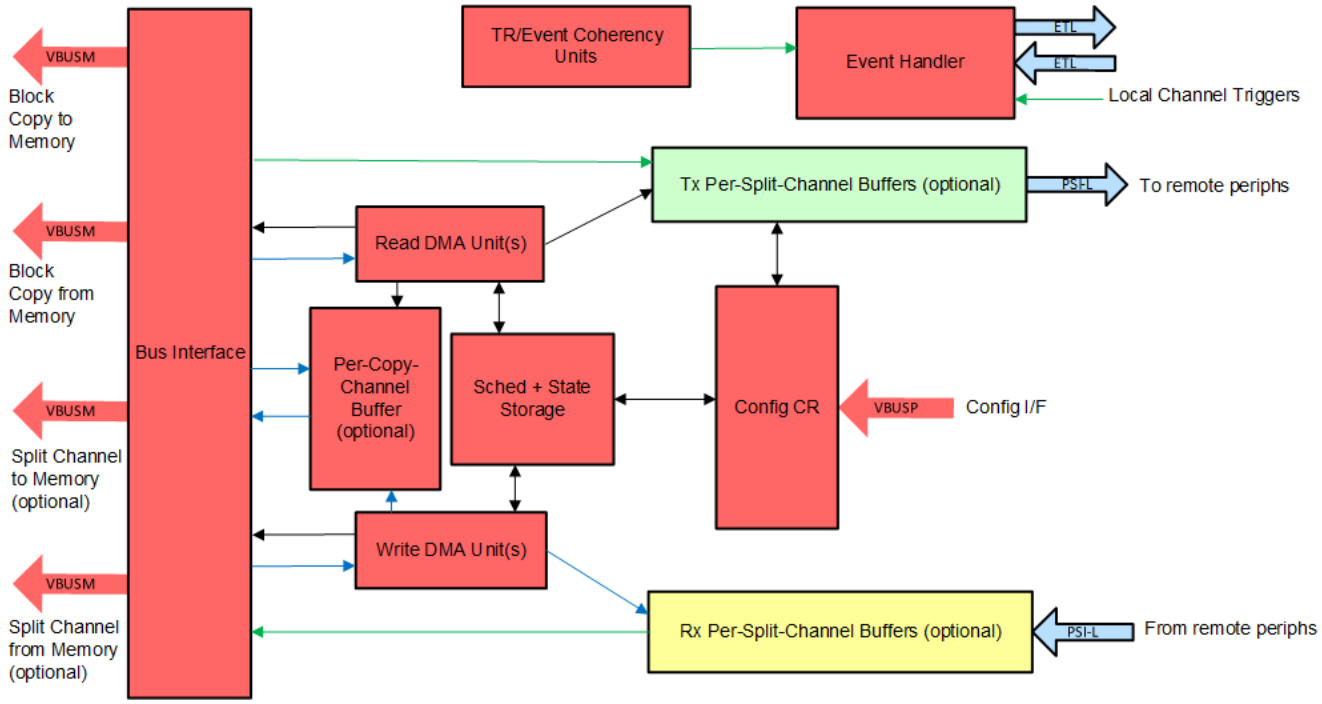


Figure 11-2. Block Copy DMA Block Diagram

11.1.1.5.1 BCDMA Submodule Descriptions

11.1.1.5.1.1 Bus Interface Unit

The Bus Interface Unit is reponsible for merging and buffering all of the transactions that originate from the various controller blocks inside the DMA controller into the 4 seperate VBUSM controller interfaces. Arbitration between the blocks to a given VBUSM interface is round robin within a single priority level. A 2 word deep retiming buffer is provided on each sub interface of each provided VBUSM bus.

11.1.1.5.1.2 Config CR

The function of the Config CR is to switch transactions which arrive on the configuration interface to the various configuration targets within the DMA. The Config CR also provides a retiming buffer on the configuration bus primary interface ports.

11.1.1.5.1.3 Configuration Registers

The Configuration Registers block is responsible for monitoring the fullness level of the Per Channel FIFOs, monitoring data transfer work which is pending, maintaining data movement thread state information, arbitrating which channel will be allowed to perform work next, issuing scheduler commands to the Read and Write DMA units, and writing back the updated state returned from those same DMA units.

The Configuration Registers block is also responsible for providing memory mapped registers for configuration of the DMA channels.

Copyright © 2026 Texas Instruments Incorporated

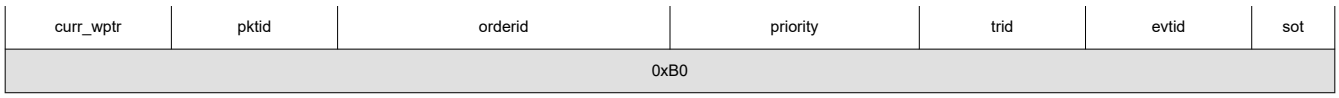
11.1.1.5.1.3.1 BCDMA Mapping Table

Table 11-3 applies to TX, RX and BC Mapping.

Copyright © 2026 Texas Instruments Incorporated

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
reload_count				waitcomp	event_size		trigger0		trigger0_type		trigger1_type		trigger1_type		spflags				eop	eol			reload_count								
0x80																															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
tr_dim1																															
0x84																															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
tr_dim2																															
0x88																															
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
tr_dim3																															
0x8C																															
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
tr_icnt0																tr_icnt1															
0x90																															
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
tr_icnt2																tr_icnt3															
0x94																															
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
icnt1																icnt2															
0x98																															
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
icnt3																rem_icnt0															
0x9C																															
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287
tr_addr																															
0xA0																															
288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
tr_addr				tr_asel				descriptor_ptr																							
0xA4																															
320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351
descriptor_ptr												RVSD				curr_index														last_index	
0xA8																															
352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383
last_index												nom_elsize			reload_index					curr_wptr											
0xAC																															
384			385			386			387			388			389			390			391			392			393			394		
395			396			397			398			399																				

Copyright © 2026 Texas Instruments Incorporated



Copyright © 2026 Texas Instruments Incorporated

Table 11-3. BCDMA Mapping Table

Total Bits	Bits	Register Offset	Field	Description
3 - 0	3 - 0	0x80	reload_count	
4	4	0x80	waitcomp	
6 - 5	6 - 5	0x80	event_size	
8 - 7	8 - 7	0x80	trigger0	
10 - 9	10 - 9	0x80	trigger0_type	
12 - 11	12 - 11	0x80	trigger1_type	
14 - 13	14 - 13	0x80	trigger1_type	
18 - 15	18 - 15	0x80	spflags	
19	19	0x80	eop	
22 - 20	22 - 20	0x80	eol	
31 - 23	31 - 23	0x80	reload_count	
63 - 32	31 - 0	0x84	tr_dim1	
95 - 64	31 - 0	0x88	tr_dim2	
127 - 96	31 - 0	0x8C	tr_dim3	
143 - 128	15 - 0	0x90	tr_icnt0	
159 - 144	31 - 16	0x90	tr_ictn1	
175 - 160	15 - 0	0x94	tr_ictn2	
191 - 176	31 - 16	0x94	tr_icnt3	
207 - 192	15 - 0	0x98	icnt1	
223 - 208	31 - 16	0x98	icnt2	
239 - 224	15 - 0	0x9C	icnt3	
255 - 240	31 - 16	0x9C	rem_icnt0	
287 - 256	31 - 0	0xA0	tr_addr	
291 - 288	3 - 0	0xA4	tr_addr	
295 - 292	7 - 4	0xA4	tr_asel	
319 - 296	31 - 8	0xA4	descriptor_ptr	
331 - 320	11 - 0	0xA8	descriptor_ptr	
349 - 336	29 - 16	0xA8	curr_index	
363 - 350	11 - 30	0xA8	last_index	
366 - 364	14 - 12	0xAC	nom_elsize	
372 - 367	20 - 15	0xAC	reload_index	
385 - 373	1 - 21	0xAC	curr_wptr	
387 - 386	3 - 2	0xB0	pktid	
391 - 388	7 - 4	0xB0	orderid	
394 - 392	10 - 8	0xB0	priority	
396 - 395	12 - 11	0xB0	trid	
398 - 397	14 - 13	0xB0	evtid	

Copyright © 2026 Texas Instruments Incorporated

Table 11-3. BCDMA Mapping Table (continued)

Total Bits	Bits	Register Offset	Field	Description
399	15	0xB0	sot	

11.1.1.5.1.4 Read Unit(s)

The Read Units implement all of the state machine functionality necessary to perform the read transfers defined in the SW supplied Transfer Request records. The read DMA unit initiates VBUSM transactions in order to read descriptor pointers from memory mapped rings, read descriptors, Transfer Request records, and payload data from memory.

11.1.1.5.1.5 TR Coherency Unit

The TR Coherency Unit is responsible for ensuring that all control structures and data have been read (and updated if applicable) by the BCDMA unit(s) and all prior TRs have completed prior to incrementing the reverse ring occupancy for the flow/channel.

11.1.1.5.1.6 Per-Copy-Channel Buffers

The Per-Copy-Channel Buffers implement a FIFO for each BCDMA generic block copy channel and is used for buffering payload data that has been fetched by read DMA unit modules. The write DMA units issue write transfers from the Per Channel Buffers to specified addresses in memory. The buffers are byte oriented on both write and read so that the data from the DMA units which may not be full words can be packed and unpacked properly. This unit is optional and is only present if one or more generic block copy channels are configured in the BCDMA instance.

11.1.1.5.1.7 Tx Per-Split-Channel Buffers

The Tx Per-Split-Channel Buffers implement a FIFO for each BCDMA Tx split mode channel and is used for buffering payload data that has been fetched by read DMA unit modules. Data is transferred from the buffer to the remote peripheral via a PSI-L interface. This unit is optional and is only present if one or more Tx split mode channels are configured in the BCDMA instance.

11.1.1.5.1.8 Rx Per-Split-Channel Buffers

The Rx Per-Split-Channel Buffers implement a FIFO for each BCDMA Rx split mode channel and is used for buffering payload data that has been received from remote peripherals via a PSI-L interface. Data is read from these FIFOs and written out to memory as directed by the write DMA unit modules. This unit is optional and is only present if one or more Rx split mode channels are configured in the BCDMA instance.

11.1.1.5.1.9 Write Unit(s)

The Write Units implement all of the state machine functionality necessary to perform the write transfers defined in the SW supplied Transfer Request records. The write DMA unit initiates VBUSM transactions in order to read descriptor pointers from memory mapped rings, read descriptors and/or Transfer Request records from memory mapped rings, read descriptor and Transfer Request records, and write payload data to memory.

11.1.1.5.1.10 Event Coherency Unit

The Event Coherency Unit is responsible for ensuring that all writes have completed prior to signaling an output event for a channel/flow.

11.1.1.5.1.11 Event Handler

The Event Handler block is responsible for generating channel completion and error events and for accepting and tracking channel triggering.

11.1.1.5.2 Channel Classes

Not all channels within a DMA controller require the same performance capabilities. Raw bandwidth and latency tolerance requirements will vary greatly between data transfers at various points in the system. Due to area concerns, several different classes of DMA channels are provided in the BCDMA and are described as follows:

Copyright © 2026 Texas Instruments Incorporated

Class Name	Description
Normal Capacity	Provides baseline amount of descriptor and TR prefetch and Tx/Rx control and data buffering. Suitable for most peripheral transfers which are communicating with on-chip memories and DDR
High Capacity	Provides an elevated amount of descriptor and TR prefetch and custom Tx/Rx control and data buffering. Suitable for applications which require moderate per-channel bandwidth with significantly increased latency (ex. Transferring data to/from PCIe)
Ultra-High Capacity	Provides identical descriptor and TR prefetch as High Capacity but with increased Tx data buffering to allow for more read data in flight. Suitable for applications requiring high per-channel bandwidth with significantly increased latency (ex. 16+Gbps transfers to/from PCIe).

BCDMA instances may provide support for any or all of the above channel classes through design time configuration parameters.

11.1.2 Definition of Terms

Host— The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.

Channel— A channel refers to the sub-division of information (flows) that is transported across a DMA engine. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (for example, CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). Information flow within a channel is a stream of strongly ordered information.

Data Buffer— A data buffer is a single data structure that contains payload information for transmission to or reception from a channel.

Buffer Descriptor— A buffer descriptor is a single data structure that contains information about one or more data buffers.

Packet Descriptor— A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. All Monolithic type descriptors are packet descriptors (and are also a Data Buffer).

Queue— A queue is a list of strongly ordered entries which is typically used to pass work between a producer and a consumer. Queue entries in most cases are references to a work payload which is being passed, but in some cases, (Transfer Request Packets for example) queue entries may actually contain data which is being transferred. Queues are used throughout DMA whenever communication is required between entities. Queues can have multiple different implementations and DMA uses two of the most common: linked lists and rings.

Linked list— A linked list is a data structure in which each entry stores not only the entry data but also a chaining pointer to the next entry in the list. The last entry in each list has its chaining pointer set to NULL (typically encoded as 0x0). The list manager maintains a pointer to the head element on the list and to the tail element on the list. Since the chaining pointer is stored with the entry data, linked lists have a length which is dynamically changeable and limited only by the ability to allocate additional entries which are to be queued/de-queued. Linked lists are present in Host Descriptors to chain multiple descriptors to form a packet.

Ring— A ring is a data structure in which a contiguous memory block defined by M N-byte entries (total size is M × N bytes) is statically allocated and sequentially written/read in order to pass data or data references. Rings are also referred to as circular buffers because when the last element in the contiguous memory array is written, the pointers wrap back to the beginning address for the ring and start the process all over again. The Ring Accelerator component uses rings in order to implement logical queues.

Copyright © 2026 Texas Instruments Incorporated

Free Descriptor/Buffer Queue— A free descriptor/buffer queue is a list of available descriptors with prelinked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Ring Accelerator.

Free Descriptor Queue— A free descriptor queue is a list of available descriptors that are not yet linked with buffers that are to be used by the receive ports for monolithic type descriptors. Free Descriptor Queues are implemented by the Ring Accelerator.

Packet Queue— A packet queue is a list of valid (that is, populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes. Packet Queues are implemented by the Ring Accelerator.

Memory— Memory is an area of data storage managed by the host. This area is visible to the port as a 64-bit addressable area.

Device Driver— A device driver is application independent software that runs on the host for purposes abstracting the low level hardware so that upper level software can use the hardware without knowing every bit field location or initialization sequence. General device driver functions include port initialization, transmit packet queuing, and receive packet processing.

SOP— Start of Packet. This refers to the descriptor/buffer that is the first buffer in a packet.

MOP— Middle of Packet. This refers to the descriptors/buffers that are neither the first or last buffers in a packet.

EOP— End of Packet. This refers to the descriptor/buffer that is the last buffer in a packet.

Copyright © 2026 Texas Instruments Incorporated

11.1.3 DMSS Hardware/Software Interface

The intent of the Data Movement architecture is to provide a uniform HW/SW interface which includes a rich set of mechanisms that SW can make use of to transfer data with low overhead and reasonable complexity. To this goal, the number of components which SW is required to interact with on an ongoing real time basis has been kept to a minimum and is as follows (in order of anticipated frequency of access):

- Interrupt Aggregator
- Packet DMA
- Block Copy DMA

When interrupts are used, the interrupt aggregator will provide the vast majority of interrupt sources from all DMA components in the system. Each non-exception/non-debug packet and TR completion signaling originates from the Interrupt Aggregator and the IA provides a uniform set of MMRs that can be queried to quickly determine the cause of a specific interrupt. Events from PSI-L/ETL components are all routed to the host via the Interrupt Aggregator.

Each PKTDMA and BCDMA instance includes an inbuilt ring control mechanism which is the primary means by which work is sent to or received from these DMAs. Each DMA instance also includes both static configuration type memory mapped registers and some real-time accessible memory mapped registers for monitoring and control of each individual channel.

Memory mapped data structures are used anytime information needs to be passed between components in the system. These components may be hardware or software. The following sections describe the data structures which are used within the DMSS for passing information. These data structures include data buffers, packet descriptors, buffer descriptors, queues (including transmit queues, transmit completion queues, and receive queues) and the configuration MMRs that are provided in the various components. The following sections provide a detailed description of these data structures.

11.1.3.1 Data Buffers

A data buffer is a byte aligned contiguous block of memory used to store payload data. Each buffer is described in an entry in either a packet descriptor or in a buffer descriptor. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the 64-bit memory space.

The Buffer Length field of the packet/buffer descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

11.1.3.2 Descriptors

Descriptors are so named because their primary function is to describe other data structures.

The PKTDMA architecture provides for 2 basic types of descriptors whose characteristics are shown in PKTDMA Descriptor Types and Attributes.

Copyright © 2026 Texas Instruments Incorporated

Table 11-4. PKTDMA Descriptor Types and Attributes

Descriptor Type	Includes Valid Packet Info	Provided # of Slots to Link In External Data	Provides Local Protocol Specific Storage	Provides Slot to Link Additional Descriptors Within Same Packet	Description
Host Packet Descriptor	✓	1	✓	✓	Used to describe packet and SOP buffer in applications that require Host OS compatible data structures (i.e. applications where the descriptors and buffers cannot be managed independently but must instead be pre-linked by the Host software). These applications inherently require a separate descriptor for each buffer.
Host Buffer Descriptor		1		✓	Used to describe non-SOP buffers in applications that require Host OS compatible data structures

As the above table shows, the Host Packet Descriptor provides packet level information that is useful to both the ports and the Host in order to properly process the packet. These descriptors are referred to as Packet Descriptors and will always appear as the first descriptor within a packet.

The PKTDMA allows descriptors to be allocated on 16-byte address boundaries. This is intended to allow for better memory utilization by allowing on-chip descriptors which are not a power of 2 in length to be packed into arrays with little wasted memory space.

Even though descriptors and buffers may be allocated on any 16-byte alignment, careful consideration of the alignment effects should be made based on the storage location and any cache related affects that may exist. If data structures are placed in off-chip SDRAM the burst size and alignment restrictions of the memory devices must be considered in order to avoid performance issues related to continually fetching mis-aligned blocks. In this case, the memory efficiency can be reduced to 50% because 2 memory bank lines are read for every line sized data fetch. Similarly, placing more than one descriptor or buffer object within a single cache line can cause the adjacent object to become corrupted during cache line writeback operations.

Software/application specific control information may be added to the end of any of the packet descriptor types using as many extra words as necessary but the above alignment rules still apply.

The Block Copy DMA architecture provides a single descriptor type whose characteristics are shown in Block Copy DMA Descriptor Types and Attributes.

Copyright © 2026 Texas Instruments Incorporated

Table 11-5. Block Copy DMA Descriptor Types and Attributes

Descriptor Type	Includes Valid Packet Info	Provided # of Slots to Link In External Data	Provides Local Protocol Specific Storage	Provides Slot to Link Additional Descriptors Within Same Packet	Description
Transfer Request Packet Descriptor		0			Used to feed transfer request sequences to the BCDMA

11.1.3.2.1 Host Packet Descriptor

Host Packet Descriptors are designed to be used when the application requires support for true, unlimited fragment count scatter/gather type operations. The Host Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a Host Packet Descriptor
- Source and Destination Tags
- Packet Type
- Packet Length
- Protocol Specific Region Size
- Protocol Specific Control / Status Bits
- Pointer to the first valid byte in the SOP data buffer
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the packet
- Software specific information

Host Packet Descriptors always contain 48 bytes of required information and may also contain optional software specific information and protocol specific information. How much optional information (and therefore the allocated size of the descriptors) is required is application dependent.

The Host Packet descriptor layout is shown below.

Table 11-6. Host Packet Descriptor Layout

Packet Info (16 bytes)
Linking Info (8 bytes)
Buffer Info (12 bytes)
Original Buffer Info (12 bytes)
Extended Packet Info Block (Optional) Includes Timestamp and Software Data (16 bytes)
Protocol Specific Data (Optional) (0 to M bytes where M is a multiple of 4)
Other SW Data (Optional and User Defined)

Host Packet Descriptors may be linked with zero or more additional Host Buffer Descriptors in a singly linked list fashion to form packets. Each Host Packet consists of a single Host Packet Descriptor followed by a chain of zero or more Host Buffer Descriptors linked together using the Next Descriptor Pointer fields in the descriptors. The last descriptor in a Host packet has a zero Next Descriptor Pointer.

Copyright © 2026 Texas Instruments Incorporated

The ‘Other SW Data’ portion of the descriptor exists after all of the defined words and is reserved for use by the host software to store completely private data. This region is not used in any way by HW components in a PKTDMA system and these modules will not modify any bytes within this region.

The contents of the Host Packet Descriptor words are detailed in Table 11-7 through Table 11-23

Table 11-7. Host Packet Descriptor Packet Information Word 0 (PD Word 0)

Bits	Name	Description	Rx Overwrite
31:30	2’d1	64-bit Host Packet Descriptor Type Identifier	Yes
29	Extended Packet Info Block Present	This field indicates the presence of the Extended Packet Info Block in the descriptor. 0 = EPIB is not present 1 = 16 byte EPIB is present	Yes
28	RESERVED	-	*Yes*
27:22	Protocol Specific Valid Word Count	This field indicates the valid # of 32-bit words in the protocol specific region. This is encoded in increments of 4 bytes as follows: 0 = 0 bytes 1 = 4 bytes … 16 = 64 bytes ... 32 = 128 bytes 33-63 = RESERVED	Yes
21:0	Packet Length	The length of the packet in bytes. If the Packet Length is less than the sum of the buffer lengths, then the packet data will be truncated. A Packet Length greater than the sum of the buffers is an error. The valid range for an exact packet length is 0 to 4M-1 bytes. If the packet length is set to 0, the port will not actually transmit any information.	Yes

Table 11-8. Host Packet Descriptor Packet Information Word 1 (PD Word 1)

Bits	Name	Description	Rx Overwrite
31:28	Error Flags	This field contains error flags that can be assigned based on the packet type	Yes
27:24	Protocol Specific Flags	This field contains protocol specific flags / information that can be assigned based on the packet type.	Yes
23:14	RESERVED	-	Yes

Copyright © 2026 Texas Instruments Incorporated

Table 11-8. Host Packet Descriptor Packet Information Word 1 (PD Word 1) (continued)

Bits	Name	Description	Rx Overwrite
13:0	Flow ID	Flow ID within which this packet is being transported. The FlowID is used by downstream blocks to make decisions about packet steering and resource allocations. FlowIDs are also used to allow specific packets to be received into specific sets of buffers.	Yes

Table 11-9. Host Packet Descriptor Packet Information Word 2 (PD Word 2)

Bits	Name	Description	Rx Overwrite
31:27	Packet Type	This field indicates the type of this packet and is encoded as follows: 0-31 = Application specific	Yes
26:16	RESERVED	-	Yes
15:0	RESERVED	-	Yes

Table 11-10. Host Packet Descriptor Packet Information Word 3 (PD Word 3)

Bits	Name	Description	Rx Overwrite
31:24	Source Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field with the value provided in the PSI-L src_tag field bits 15:8.	Yes
23:16	Source Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field with the value provided in the PSI-L src_tag field bits 7:0.	Yes
15:8	Dest Tag – Hi	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field with the value provided in the PSI-L dst_tag field bits 15:8.	Yes
7:0	Dest Tag – Lo	This field is application specific. During Packet reception, the DMA controller in the port will overwrite this field with the value provided in the PSI-L dst_tag field bits 7:0	Yes

Copyright © 2026 Texas Instruments Incorporated

Table 11-11. Host Packet Descriptor Linking Word 0 (PD Word 4)

Bits	Name	Description	Rx Overwrite
31:0	Next Descriptor Pointer LSB	The 32 LSBs of the 48-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero then the current buffer is the last buffer in the packet. The host sets the Next Descriptor Pointer.	No

Table 11-12. Host Packet Descriptor Linking Word 1 (PD Word 5)

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		No
19:16	Next Descriptor Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	No
15:0	Next Descriptor Pointer MSB	The 16 MSBs of the 48-bit next descriptor pointer. If a specific SoC uses less than 48 bits of address reach the width of this field may be reduced.	No

Table 11-13. Host Packet Descriptor Buffer 0 Info Word 0 (PD Word 6)

Bits	Name	Description	Rx Overwrite
31:0	Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will be written during reception. These are the 32 LSBs of the 48-bit buffer pointer. **THIS Value along with the MSB bits cannot be 0 or the buffer will be seen as a NULL PTR**	Yes

Table 11-14. Host Packet Descriptor Buffer 0 Info Word 1 (PD Word 7)

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		Yes

Copyright © 2026 Texas Instruments Incorporated

Table 11-14. Host Packet Descriptor Buffer 0 Info Word 1 (PD Word 7) (continued)

Bits	Name	Description	Rx Overwrite
19:16	Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	Yes
15:0	Buffer 0 Pointer MSB	The 16 MSBs of the 48-bit buffer pointer. If a specific SoC uses less than 48 bits of address reach the width of this field may be reduced.	Yes

Table 11-15. Host Packet Descriptor Buffer 0 Info Word 2 (PD Word 8)

Bits	Name	Description	Rx Overwrite
31:22	RESERVED	Written to 0	Yes
21:0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer.	Yes

Table 11-16. Host Packet Descriptor Reserved Word 0 (PD Word 9)

Bits	Name	Description	Rx Overwrite
31:0	RESERVED		No

Table 11-17. Host Packet Descriptor Reserved Word 1 (PD Word 10)

Bits	Name	Description	Rx Overwrite
31:0	RESERVED		No

Table 11-18. Host Packet Descriptor Reserved Word 2 (PD Word 11)

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		No
15:0	RESERVED		No

This word is only present if the Extended Packet Info Block present bit is set in Word 0

Copyright © 2026 Texas Instruments Incorporated

Table 11-19. Host Packet Descriptor Extended Packet Info Block Word 0 (Optional)

Bits	Name	Description	Rx Overwrite
31:0	Timestamp Info	This field contains an application specific timestamp which can be used for traffic shaping in a QoS enabled system.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

Table 11-20. Host Packet Descriptor Extended Packet Info Block Word 1 (Optional)

Bits	Name	Description	Rx Overwrite
31:0	Software Info 0	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

Table 11-21. Host Packet Descriptor Extended Packet Info Block Word 2 (Optional)

Bits	Name	Description	Rx Overwrite
31:0	Software Info 1	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

This word is only present if the Extended Packet Info Block present bit is set in Word 0.

Table 11-22. Host Packet Descriptor Extended Packet Info Block Word 3 (Optional)

Bits	Name	Description	Rx Overwrite
31:0	Software Info 2	This field stores software centric information that needs to travel with the packet through the stack. This information will be copied from the source descriptor to the destination descriptor whenever a prefetch operation is performed or when transferring through an infrastructure DMA node.	Configurable

Copyright © 2026 Texas Instruments Incorporated

Table 11-23. Host Packet Descriptor Protocol Specific Word N (Optional)

Bits	Name	Description	Rx Overwrite
31:0	Protocol Specific Data N	This field stores information which varies depending on the block and packet type.	Configurable

11.1.3.2.2 Host Buffer Descriptor

The Host Buffer Descriptor is identical in size and organization to a Host Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. Host Buffer Descriptors are designed to be linked onto a Host Packet Descriptor or another Host Buffer Descriptor to provide support for unlimited scatter / gather type operations. Host Buffer Descriptors provide information about a single corresponding data buffer. Every Host buffer descriptor stores the following information:

- Pointer to the first valid byte in the data buffer
- Length of the data buffer
- Pointer to the next buffer descriptor in the packet

Host Buffer Descriptors always contain 48 bytes of required information. Since it is a requirement that it is possible to convert a Host descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Host Buffer Descriptors will be allocated using the same sizes as Host Packet Descriptors. The Host Buffer Descriptor layout is shown below.

Table 11-24. Host Buffer Descriptor Layout

Buffer Reclamation Info (16 bytes)
Linking Info (8 bytes)
Buffer Info (12 bytes)
Original Buffer Info (12 bytes)

A Host Packet Descriptor and zero or more Host Buffer Descriptors may be linked together using the Next Descriptor Pointer fields to form packets. The last descriptor in a packet has a zero Next Descriptor Pointer. Each Host Buffer descriptor also points to a single data buffer.

The contents of the Host Buffer Descriptor words are detailed in : Table 11-25 through Table 11-33

Table 11-25. Host Buffer Descriptor Reserved Word 0 (BD Word 0)

Bits	Name	Description	Rx Overwrite
31:0	RESERVED	-	No

Table 11-26. Host Buffer Descriptor Reserved Word 1 (BD Word 1)

Bits	Name	Description	Rx Overwrite
31:0	RESERVED	-	No

Table 11-27. Host Buffer Descriptor Reserved Word 2 (BD Word 2)

Bits	Name	Description	Rx Overwrite
31:16	RESERVED	-	No
15:0	RESERVED	-	No

Copyright © 2026 Texas Instruments Incorporated

Table 11-28. Host Buffer Descriptor Reserved Word 3 (BD Word 3)

Bits	Name	Description	Rx Overwrite
31:0	RESERVED		No

Table 11-29. Host Buffer Descriptor Linking Word 0 (BD Word 4)

Bits	Name	Description	Rx Overwrite
31:0	Next Descriptor Pointer LSB	The 32 LSBs of the 48-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero then the current buffer is the last buffer in the packet. The host sets the Next Descriptor Pointer.	No

Table 11-30. Host Buffer Descriptor Linking Word 1 (BD Word 5)

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		No
19:16	Next Descriptor Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other ‘tiles’ on large devices.	No
15:0	Next Descriptor Pointer MSB	The 16 MSBs of the 48-bit next descriptor pointer. If a specific SoC uses less than 48 bits of address reach the width of this field may be reduced.	No

Table 11-31. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 6)

Bits	Name	Description	Rx Overwrite
31:0	Buffer 0 Pointer LSB	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value will be written during reception. These are the 32 LSBs of the 48-bit buffer pointer.	No

Copyright © 2026 Texas Instruments Incorporated

Table 11-32. Host Buffer Descriptor Buffer N Info Word 1 (BD Word 7)

Bits	Name	Description	Rx Overwrite
31:20	RESERVED		No
19:16	Buffer 0 Pointer Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other 'tiles' on large devices.	No
15:0	Buffer 0 Pointer MSB	The MSBs of the 48-bit buffer pointer. If a specific SoC uses less than 48 bits of address reach the width of this field may be reduced.	No

Table 11-33. Host Buffer Descriptor Buffer N Info Word 2 (BD Word 8)

Bits	Name	Description	Rx Overwrite
31:22	RESERVED		No
21:0	Buffer N Length	The Buffer Length field indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. This value will be overwritten during reception.	No

11.1.3.2.3 Transfer Request Descriptor

The Transfer Request Descriptor contains the following information:

- Indicator which identifies the descriptor as a TR Descriptor
- Set of one or more Transfer Request Records
- Set of one or more Transfer Response Records

Transfer Request Packet Descriptors always contain 16 bytes of required information and a variable number of Transfer Request / Transfer Response Records (request and response counts match).

The Transfer Request descriptor layout is shown below.

Table 11-34. Transfer Request Packet Descriptor Layout

Packet Info Given via Transfer Request Packet Descriptor Word 0-3 (16 bytes)
Null (0-102 bytes to bring start of TR request array into natural alignment for memory fetch efficiency)

Copyright © 2026 Texas Instruments Incorporated

Table 11-34. Transfer Request Packet Descriptor Layout (continued)

Array of Transfer Request Records
Array of Transfer Response Records

Table 11-35. Transfer Request Packet Descriptor Word 0

Bits	Name	Description	Rx Overwrite
31:30	2'd3	TR Packet Descriptor type.	No
29	RESERVED		No
28:20	Reload Count	Specifies what to do when the last entry is processed in this packet. This field specifies how many times to return to the Reload Index upon reaching the Last Entry. When an internal count is incremented to this value and the TR indicated by the Last Entry has been processed, this packet will be considered complete and the descriptor will be placed back on the return queue specified in Word 2. A value of 0x1FF indicates that a perpetual loop is desired. In this case, the loop count is considered infinite and the internal count will not be incremented. A teardown operation on the channel will cause the loop to be broken at the nearest iteration boundary.	No
19:14	Reload Index	Specifies the value to set the current processing index to when the last entry is processed and the Reload Enable is set to 1. This is basically an absolute index to jump to on the 2**nd** and following passes through the TR packet.	No
13:0	Last Entry	Specifies the index of the last valid entry in this packet	No

Table 11-36. Transfer Request Packet Descriptor Word 1

Bits	Name	Description	Rx Overwrite
31:28	RESERVED	-	No
27	RESERVED		No

Copyright © 2026 Texas Instruments Incorporated

Table 11-36. Transfer Request Packet Descriptor Word 1 (continued)

Bits	Name	Description	Rx Overwrite
26:24	Transfer Request Nominal Element Size	Specifies the stride between TR entries. The value in this field must be set large enough that any TR in the buffer will fit within the given dimension. TRs are expected to be placed on boundaries as given in this dimension. This field is also used to calculate the location where the Transfer Responses will be written back. This field is encoded as follows: 0 = 16-byte Transfer Request record size 1 = 32-byte Transfer Request record size 2 = 64-byte Transfer Request record size 3 = 128-byte Transfer Request record size 4-7 = RESERVED	No
23:14	RESERVED	-	No
13:0	RESERVED	-	No

Table 11-37. Transfer Request Packet Descriptor Word 2

Bits	Name	Description	Rx Overwrite
31:17	RESERVED		Yes
16	RESERVED	-	Yes
15:0	RESERVED	-	Yes

Table 11-38. Transfer Request Packet Descriptor Word 3

Bits	Name	Description	Rx Overwrite
31:24	RESERVED	-	No
15:0	RESERVED	-	No

Table 11-39. Transfer Request Packet Descriptor Payload Words 0-N

Bits	Name	Description	Rx Overwrite
31:0	Transfer Control Record Array	Transfer Control Records. This is an array of records which encapsulate the Transfer Request and Transfer Response messages which are used by the BCDMA	Yes (partial)

Copyright © 2026 Texas Instruments Incorporated

11.1.3.3 Transfer Request Record

11.1.3.3.1 Overview

This section describes the standard transfer request (TR) format to initiate a DMA transfer. The TRs can support both half and full duplex with multiple dimensions. Each TR will also generate a TR response. A TR can be sent to the BCDMA either through a TR descriptor using a pass-by-reference ring or as a Direct TR submission using a pass-by-value ring.

Table 11-40. Terms and Definitions

Term	Definition
TR	A transfer request to move data.
CR	A cache operation request. A request to send messages to a specified cache controller to prepare the cache for a future operation.

11.1.3.3.2 Addressing Algorithm

For a basic TR request the same algorithm is used. The innermost loop always consumes physically contiguous elements from memory. Its implicit dimension is 1 byte. The pointer itself moves from fetch to fetch on the maximum byte alignments specified for the BCDMA, in increasing order. In each level outside the inner loop, the loop moves the pointer to a new location based on the size of that loop level's dimension.

11.1.3.3.2.1 Linear Addressing (Forward)

The following code illustrates the basic algorithm for a 4-level loop nest block move. This model assumes that it transfers data 1 byte at a time and the input and output block are the same size in all dimensions.

```
// sptr is a byte pointer.
// dptr is a byte pointer.
// Source address is indirect
if (TR_ISA) {
    sptr = *sptr;
}
// Destination address is indirect
if (TR_IDA) {
    dptr = *dptr;
}
// TR_TRIGX can be selected to come from Global events, Local event, or none.
// Check for trigger of TYPE0
if (TR_TRIG0_TYPE == TYPE0) while(!TR_TRIG0);
if (TR_TRIG1_TYPE == TYPE0) while(!TR_TRIG1);
for (i3 = 0; i3 < ICNT3; i3++)
{
    sptr3 = sptr; // save current position before entering next level
    dptr3 = dptr; // save current position before entering next level
    // Check for trigger of TYPE1
    if (TR_TRIG0_TYPE == TYPE1) while(!TR_TRIG0);
    if (TR_TRIG1_TYPE == TYPE1) while(!TR_TRIG1);
    for (i2 = 0; i2 < ICNT2; i2++) {
        sptr2 = sptr; // save current position before entering next level
        dptr2 = dptr; // save current position before entering next level
        // Check for trigger of TYPE2
        if (TR_TRIG0_TYPE == TYPE2) while(!TR_TRIG0);
        if (TR_TRIG1_TYPE == TYPE2) while(!TR_TRIG1);
        for (i1 = 0; i1 < ICNT1; i1++) {
            sptr1 = sptr; // save current position before entering next level
            dptr1 = dptr; // save current position before entering next level
            // Check for trigger of TYPE3
            if(TR_TRIG0_TYPE == TYPE3) while(!TR_TRIG0);
            if(TR_TRIG1_TYPE == TYPE3) while(!TR_TRIG1);
            for (i0 = 0; i0 < ICNT0; i0++) {
                // BCDMA can combine these in optimized burst aligned accesses.
                *dptr = *sptr;
                sptr = sptr++;
                dptr = dptr++;
            }
        // Update based on saved pointer for this level
        sptr = sptr1 + SDIM1;
```


Copyright © 2026 Texas Instruments Incorporated

```
        dptr = dptr1 + DDIM1;
        }
    // Update based on saved pointer for this level
    sptr = sptr2 + SDIM2;
    dptr = dptr1 + DDIM2;
    }
```

This form of addressing allows programs to specify regular paths through memory in a small number of parameters. Additionally, it allows for various stall points through the loop to allow for hardware or software induced pausing of the transfer. The following section defines these parameters in detail.

11.1.3.3.3 Transfer Request Formats

The BCDMA defines a four-level loop nest for addressing elements within the TR, using the algorithms described in the Addressing algorithm. Most of the fields in the TR template map directly to the parameters in those algorithms.

Table 11-41. Transfer Request Fields

Field Name	Description	Size
FLAGS	TR flags that specify type of TR and how the TR should be handled. It also supports the TR triggering and output events.	32 bits
ICNT0	Total loop iteration count for level 0 (innermost)	16 bits
ICNT1	Total loop iteration count for level 1	16 bits
ADDR	Starting address for the source data or destination data if it is a half-duplex write.	64 bits
DIM1	Signed dimension for loop level 1 for the source data	32 bits
ICNT2	Total loop iteration count for level 2	16 bits
ICNT3	Total loop iteration count for level 3 (outermost)	16 bits
DIM2	Signed dimension for loop level 2	32 bits
DIM3	Signed dimension for loop level 3	32 bits
DDIM1	Signed dimension for loop level 1 for the destination data	32 bits
DADDR	Starting address for the destination of the data	64 bits
DDIM2	Signed dimension for loop level 2 for the destination data	32 bits
DDIM3	Signed dimension for loop level 3 for the destination data	32 bits
DICNT0	Total loop iteration count for level 0 (innermost) used for destination	16 bits
DICNT1	Total loop iteration count for level 1 used for destination	16 bits
DICNT2	Total loop iteration count for level 2used for destination	16 bits

Copyright © 2026 Texas Instruments Incorporated

Table 11-41. Transfer Request Fields (continued)

Field Name	Description	Size
DICNT3	Total loop iteration count for level 3 used for destination	16 bits

The TR assumes all iteration counts as unsigned integers, and all dimensions as signed integers representing byte offsets.

The template above fully specifies the type of elements, length, and dimensions of the transfer.

Since all transfers will not require all the fields the TRs have the type field which allow the number of words required to be sent for the TR to be reduced.

11.1.3.3.4 Flags Field Definition

The next diagram expands the 32-bit FLAGS field. The numbers above each field here indicate bit numbers within the field.

Table 11-42. Transfer Request Template FLAGS Field

31			24	23			16
CONFIGURATION SPECIFIC FLAGS				Reserved			
15			8	7	5	4	3 0
TRIGGER_SIZE1	TRIGGER1	TRIGGER_SIZE0	TRIGGER0	EVENT_SIZE	WAIT	STATIC	TYPE

The flags field fills in the remaining details about the stream, as follows:

Table 11-43. FLAGS Field Descriptions

Bit	Field	Description
0-3	**TYPE**	The TYPE of TR that is being sent 0: One dimensional data move 1: Two dimensional data move 2: Three dimensional data move 3: Four dimensional data move 4: Four dimensional data move with data formatting 5: Four dimensional Cache Warm 6-7: Reserved 8: Four Dimensional Block Move 9: Four Dimensional Block Move with Repacking 10: Two Dimensional Block Move 11: Two Dimensional Block Move with Repacking 12-14: Reserved 15 Four Dimensional Block Move with Repacking and Indirection
4	Reserved	Reserved for Future use.

Copyright © 2026 Texas Instruments Incorporated

Table 11-43. FLAGS Field Descriptions (continued)

Bit	Field	Description
5	WAIT	This field is only valid on Type 15 TRs. This field indicates whether or not this TR should ensure it completes before allowing the next TR to start on this channel. The encoding is as follows: 0 = Allow next TR to start immediately 1 = Wait for this TR to complete before allowing next TR to start When set, the next TR will not be considered triggered (regardless of any other trigger conditions) until all write status responses for the current TR have landed.
6-7	EVENT_SIZE	This is how often the TR will generate an output event. 0: Event is only generated with the TR is complete 1: Event is generated when the second inner most loop (ICNT1) is decremented by 1. 2: Event is generated when the third inner most loop (ICNT2) is decremented by 1. 3: Event is generated when the outer most loop (ICNT3) is decremented by 1.
8-9	TRIGGER0	This is one of two selectable triggers. The receipt of this trigger will enable the TR to be active for enough data transfer as specified by the TRIGGER0_TYPE Field. 0: No Trigger 1: Global Trigger 0 for the channel 2: Global Trigger 1 for the channel 3: Local Event for the channel
10-11	TRIGGER0_TYPE	This is the type of data transfer that will be enabled by receiving a trigger. 0: The second inner most loop (ICNT1) will be decremented by 1. 1: The third inner most loop (ICNT2) will be decremented by 1. 2: The outer most loop (ICNT3) will be decremented by 1. 3: The entire TR will be allowed to complete.
12-13	TRIGGER1	This is one of two selectable triggers. The receipt of this trigger will enable the TR to be active for enough data transfer as specified by the TRIGGER1_TYPE Field. 0: No Trigger 1: Global Trigger 0 for the channel 2: Global Trigger 1 for the channel 3: Local Event for the channel This field is reserved for a Direct TR.

Copyright © 2026 Texas Instruments Incorporated

Table 11-43. FLAGS Field Descriptions (continued)

Bit	Field	Description
14-15	**TRIGGER1_TYPE**	This is the type of data transfer that will be enabled by receiving a trigger. 0: The second inner most loop (ICNT1) will be decremented by 1. 1: The third inner most loop (ICNT2) will be decremented by 1. 2: The outer most loop (ICNT3) will be decremented by 1. 3: The entire TR will be allowed to complete.
16-23	Reserved	Reserved for Future use.
24-31	**Configuration Specific Flags**	These are flag bits that can be specified by the specific BCDMA configuration. These bits will remain undefined in the global TR format to allow for customization requirements that might be required in the specific BCDMA implementation.

The following sections expand on each of these fields.

11.1.3.3.4.1 Type: TR Type Field

The TR Type field gives the size of the TR and which fields are expected in the TR. Based on the type the number of words required to be sent to complete the TR can be decreased. The tables below show the minimum size for each TR as well if any fields will be treated as reserved.

Table 11-44. Transfer Request Minimum Size Type 0 One Dimensional Transfer

word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
FLAGS	ICNT0	ADDR		RESERVED/NOT REQUIRED											

Table 11-45. Transfer Request Minimum Size Type 1 Two Dimensional Transfer

word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
FLAGS	ICNT0/ 1	ADDR		DIM1	RESERVED/NOT REQUIRED										

Table 11-46. Transfer Request Minimum Size Type 2 Three Dimensional Transfer

word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2	DIM2	RESERVED/NOT REQUIRED								

Table 11-47. Transfer Request Minimum Size Type 3 Four Dimensional Transfer

word 0	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
FLAGS	ICNT0/ 1	ADDR		DIM1	ICNT2/ 3	DIM2	DIM3	RESERVED/NOT REQUIRED							

Copyright © 2026 Texas Instruments Incorporated

Table 11-48. Transfer Request Minimum Size Type 15 Four Dimensional Block Copy with Repacking and Indirection Support

w0	word 1	word 2	word 3	word 4	word 5	word 6	word 7	word 8	word 9	word 10	word 11	word 12	word 13	word 14	word 15
FLAGS	ICNT0/1	ADDR		DIM1	ICNT2/3	DIM2	DIM3	RESERVED	DDIM1	DADDR		DDIM2	DDIM3	DICNT0/1	DICNT2/3

11.1.3.3.4.2 EVENT_SIZE: Event Generation Definition

The BCDMA allows for an event to be generated at specified intervals for each TR. As the TR passes through the various loops in the addressing algorithm it will generate an event that then can be routed to other event receivers, such as other channels or interrupts to processors.

Table 11-49. Event Size Encoding

Value	Event Type	When it Fires
0	Completion Event	When the TR is complete and all status for the TR has been received.
1	ICNT1 Decrement	Type 0: When the last data transaction is sent for the TR. Type 1-11: When ICNT1 is decremented
2	ICNT2 Decrement	Type 0 – 1,10-11: When the last transaction is sent for the TR. All Other Types: When ICNT2 is decremented
3	ICNT3 Decrement	Type 0 – 2,10-11: When the last transaction is sent for the TR. All Other Types: When ICNT3 is decremented

11.1.3.3.4.3 TRIGGER_INFO: TR Triggers

The TR allows for two different triggers to be set to allow for the data transfer to be prevented or halted through the process of transferring the data. The triggers are specified by selecting a type of trigger and the size of the transfer that can be allowed for the receipt of a given trigger. The triggers themselves in the BCDMA are collected on a per channel basis and for each of three sources. Anytime the trigger event is received the internal counter is incremented. The trigger event will not be cleared until the specified block has started its transfer and it will **only** clear the triggers that are active.

CAUTION

This does allow for one TR in the channel to use global event 0 and the next TR to use global event 1. If while the first TR is running global event 0 can increment and will decrement each time the TR reaches the specified level. At the same time global event 1 will be incremented whenever it is received but it will not decrement until the next TR runs. If the global event 1 counter reaches overflows then an error event will NOT be generated but the event will be lost.

11.1.3.3.4.4 TRIGGERX_TYPE: Trigger Type

The trigger type sets the value of TR_TRIGX as shown in the addressing algorithm description. The TR_TRIGX value allows for the temporary halting of the TR until the expected event has been received. After the given loop has been entered the selected trigger will be decremented.

Copyright © 2026 Texas Instruments Incorporated

11.1.3.3.4.5 TRIGGERX: Trigger Selection

The TR is able to select up to 2 of 3 trigger sources for a given TR. The events can come from the dedicated local event on the BCDMA itself or from two assigned global events that come from the PSI-L interface on the BCDMA. The trigger counters are always enabled so that events can be received prior to the TR getting loaded. The trigger counters are only decremented when the trigger is active and the TR has scheduled the first transfer of the transaction.

11.1.3.3.4.6 Configuration Specific Flags Definition

The configuration specific flags are specific to a given TR type. If a TR type does not require additional flags the field should be filled with 0's. Currently only Types 0-4 and Type 15 support the following configuration specific flags:

Bit	Field	Description
0	**ISA**	Indirect Source Address 0: Source address in TR is a directly usable pointer to the data 1: Source address in TR is a pointer to a 64-bit location that contains the actual pointer to the data
1	**IDA**	Indirect Destination Address 0: Destination address in TR is a directly usable pointer to the data 1: Destination address in TR is a pointer to a 64-bit location that contains the actual pointer to the data
2	**SUPR_EVT**	Suppress Event Output: 0 = Output events will be generated according to FLAGS.EVENT_SIZE field 1 = No output events will be generated for the duration of this TR execution This field is only valid on split and type 15 TRs (Type 0-3,15)
3	**Reserved**	Reserved for Future use.

Copyright © 2026 Texas Instruments Incorporated

Bit	Field	Description
6:4	EOL	This field is only valid on split TRs (Type 0-3). On source (Read) split TRs, this field specifies whether or not the EOL delimiters should be produced on the Tx PSI-L bus. The encodings of this field for source split TRs is as follows: 0 = SOL/EOL match SOP/EOP 1 = SOL/EOL boundaries are each ICNT0 bytes 2 = SOL/EOL boundaries are each ICNT0*ICNT1 bytes 3 = SOL/EOL boundaries are each ICNT0*ICNT1*ICNT2 bytes 4 = SOL/EOL boundaries are each ICNT0*ICNT1*ICNT2*ICNT3 bytes On destination (Write) split TRs, this field specifies how to handle EOL delimiters when they are encountered on the Rx (write) side of a TR. EOL delimiters can be produced from the Tx (read) side of a block copy operation or from a remotely paired peripheral when operating in split TR mode. The encodings of this field are as follows: 0: Ignore EOL 1: Line length is icnt0 bytes. Clear any remaining ICNT0 bytes and increment ICNT1 by 1 2: Line length is icnt0*icnt1 bytes. Clear any remaining ICNT0/1 bytes and increment ICNT2 by 1 3: Line length is icnt0*icnt1*icnt2 bytes. Clear any remaining ICNT0/1/2 bytes and increment ICNT3 by 1 4: Line length is icnt0*icnt1*icnt2*icnt3 bytes. Move on to next TR 5-7: RESERVED
7	EOP	This TR should generate an EOP on the streaming interface for any downstream packet oriented consumers to denote that a ‘packet’ of data is complete 0: No EOP flag will accompany the last PSI-L data phase associated with transfers from this TR 1: On egress the DMA will set the EOP flag coincident with transferring the last of the data for this TR. If the TR data does not complete an entire PSI-L data phase then the remaining bytes in the data phase will be skipped and the internal FIFO pointer will be updated to begin packing new data on a new data phase boundary.

Copyright © 2026 Texas Instruments Incorporated

11.1.3.3.5 TR Address and Size Attributes

The fields described below all describe the data transfer that needs to be made as described in the Linear Memory Addressing Block Move Algorithm.

11.1.3.3.5.1 ICNT0

The ICNT0 is the number of elements to transfer in the inner loop of the TR. If the TR type does not contain a FMTFLAGS field then this count is assumed to be the number of bytes to transfer. If the FMTFLAGS field is included in the types then the number of bytes to be transferred will be ICNT0 times the Element size rounded up to the nearest byte.

11.1.3.3.5.2 ICNT1

The ICNT1 field is the loop count for the second innermost loop count as defined in the Addressing algorithm.

11.1.3.3.5.3 ADDR

The address location is the initial address that will be accessed at the start of the transfer. All of the dimensions will also be based off of this value. This address can either physical or virtual based upon settings in the DMA channel Configuration register.

While the ADDR field is 64 bits wide, the usable extent of the address is up to 48 bits of absolute offset plus a 4-bit address space selector which indicates 1 of 16 different orthogonal address spaces that the pointer is referencing within. The format of the ADDR field is given as follows:

Bits	Subfield	Description
63:52	Reserved	Reserved
51:48	Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other ‘tiles’ on large devices.
47:0	Address	The 48-bit source or source starting address for the transfer. This address is assumed to be a physical address. The actual width implemented for the address field on any given DMA instance may be adjusted to the needs of each specific KSLC SoC. 48 bits is just the maximum size which is envisioned to be supported. Some systems may have address widths as low as 32 or 36 bits. The HW implementation may choose to provide address width configurability in order to reduce costs.

11.1.3.3.5.4 DIM1

This is the offset of the address from the initial address for the first access of the second loop. This will be added to the address from the loop before. This is a signed value so that the address can be both above and below the initial address.

Copyright © 2026 Texas Instruments Incorporated

11.1.3.3.5.5 ICNT2

This is the count for the third innermost loop in the addressing algorithm.

11.1.3.3.5.6 ICNT3

This is the count for the outermost loop in the addressing.

11.1.3.3.5.7 DIM2

This is the offset of the address from the initial address to the next address in the third innermost loop. This is a signed value so that the address can be both above and below the previous address.

11.1.3.3.5.8 DIM3

This is the offset of the address from the initial address to the next address in the outermost loop. This is a signed value so that the address can be both above and below the previous address.

11.1.3.3.5.9 DDIM1

This is the offset of the destination address from the initial destination address for the first access of the second loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

11.1.3.3.5.10 DADDR

The destination address location is the initial destination address that will be accessed at the start of the transfer. All of the dimensions will also be based off of this value. This address can either physical or virtual based upon settings in the DMA channel Configuration register

While the DADDR field is 64 bits wide, the usable extent of the address on a KSLC system is up to 48 bits of absolute offset plus a 4 bit address space selector which indicates 1 of 16 different orthogonal address spaces that the pointer is referencing within. The format of the DADDR field is given as follows:

Bits	Subfield	Description
63:52	Reserved	Reserved
51:48	Address Space Select	Effectively bits 51:48 of the address. The value given in this field will be output by the DMA initiators on the casel pin which is used by the infrastructure as an identifier for which address space this particular memory region is located within. Address space 0 is the default unified address space for a given device. Address spaces 1-15 are used for alternate address maps which may be external to the device (PCIe/Hyperlink) or in other ‘tiles’ on large devices.
47:0	Address	The 48-bit starting destination address for the transfer. This address is assumed to be a physical address. The actual width implemented for the address field on any given DMA instance may be adjusted to the needs of each specific KSLC SoC. 48 bits is just the maximum size which is envisioned to be supported. Some systems may have address widths as low as 32 or 36 bits. The HW implementation may choose to provide address width configurability in order to reduce costs.

Copyright © 2026 Texas Instruments Incorporated

11.1.3.3.5.11 DDIM2

This is the offset of the destination address from the initial destination address for the first access of the third loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

11.1.3.3.5.12 DDIM3

This is the offset of the destination address from the initial destination address for the first access of the outer loop. This will be added to the destination address from the loop before. This is a signed value so that the destination address can be both above and below the initial destination address.

11.1.3.3.5.13 DICNT0

This is the number of elements to use in the destination if the TR is repacking the data. This number reflects the number of elements to be transferred.

11.1.3.3.5.14 DICNT1

This is the number of times to execute the second loop to use in the destination transfer.

11.1.3.3.5.15 DICNT2

This is the number of times to execute the third loop to use in the destination transfer.

11.1.3.3.5.16 DICNT3

This is the number of times to execute the fourth loop to use in the destination transfer. The value of DICNT0 x DICNT1 x DICNT2 x DICNT3 must equal ICNT0 x ICNT1 x ICNT2 x ICNT3. If any of the values are zero they are NOT included in the multiplication.

11.1.3.4 Transfer Response Record

Upon completing a TR, the BCDMA will send back a TR Response. The BCDMA TR response format is a single 32 bit word as shown in the Table 11-51 below.

Table 11-50. Transfer Request Response Template STATUS FLAGS Field

31	24	23	16
CONFIGURATION SPECIFIC STATUS		RESERVED	
15	8	7	0
RESERVED		STATUS_INFO	STATUS_TYPE

Table 11-51. STATUS FLAGS Field Descriptions

Bit	Field	Description
3:0	**STATUS_TYPE**	The completion status of the TR.
7:4	**STATUS_INFO**	Information that is unique based on the STATUS_TYPE returned
15:8	**Reserved**	Reserved for Future use
23:16	**Reserved**	Reserved for Future use

Copyright © 2026 Texas Instruments Incorporated

Table 11-51. STATUS FLAGS Field Descriptions (continued)

Bit	Field	Description
31:24	**Configuration Specific Flags**	These are flag bits that can be specified by the specific BCDMA configuration. These bits will remain undefined in the global TR format to allow for customization requirements that might be required in the specific BCDMA implementation. For BCDMA channels which are capable of interfacing to endpoint peripherals (or via a PDMA) the format of these flags are as follows: 31:28 error_flags – the error_flags field as is provided across PSI-L 27:24 ps_flags – the ps_flags field as is provided across PSI-L

11.1.3.4.1 STATUS Field Definition

The Status field is made up of two sub-fields STATUS_TYPE and STATUS_INFO. A value of 0 means it completed as requested any other value means an error has occurred. The Table below states the legal STATUS_TYPEs and the use of the STATUS_INFO in each case.

11.1.3.4.1.1 STATUS_TYPE Definitions

The STATUS_TYPE field is used to determine what type of status is being returned. Depending on the type of Status it might additionally have information in the STATUS_INFO to give more details about the specific reason why the status was returned.

Table 11-52. STATUS_TYPE Encoded Values

Value	Error Type	Completion	STATUS INFO Definition
0	None	Complete	None
1	Transfer Error	None to Partial	CBA Non-Complete Status Received
2	Aborted Error	None to Partial	None
3	Submission Error	None	Submission Error Type
4	Unsupported Feature	None	Feature Type
5	Transfer Exception	Partial	Exception Type
6	Teardown Flush	None	None
7-15	Reserved	Unknown	Reserved

11.1.3.4.1.1.1 Transfer Error

A transfer error occurs anytime that one of the CBA transactions performed by the BCDMA returns a status other than complete. The BCDMA may continue the transfer or may abort the transfer and return back to an IDLE state. Once the transfer is finished (aborted or complete) the BCDMA will send back the TR Response with the STATUS TYPE of Transfer Error. The STATUS INFO will contain the 3 bit CBA Status field the upper bit specifies if it was a read status (1) or a write status (0).

11.1.3.4.1.1.2 Aborted Error

An aborted error occurs if the PSI-L interface asserts the drop signal prior to the completion of the TR. The BCDMA will return back to an IDLE state and send back the TR Response with the STATUS TYPE. If the

Copyright © 2026 Texas Instruments Incorporated

BCDMA receives a transfer error after receiving the abort the transfer error should be reported instead of the abort error.

11.1.3.4.1.1.3 Submission Error

A submission error is returned for a TR that is received that cannot be run. The STATUS_INFO field specifies the type of submission errors.

Table 11-53. Submission Error STATUS_INFO Encodings

Value	Submission Error Type
0	ICNT0 was 0
1	Channel FIFO was full when TR received
2	Channel is not owned by the submitter. Either CC submitting to Direct or Direct submitting to a CC channel
3	Reserved
4	Reserved
5	Bad descriptor type
6-15	Reserved

11.1.3.4.1.1.4 Unsupported Feature

An unsupported feature error is returned for a TR that is received that cannot be run because it specifies a feature that is optional and not supported by the BCDMA that received the TR. The STATUS INFO specifies the feature that was not supported. If it specifies multiple features that were not supported the BCDMA implementation can determine which type it wants to return.

Table 11-54. Unsupported Feature STATUS_INFO Encodings

Value	Submission Error Type
0	TR Type not supported
1	STATIC not supported
2	EOL not supported
3	CONFIGURATION SPECIFIC not supported
4	AMODE not supported
5	ELTYPE not supported
6	DFMT not supported
7	SECTR not supported
8	AMODE SPECIFIC Field not supported
9-15	Reserved

11.1.3.4.1.1.5 Transfer Exception

A transfer exception is returned for a TR that completed but experienced a known exception during reception. The STATUS_INFO field specifies the type of transfer exception that was encountered.

Table 11-55. Transfer Exception STATUS_INFO Encodings

Value	Transfer Exception Type
0	EOP on incoming data stream was encountered prematurely (short packet)
1	EOP on incoming data stream was encountered late (long packet)

Copyright © 2026 Texas Instruments Incorporated

Table 11-55. Transfer Exception STATUS_INFO Encodings (continued)

Value	Transfer Exception Type
2-15	Reserved

11.1.3.4.1.1.6 Teardown Flush

Split mode BCDMA Rx channels (write channels) can respond with a teardown flush in the case that they have received a teardown message, all data has been transferred , and TRs have been prefetched (in anticipation of more data being received). In this case, the BCDMA will simply return the prefetched TR to the completion queue with the status type set to teardown flush.

11.1.3.5 Channels

Each DMA controller instance contains one or more channels. A channel represents a single thread of strongly ordered operations whose purpose is to move data from one interface to another. Operations between channels are orthogonal and have no assumption of ordering but operations within a channel must be completed in order. The DMA controller uses time division multiplexing to allow work from channels to momentarily use shared data transfer units and data paths.

11.1.3.6 Flows

Each DMA channel will provide one or more flows which allow communication with one or more SW hosts. Each flow contains a single queue pair implemented within a shared circular buffer (ring) in memory. One queue provides uni-directional communication from SW to the DMA while the other queue provides uni-directional communication from the DMA back to SW. Flows are intended to allow traffic which is produced or consumed by a different Host SW process or traffic which is on different priorities to share a single physical DMA channel. A DMA channel will perform time division multiplexing between flows on specific work boundaries. Once a channel is in work on a specific flow, that entire unit of work will be completed before allowing work from a different flow to begin.

11.1.3.7 Queues

Queues are used to hold either values or references that need to be passed between SW and HW components in the system. Queues are implemented in the PKTDMA and BCDMA modules using memory mapped rings. Each ring is comprised of a single contiguous memory block which contains an array of elements. Each element is 8 bytes. The size of the ring is selectable from 1 entry up to 64K entries with single entry granularity. Each ring is used to implement a forward and a reverse queue which share the same data elements but which have independent pointers and occupancies. The forward queue is used to pass work from SW to HW and the reverse queue is used to pass completed work back from HW to SW. A doorbell MMR is provided per ring to allow SW to add elements to the forward queue. A separate doorbell MMR is provided per ring to allow SW to acknowledge the popping of elements from the reverse queue. An occupancy MMR is provided for the reverse queue so that SW can know how many entries are currently completed. Entries are returned on the reverse queue in the exact same order and in the exact same index as the work was provided on the forward queue and because of this, the DMA controllers do not actually modify the ring entry for completion operations (the only exception being for acknowledgement of a teardown operation). Completion is indicated by incrementing of the reverse occupancy only.

11.1.3.7.1 Queue Types

11.1.3.7.1.1 Transmit Queues

Tx channels use packet queues referred to as “transmit queues” to store the packets that are waiting to be transmitted. Tx Queues only pass a pointer to a descriptor. For the PKTDMA, Tx Queues contain pointers to Host descriptors. For the BCDMA, pass Tx Queues contain a pointer to a TR packet descriptor. Transmit Queues are implemented as the forward queue on the shared ring structure for a Tx flow.

Copyright © 2026 Texas Instruments Incorporated

11.1.3.7.1.2 Transmit Completion Queues

Tx channels also use packet queues referred to as “transmit completion queues” to return packets to the host after they are transmitted. A Tx Completion Queue is provided for each Tx Queue. The DMA controllers do not actually write anything back to completion queues (except for a teardown acknowledgement) but instead just increment the reverse occupancy. Transmit completion queues are implemented as the reverse queue on the shared ring structure for a Tx flow.

11.1.3.7.1.3 Free Descriptor / Buffer Queues

Free descriptor/buffer queues pass an empty list of pre-chained descriptors, each pointing to a single buffer, from SW to the DMA to use for receiving incoming packets. Free Descriptor/Buffer Queues are implemented as the forward queue on the shared ring structure for an Rx flow.

11.1.3.7.1.4 Receive Queues

Rx queues pass a packet which has been received from the DMA back to the SW. Receive queues are implemented as the reverse queue on the shared ring structure for an Rx flow.

11.1.3.7.1.5 Ring Based Queues Implementation

Work packets which are stored in a queue are shown in the following diagram:

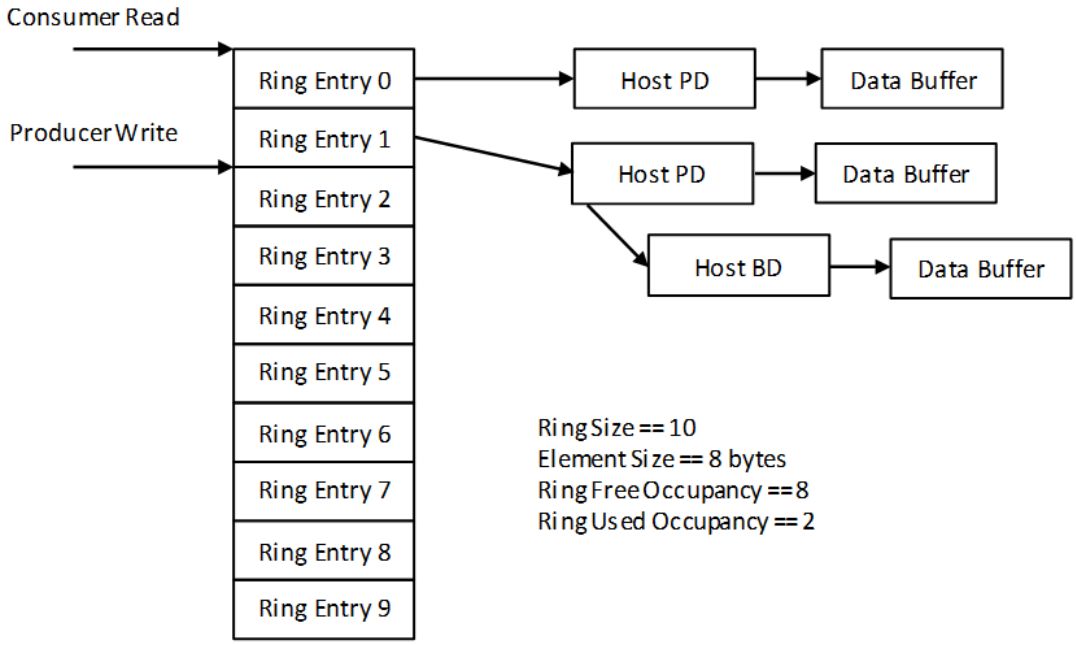


Figure 11-3. Ring Based Queue Structure

11.1.4 Operational Description

11.1.4.1 Resource Allocation

Rings, memory, interrupts, DMA channels, and flow table entries are shared resources within the data movement architecture and their ownership and use must be managed in order to provide stable, safe, and secure operation. The DMSS provides channelized firewalls which can be used to protect arrayed resources (like rings, channel control MMRs, etc.) from unwanted/unauthorized use as well as restricted access to the firewall configurations themselves. It is outside the scope of this document to define how the resource management system functions but it is assumed that all of these resources are isolate-able and secure-able.

Copyright © 2026 Texas Instruments Incorporated

11.1.4.2 PKTDMA/BCDMA - Ring Operation

11.1.4.2.1 Queue Initialization

The base address and size for each ring must be initialized via the provided MMRs prior to using the ring. Ring initialization is typically done only on channel / flow setup as once initialized the rings can be used indefinitely.

11.1.4.2.2 Queueing Entries

Entries can only be queued by SW onto the forward queue of a ring. To queue an entry the producer will write the entry contents (typically a pointer to the Packet Descriptor or a TR) to the memory location pointed to by the current write pointer and will decrement its internally maintained free entry occupancy. After the producer has confirmed that the data has actually landed it memory, it will then write to the forward doorbell register (RINGRT[a]_RT_FDB) for the ring to increment the pending forward occupancy count for the ring.

More than one entry can be queued with a single doorbell write as the entry count in the doorbell write indicates how many entries have been queued.

11.1.4.2.3 De-queueing Entries

Entries can only be de-queued by SW from the reverse queue of a ring. The consumer (entity which is going to de-queue an entry from the ring) maintains a current read pointer for each ring it controls. To de-queue an entry the consumer will read the occupancy from the reverse occupancy register (RINGRT[a]_RT_ROCC) and will then read the entry contents from the memory location pointed to by the current read pointer. The consumer will then write a decrement value to the reverse doorbell register (RINGRT[a]_RT_ROCC) for the ring to decrement the waiting occupancy count for the reverse queue within the ring and will increment its internally maintained free entry occupancy for the ring.

11.1.4.3 PKTDMA/BCDMA - Output Event Generation

Each channel and flow within a DMA has the ability to generate specific types of events which can be routed to the IA in the system and from there re-routed to any event consumer. Events are output from the DMA controllers using a simple integer encoding. Up to 4 events can be generated from any given channel but only event types which have functional value are supported on each specific type of channel. The DMA channel event types are listed as follows:

Table 11-56. Per-Channel/Flow Event Support and Encoding

Name	Description	Per	BCDMA BC	BCDMA Tx	BCDMA Rx	PKTDMA Tx	PKTDMA Rx
Ring Entry Zero/Non-zero	Asserts an up event anytime the ring transitions to a non-zero occupancy / teardown state. Asserts a down event anytime the ring transitions to a zero occupancy / teardown state.	Flow	X	X	X	X	X

Copyright © 2026 Texas Instruments Incorporated

Table 11-56. Per-Channel/Flow Event Support and Encoding (continued)

Name	Description	Per	BCDMA BC	BCDMA Tx	BCDMA Rx	PKTDMA Tx	PKTDMA Rx
Channel Error	Asserts an up event anytime an error occurs on the channel. Asserts a down event when the channel error bit is cleared.	Channel	First flow in channel only	First flow in channel only	First flow in channel only	First flow in channel only	First flow in channel only
Rx Starvation	Asserts an up event when starvation occurs on a flow. Asserts a down event when entries are added to the ring for that flow.	Flow					First flow in channel only
RX Flow FW Error	Asserts an up event when a flow id arrives on the RX interface that is not between the flowid_start and flowid end values for the channel. Asserts a down event when the RX Flow ID Firewall Status Register pend bit is cleared	Pktdma					X
Data Events	Asserts an up event when the selected loop counter has decremented. If the suppress event bit is not set in the TR	Channel	X	X	X		

Each BCDMA/PKTDMA instance will provide an event transport-lane (ETL) interface to pass outgoing events. The index field on this ETL will be calculated based on the event type and channel or flow offset. Each PKTDMA or BCDMA instance is configured with base values to which a channel or flow index is added for each type of event which can be generated. These base index values are listed in the specification for that specific instantiation of each DMA and are named as follows:

Copyright © 2026 Texas Instruments Incorporated

Table 11-57. BCDMA/PKTDMA Event Base Index Parameters

DMA	Event Base Parameter Name	Function	Offset
PKTDMA	tcomp_evtbase	Base index for Tx flow reverse ring completion events	Tx flow number
	rcomp_evtbase	Base index for Rx flow reverse ring completion events	Rx flow number
	terr_evtbase	Base index for Tx channel error events	Tx channel number
	rerr_evtbase	Base index for Rx channel error events	Rx channel number
	rstarve_evtbase	Base index for Rx flow starvation events	Rx flow number
	rflowfw_evtbase	Base index for Rx Flow Firewall event	None
BCDMA	tcomp_evtbase	Base index for split Tx flow reverse ring completion events	Tx flow number
	rcomp_evtbase	Base index for split Rx flow reverse ring completion events	Rx flow number
	bcomp_evtbase	Base index for normal Block Copy flow reverse ring completion events	BC flow number
	terr_evtbase	Base index for split Tx channel error events	Tx channel number
	rerr_evtbase	Base index for split Rx channel error events	Rx channel number
	berr_evtbase	Base index for normal Block Copy channel error events	BC channel number
	tdcomp_evtbase	Base index for Tx channel TR data completion events	Tx channel number
	rdcomp_evtbase	Base index for Rx channel TR data completion events	Tx channel number
	bdcomp_evtbase	Base index for Block Copy channel TR data completion events	BC channel number

Example:

The producer event map is set up so that the Tx flows from a PKTDMA instance are to start at 64 and proceed upward from there. The tcomp_evtbase for that PKTDMA instance is set to 64. In this configuration, the reverse ring entry completion event for flow 0 will produce an event on index 64, flow 1 will produce an event on index 65, etc. The terr_evtbase for the same PKTDMA instance is set to 128. In this configuration, error events for channel 2 would produce events with an index of 130.

11.1.4.4 PKTDMA - Transmit Channel Setup

This section describes the setup procedure for Tx channels within the PKTDMA. After a reset or a previous teardown operation but before queuing packets to a channel the host must initialize the channel's Tx Port DMA State. The host initializes the channel Tx Port DMA State by configuring all of the Tx flow entries for the channel and then writing to the Tx Channel Configuration Register A (TCHAN[a]_TCFG). The Host may choose to write

Copyright © 2026 Texas Instruments Incorporated

the enable bit in the Tx Channel Configuration Register A (TCHAN[a]_TCFG) at the same time or after it has written all of the channel parameters but note that every write to the Tx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

After a Transmit channel has been set up, packets can be added to the Queues for the channel to begin the transmit operation. The following sections describe how the transmit operations are performed for the various descriptor types.

11.1.4.5 PKTDMA - Transmit Channel Pause

Setting the tx_pause bit in the Tx Channel N Real-time Control Register (<TCHANRT[a]_TRT_CTL > [29] TX_PAUSE) will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potential underrun/overrun conditions from a downstream data sink or upstream data source.

11.1.4.6 PKTDMA - Transmit Channel Teardown

This section describes the teardown procedure for internal Tx channels within the PKTDMA. A Tx channel teardown is initiated by the host by writing the tx_teardown bit in the Tx Channel N Real-time Control Register (<TCHANRT[a]_TRT_CTL > [30] TX_TEARDOWN).

Once the host has written the tx_teardown bit, the PKTDMA will do the following:

1. Stops performing any additional descriptor fetches for the channel.
2. Completes any packets normally for which a pointer/descriptor/TR fetch has already been performed.
3. Sets the tdown bit on the EOP data phase of the last packet to be sent (if any traffic was pending when teardown was initiated) or sends a zero byte packet with the tdown, sop, and eop bits asserted. This signals the destination thread that all of the data has been sent.
4. Clears the channel enable in the Tx Channel N Global Configuration Register (<TCHANRT[a]_TRT_CTL> [31] TX_ENABLE).
5. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
6. Sets bit 31 of the Tx Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
7. If the current Tx reverse ring occupancy is 0, issues an Tx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.
8. If the current TX Reverse ring occupancy is not 0 then a write to the doorbell with a negative value of the current occupancy will clear any remaining work but the forward ring pointer will remain the same. If the ring pointer needs to be reset then perform a write to any of the ring configuration registers.
9. The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not.

The Host can determine that a teardown is complete by using either of the following methods:

1. Periodically polling the teardown and enable bits for the channel
2. Waiting for an interrupt and observing the teardown complete bit is set in the channel's default flow (first flow in channel) reverse occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) for the channel.

11.1.4.7 PKTDMA - Transmit Operation

After a channel has been set up it can begin to be used to transmit packets. Packet transmission involves the following steps:

1. The Host is made aware of one or more chunks of data in memory that need to be transmitted as a packet. This may involve directly sourcing data from the Host or it may involve data which has been forwarded from another data source in the system.
2. The Host allocates and populates a host packet descriptor. The host will initialize the following fields within the packet descriptor:

Copyright © 2026 Texas Instruments Incorporated

a. Descriptor Type (set to Host).
b. Packet Length indicating the total number of bytes that are to be read from all of the buffers for this packet.
c. Source Tag.
d. Destination Tag.
e. Packet Type.
f. Any protocol specific flags for the given packet type.
g. Buffer Pointer with the byte aligned address of the first chunk of buffer data.
h. Buffer Length with the number of bytes in the first chunk of buffer data.
i. The Next Descriptor Pointer with the 16-byte aligned address of the next descriptor in this packet. If this is the last chunk of data in the packet, this field must be set to zero.
j. Any protocol specific descriptor sections that are required for the given packet type or system configuration.

3. The Host allocates and populates host buffer descriptors as necessary to point to any remaining chunks of data that belong to this packet. The host will initialize the following fields within the host buffer descriptor:

   a. Buffer Pointer with the byte aligned address of the given chunk of buffer data.
   b. Buffer Length with the number of bytes in the given chunk of buffer data.
   c. The Next Descriptor Pointer with the 16-byte aligned address of the next descriptor in this packet. If this is the last chunk of data in the packet, this field must be set to zero.

4. The Host writes queues the packet onto one of the Transmit Queues for the desired DMA channel. Channels may provide more than one Tx Queue (if more than 1 flow is provided) and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller / scheduler implementation.
5. The PKTDMA internally generates a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
6. The PKTDMA Tx engine is eventually brought into context for the corresponding channel and begins to process the packet.
7. The PKTDMA reads the packet descriptor pointer from the memory mapped ring for that channel/flow.
8. The PKTDMA reads the packet descriptor from memory.
9. The PKTDMA empties each buffer in sequence by transmitting the contents in one or more block data moves. The size of these blocks is application specific. The PKTDMA module specification is intended to document the block size used by a given implementation. As each buffer is emptied, the PKTDMA will read the next buffer descriptor to obtain the pointer and size of the data buffer as well as the pointer to the next descriptor in the chain.
10. When all data for the packet has been transmitted as specified in the packet size field, the DMA will increment the occupancy of the reverse queue (Tx Completion Queue).
11. After the occupancy is incremented, the PKTDMA will indicate the status of the Tx Completion Queue by sending an up event.
12. The Interrupt Aggregator receives the up event and sets the corresponding bit in the interrupt status register (VINT[a]_STATUS_SET) as programmed in the interrupt mapping registers. This in turn causes an interrupt to the Host to be generated.
13. The Host responds to the status change from the PKTDMA (via the Interrupt Aggregator) and performs garbage collection as necessary for the packet.
14. During garbage collection the Host will write to the reverse queue Doorbell register (RINGRT[a]_RT_RDB) for the channel/flow to acknowledge the popping of those completed packets. The doorbell writes eventually cause the queue to become empty.
15. The PKTDMA will send a down event to the Interrupt Aggregator which will clear the corresponding bit in the Interrupt Status Register (VINT[a]_STATUS) and potentially de-assert the interrupt line.

Copyright © 2026 Texas Instruments Incorporated

11.1.4.8 PKTDMA - Receive Free Descriptor / Buffer Queue Setup

The Host is ultimately responsible for providing all of the free descriptors and buffers that the port uses as it receives packets. Each entry on the Rx Free Descriptor/Buffer Queue consists of one or more descriptors chained together with each descriptor also pointing to a single contiguous buffer. Each entry on the Rx Free Descriptor/Buffer queue is intended to receive one packet of data. All chaining of descriptors within a packet is performed prior to placing the Rx resources onto the free queues.

Before the Host adds each Rx buffer descriptor chain to the queue, it must first initialize the Rx buffer descriptor values as follows:

- Write the Buffer Pointer with the byte aligned address of the buffer data
- Write the Buffer Size
- Write the next descriptor pointer to the next descriptor in the chain (non-eop) or to 0x0 (eop)

All other fields in the Descriptor do not need to be initialized as they will be overwritten by the Port on reception.

11.1.4.9 PKTDMA - Receive Channel Setup

After a reset or a previous teardown operation but before receiving packets on a channel the host must initialize the channel's Rx Port DMA State. The host initializes the channel Rx Port DMA State by writing to the Rx Channel Configuration Registers. The Host may choose to write the enable bit in the Rx Global Channel Configuration Register (<RCHANRT[a]_RRT_CTL> [31] RX_ENABLE) at the same time or after it has written all of the channel parameters but note that every write to the Rx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

11.1.4.10 PKTDMA - Receive Channel Teardown

An Rx channel teardown is intended to be initiated at the data source (the source thread). Initiation is commanded by the host by writing the teardown bit in the PSI-L source thread register 0x408 . This will cause the source of the data to gracefully terminate any packet that is in flight and send a PSI-L data phase with the tdown signal asserted. When the PKTDMA receives a data phase with the tdown attribute asserted it will immediately set the internal rx_teardown bit in the Rx Channel N Real-time Control Register (<RCHANRT[a]_RRT_CTL> [30] RX_TEARDOWN).

Once a teardown has been initiated, the PKTDMA will do the following:

1. Completes any packets normally which may be pending in the ingress port buffers (what the port considers as pending packets is application specific).
2. Clears the channel enable in the Rx Channel Global Configuration Register (<RCHANRT[a]_RRT_CTL> [31] RX_ENABLE)**>**.
3. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
4. Sets bit 31 of the Rx Flow Reverse Ring Occupancy register to indicate that a teardown has completed (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE).
5. If the current Rx reverse ring occupancy is 0, issues an Rx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.
6. If the current RX Reverse ring occupancy is not 0 then a write to the doorbell with a negative value of the current occupancy will clear any remaining work but the forward ring pointer will remain the same. If the ring pointer needs to be reset then perform a write to any of the ring configuration registers.

The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not. The normally intended method of issuing a teardown on a packet mode channel is to initiate the teardown at the remote PSI-L data source and allow it to flow into the PKTDMA. If that is not possible, the host may write the rx_teardown bit directly to a 1.

The Host determines that a teardown is complete by using either of the following methods:

1. Periodically polling the teardown and enable bits for the channel

Copyright © 2026 Texas Instruments Incorporated

2. Waiting for an interrupt and observing the teardown complete bit is set in the channel's default flow (first flow in channel) reverse occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) for the channel.

11.1.4.11 PKTDMA - Receive Channel Pause

Setting the rx_pause bit in the Rx Channel N Control Register (<RCHANRT[a]_TRT_CTL> [29] RX_PAUSE) will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potential underrun/overrun conditions from a downstream data sink or upstream data source.

11.1.4.12 PKTDMA - Receive Operation

When packet reception begins on a given channel, the port will begin by fetching the descriptor and buffer from the ring for the channel / flow. If the SOP Buffer Offset in the Rx Flow Table entry is nonzero, then the port will begin writing data after the offset number of bytes in the SOP buffer. The port will then continue filling that buffer and will fill additional descriptors + buffers as needed using the pre-linked buffers in the packet.

A detailed summary is as follows:

1. Host allocates, populates, and places pointers to free descriptor/buffer structures onto Rx Free Descriptor/ Buffer Queues
2. PKTDMA fetches packet descriptor pointer from forward queue of ring for specified channel.
3. PKTDMA reads the packet descriptor to obtain the packet info, buffer pointer, buffer size, and next descriptor pointer.
4. PKTDMA fills the buffer with received data

If packet is not complete and the rx_chan_type field in the Rx Channel Configuration Register (<RCHAN[a]_RCFG> [19:16] RX_CHAN_TYPE) is set to 2 (Normal Rx Host Mode). The machine will proceed to step 5 and will repeat steps 5-6 until all packet data is received at which point the Rx engine proceeds to step 8. If the rx_desc_type is set to 3 (Single Buffer Packet Host Mode) the Rx engine will proceed directly to step 7 regardless of whether or not an end of packet terminator has been reached on the incoming data stream. The current packet will be completed following the sequence through step 12 and a new packet will be started back at step 1.

5. PKTDMA fetches next buffer descriptor using next descriptor pointer obtained from previous descriptor.

6. PKTDMA fills the buffer with received data.

The PKTDMA performs the following operations when the entire packet has been received:

7. PKTDMA writes the packet descriptor to memory. This includes the following fields:

a. Descriptor Type (set to Host)

b. Packet Length indicating the total number of bytes that are to be read from all of the buffers for this packet.

c. Source Tag

d. Destination Tag

e. Packet Type

f. Any protocol specific flags for the given packet type

g. Any protocol specific words that are required for the given packet type

h. Extended Packet Information (optional)

8. PKTDMA increments the occupancy of the reverse queue for the channel / flow.

9. Once the occupancy is incremented it will send an up event to the Interrupt Aggregator

Copyright © 2026 Texas Instruments Incorporated

10. The Interrupt Aggregator upon receiving the up event will set the appropriate bit in the Interrupt Status Register (VINT[a]_STATUS_SET) indicated by the interrupt mapping table which will cause an interrupt to be asserted to the Host

11.The Host will pop the packet pointer from the Rx Queue by reading the ring contents and then pushing a decrement to the reverse occupancy for the ring.

12. If the pop causes the queue to become empty, the PKTDMA will send a down event to the Interrupt Aggregator thus causing the associated bit to be cleared and also potentially clearing the interrupt to the Host.

11.1.4.13 BCDMA - Block Copy Channel Setup

This section describes the setup procedure for channels within the BCDMA. After a reset or a previous teardown operation but before queuing packets to a channel the host must initialize the channel's Port DMA State. The host initializes the channel Port DMA State by initializing all of the channel flow entries and then writing to the Block Copy Channel Configuration Register A (BCHANRT[a]_TRT_CTL). The Host may choose to write the enable bit in the Block Copy Channel Configuration Register A (<BCHANRT[a]_TRT_CTL> [31] TX_ENABLE) at the same time or after it has written all of the channel parameters but note that every write to the Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

After a channel has been set up, packets can be added to the Queues for the channel to begin the copy operation.

For general purpose block copy channels, only the Block Copy DMA configuration, flow, and real-time registers will be present. For split mode channels, separate Tx and Rx configuration, flow, and real-time registers are provided. In all cases, configuration of all fields must be completed before a channel is enabled.

11.1.4.14 BCDMA - Block Copy Channel Pause

Setting the pause bit in the Channel N Control Register (<BCHANRT[a]_TRT_CTL> [29] TX_PAUSE]) will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potentially overflowing trigger events.

11.1.4.15 BCDMA - Block Copy Channel Teardown

This section describes the teardown procedure for generic block copy channels within the BCDMA. A channel teardown is initiated by the host by writing the teardown bit in the Channel N Real-time Control Register (<BCHANRT[a]_TRT_CTL> [30] TX_TEARDOWN). When the host initiates teardown, it can choose to perform either a graceful (no data loss) or forced teardown of the channel depending on the setting of the forced_teardown bit in the Channel N Real-time Control Register (<BCHANRT[a]_TRT_CTL> [28] TX_FORCED_TEARDOWN).

If the forced_teardown bit is clear, a normal teardown has been initiated. In this case, the BCDMA will do the following:

1. Stops performing any additional fetches for the channel.
2. Completes any packets normally for which a pointer/descriptor/data fetch has already been performed
3. Sets the tdown bit on the EOP data phase of the last packet to be sent (if any traffic was pending when teardown was initiated) or sends a zero byte packet with the tdown, sop, and eop bits asserted. This signals the destination thread that all of the data has been sent.
4. Clears the channel enable in the Channel N Global Configuration Register (<BCHANRT[a]_TRT_CTL> [31] TX_ENABLE).
5. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
6. Sets bit 31 of the BC Flow Reverse Ring Occupancy register to indicate that a teardown has completed.
7. If the current BC reverse ring occupancy is 0, issues an BC reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.
8. If the current BC Reverse ring occupancy is not 0 then a write to the doorbell with a negative value of the current occupancy will clear any remaining work but the forward ring pointer will remain the same. If the ring pointer needs to be reset then perform a write to any of the ring configuration registers.

Copyright © 2026 Texas Instruments Incorporated

9. If the forced_teardown bit is set, a destructive teardown has been initiated. In this case, the BCDMA will do the following:
   a. Suspends any triggers to allow the channel to operate even if the trigger source is no longer functioning.
   b. Completes any packets which have been fetched with or without transferring any data or generating events (the implementation has the option of doing whichever is simplest for that implementation).
   c. Clears the channel enable in the Channel Global Configuration Register.
   d. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
   e. Sets bit 31 of the BC Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
   f. If the current BC reverse ring occupancy is 0, issues an BC reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.

   The host may issue a teardown on any channel at any time, regardless of whether the channel is actively transferring data or not.

   The Host determines that a teardown using either of the following methods:

   a. Periodically polling the teardown and enable bits for the channel
   b. Waiting for an interrupt and observing the teardown complete bit is set in the channel's default flow (first flow in channel) reverse occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) for the channel.

   Note that the teardown message will actually be written to the next available location in the ring following the last completed packet.

11.1.4.16 BCDMA - Block Copy Operation (TR Packet)

Packet transmission in TR Packet mode involves the following steps:

1. The Host allocates and populates a type 15 TR packet descriptor. The host will initialize the following fields within the packet descriptor:
   a. Descriptor Type (set to TR)
   b. Reload Enable to 1 if looping is required, otherwise 0
   c. Reload Index to an appropriate offset if Reload Enable set, otherwise 0
   d. Last Entry to TR count minus 1
   e. TR Nominal Element Size to a value that is as large as required for any given TR in the buffer
   f. A set of one or more valid Transfer Request records whose quantity matches the last index specified previously.
2. The Host queues the packet onto one of the Block Copy Queues for the desired BCDMA channel. Channels may provide more than one Queue (1 queue per provided flow) and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/ scheduler implementation.
3. The BCDMA internally provides a level sensitive status signal for the queue which indicates if any work is currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
4. The DMA controller is eventually brought into context for the corresponding read channel and begins to process the packet.
5. The DMA controller reads the packet descriptor pointer from the ring in memory.
6. The DMA controller reads the packet descriptor from memory
7. The DMA controller sequentially empties the data region in the descriptor by reading the contents in one or more nominal TR sized block data moves. As a TR is read from the descriptor, it is stored in the internal state of the DMA channel until it is completed.

Copyright © 2026 Texas Instruments Incorporated

8. All of the data transfers specified in a TR will be completed as a series of reads followed later on by a set of corresponding writes (which may or may not be the same size based on the parameters in the block copy TR).
9. The BCDMA will wait until write status returns for the final write operation that was initiated for each TR. When the final write status is returned within each TR, the BCDMA will write the response into the TR buffer in the Transfer Response records array. Each response is written into an array index which directly matches the index of the request record to which is corresponds.
10. When all Transfer Requests in the packet have been processed and all Transfer Responses have been written back and confirmed to have landed in memory, the BCDMA will increment the reverse occupancy for the channel/flow.
11. After the occupancy is incremented, the BCDMA will indicate the status of the Tx Completion Queue by sending an up event.
12. The Interrupt Aggregator receives the up event and sets the corresponding bit in the interrupt status register (VINT[a]_STATUS_SET) as programmed in the interrupt mapping registers. This in turn causes an interrupt to the Host to be generated.
13. The Host responds to the status change from the BCDMA (via the Interrupt Aggregator) and performs garbage collection as necessary for the packet.
14. During garbage collection the Host will write to the reverse queue Doorbell register (RINGRT[a]_RT_RDB) for the channel/flow to acknowledge the popping of those completed packets. The doorbell writes eventually cause the queue to become empty.
15. The BCDMA will send a down event to the Interrupt Aggregator which will clear the corresponding bit in the Interrupt Status Register (VINT[a]_STATUS) and potentially de-assert the interrupt line

11.1.4.17 BCDMA - Block Copy Error/Exception Handling

There are 3 specific errors which may be encountered during generic block copy operation and which are trapped by the BCDMA. The following table lists the errors and how the BCDMA will process them:

Table 11-58. BCDMA Generic Block Copy Mode Error / Exception Handling

Condition	Pauses Channel	Error Flag Set in Channel	Data Transfer	TR Flush	Outputs Transfer Events
TR null icnt0	Selectable(1)	Yes	No	Just that TR	No
Unsupported TR Type	Selectable(1)	Yes	No	Just that TR	No
Bus Errors	Selectable(1)	Yes	Yes	No	Yes

(1) If pause_on_error bit is set in channel configuration

11.1.4.17.1 Null Icnt0 Error

The icnt0 parameter in a TR is never allowed to be 0 as this would cause zero byte count transactions to be issued on the bus and is unproductive wrt completion of any useful work. The Block Copy engine in the BCDMA will trap any TR for which the icnt0 is zero and will immediately return a transfer response with the TR icnt0 error conditions set without performing any data transfer and without outputting any events (including end of TR event).

11.1.4.17.2 Unsupported TR Type

If a TR is provided to a channel which is of a type which is not supported by the channel, the Block Copy engine in the BCDMA will trap that TR and immediately return a transfer response with the Unsupported TR error conditions set without performing any data transfer and without outputting any events (including end of TR event).

Copyright © 2026 Texas Instruments Incorporated

11.1.4.17.3 Bus Errors

If a bus error is encountered during transmit the BCDMA will log the error by asserting the tx_error bit for the channel (and optionally sending an error event if enabled) but the DMA will continue to attempt to complete the transfer using the returned (and potentially incorrect) data.

11.1.4.18 BCDMA - Split Transmit Channel Setup

This section describes the setup procedure for split Tx channels within the BCDMA. After a reset or a previous teardown operation but before queuing packets to a channel the host must initialize the channel's Tx Port DMA State. The host initializes the channel Tx Port DMA State by configuring all of the Tx flow entries for the channel and then writing to the Tx Channel Configuration Register A (TCHANRT[a]_TRT_CTL). The Host may choose to write the enable bit in the Tx Channel Configuration Register A (<TCHANRT[a]_TRT_CTL> [31] TX_ENABLE) at the same time or after it has written all of the channel parameters but note that every write to the Tx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

After a Transmit channel has been set up, packets can be added to the Queues for the channel to begin the transmit operation. The following sections describe how the transmit operations are performed for the various descriptor types.

11.1.4.19 BCDMA - Split Transmit Operation Pause

Setting the pause bit in the Channel N Control Register (<TCHANRT[a]_TRT_CTL> [29] TX_PAUSE) will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potentially overflowing trigger events.

11.1.4.20 BCDMA - Split Transmit Channel Teardown

This section describes the teardown procedure for split mode Tx channels within the BCDMA. A Tx channel teardown is initiated by the host by writing the tx_teardown bit in the Tx Channel N Real-time Control Register (<TCHANRT[a]_TRT_CTL> [30] TX_TEARDOWN). When the host initiates teardown, it can choose to perform either a graceful (no data loss) or forced teardown of the channel depending on the setting of the tx_forced_teardown bit in the Tx Channel N Real-time Control Register (<TCHANRT[a]_TRT_CTL> [28] TX_FORCED_TEARDOWN) .

If the tx_forced_teardown bit is clear, a normal teardown has been initiated. In this case, the DMA will do the following:

1. Stops performing any additional descriptor fetches for the channel.
2. Completes any packets normally for which a pointer/descriptor/TR fetch has already been performed
3. Sets the tdown bit on the EOP data phase of the last packet to be sent (if any traffic was pending when teardown was initiated) or sends a zero byte packet with the tdown, sop, and eop bits asserted. This signals the destination thread that all of the data has been sent.
4. Clears the channel enable in the Tx Channel N Global Configuration Register (<TCHANRT[a]_TRT_CTL> [31] TX_ENABLE).
5. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
6. Sets bit 31 of the Tx Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
7. If the current Tx reverse ring occupancy is 0, issues a Tx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.
8. If the current TX Reverse ring occupancy is not 0 then a write to the doorbell with a negative value of the current occupancy will clear any remaining work but the forward ring pointer will remain the same. If the ring pointer needs to be reset then perform a write to any of the ring configuration registers.
9. If the tx_forced_teardown bit is set (only valid on BCDMA), a destructive teardown has been initiated. In this case, the BCDMA will do the following:

1. Suspends any triggers to allow the channel to operate even if the trigger source is no longer functioning.
2. Completes any packets which have been prefetched with or without transferring any data or generating events (the implementation has the option of doing whichever is simplest for that implementation).

Copyright © 2026 Texas Instruments Incorporated

3. Clears the channel enable in the Tx Channel Global Configuration Register (<TCHANRT[a]_TRT_CTL> [31] TX_ENABLE).
4. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
5. Sets bit 31 of the Tx Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
6. If the current Tx reverse ring occupancy is 0, issues a Tx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.

The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not.

The Host determines that a teardown using either of the following methods:

1. Periodically polling the teardown and enable bits for the channel
2. Waiting for an interrupt and observing the teardown complete bit is set in the channel's default flow (first flow in channel) reverse occupancy register (RINGRT[a]_RT_ROCC) for the channel.

11.1.4.21 BCDMA - Split Transmit Operation (TR Packet)

Packet transmission in TR Packet mode involves the following steps:

1. The Host allocates and populates a TR packet descriptor. The host will initialize the following fields within the packet descriptor:
    a. Descriptor Type (set to TR)
    b. Reload Enable to 1 if looping is required, otherwise 0
    c. Reload Index to an appropriate offset if Reload Enable set, otherwise 0
    d. Last Entry to TR count minus 1
    e. TR Nominal Element Size to a value that is as large as required for any given TR in the buffer
    f. A set of one or more valid Transfer Request records whose quantity matches the last index specified previously.
2. The Host queues the packet onto one of the Transmit Queues for the desired BCDMA channel. Channels may provide more than one Queue (1 queue per provided flow) and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller/ scheduler implementation.
3. The BCDMA internally provides a level sensitive status signal for the queue which indicates if any work is currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
4. The DMA controller is eventually brought into context for the corresponding read channel and begins to process the packet.
5. The DMA controller reads the packet descriptor pointer from the ring in memory.
6. The DMA controller reads the packet descriptor from memory
7. The DMA controller sequentially empties the data region in the descriptor by reading the contents in one or more nominal TR sized block data moves. As a TR is read from the descriptor, it is stored in the internal state of the DMA channel until it is completed.
8. All of the data transfers specified in a TR will be completed as a series of reads.
9. The BCDMA will wait until data returns for each read operation that has been initiated. When the final read operation is returned within each TR, the BCDMA will write the response into the TR buffer in the Transfer Response records array. Each response is written into an array index which directly matches the index of the request record to which is corresponds.
10. When all Transfer Requests in the packet have been processed and all Transfer Responses have been written back and confirmed to have landed in memory, the BCDMA will increment the reverse occupancy for the channel / flow.
11. After the occupancy is incremented, the BCDMA will indicate the status of the Tx Completion Queue by sending an up event.

Copyright © 2026 Texas Instruments Incorporated

12. The Interrupt Aggregator receives the up event and sets the corresponding bit in the interrupt status register (VINT[a]_ENABLE_SET) as programmed in the interrupt mapping registers. This in turn causes an interrupt to the Host to be generated.
13. The Host responds to the status change from the BCDMA (via the Interrupt Aggregator) and performs garbage collection as necessary for the packet.
14. During garbage collection the Host will write to the reverse queue Doorbell register (RINGRT[a]_RT_RDB) for the channel/flow to acknowledge the popping of those completed packets. The doorbell writes eventually cause the queue to become empty.
15. The BCDMA will send a down event to the Interrupt Aggregator which will clear the corresponding bit in the Interrupt Status Register (VINT[a]_STATUS) and potentially de-assert the interrupt line.

11.1.4.22 BCDMA - Split Transmit Error / Exception Handling

There are 3 specific errors which may be encountered during Tx split operation and which are trapped by the BCDMA. The following table lists the errors and how the BCDMA will process them:

Table 11-59. BCDMA Tx Error / Exception Handling

Condition	Pauses Channel	Error Flag Set in Channel	Data Transfer	TR Flush	Outputs Transfer Events
TR null icnt0	Selectable(1)	Yes	No	Just that TR	No
Unsupported TR Type	Selectable(1)	Yes	No	Just that TR	No
Bus Errors	Selectable(1)	Yes	Yes	No	Yes

(1) If pause_on_error bit is set in channel configuration

11.1.4.22.1 Null Icnt0 Error

The icnt0 parameter in a TR is never allowed to be 0 as this would cause zero byte count transactions to be issued on the bus and is unproductive wrt completion of any useful work. The Block Copy engine in the BCDMA will trap any TR for which the icnt0 is zero and will immediately return a transfer response with the TR icnt0 error conditions set without performing any data transfer and without outputting any events (including end of TR event).

11.1.4.22.2 Unsupported TR Type

If a TR is provided to a channel which is of a type which is not supported by the channel, the Block Copy engine in the BCDMA will trap that TR and immediately return a transfer response with the Unsupported TR error conditions set without performing any data transfer and without outputting any events (including end of TR event).

11.1.4.22.3 Bus Errors

If a bus error is encountered during transmit the BCDMA will log the error by asserting the tx_error bit for the channel (and optionally sending an error event if enabled) but the DMA will continue to attempt to complete the transfer using the returned (and potentially incorrect) data.

11.1.4.23 BCDMA - Split Receive Channel Setup

After a reset or a previous teardown operation but before receiving packets on a channel the host must initialize the channel's Rx Port DMA State. The host initializes the channel Rx Port DMA State by writing to the Rx Channel Configuration Registers. The Host may choose to write the enable bit in the Rx Global Channel Configuration Register (<RCHANRT[a]_RRT_CTL > [31] RX_ENABLE) at the same time or after it has written all of the channel parameters but note that every write to the Rx Channel Configuration Registers will overwrite the channel state for all bytes that are enabled for the write transaction.

Copyright © 2026 Texas Instruments Incorporated

11.1.4.24 BCDMA - Split Receive Channel Pause

Setting the pause bit in the Channel N Control Register (<RCHANRT[a]_RRT_CTL > [29] RX_PAUSE) will suspend the channel from arbitration resulting in a halting of the flow of data. Clearing this bit will cause the channel to be added back into the arbitration list. Pausing a channel has no other destructive side effects (other than potentially overflowing trigger events.

11.1.4.25 BCDMA - Split Receive Channel Teardown

An Rx channel teardown is intended to be initiated at the data source (the source thread). Initiation is commanded by the host by writing the teardown bit in the PSI-L source thread register 0x408 . This will cause the source of the data to gracefully terminate any packet that is in flight and send a PSI-L data phase with the tdown signal asserted. When the BCDMA receives a data phase with the tdown attribute asserted it will immediately set the internal rx_teardown bit in the Rx Channel N Real-time Control Register (<RCHANRT[a]_RRT_CTL > [30] RX_TEARDOWN).

Upon seeing the rx_teardown bit asserted the port can perform either a graceful (no data loss) or forced teardown of the channel depending on the setting of the rx_forced_teardown bit in the Rx Channel N Real-time Control Register (<RCHANRT[a]_RRT_CTL > [28] RX_FORCED_TEARDOWN).

If the rx_forced_teardown bit is clear, a normal teardown has been initiated. In this case, the BCDMA will do the following:

1. Stops performing any additional prefetches for the channel (TR mode channels)
2. Completes any packets normally which may be pending in the ingress port buffers (what the port considers as pending packets is application specific).
3. Clears the channel enable in the Rx Channel Global Configuration Register (<RCHANRT[a]_RRT_CTL > [31] RX_ENABLE).
4. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
5. Sets bit 31 of the Rx Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
6. If the current Rx reverse ring occupancy is 0, issues an Rx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.
7. If the current RX Reverse ring occupancy is not 0 then a write to the doorbell with a negative value of the current occupancy will clear any remaining work but the forward ring pointer will remain the same. If the ring pointer needs to be reset then perform a write to any of the ring configuration registers.
8. If the rx_forced_teardown bit is set, a destructive teardown has been initiated. In this case, the BCDMA will do the following:
    a. Suspends any triggers to allow the channel to operate even if the trigger source is no longer functioning.
    b. Completes any packets which may be pending in the ingress port buffers with or without transferring any data or generating events (the implementation has the option of doing whichever is simplest for that implementation) . Packets will be drained from the Rx FIFO either using ‘null’ write transfers or real transfers and the Packet Descriptors will be returned with an accurate accounting of actual bytes transferred.
    c. Clears the channel enable in the Rx Channel Global Configuration Register (<RCHANRT[a]_RRT_CTL > [31] RX_ENABLE).
    d. Resets the channel state (including scoreboards, FIFOs, counters, statistics, etc.) to their after reset values.
    e. Sets bit 31 of the Rx Flow Reverse Ring Occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) to indicate that a teardown has completed.
    f. If the current Rx reverse ring occupancy is 0, issues an Rx reverse ring completion up event which is then routed to an IA and can be further routed to any event consumer.

The host may issue a teardown on any channel at any time, regardless of whether the channel is actively receiving a packet or not. The normally intended method of issuing a teardown on a packet mode channel is to

Copyright © 2026 Texas Instruments Incorporated

initiate the teardown at the remote PSI-L data source and allow it to flow into the BCDMA. If that is not possible, the host may write the rx_teardown bit directly to a 1.

The Host determines that a teardown using either of the following methods:

1. Periodically polling the teardown and enable bits for the channel
2. Waiting for an interrupt and observing the teardown complete bit is set in the channel's default flow (first flow in channel) reverse occupancy register (<RINGRT[a]_RT_ROCC> [31] TDOWN_COMPLETE) for the channel.

11.1.4.26 BCDMA - Split Receive Operation (TR Packet)

Packet reception in TR Packet mode involves the following steps:

1. The Host allocates and populates a TR packet descriptor. The host will initialize the following fields within the packet descriptor:
    a. Descriptor Type (set to TR)
    b. Reload Enable to 1 if looping is required, otherwise 0
    c. Reload Index to an appropriate offset if Reload Enable set, otherwise 0
    d. Last Entry to TR count minus 1
    e. TR Nominal Element Size to a value that is as large as required for any given TR in the buffer
    f. A set of one or more valid Transfer Request records whose quantity matches the last index specified previously.
2. The Host queues the packet onto one of the Rx TR Queues for the desired BCDMA channel. Channels may provide more than one Rx TR Queue (one per flow) and may provide a particular prioritization policy between the queues. This behavior is application specific and is controlled by the DMA controller / scheduler implementation.
3. The Ring Accelerator provides a level sensitive status signal for the queue which indicates if any packets are currently pending. This level sensitive status line is sent to the hardware block which is responsible for scheduling DMA operations.
4. The DMA controller is eventually brought into context for the corresponding channel and begins to process the packet.
5. The DMA controller reads the packet descriptor pointer from the ring in memory.
6. The DMA controller reads the packet descriptor from memory
7. The DMA controller empties the data region in the descriptor by reading the contents in one or more nominal TR sized block data moves.
8. All of the data transfers specified in a TR will be completed as a series of writes and a Transfer Response will be returned indicating the completion and status of the transfer.
9. The BCDMA will wait until Transfer Responses have been returned for each Transfer Request that it issued. As each Transfer Response is returned, the BCDMA will write the response into the TR buffer in the Transfer Response records array. Each response is written into an array index which directly matches the index of the request record to which is corresponds.
10. When all Transfer Requests in the packet have been processed and all Transfer Responses have been written back and confirmed to have landed in memory, the DMSS will write the pointer to the packet descriptor to the queue specified in the return queue number fields of the packet descriptor.
11. After the Packet Descriptor pointer has been written, the Ring Accelerator indicates the status of the Rx Completion Queues to other ports / processors / prefetcher blocks using events sent to the Interrupt Aggregator. These events are then converted into standard interrupts.

11.1.4.27 BCDMA - Split Receive Error / Exception Handling

Several types of errors/exceptions may be encountered during reception of split Rx data. The following table outlines the various errors and exceptions that can occur:

Copyright © 2026 Texas Instruments Incorporated

Table 11-60. Rx Error / Exception Handling

Condition	Channel Type	Severity	Pauses Channel	Error Flag Set	Data Transfer	Data Flush	TR Flush	Event Output
Descriptor Starvation	All Packet	Exception	No	No	Option	Option	-	No
Protocol Errors	All Packet	Info	No	No	Normal	No	-	No
Drop	Host – normal	Exception	No	No	Partial(1)	Until EOP	-	No
EOP Asserted Late (Long Packet)	BCDMA – single ended	Exception	No	No	Partial	Excess	No	Yes
TR null icnt0	All BCDMA	Error	Selectable(1)	Yes	No	No	Just that TR	No
Unsupported TR Type	All BCDMA	Error	Selectable(1)	Yes	No	No	Just that TR	No

(1) Data will not be transferred after drop is detected. All data after that point is guaranteed to be flushed and some data prior to the exact data phase on PSI-L where drop was asserted may also be flushed since data is constantly accumulated so it can be transferred in chunks.

11.1.4.27.1 PKTDMA Exception Conditions

There are a few cases which can occur when processing incoming packets which are not considered errors but which will require handling in order to avoid locking up the PKTDMA engines.

Packet mode channel exceptions include:

- Descriptor Starvation
- Protocol Errors
- Dropped Packets
- Long Packets

11.1.4.27.1.1 Descriptor Starvation

Descriptor starvation occurs when the Port needs to fetch a free descriptor from the RX ring and none are available. When the port detects that descriptor starvation has occurred it will react based on the value of the rx_error_handling bit which was programmed into the Rx Flow N Configuration Register A (<RFLOW[a]_RFA> [28] RX_ERROR_HANDLING).

If the rx_error_handling bit is cleared:

- The Port will increment an internal per-channel starvation counter and will then flush the packet.

If the rx_error_handling bit is set:

- The Port will assert a RX descriptor starvation event (which may cause an interrupt to the Host) and will then wait for the doorbell for the selected flow to be written with a positive value. Note that the intention is that the port will wait for an entry to be added to the ring. It is assumed that when in this mode, if data loss is not desired that the host will guarantee that a chain of buffers will be provided that can receive the entire packet.

11.1.4.27.1.2 Protocol Errors

Protocol errors occur when the port logic detects that the received packet did not pass protocol specific criteria that was checked during reception of the packet. Examples of protocol errors include packet length errors, CRC errors, or alignment errors.

When protocol errors are detected, the port may choose whether or not it will drop the packet. The mechanism that is used to determine which packets to drop and which to forward is application specific. If the port determines that it needs to forward the packet to the Host, it will set the Packet Error bit in the Descriptor

Copyright © 2026 Texas Instruments Incorporated

indicating that this is a packet which experienced a protocol related error. The type of protocol related error is generally indicated in either the Protocol Specific bits in the Host descriptor or in the Protocol Specific bytes region. The packet length information and certain portions of the Protocol Specific region may not be correct for packets which encounter errors.

11.1.4.27.1.3 Dropped Packets

Packets can be dropped within an Rx Port for any number for reasons which with the exception of the starvation case which was covered above are application specific.

When a Port determines that it needs to drop a packet after reception of the packet has begun, it must flush the remainder of the packet, increment the dropped packets statistic, and rewind any fetched descriptor/buffer chain so that the it can be use the same descriptor/buffer chain on the next packet.

11.1.4.27.1.4 Long Packet

For Rx channels which are not configured in single buffer mode, if the host provides a buffer chain whose total length is insufficient to receive the entire packet, whatever portion of the packet which has not been received will be flushed until an end of packet indicator is received on the PSI-L interface. This behavior will occur regardless of which error handling mode has been selected for the channel.

11.1.4.27.2 BCDMA Exception Conditions

Since the TR mode engine performs transfer sequencing using a count based mechanism it is susceptible to becoming out of synchronization if the source of the data and the destination for the data do not exactly match in their expectations for how much data will be transferred. The following sections outline these scenarios.

TR mode channel exceptions include:

- Reception of EOL delimiter
- Short Packets
- Long Packets
- Descriptor Starvation

11.1.4.27.2.1 Reception of EOL Delimiter

If an end of line delimiter is received by the Rx (write) side of the BCDMA the EOL_ADV field in the TR application specific flags field is used to advance the appropriate indices of the TR to the next level. Each EOL that is received will cause the TR to advance to the next specified loop iteration breaking as many intermediate loops as are required.

11.1.4.27.2.2 EOP Asserted Prematurely (Short Packet)

Split TR mode channels can experience an incoming packet whose length is shorter than what was expected based on one or more TRs which form the control description for the expected packet. If an EOP is received for a TR and the TR has not been completely executed, the port is required to receive as much data as is actually provided in the incoming data placing it in accordance with the instructions in the TR and the to continue executing any remaining TR(s) such that the required events are produced to keep downstream consumers in sync. In this case, no data will be transferred to the buffers described by the TR beyond what was provided in the incoming packet.

11.1.4.27.2.3 EOP Asserted Late (Long Packets)

Split TR mode channels can experience an incoming packet whose length is longer than what was expected based on one or more TRs which form the control description for the expected packet. If a TR completes and is marked as EOP but the incoming packet is not finished, then the port must throw away any additional received data until the incoming packet reaches an EOP condition. At this point, reception will start with a new TR and a new incoming packet.

11.1.4.27.2.4 Descriptor Starvation

Descriptor starvation occurs when the Port receives enough data for a burst but the ring for the channel is currently empty. This causes push back on the receive port and can cause lost data.

Copyright © 2026 Texas Instruments Incorporated

The Port will assert a starvation bit in the RX Receive Channel Status register (RCHANRT[a]_RRT_STATUS[1-0]). When the doorbell register (RINGRT[a]_RT_DB) is written to populate the ring the starvation bit will clear. Note that the intention is that the port will wait for an entry to be added to the ring. It is assumed that if data loss is not desired that the host will guarantee that a descriptor is present.

Copyright © 2026 Texas Instruments Incorporated

11.2 Data Movement Subsystem (DMSS)

This chapter describes the Data Movement Subsystem (DMSS) module in the device.

Copyright © 2026 Texas Instruments Incorporated

11.2.1 Data Movement Subsystem (DMSS)

This chapter describes the features and functions of the Data Movement Subsystem (DMSS) hardware modules in the device.

11.2.1.1 DMSS Overview

The DMSS module on provides data movement (DMA) and bridges between the CBA switched interconnect and the packet streaming fabric (network on chip) on the device.

The Data Movement Subsystem (DMSS) consists of DMA/Queue Management components and Peripherals:

- Packet DMA
- Block Copy DMA
- Ring Accelerator
- Packet Streaming Interface (PSILSS)
- Infrastructure components such as CBASS, secure proxy, and an interrupt aggregator

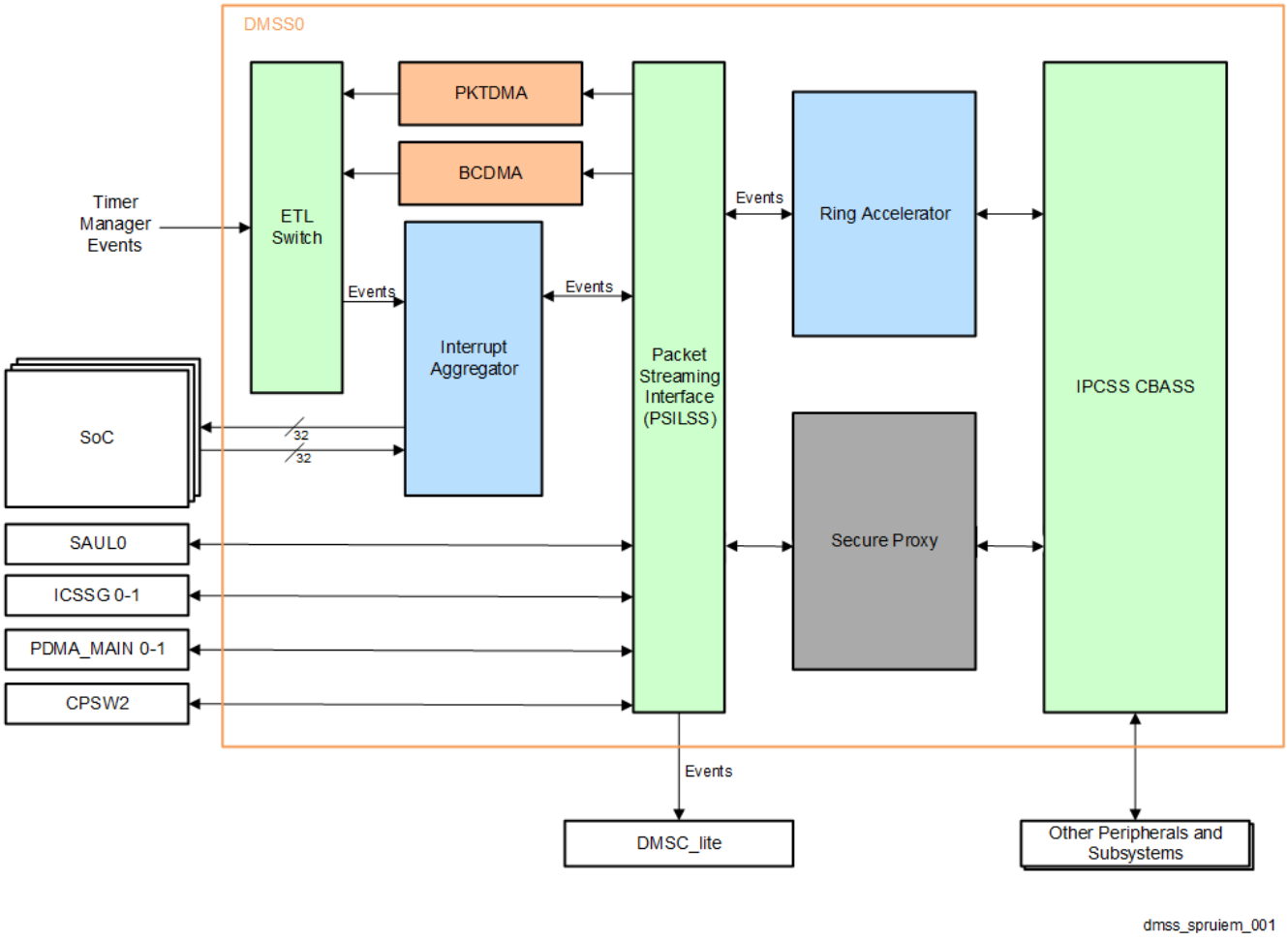


Figure 11-4. DMSS Top-Level Block Diagram

Copyright © 2026 Texas Instruments Incorporated

Table 11-61. DMSS Allocation Within Device Domain

Module Instance	Domain	
	MCU	MAIN
DMSS0	-	✓(DMSS)

Note

Some features may not be available. See *Module Integration* for more information.

11.2.1.2 DMSS Functional Description

See Section Chapter 11, DMSS Architecture, for the TI Data Movement Architecture (DMA) specification.

See the following sections for the respective module description:

- PKTDMA - Section 11.1.1.4
- BCDMA - Section 11.1.1.5
- Ring Accelerator:
  - See Section 11.1.1.1 for a summary overview of the Ring accelerator.
  - SeeSection 11.2.2 for a full chapter breakdown of the Ring accelerator.
  - See Section 11.2.2.1 for a block diagram overview.
  - See Section 11.2.2.1.1 for a list of features.
  - See Section 11.2.2.2 for a functional description of the Ring Accelerator.
- Infrastructure Components:
  - CBASS - Chapter 3
  - Secure Proxy:
    - See Section 11.1.1.2 for a summary overview of the Secure Proxy.
    - See Section 11.2.3 for a full chapter breakdown of the Secure Proxy.
    - SeeSection 11.2.3.1.1 for a list of features.
    - See Section 11.2.3.2 for a functional description of the Secure Proxy.
  - Interrupt Aggregator
    - See Section 11.1.1.3 for a summary overview of the Interrupt Aggregator.
    - See Section 11.2.4 for a full chapter breakdown of the Interrupt Aggregator.
    - SeeSection 11.2.4.1.1 for a list of features.
    - See Section 11.2.4.2 for a functional description of the Interrupt Aggregator.

11.2.1.3 DMSS Interrupt Configuration

This section describes the actions needed to configure interrupts within the DMSS.

Table 11-62. DMSS Parameters

Parameter	Value DMSS	Description
RA_RING_CNT	20	Number of total rings supported (specific to each ring accelerator)
IA_SEVI	1536	Interrupt Aggregator Source Event Input (SEVI) count (specific to each interrupt aggregator)
IS_VINTR	184	Interrupt Aggregator Virtual Interrupt (VINTR) count (specific to each interrupt aggregator)
EO	See *Global Event Map* in Module Integration	Event offset

DMSS Software Variables lists software-configurable variables used in the examples and descriptions.

Copyright © 2026 Texas Instruments Incorporated

Table 11-63. DMSS Software Variables

Variable	Valid Range	Description
R#	0 to RA_RING_CNT-1	Ring number
E#	0 to (destination module specific)	Event number
GE#	= EO + E#	Global event number
SB#	0 to IA_SEVI-1	Interrupt aggregator status bit number
VI#	0 to IA_VINTR-1	Virtual interrupt number

11.2.1.3.1 DMSS Event and Interrupt Flow

DMSS Interrupt Flow illustrates the event and interrupt flow within the DMSS and output interrupts to a CPU.

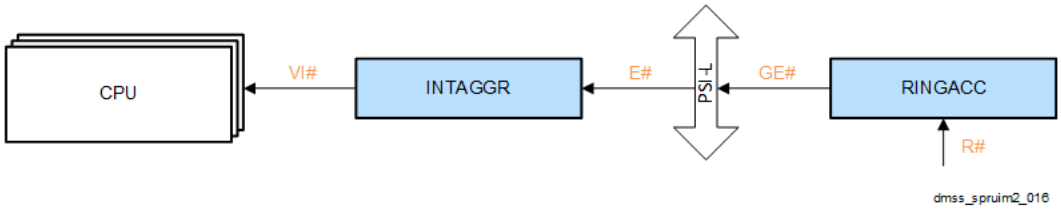


Figure 11-5. DMSS Interrupt Flow

11.2.1.3.1.1 DMSS Interrupt Description

This section describes the interrupt-related information that flows within DMSS and how this information is configured or generated.

1. **Ring Number (R#):**
   - Data is read from and written to a specific Ring Accelerator ring using direct ring-mode, or secure proxy access
   - The Ring Accelerator has specific ring number ranges for specific purposes. The R# must match the intended purpose of the ring (see Table 10-578, RINGACC Ring Mapping)
2. **Global Event and Event Numbers (GE# and E#):**
   - GE# = EO + E#
   - EO determines the destination module for the event. The PSILSS will route the GE# to the destination module per the DMSS event mapping, removing EO in the process. The destination module sees only E#. See *Global Event Map* in *Module Integration* for details.
   - For the INTAGGR module, E# ranges from 0 to (IA_SEVI - 1)
3. **Virtual Interrupt (VI#)**
   - The INTAGGR maps E# (via software configuration) to any SB#
   - ranges from 0 to (IA_SEVI - 1)
   - Each VI# (0 – (IA_VINTR-1)) is driven by a group of 64 contiguous SB# numbers (VI# driven by SB#'s VI#×64 – (VI#×64)+63)

11.2.1.3.1.2 DMSS Event Description

1. Interrupt aggregator includes three orthogonal functions:
   a. Event mapping block which converts local/umapped events to global events, and sends those events out.
   b. Aggregating global events and convert them into interrupt outputs.
   c. L2G block: This is a block to convert the event/interrupt from SoC level into global events. L2G block takes wired signal from SoC (interrupt signal or DMA signal) converted into local events.

Copyright © 2026 Texas Instruments Incorporated

2. L2G Block: This block converts the event/interrupt from SoC level into global events. L2G block takes wired signal from SoC (interrupt signal or DMA signal) converted into local events. Those local events are sent to ETL and sent to event mapping logic inside interrupt aggregator to convert them into global event. And then those global events are sent out again.
3. PSIL and ETL: PSIL is a streaming protocol which carries both data information and event information. ETL is the PSIL portion which only carries event. PSIL data is transported independently. When you see PSIL, it must include ETL.
4. Local events and unmapped events are in the same category. All those unmapped/local events are routed to the mapping block inside Interrupt aggregator to be converted into global events
5. Other than L2G block, Currently there are only three components generating unmapped events: BCDMA, PktDMA and timer manager. All those unmapped events are routed to the event mapping block inside interrupt aggregator first. All the other components produce global events.
6. After the local/unmapped events are mapped to global events in interrupt aggregator, those global events are sent out by interrupt aggregator and can be transported to different locations (for example, BCDMA to trigger BCDMA or pktDMA or going back to interrupt aggregator's aggregating global events into interrupts.

Copyright © 2026 Texas Instruments Incorporated

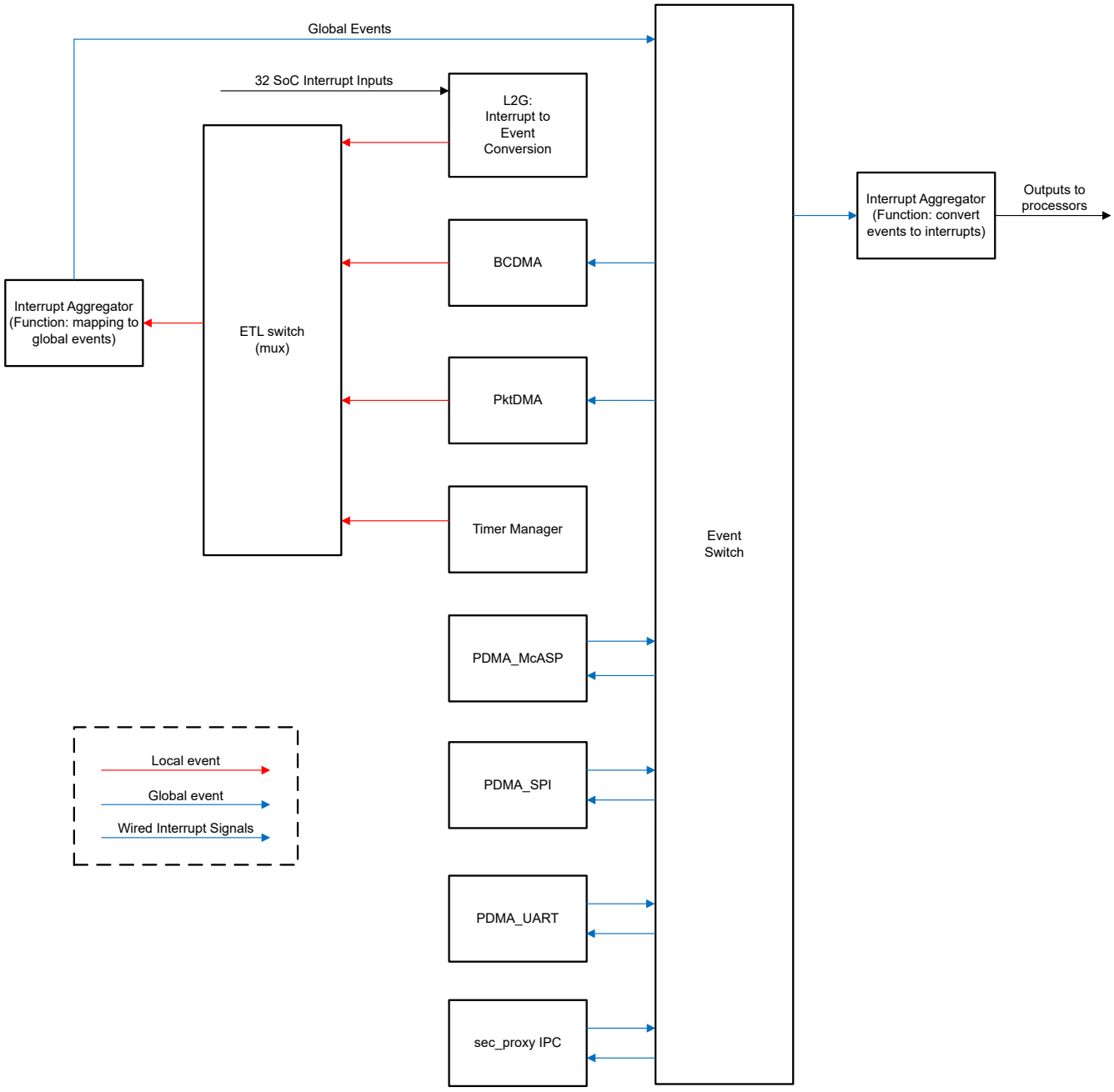


Figure 11-6. DMSS Event Flow

Copyright © 2026 Texas Instruments Incorporated

11.2.2 Ring Accelerator (RINGACC)

This chapter describes the RINGACC module, part of the DMSS.

Note

Not all of the RINGACC Functions specified in the document are supported in the device. Updates will be included in the next revision to reflect only supported modes of operation.

11.2.2.1 RINGACC Overview

The Ring Accelerator (RINGACC or RA) provides hardware acceleration to enable straightforward passing of work between a producer and a consumer.

Table 11-64. RINGACC Allocation Within Device Domain

Module Instance	Domain	
	MCU	MAIN
DMSS0_RINGACC0	-	✓(DMSS)

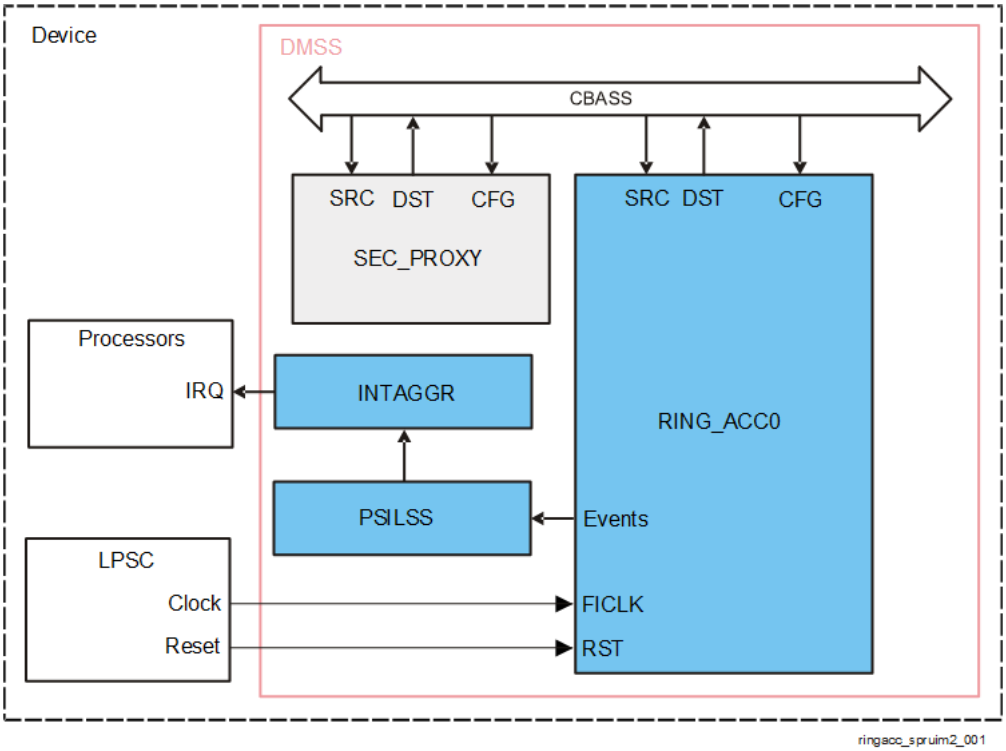


Figure 11-7. RINGACC Overview

11.2.2.1.1 RINGACC Features

- Ring Accelerator implementation
  - Supports up to 32 independent memory mapped ring structure
  - Supports various modes for each ring based on usage and compatibility
  - Provides single word deep shared incoming Transfer Response FIFO
  - Optionally supports dynamic clock gating
  - Provides bit wide source VBUSM read/write target interface for accesses from DMA controller entities
    - Provides 2 word deep command FIFO
    - Provides 2 word deep write data FIFO
    - Provides 2 word deep read data FIFO

Copyright © 2026 Texas Instruments Incorporated

- Provides 2 word deep write status FIFO
- Provides bit wide destination VBUSM read/write initiator interface for accesses to ring structures in memory
- Supports up to 16 outstanding writes
- Supports up to 16 outstanding reads
- Provides 2 word deep command FIFO
- Provides 2 word deep write data FIFO
- Provides 2 word deep read data FIFO
- Provides 2 word deep write status FIFO
- Source interface provides an array of 32x512-byte long address windows (four for each ring) which are packed into a single contiguous address range.
  - Read and write addresses which target a specific window are mangled to redirect the read or write transaction to an effective address calculated from the base address for the ring plus current ring offset.
  - Each read or write access presented on VBUSM target interface is modified and bridged onto the VBUSM initiator interface.

Note

Some features may not be available. See *Module Integration* for more information.

11.2.2.1.2 RINGACC Parameters

RINGACC Configuration Parameters shows the RINGACC configuration parameters in this SoC

Table 11-65. RINGACC Configuration Parameters

Module Instance	Parameters		
	Ring Count	Number of Monitors	Proxy Target Base
DMSS0_RINGACC0	32	0	0x4e000000

Copyright © 2026 Texas Instruments Incorporated

11.2.2.2 RINGACC Functional Description

The Ring Accelerator (RINGACC) converts constant-address read and write accesses to equivalent read or write accesses to a circular data structure in memory. The RINGACC eliminates the need for each DMA controller which needs to access ring elements from having to know the current state of the ring (base address, current offset). The DMA controller performs a read or write access to a specific address range (which maps to the source interface on the RINGACC) and the RINGACC replaces the address for the transaction with a new address which corresponds to the head or tail element of the ring (head for reads, tail for writes). Since the RINGACC maintains the state, multiple DMA controllers or channels are allowed to coherently share the same rings as applicable. The RINGACC serves a very similar function to the Queue Manager in the earlier SoCs. The RINGACC uses less memory and is able to place data which is destined towards software into cached memory directly.

11.2.2.2.1 Block Diagram

Ring Accelerator Block-Diagram shows ring accelerator's main internal components.

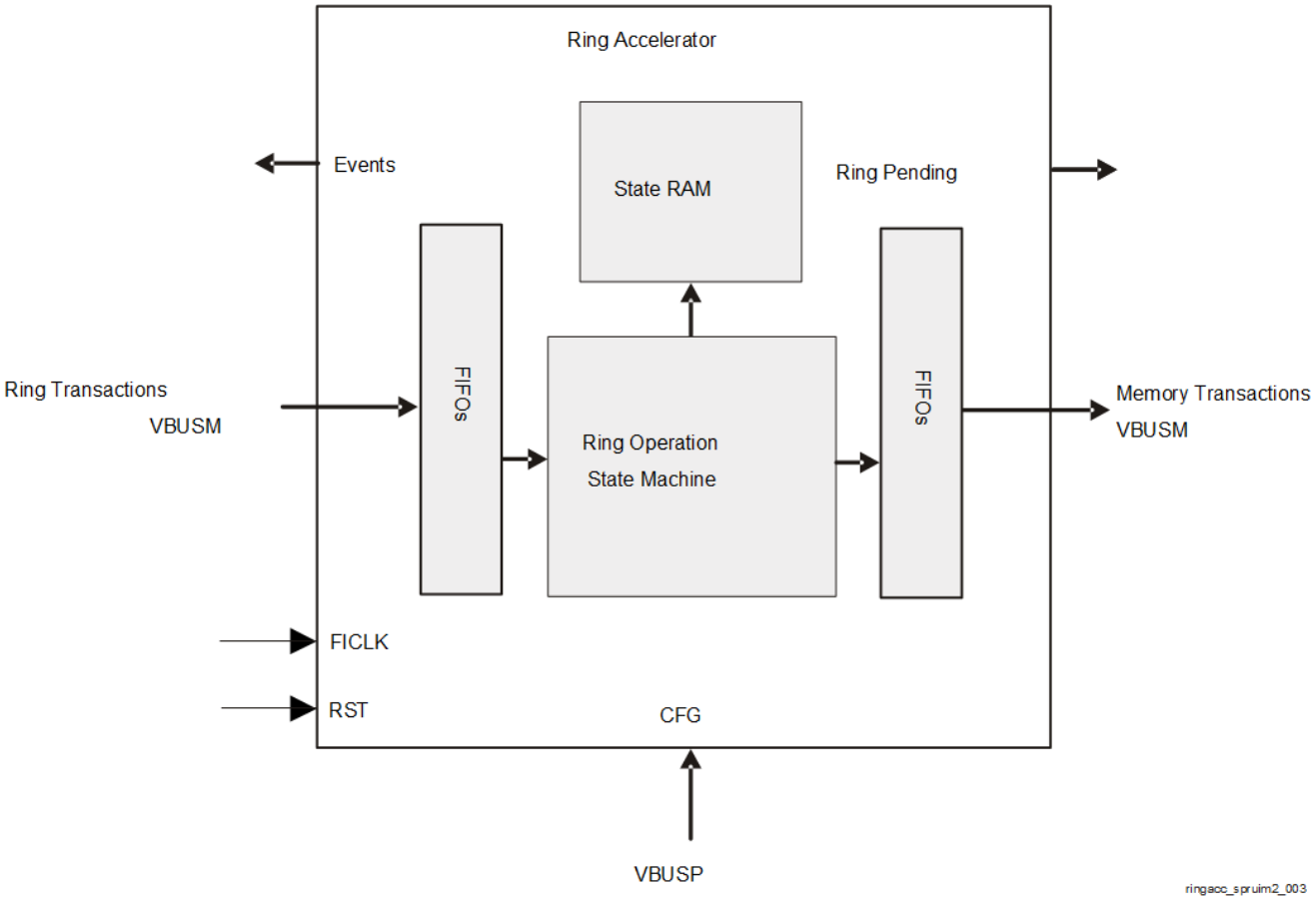


Figure 11-8. Ring Accelerator Block-Diagram

11.2.2.2.1.1 Configuration Registers

The Configuration Registers block is responsible for providing memory mapped registers for configuration of the RINGACC functions. The configuration registers are divided into:

- Static configuration fields , which are typically initialized and then left at specified values for long time periods (or until a reset occurs)
- Real-time (RT) registers , which are modified frequently during runtime. Configuration registers are listed and described in *RINGACC Registers*.

Copyright © 2026 Texas Instruments Incorporated

11.2.2.2.1.2 Source Command FIFO

The Source Command FIFO buffers VBUSM command phases which are accepted from the VBUSM source interface.

11.2.2.2.1.3 Source Write Data FIFO

The Source Write Data FIFO buffers VBUSM write data phases which are accepted from the VBUSM source interface.

11.2.2.2.1.4 Source Read Data FIFO

The Source Read Data FIFO buffers VBUSM read data phases which have been returned from the VBUSM destination interface (via the main state machine) and which are to be output on the VBUSM source interface.

11.2.2.2.1.5 Source Write Status FIFO

The Source Write Status FIFO buffers VBUSM write status data phases which have been returned from the VBUSM destination interface and which are to be output on the VBUSM source interface.

11.2.2.2.1.6 Main State Machine

The Main State Machine (MSM) block implements all of the control logic which is necessary to accept a command from the source interface, perform a lookup into the ring state RAM, determine the effective address for the destination interface transaction, modify the address (and byte count for reads) for the transaction, and place that modified transaction into the outgoing destination FIFOs. The MSM block is also responsible for updating the ring index and occupancy values and for producing output status to indicate changes to the ring state. The MSM also modifies the ring state whenever the read data or write status for a previous ring transaction return and forwards those responses back to the originating initiator.

11.2.2.2.1.7 Destination Command FIFO

The Destination Command FIFO stores VBUSM transaction commands which are output from the MSM as modified versions of transactions that were popped from the Source Command FIFO. The Destination command FIFO allocates a new command ID for each read or write command that is pushed to the FIFO. Read IDs are allocated from a scoreboard maintained in the Destination Read Data FIFO. Write IDs are allocated from a scoreboard maintained in the Destination Write Status FIFO. When a new command is allocated, the Destination Command FIFO also pushes the original cid and crouteid values from the source side transaction into a scoreboard in either the Destination Read Data or Destination Write Status FIFOs depending on the applicable transaction direction. The read interface of the Destination Command FIFO directly drives commands onto the destination VBUSM command interface.

11.2.2.2.1.8 Destination Write Data FIFO

The Destination Write Data FIFO stores write data phases for write commands that are passing through the RINGACC towards memory. The write interface of the Destination Write Data FIFO is directly connected to the read interface of the Source Write Data FIFO and the read interface directly drives write data onto the destination VBUSM write data interface.

11.2.2.2.1.9 Destination Read Data FIFO

The Destination Read Data FIFO provides a read command translation scoreboard and also stores read data phases that are passing through the RINGACC towards the originating DMA controller. The write interface of the Destination Read Data FIFO is directly connected to the destination VBUSM read data interface. As read data passes through the Destination Read Data FIFO, the *rid* and *rrouteid* are changed back to their original values using information which was stored in read command translation scoreboard.

11.2.2.2.1.10 Destination Write Status FIFO

The Destination Write Status FIFO provides a write command translation scoreboard and also stores write status phases that are passing through the RINGACC towards the originating DMA controller. The write interface of the Destination Writes Status FIFO is directly connected to the destination VBUSM write status interface. As write

Copyright © 2026 Texas Instruments Incorporated

status words pass through the Destination Write Status FIFO, the *sid* and *srouteid* are changed back to their original values using information which was stored in the write command translation scoreboard.

11.2.2.2.2 RINGACC Functional Operation

11.2.2.2.2.1 Queue Modes

Each ring or queue can be in one of four modes that determines how it must be accessed and the compatibility it has with hardware and software.

11.2.2.2.2.1.1 Ring Mode

Ring mode is used when software owns one side of the ring, and hardware or another software owns the other side. This mode allows the software that owns a side of the ring to control that side including accessing the memory directly rather than going through hardware, and software updates the hardware on any changes through the Doorbell registers. These doorbell accesses indicate when software has pushed or popped entries, allowing the entity on the other side of the ring to know when there are elements ready. The ring mode has limitations that the exposed side of the ring is completely under software control, and any atomicity needed must also be performed by software, which is why the exposed side of a ring is usually owned by a single thread of software. This limitation also means that any hardware configuration that would normally access the same ring for both pushes and pops cannot use ring mode as hardware cannot access the exposed side of the ring, such as free queues that are normally read by hardware but could also be pushed by hardware on an error.

When software is popping from the ring, it must guarantee that it never pops more elements through the Doorbell register (RINGRT[a]_RT_DB) than what is valid in the Ring Occupancy (RINGRT[a]_RT_OCC) register, which holds how many elements are ready for software consumption. There could be some elements which have data written to memory but have not received status back from memory and those are not considered ready to pop yet as the ring status has not fully updated. Therefore, just reading from memory to determine the number of elements to pop is not sufficient. If software attempts to pop more than the RINGRT[a]_RT_OCC register value, then the occupancies could go negative resulting in missed down events and spurious interrupts. An optimization for software is to read the RINGRT[a]_RT_OCC register less frequently and keep a local copy which guarantees the maximum number of allowed pops. When the software copy of RINGRT[a]_RT_OCC reaches 0, or periodically, the software can read the register to refresh the value. This must reduce the frequency of register reads instead of reading the register for every software pop.

11.2.2.2.2.1.2 Messaging Mode

Messaging mode requires that all accesses to the queue must go through RINGACC so that all accesses to the memory are controlled and ordered. RINGACC then controls the entire state of the queue, and software has no directly control, such as through doorbells and cannot access the storage memory directly. This is particularly useful when more than one software or hardware entity can be the producer and/or consumer at the same time. Software must access the data through bus accesses to the RINGACC. As with the earlier example, if there are free queues that need the hardware to both push and pop, then they must be configured as at least messaging mode (or a later mode) and not ring mode.

11.2.2.2.2.1.3 Credentials Mode

Credentials mode adds per element credentials storage to messaging mode. This allows multiple producers with their own credentials to have those credentials stored with the element. This allows the consumer to inherit the credentials of each element rather than a single set for the entire queue, such as for DMA TR credential inheritance. This mode requires each operation to use two elements of storage since the credentials are stored in the second element, so the element count must be scaled appropriately. The credentials field is located in the same location as all modes, the word just before the first word of the message.

11.2.2.2.2.1.4 Peek Support

All modes except ring mode allow the peek operations so that software can view the next element without actually popping it from the queue. This is done by reading from a different offset from the normal pop operation. All the same fields are read but the element is not actually popped off of the queue. This is useful for algorithms

Copyright © 2026 Texas Instruments Incorporated

that need to know about the overall queue and head element to make decisions but have not yet decided to pop from the queue.

11.2.2.2.2.1.5 Index Register Operation

In ring mode, the Index (RINGRT[a]_RT_INDX) register follows the software control of the ring through the doorbell registers, while the hardware index (RINGRT[a]_RT_HWINDX) register follows the hardware access of the ring through bus transactions. It works for a ring in either direction as the doorbell register (RINGRT[a]_RT_DB) update indicates whether elements were produced or consumed. But in the other queue modes, there is no simple manner to determine which index is for software or hardware since they both use bus transactions which the module cannot differentiate. So for these other queue modes, the index register is always the read index, and the hardware index register is the write index.

11.2.2.2.2.2 VBUSM Target Ring Operations

Each ring contains four memory spaces on the VBUSM target bus for access, each of which is 512 bytes long, with a total of 4 kB per ring. Each ring space starts immediately after the preceding ring. The allocation per ring is as in Table 11-66.

Table 11-66. Ring Memory Partition

Offset	Operation	Supported Ring Modes
0x0 - 0x1FF	Reads pop from head. Writes push to head.	Read supported in all modes, write supported in all modes except Ring mode
0x200 - 0x3FF	Writes push to tail. Reads pop from tail.	Write supported in all modes, read supported in all modes except Ring mode
0x400 - 0x5FF	Reads peek from head. Writes ignored.	All modes except for Ring mode.
0x600 - 0x7FF	Reads peek from tail. Writes ignored.	All modes except for Ring mode.
0x800 - 0xFFF	Reserved	Reserved

Within each 512-byte (0x200) space, the message data is right justified so that the last byte is always the 511-th byte. The element size of the ring defines which words are valid. The word just before the first valid message word is the credentials field. Access must not be made before the credentials register (CRED[a]_CRED).

11.2.2.2.2.3 VBUSM Initiator Interface Command ID Selection

The RINGACC keeps a transaction command scoreboard for reads and writes. Each scoreboard tracks which command indexes are currently outstanding from the RINGACC and which are free to use. When a read or write is requested by the Main State Machine, the scoreboard corresponding to the specified direction (read/write) is queried starting at index 0 and extending up to the maximum index for that scoreboard. The lowest index that is found and which is not currently allocated is selected and is directly used as the command ID (*cid*) for the outgoing transaction on the destination interface. 15 is the maximum index value for each scoreboard. Entries are freed for re-use in a write scoreboard when write status responses are received which account for all of the outstanding write data. Entries are freed for re-use in the read scoreboard only when all data has been returned for the read.

11.2.2.2.2.4 Ring Push Operation (VBUSM Write to Source Interface)

The following sequence will occur for each VBUSM write which is sent to the source interface:

1. RINGACC extracts ring number from incoming write transaction address
2. RINGACC looks up ring state using *ring number*
3. RINGACC calculates effective write address using *ring base + ring_index*
4. RINGACC re-evaluates pending bit for ring
5. RINGACC increments HW index and HW occupancy for ring
6. RINGACC allocates *cid* from scoreboard and places original *crouteid, cid, and ring number* into scoreboard
7. RINGACC pushes altered *caddress, cid* and unaltered remainder of command attributes to output FIFO (note *routeid* is not included)

Copyright © 2026 Texas Instruments Incorporated

At a later time when write status returns:

1. RINGACC recovers original *cid, crouteid,* and *ring number* from scoreboard
2. RINGACC increments SW index and occupancy for ring
3. RINGACC pushes restored *srouteid, sid*, and unaltered write status to output write status FIFO

11.2.2.2.2.5 Ring Pop Operation (VBUSM Read from Source Interface)

The following sequence will occur for each VBUSM read which is sent to the source interface:

1. RINGACC extracts ring number from incoming read transaction address
2. RINGACC looks up ring state using *ring number*
3. RINGACC calculates effective read address using *ring base* + *ring_index*
4. RINGACC increments HW index and decrements HW occupancy for ring
5. RINGACC re-evaluates pending bit for ring
6. RINGACC munges *cid* to add indicator in 4 MSBs to reconstitute *routeid* for returning read data
7. RINGACC pushes altered *caddress, cid* and unaltered remainder of command attributes to output FIFO (note routeid is not included)

At a later time when read data returns:

1. RINGACC recovers original *cid, crouteid*, and *ring number* from scoreboard
2. RINGACC increments SW index and decrements SW occupancy for ring
3. RINGACC pushes restored *rrouteid, rid,* and unaltered read data, status, and other control signals to output read FIFO

11.2.2.2.2.6 Host Doorbell Access

The following sequence will occur for each VBUSP write to the doorbell register (RINGRT[a]_RT_DB) for a ring:

1. RINGACC extracts ring number from config address
2. RINGACC looks up ring state using ring number
3. RINGACC adds doorbell ring value to both HW and SW occupancies for ring
4. RINGACC re-evaluates pending bit for ring

11.2.2.2.2.7 Queue Push Operation (VBUSM Write to Source Interface)

The following sequence will occur for each VBUSM write which is sent to the source interface:

1. RINGACC extracts ring number from incoming write transaction address
2. RINGACC looks up ring state using *ring number*
3. RINGACC calculates effective write address using *ring base* + *ring_index*
4. RINGACC re-evaluates pending bit for ring
5. RINGACC increments WR index and RD and WR occupancy for ring
6. RINGACC allocates *cid* from scoreboard and places original *crouteid, cid*, and *ring number* into scoreboard
7. RINGACC pushes altered *caddress, cid* and unaltered remainder of command attributes to output FIFO (note routeid is not included)

At a later time when write status returns:

1. RINGACC recovers original *cid, crouteid*, and ring number from scoreboard
2. RA pushes restored srouteid, sid, and unaltered write status to output write status FIFO

11.2.2.2.2.8 Queue Pop Operation (VBUSM Read from Source Interface)

The following sequence will occur for each VBUSM read which is sent to the source interface:

- RINGACC extracts ring # from incoming read transaction address
- RINGACC looks up ring state using *ring number*
- RINGACC calculates effective read address using *ring base* + *ring_index*

Copyright © 2026 Texas Instruments Incorporated

- RINGACC increments RD index and decrements RD and WR occupancy for ring
- RINGACC re-evaluates pending bit for ring
- RINGACC munges cid to add indicator in 4 MSBs to reconstitute routeid for returning read data
- RINGACC pushes altered *caddress, cid* and unaltered remainder of command attributes to output FIFO (note routeid is not included)

1. At a later time when read data returns:

- RINGACC recovers original *cid, crouteid*, and *ring number* from scoreboard
- RINGACC pushes restored *rrouteid, rid*, and unaltered read data, status, and other control signals to output read FIFO

11.2.2.2.2.9 Mismatched Element Size Handling

When less data is written than defined by the element size for the ring for RING or MESSAGE modes, only the written data is stored in the memory and any remaining bytes maintain their old values. The index and occ are still incremented normally, and any special fields that are unwritten are assumed to be 0 (such as the length for QM mode rings).

Writing less data for CREDENTIALS mode is illegal, as those types need the full data to append the additional words to the correct offsets in memory.

When more data is written than defined by the element size for the ring, it will be an error and the write will be ignored and no index or occ increments occur. The only exception is for writes to an 8 byte element size ring, where the additional fields up to another 8 bytes are allowed but not written (for old Queue Manager compatibility).

When less data is read than defined by the element size for the ring, only the read data is fetched from memory, and any remaining bytes are lost. The index and occ are still incremented normally.

When more data is read than defined by the element size for the ring, it will be an error and the read will not affect the ring, the read data will be 0s, and no index or occ decrements occur. Access beyond the credentials word should not be attempted and can result in unpredictable behavior.

All accesses must be for multiples of 32 bit words, and partial bytes are not supported for all ring modes.

11.2.2.2.3 Events

The module outputs events about queue levels that can be used by an external module for statistics or interrupts.

Each queue has:

- an up event when the queue goes from 0 elements to 1 or more elements
- and a down event when the queue goes from 1 or more elements to 0 elements. The event number is programmable per queue in the RING[a]_EVT register. This event occurs when the state machine updates the final occupancy for the queue. In ring mode this is when the response from the memory access returns, while in other queue modes this is when the operation is processed.

An event number programmed to 0xFFFF indicates to not produce an event for that ring when an event is not necessary. The register event number fields reset to 0xFFFF, so they must be programmed to a valid value before events will be generated. This applies to the monitor event number fields as well.

11.2.2.2.4 Bus Error Handling

If a bus error is detected on a response to a ring memory transaction, an error event is triggered to alert software. Since the ring contents may be corrupted as the push or pop was not completed correctly, software should consider resetting the ring and any software or hardware elements using the ring. A log register (RINGACC_ERROR_LOG) captures the ring that had the error and whether it was from a push or pop, and an error up event is sent to the programmed event number. After an error is logged, another will not be captured until the first is read out. When read, if an error is pending an error down event is sent to clear any interrupt,

Copyright © 2026 Texas Instruments Incorporated

and then the next bus error log can be captured. A read of the RINGACC_ERROR_LOG register when there is no error log pending will not produce any down events. The event that is triggered is also programmable. Only legal push and pop operations will capture this error, and not for any peek, emudbg read, flush command, or any operations that is illegal and causes the resulting memory access to be flushed.

11.2.2.2.5 Monitors

Monitors can be configured that allow various functions to be tracked of programmed queues. Each monitor is programmed to monitor a particular queue, the function to provide, and the data source to operate on. The supported functions are: to check the queue against programmed thresholds, to keep track of watermarks of the queue, to keep track of starvation occurrences (a read of an empty queue), or statistics capture. The supported data sources are: the queue element count, the head element size value, or the accumulated size value. Each monitor can also be programmed to produce an event should it have a triggering condition.

11.2.2.2.5.1 Threshold Monitor

A threshold monitor uses a programmed low threshold value and a programmed high threshold value to track the data source in the queue and cause an interrupt or status when a threshold is broken, either below the low value or above the high value. This is useful for software to replenish empty buffers when a queue has too few, or for a queue to accumulate enough elements so that software must start processing them, or for debug. The threshold being broken will cause the up or down event.

11.2.2.2.5.2 Watermark Monitor

A watermark monitor tracks the low and high values of the data source since the last read of the monitor. After initial setup or clearing, the watermark values will load the current value of the data source as the low and high. Then for successive operations if the resulting data source of the queue is below the low watermark, or above the high watermark, the associated watermark will be updated. When the monitor value registers are read, the value is cleared and starts tracking again using the current data source value. This can be useful for tracking low and high values of queue element counts or sizes for data tracking, debug or future optimizations, such as allocating items to a queue or buffer sizes based on counts.

11.2.2.2.5.3 Starvation Monitor

A starvation monitor tracks the number of starvation events (a read to an empty queue) that occurred on the queue. The data source is ignored, and the starvation count is always incremented whenever there is a read to the queue and the queue is empty. The count is cleared when the monitor value is read. This can be useful to trigger if any starvation event occurs, or to track how many of these events occur in a timeframe, and can be used for future optimizations or debug.

11.2.2.2.5.4 Statistics Monitor

The statistics monitor can track some simple statistics on the queue operations. The data source is ignored. The first value is the count of the number of writes to the queue, and the second value is the count of the number of reads from the queue. The counts are cleared when the monitor values are read. This can be useful to track the activity on a queue, and can be used for future optimizations or debug.

11.2.2.2.5.5 Overflow

RINGACC provides an overflow function where a push to a full ring will result in a push to a special overflow ring, rather than just losing the pushed data. The overflow ring is defined through RINGACC_OVERFLOW register, and that ring must be defined with a large enough element size to cover the element size of any other ring, as well as enough elements to hold enough overflow data (as an overflow to the overflow ring will be lost). When there is a push to a full ring, there will be two elements written to the overflow ring, first the push data, and then the ring number that was full. A supervisor entity can own the overflow ring and perform what actions are needed from the overflow data. The size of the overflow ring must be an even number since two elements will be pushed for each overflow event. The overflow ring must be configured in ring or message modes.

An overflow queue value of 0xFFFF will disable the overflow function and will not record queue overflows. (SR2.0)

Copyright © 2026 Texas Instruments Incorporated

11.2.2.2.5.6 Ring Update Port

To enhance performance for DMAs, the RINGACC provides a bus that indicates when ring updates occur, allowing the DMA to track their own rings and detect updates before the responses arrive. The bus will go valid whenever there is an update to the state of a ring, which can be from a push, pop, or doorbell write. It will not go valid for a peek, an emudbg pop, or a push to a full ring, as those do not update the ring state. When the bus is valid, the other signals are valid and indicate whether the operation was a push or a pop, whether the resulting state of the ring is empty, the ring number, and the number of elements updated to the ring (the push determines whether it is subtracted or added to the current count). One special case is a pop from an empty ring, which indicates an update but the element count is 0 since no items were actually popped, but a DMA using the cnt to add or subtract must handle this correctly by default. The bus will go valid once the operation command has been completed, and not after the response has been receivied, even for ring mode rings, allowing the DMA to see the update as soon as it is known by the RINGACC even before it is known to software.

11.2.2.2.5.7 Tracing

This module provides a trace output of all operations so that the traffic can be viewed at a later time. The trace uses a simple write-only 32-bit VBUSP bus, with a defined packet header. The msgtype field defines the type of operation, and the message data is simply the data involved in the operation. It is likely the trace output will be read slower than new operations could provide data, so there will be a trace buffer able to hold two complete messages plus trace headers. If a new operation starts while the trace buffer does not have enough space, then that trace message will not be sent. And RINGACC_TRACE_CTL register controls whether to enable the trace output, whether to trace a single queue or all queues, and whether to include the message data in the trace output.

Table 11-67 shows the trace message for a push.

Table 11-67. Trace Message for a Push

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x1	Msgtype for push
1	8:0	0x009	Standard Trace Header
2	31	push_to_head	Push to head flag, 0 = to tail, 1 = to head
2	30:16	0x0	Unused
2	15:0	queue	Queue for push
3-N	31:0	message data	Optional message data for push, length dependent on message size

Table 11-68 shows the trace message for a pop.

Table 11-68. Trace Message for a Pop

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x2	Msgtype for pop
1	8:0	0x009	Standard Trace Header
2	31	empty	empty pop, 0 = valid message, 1 = empty
2	30:16	0x0	unused

Copyright © 2026 Texas Instruments Incorporated

Table 11-68. Trace Message for a Pop (continued)

Word	Bits	Data	Comment
2	15:0	queue	queue for pop
3-N	31:0	message data	Optional message Data for pop, length dependent on message size

Table 11-69 shows the trace message for a peek

Table 11-69. Trace Message for a Peek

Word	Bits	Data	Comment
1	31:16	0x0	Time to be replaced by trace hardware
1	15:9	0x3	Msgtype for peek
1	8:0	0x009	Standard Trace Header
2	31	empty	empty peek, 0 = valid message, 1 = empty
2	30:16	0x0	unused
2	15:0	queue	queue for peek
3-N	31:0	message data	Optional message Data for peek, length dependent on message size

Copyright © 2026 Texas Instruments Incorporated

11.2.3 Secure Proxy (SEC_PROXY)

This chapter describes the Secure Proxy (SEC_PROXY) module in the DMSS.

11.2.3.1 Secure Proxy Overview

The Secure Proxy module provides proxy buffers for hosts that need to create large data bursts but can only perform small accesses.

11.2.3.1.1 Secure Proxy Features

Secure Proxy supports the following features:

- Provides proxy function to store large data bursts that a host can only access in smaller amounts
- Supports a configurable number of threads, where each has their own independent proxy function
- Keeps the large data burst coherent until the complete data has been accessed
- Allows for interleaved access between multiple hosts or tasks using multiple proxy threads
- Supports a configurable target resource. The target has a configurable number of channels, size of each channel and base address
- Supports a programmable fixed queue for each proxy thread
- Supports multiple producers all writing to the same queue
- Supports a max message count for outbound proxy threads limiting the number of messages a thread can produce
- Supports programmable thresholds for when to generate events
- Optionally supports dynamic clock gating

Table 11-70. Secure Proxy Allocation Within Device Domain

Module Instance	Domain	
	MCU	MAIN
DMSS0_SEC_PROXY0	-	✓(DMSS)

11.2.3.1.2 Secure Proxy Parameters

Secure Proxy Configuration Parameters shows the Secure Proxy configuration parameters set during SoC design.

Table 11-71. Secure Proxy Configuration Parameters

Module Instance	Parameters				
	Proxies(1)	Message Size(2)	Target	Channels(3)	Sizes(4)
DMSS0_SEC_PROXY0	76	64	DMSS0_RINGACC	20	4096

(1) Number of proxy threads supported. Can be read off from the SEC_PROXY_CONFIG read-only register
(2) Number of bytes for message. Can be read off from the SEC_PROXY_CONFIG read-only register
(3) Number of channels for the target
(4) Number of bytes per channel supported for the target

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

11.2.3.2 Secure Proxy Functional Description

This module provides proxy functions for hosts that need to create large data bursts but can only perform small accesses. Examples of this need are:

- large messages (for power control, security control, or IPC)
- data descriptors that are more than a pointer
- DMA Transfer Requests (TR)

All of these require larger data sets, from 12 to 128 bytes, and the storage hardware requires the entire data on the bus in a single burst. Unfortunately, many CPUs cannot produce bursts to device memory, such as registers, and the largest CPU access is typically 8 bytes. So the proxy provides the ability to access the hardware with a single burst, and at the same time allows the CPU to access the same data in smaller accesses. The proxy itself does not contain the resources, but will front other hardware that do contain the resources. The proxy will access this other hardware with the entire data in a single burst, once the host has completed the data.

Secure Proxy Block Diagram describes Secure Proxy major building block

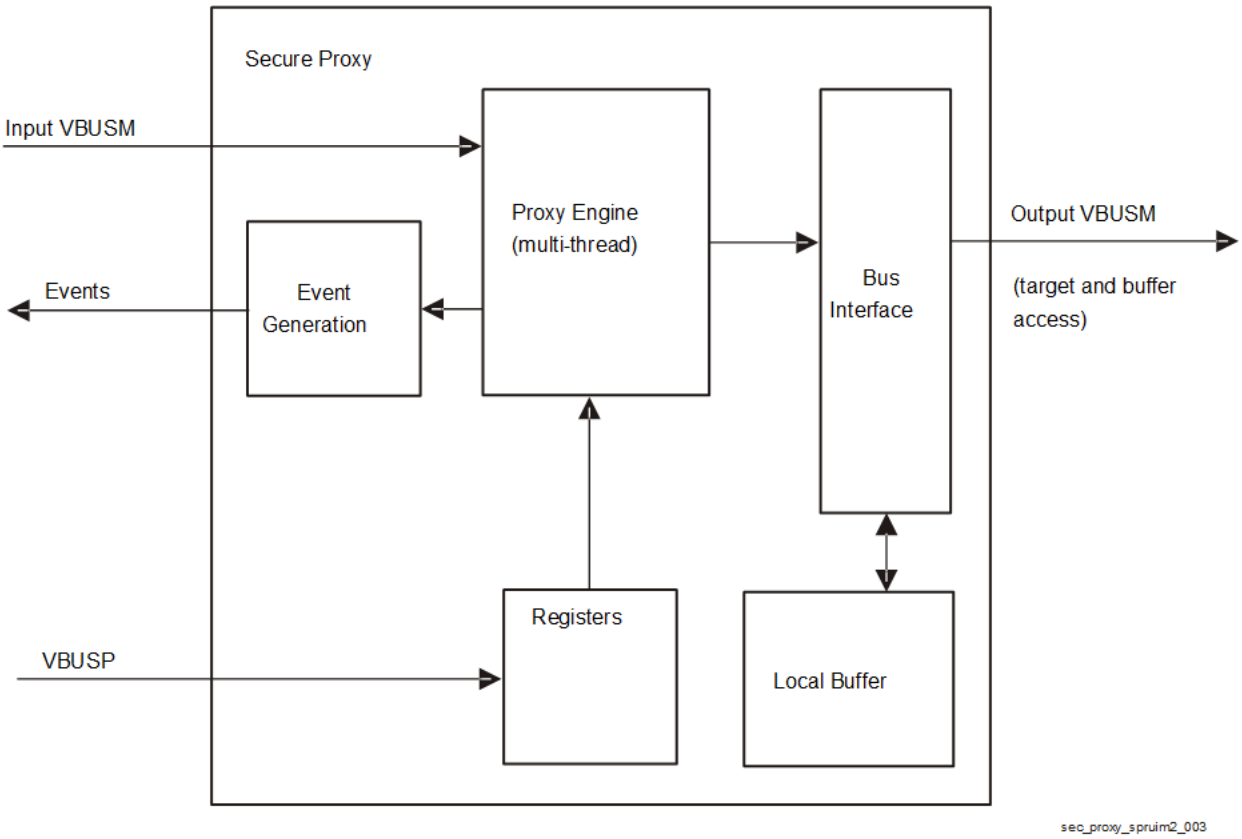


Figure 11-9. Secure Proxy Block Diagram

11.2.3.2.1 Targets

11.2.3.2.1.1 Ring Accelerator

The RINGACC provides four distinct offsets to use per ring/queue shown in RINGACC Offset per Ring/Queue, totaling 4 KB space per ring/queue.

Secure Proxy always writes to tail, which is the offsets from 0x200 to 0x3FF. Secure Proxy always reads from the head, which is the offsets from 0x000 to 0x1FF. The latter two offsets in RINGACC are not used by this proxy. And if the proxy message size is less than 512 bytes, then it always accesses the upper bytes of those regions, so that the message always ends on the last byte of each region, byte 511.

Copyright © 2026 Texas Instruments Incorporated

Table 11-72. RINGACC Offsets per Ring/Queue

Offset	Operation
0x0 - 0x1FF	Reads pop from head. Writes push to head.
0x200 - 0x3FF	Writes push to tail. Reads pop from tail.
0x400 - 0x5FF	Reads peek from head. Writes ignored.
0x600 - 0x7FF	Reads peek from tail. Writes ignored.
0x800 - 0xFFF	Reserved.

This means that target writes will use the following calculation:

target base address + (queue × 4KB) + 0x400 - msg_size (17), and burst until offset 0x3FF.

And target reads will use the following calculation:

target base address + (queue × 4KB) + 0x200 - msg_size (18), and burst until offset 0x1FF.

11.2.3.2.2 Buffers

The proxy does not store all the buffers internally. It is programmed with a buffer base address (SEC_PROXY_BUFFER_L, SEC_PROXY_BUFFER_H), and all accesses to the buffer are made through the output VBUSM port using that base. The external buffer must be enough to store one msg_size per proxy thread. The proxy only includes a small temporary buffer to save one entire message when accessing the target. Otherwise all host accesses are made to the external buffer. This buffer storage must be considered private for the proxy hardware, so any other access can cause unpredictable results.

11.2.3.2.2.1 Proxy Credits

To support outbound proxy limits, the proxy will use internal credits for tracking resource usage. SEC_PROXY_THREAD[a]_CTL register sets the max credits each outbound proxy thread contains. When a proxy thread sends a message, the current credit count (SEC_PROXY_THREAD[a]_STATUS) decrements by 1. If the current credit count is 0, then the proxy thread is not allowed to send any more messages, and they just remain inside the proxy (so that the host can come back later when a credit has become available and quickly send the same message). When a message that was send by this proxy thread is read by an inbound proxy thread then the credit is returned to this proxy thread and the current credit count increments by 1.

For inbound proxy threads, the credits are used to track pending messages. When an outbound proxy thread sends a message that is read by this proxy thread, then the current credit count of this proxy thread is incremented by 1. A current credit count that is non zero indicates a message is pending and can be read. When the proxy thread completes the read of a message the current credit count is decremented by 1. If the current credit count is zero, then the proxy thread will read an empty message where all the data is 0s. The credit count saturates at 255, so if there can be more messages in the system for that proxy thread than 255 then multiple proxy threads should be used, or the count can become mismatched after a saturate.

11.2.3.2.2.2 Proxy Private Word

To support tracking of credits from each proxy thread, the proxy will extract the first word from the total message size for private tracking usage. This will include saving the proxy thread ID used within the proxy. This is necessary so that when a proxy reads a new message from the target it can inspect this word and return the credit to the source proxy thread. This allows the outbound proxy threads to have a set number of message credits, which decrement when a new message is written, and increment when a message from that proxy thread is read, thus keeping the credits in the system coherent. Any data written to this first word of the message will be ignored, so software should not write to the first word of messages through the proxy. Consumer software can use the first word to identify the source proxy thread of the message (SEC_PROXY_DATA_j).

Copyright © 2026 Texas Instruments Incorporated

11.2.3.2.2.3 Completion Byte

The proxy completes a message when the completion byte is accessed. This completion byte is the final byte in the full size of the message. This byte can be written by any size access, whether byte, 32-bit, or 64-bit word. The write of a message is not complete until this final byte of the message is written, and only then will the message be sent to the target. The read of a message is not complete until this final byte of the message is read by the host, and only then will reading a new message be allowed. Even if the final byte of the message is not needed, it must be accessed to complete the message inside the proxy hardware.

11.2.3.2.3 Proxy Thread Sizes

There are a configurable number of proxy threads in the module. Each thread has its own address space to the module and determining which proxy thread is being accessed is simply based on the address bits above the final size of each proxy thread space. Since each proxy thread will be programmed to a single resource, the only space requirement is that for using 4 KB to match typically used MMU page sizes. Only the first N bytes, matching the target channel size, is usable.

11.2.3.2.4 Proxy Thread Interleaving

Each proxy thread must have a single owner, as each proxy thread only has a single buffer to work on the current data. If there are multiple owners in the system, they must each use their own proxy thread. Then they will have their own buffer and each can access a different resource at the same time. Each proxy thread is completely independent and kept coherent regardless of accesses to other proxy threads or regardless of the order of accesses. So the proxy support interleaving of multiple hosts' accesses.

11.2.3.2.5 Proxy States

Each proxy thread uses a simple state machine to track the currently usage by that host. This state machine is used to track when a proxy thread is in use or not, but also for error detection. The proxy thread always starts in idle state, when there is no currently activity. There is a write state for when the proxy thread has been written and contains data that has yet to be written to a resource. There is a read state for when the proxy thread has read data from a resource and that entire data has not been fully read by the host. When a proxy thread access completes, final write or final read, then the state will go back to idle. The following sections show the utilization of these states.

11.2.3.2.6 Proxy Host Access

A proxy thread begins in an idle state, where there is no data in the buffer. In this state the proxy thread will accept an access to the resource.

11.2.3.2.6.1 Proxy Host Writes

The first write to an idle proxy thread puts the proxy thread into write mode for the selected channel. Further legal writes are stored in the buffer for the enabled bytes and offset within the buffer. A write to a previously written location will overwrite the previous data. The buffer is not cleared initially, so if a byte is not written, the message will include previous buffer data. When there is a legal write to the last byte of the resource, then the entire data is written by the proxy to the target and channel as a single write burst. The proxy will wait for the write status from the target write before it sends the write status for this final byte host write. This allows the host to use the write status to guarantee the entire write data from the proxy has landed at the target. The proxy thread is then put back into idle state. A write completion when there are no credits available results in the message not being written to the resource and the proxy thread remains in write mode (so that the message can be quickly completed when a credit becomes available).

11.2.3.2.6.2 Proxy Host Reads

When in idle state, the first read to the proxy thread will put the proxy into read state, and the proxy will read the target and channel requested in a single burst of the target's channel size. When the data is returned it is stored in the buffer and the accessed part is returned to the host. Further legal reads are allowed and return the accessed data in the buffer. When there is a legal read to the last byte of the resource, the accessed data is returned, and the proxy thread is put back into idle state. If an initial read is made and there are no pending messages, the proxy thread will return an empty message, where the data is all 0s.

Copyright © 2026 Texas Instruments Incorporated

11.2.3.2.6.3 Buffer Accesses

Each read or write will actually trigger a read or write to the external buffer space allocated for that proxy thread, where the real message data is stored. In addition, when a write message completes, the external buffer will be read for the entire message for that proxy thread so that it can then be written to the target. Similarly, after a read causes a read from the target to get a new message, the entire message is written to the external buffer to store the message to the buffer.

11.2.3.2.6.4 Target Access

After the write has completed and the external buffer read for the entire message, the entire message is then written to the target as a single burst. Similarly, for a read that needs a new message, the target is read for a full message as a single burst.

11.2.3.2.6.5 Error State

If an error is detected in a proxy thread, the proxy will put that proxy thread into error state, and the current buffer will be lost. A status bit will be set that indicates the proxy thread is in error. All accesses to that proxy thread will be ignored until the status bit is cleared. Once cleared the proxy thread is put back into idle state. Errors are an access to an offset beyond the length of the message, or a mismatch in the direction of access against what was programmed such as reading an outbound thread or writing an inbound thread.

11.2.3.2.7 Permission Inheritance

The proxy will automatically inherit the permissions from the access and pass these to the target upon the target burst access. The exact transactions that are inherited from are the final byte write or the first read, and these exact permissions are used on the output bus transaction that accesses the target.

11.2.3.2.8 Resource Association

Each proxy thread has a register (SEC_PROXY_THREAD[a]_CTL) to define the resource within the target that the proxy thread is mapped to. The host has no ability to access any other resource in the target. This allows the setup of the proxy to determine the resources that all the hosts use.

11.2.3.2.9 Direction

Each proxy thread has a register (SEC_PROXY_THREAD[a]_CTL) to define the direction for access. It can either be an outbound proxy thread for writing messages to resources. Or it can be an inbound proxy thread for reading messages from resources. The host must match the direction programmed or it will result in a proxy thread error.

11.2.3.2.10 Threshold Events

Each proxy thread can produce events based on the state of the resource it uses. For outbound proxy threads, a threshold can be set so that an event is produced when there are now at least N messages available to write. For inbound proxy threads, a threshold can be set so that an event is produced when there are now at least N messages available to read. These threshold values are programmed in SEC_PROXY_EVT_MAP_j, allowing the host to determine how often it gets events. Programming the count values while the threshold is set to 0 will not trigger events, but if the threshold is not 0 then programming the counts will trigger events based on whether the threshold is reached or not. No event is generated if the event is programmed to 0xFFFF.

11.2.3.2.11 Error Events

Each proxy thread can produce an event when it has detected an error. This error occurs when the host violates the programmed setup of the proxy thread or accesses beyond the message size. There is a separate status bit per proxy thread for these errors. No event is generated if the event is programmed to 0xFFFF.

11.2.3.2.12 Bus Error and Credits

If a target read for a message returns a bus error, this indicates the read was bad and the data is corrupt and usually 0s. Because the private word in contained in the message, if the private word is corrupted then the source proxy thread is not valid and the credit return will not be correct. Since the data is usually 0, this means the credit would be returned to proxy thread 0 incorrectly most of the time. And the credit will be lost for the

Copyright © 2026 Texas Instruments Incorporated

actual source proxy thread of the message that was not read correctly. The hardware has no way to correct for this, as the message from the target is corrupt so it does not know the correct owner of the credit. If software detects that a read message is corrupt, it can attempt to correct the situation by resetting the affected proxy threads using that queue, to restore their credit counts back to the default. Software may also want to reset proxy thread 0 since it may be getting extra credits.

11.2.3.2.13 Debug

When the transaction has the emudbg set for a read, all side effects will not occur. This means that the read will not cause a target read since that would affect the target, and instead current buffer will be read even if the previous message has completed.

Copyright © 2026 Texas Instruments Incorporated

11.2.4 Interrupt Aggregator (INTAGGR)

This chapter describes the INTAGGR module, part of DMSS

11.2.4.1 INTAGGR Overview

The Interrupt Aggregator (INTAGGR or INTA) provides a centralized machine which handles the termination of system events to that they can be coherently processed by the host(s) in the system. The main function of the module is to convert between 'global events' (assertion and de-assertion events transmitted on the event-transport-lane (ETL) bus) and 'local events' (level sensitive signals on output, and level or edge sensitive signals on input).

Table 11-73. INTAGGR Allocation Within Device Domain

Module Instance	Parameters	
	MCU	MAIN
DMSS0_INTAGGR0	-	✓(DMSS)

11.2.4.1.1 INTAGGR Features

The DMSS0_INTAGGR0 supports the following features:

- 64-bit VBUSP target using 64-bit registers
- Provides a set of TI Interrupt Architecture compliant interrupt status and mask register sets which are used to pass specific event status to one or more Host blocks.
  - Maps a collection of DMA Messaged Events which are input through an Event Transport Lane into specific bit locations in one or more interrupt cause registers
  - Mapping is performed based on a programmable table which provides a single location for each ordinal input event number (0 through sevt_cnt-1)
    - Table specifies a specific bit in a specific cause register that the event should set or clear depending on the type of event (up vs down)
    - Tracks a single 'event up' / 'event down' condition. There is no event counting. (The 'cnt' field of the ETL is ignored.)
  - Tracked status interrupts are presented to the user through a standard interrupt register interface.
    - Provides separate 'enable set' and 'enable clear' registers.
    - Provides both masked and unmasked interrupt status.
    - Interrupt status bits can be manually cleared using 'write 1 to clear'.
- Provides an optional set of Unmapped Event (UNMAP) which can take an 'unmapped' event from the ingress ETL and generate a Global event on the egress Event Transport Lane (ETL) interface.
  - Each ingress event can generate an egress Global event on the egress ETL, controllable through a mapping register.
  - Each mapping register has the option of alternatively directly setting an interrupt status bit.
- Provides an optional set of Global Event Input (GEVI) counters which can count events which were delivered via an ingress Event Transport Lane (ETL).
  - Each counted event provides an MMR by which the count can be read and an MMR by which a specified amount can be decremented by the host.
  - Each counted event generates an egress Global event on a zero to non-zero and non-zero to zero transition, controllable through a mapping register.
  - Each mapping register has the option of alternatively directly setting an interrupt status bit.
- Provides an optional set of Global Event Input (GEVI) 'Multicast' registers which can take a Global event from an ingress ETL and generate two egress Global events on the egress ETL interface.
  - Each mapping register has the option of alternatively directly setting an interrupt status bit.

Copyright © 2026 Texas Instruments Incorporated

- Provides an optional set of Local Event Input (LEVI) to Global event registers which can be used to convert pulsed discrete interrupt inputs or clock synchronous rising edge events into Global events on an egress ETL.
  - Each ingress event index provides a configurable 'pulse' or 'rising edge' bit, plus the configurable index of the generated Global event.
- Supports dynamic clock gating via

Note

Some features may not be available. See *Module Integration* for more information.

11.2.4.1.2 INTAGGR Parameters

Interrupt Aggregators Parameters shows the Interrupt Aggregator 0, parameters set during design time.

Table 11-74. Interrupt Aggregators Parameters

Module Instance	Parameters				
	VINTR[1]	SEVI[2]	GEVI[3]	LEVI[4]	MEVI[5]
DMSS0_INTAGGR0	184	1536	256	32	128

(1) VINTR – Number of Virtual Interrupt outputs. These are connected to the interrupt router inputs. Value can be read from INTAGGR_INTCAP register.

(2) SEVI – Number of Main (Steerable) Events. Value can be read from INTAGGR_INTCAP register.

(3) GEVI – Number of Countable Events. Value can be read from INTAGGR_AUXCAP register.

(4) LEVI – Number of Local Event inputs. Value can be read from INTAGGR_AUXCAP register.

(5) MEVI – Number of Multicast Events. Value can be read from INTAGGR_AUXCAP register.

Copyright © 2026 Texas Instruments Incorporated

11.2.4.2 INTAGGR Functional Description

The Interrupt Aggregator (INTAGGR) provides a centralized machine which handles the termination of system events to that they can be coherently processed by the Host(s) in the system. Both the signaling and content of TI system events are incompatible with standard interrupt controllers. The INTAGGR provides mechanisms which convert the TI proprietary system events to standard level sensitive pending bits which can be used by all downstream interrupt infrastructure. Events can be presented to the INTAGGR in two different forms:

- Events may be active high pulses, or clock synchronous rising edges, which are input on separate orthogonal pins. These are called 'Local Events'.
  - Local event signals that are used in pulse counting mode must be sourced by the same clock as the interrupt aggregator.
  - Local event signals that are used in edge counting mode must be sourced by pseudo-synchronous sources (and be a slower clock multiple) to the interrupt aggregator clock, and they must remain high for at least 1 'slow' clock cycle.
- Events may be messages which are input in time division multiplexed sequential order from an Event Transport Lane. These are called 'Global Events'.

In both cases, these events need to be conditioned to transition them from a transient indicator that something occurred to a persistent and reliable indication of the current state of the system. The IA is architected to perform this conversion in an efficient and cost effective manner.

11.2.4.2.1 Submodule Descriptions

11.2.4.2.1.1 Status/Mask Registers

The Status and Mask Registers block is responsible for providing persistent storage for the interrupt status and mask bits and for formatting those in a way that is compliant to the TI Interrupt Architecture requirements. These requirements include the ability to set and clear bits orthogonally and to provide a masked version of the status register that corresponds to the supplied bit mask for each register.

11.2.4.2.1.2 Interrupt Mapping Block

The Interrupt Mapping Block accepts ordinally numbered events (0-N) and converts those ordinal event numbers into a status register number and bit number pair that is then used to manipulate the specified bit in the Status Registers block.

11.2.4.2.1.3 Global Event Input (GEVI) Counters

The GEVI Counters block is responsible for accepting an Event Transport Lane events and summing the total message counts for each received event index. The module creates global egress events for zero to non-zero count up' events and non-zero to zero 'count down' events. The egress global event index is configured via GEVI[a]_MCMAP register, so count status egress events can optionally be fed back into the INT_AGGR, via the ETL switch fabric.

11.2.4.2.1.4 Local Event Input (LEVI) to Global Event Conversion

The Local to Global event block is responsible for accepting an array of input pins and independently counting the number of cycles for which those pins are asserted high, or by counting individual clock synchronous rising edge events. The counting mode is configurable on a 'per pin' basis. The events are converted to egress Global events on an egress ETL. Global events can optionally be fed back into the INT_AGGR, via the ETL switch fabric.

11.2.4.2.1.5 Global Event Multicast

The Global event multicast block takes an ingress global event from an ingress ETL, and maps it into two egress Global events on two egress ETL interfaces. Global events can optionally be fed back into the INT_AGGR, via the ETL switch fabric.

11.2.4.2.2 General Functionality

Copyright © 2026 Texas Instruments Incorporated

11.2.4.2.2.1 Event to Interrupt Bit Steering

Each time an event message is received on the Main Event Transport Lane interface, the interrupt mapping block performs a direct lookup into an SRAM using the event number as the address. The SRAM stores a corresponding interrupt status (INTR_STATUSM) register and bit number within that register (VINT[a]_STATUS_MSKD) which are to be manipulated anytime a message is received indicating something occurred on that specific event. When an up event is received, the specified bit in the [a] Status register will be set. When a down event is received, that same bit will be cleared. This block is what allows flexible aggregation of various system events into an array of bits in the interrupt status registers. It is intended that this mapping is essentially static - set up when a resource is allocated and left untouched until the resource is no longer needed. Note that the 'cnt' field of the ETL is ignored and no interrupt counting is performed here. When 'UpDn=1', the interrupt flag bit is set, and when 'UpDn=0', the flag bit is cleared.

11.2.4.2.2.2 Interrupt Status

The event messages which are generated from the event to interrupt bit steering logic are input to the Interrupt Status registers. Each time an event is received, the interrupt status registers machine will assert or de-assert the specified bit in the specified register. The assertion and de-assertion in the interrupt status register is unaffected by the interrupt enable state. When an up event is received, the corresponding bit is set and when a down event is received, the corresponding bit is cleared. Some sources of input events will not include the ability to send a down event. In these cases, the interrupt router provides the ability to clear the status bits through the Interrupt Source Clear register (VINT[a]_STATUS). The host will write a one to the specific bit in the register which is to be cleared and the interrupt status machine will clear the bit internally. It is not intended that the Host directly clear bits which are automatically cleared via down events from the source itself.

11.2.4.2.2.3 Interrupt Masked Status

Interrupt enable bits are used in conjunction with the interrupt status bits to create the interrupt masked status register values. The interrupt masked status register (VINT[a]_STATUS_MSKD) contains the value of the interrupt status ANDed with the value of the interrupt enable register (VINT[a]_ENABLE_SET). Each time a new event message is received from the event to interrupt bit steering logic or the interrupt enable register is modified, the interrupt masked status register is re-evaluated.

11.2.4.2.2.4 Enabling/Disabling Individual Interrupt Source Bits

Separate (VINT[a]_ENABLE_SET and VINT[a]_ENABLE_CLR) registers are provided to allow individual enable bits to be enabled or disabled without the need for a read-modify-write operation. When the VINT[a]_ENABLE_SET register is written, all bits within written bytes which are 1 will cause corresponding bits in the internal enable register to be set. When the VINT[a]_ENABLE_CLEAR register is written, all bits within written bytes which are 1 will cause corresponding bits in the internal enable register to be cleared.

11.2.4.2.2.5 Global Event Counting

When enabled, the INTAGGR will provide a set of counters which will track how many outstanding messages have been observed for each ingress event index from the ingress Counted Global Event Transport Lane (ETL) interface. For each message received where the 'UpDn' flag is set, the corresponding counter will be incremented by value of the 'cnt' field of ingress message. Events where the 'UpDn' flag is cleared are ignored. The counter is made visible to software in the GEVI[a]_COUNT register. When software has read the count, it can acknowledge that count has been seen and processed by writing back an integer value specifying the amount by which the counter should be decremented, and the counter will subtract that value from the current count (which may have updated since it was read). The count will saturate at a value of 0xFFFFFFFF. Writing a count 'ack' value higher than the one read is not supported and will produce non-deterministic results.

When a count transitions from zero to a non-zero value, a Global Up 'UpDn=1' event is sent out an egress ETL interface. When a count transitions from a non-zero value to zero, a down 'UpDn=0' event will be sent. The index of the Global event is mappable on a 'per-counter' basis, using the GEVI[a]_MAP register. The event may be mapped such that it then re-enters INTAGGR's 'Event to Interrupt Bit Steering Block' to generate an interrupt to the host. The event routing is handled by an external event switch fabric. Note that writing a new egress event index via the GEVI[a]_MAP register does not alter the stored event count for the ingress event register index.

Copyright © 2026 Texas Instruments Incorporated

11.2.4.2.2.6 Local Event to Global Event Conversion

When enabled, the INTAGGR will provide a set of Local to Global event conversion registers. Local events may be discrete clock synchronous pulses or clock synchronous rising edges. The counting mode is configurable on a 'per pin' basis. In pulsed mode, the module will track how many outstanding clock cycle long pulses have been observed on each of the provided levi interface pins. For each cycle in which a LEVI input pending bit is asserted the corresponding counter will be incremented by 1. In 'rising edge' mode, the number of rising edge transitions on the pin is counted.

The module will generate egress Global events on its egress ETL, with a different global index configured for each ingress LEVI pin. The LEVI pins numbers (1-N) are converted to arbitrary global event indices via the global event mapping LEVI[a]_MAP registers. The egress Global events can themselves be mapped to re-enter the INTAGGR's Global Event Counting block, so that the counts can then be made visible to software in the GEVI count registers as described in Section 10.2.7.3.1.3. The event routing is handled by an external event switch fabric.

11.2.4.2.2.7 Global Event Multicast

In DMSS0_INTAGGR0, the INTAGGR will provide a set of Global event multicast (GEVI[a]_MCMAP) registers. These registers allow an ingress Global event on its ingress ETL interface to be mapped into two egress Global events on two separate egress ETL interfaces. A set of registers map the ingress Global event index (1-N) into two arbitrary egress event indices.

Copyright © 2026 Texas Instruments Incorporated

11.2.5 Packet Streaming Interface Link (PSI-L)

This section describes the Packet Streaming Interface Link (PSI-L) for the device.

11.2.5.1 PSI-L Overview

PSI-L is a packet based protocol used to transfer data between several device modules. Each transaction is routed based on thread ID value instead of physical address. All PSI-L interfaces are point to point direct connections to DMSS0. All switching functions among the PSI-L based interfaces are done inside DMSS0.

The PSI-L also has an event interface referred to as ETL, which is purely used to transfer event information.

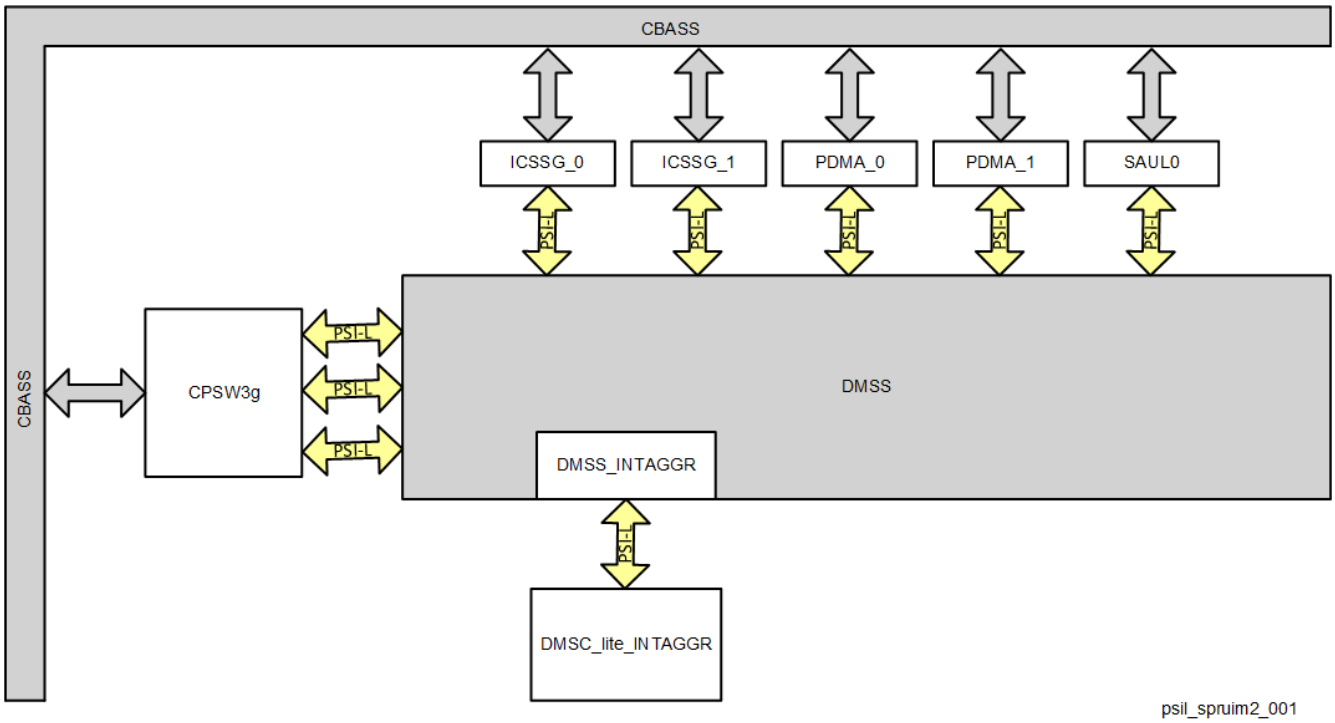


Figure 11-10. PSI-L Overview

Copyright © 2026 Texas Instruments Incorporated

11.2.5.2 PSI-L Functional Description

11.2.5.2.1 PSI-L Introduction

The PSI-L is used to transfer words of packet data and control information between two peer entities via direct connection. There are credits used for flow control to guarantee that blocking does not occur between threads. Multiple packet transfers can be ongoing simultaneously across a PSI-L interface each on a separate logical thread but all sharing a single data path through time division multiplexing. Each packet which is transferred is accompanied by a destination thread ID which indicates the logical destination thread to which the packet is being sent.

There is low level hardware handshake between PSI-L initiator and target used for transferring data. As it is blocking by nature, a higher level protocol is also used in which credits are maintained for each thread connection. Moments of non-readiness are allowed on the interface but there is no possibility that one thread can block another for an indefinite time since a thread must have credit for the destination before it is allowed to use the interface.

PSI-L Connection illustrates the point to point PSI-L connection.

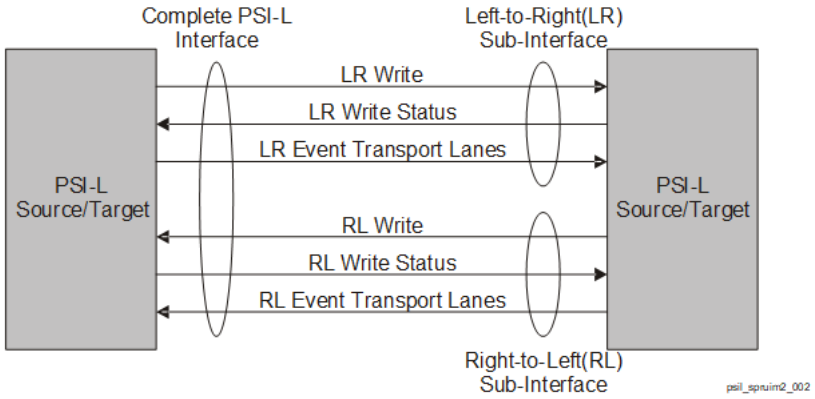


Figure 11-11. PSI-L Connection

PSI-L is intended to be a versatile interface which can carry various types of communications. The following table describes several types of threads which have been defined.

Table 11-75. Types of PSI-L Threads

Type	Description	Example Endpoints
Queue Management (QM) Messaging	Provides transport for queue push, pop, and divert messages to support distributed queue based work passing	Centralized queue manager or message manager and distributed DMA clients
Direct IO	Allows for tunneling CBASS compatible bus transactions through a PSI interconnect	Module which has no CBASS initiator to a remote proxy block
DMA Control Transport	Provides transport for DMA transfer descriptions from a source thread to a destination thread	DMA channel controller to DMA transfer controller
DMA Data Transport	Provides transport for packed data from a source thread to a destination thread	Packet oriented module like networking adapters to or from centralized DMA

Each thread on PSI-L is established between a single thread initiator and a single thread target using a connection oriented transfer mechanism. A thread initiator and a thread target are "paired" to create a logical connection between them referred to as a thread. Pairing is accomplished using a Pair Configuration Message which is valid in all thread types. Pairing can only occur between a thread initiator and a thread target of the same type.

Copyright © 2026 Texas Instruments Incorporated

11.2.5.2.2 PSI-L Operation

The PSI-L allows multiple independent streams of packet data (threads) to share a single interface using data phase granularity time division multiplexing. This time division multiplexing is accomplished by dividing each stream into single data transfers. In this way a stream is comprised of a sequence of strongly ordered data transfers but packet transfers between different streams are allowed to be interleaved. Each data phase is qualified with a type identifier which indicates whether the data is packet info, direct IO write or read info, timestamp, software info, control data, payload data, or status.

11.2.5.2.3 Event Transport

PSI-L provides a mechanism by which events can be transported within one or more optional subinterfaces on a link. These sub-interfaces are referred to as event transport lanes (ETLs) and are independent of any PSI-L packet or message transfers which may be ongoing on the other interfaces within the PSI-L. A maximum of one event can be transferred on each event transport lane in each clock cycle. The event transport protocol is a simple request-ready type of hardware handshake.

11.2.5.2.4 Threads

A thread is a complete flow-controlled stream of communication. The PSI-L thread space is divided into a 32K contiguous region representing all of the source threads (0x0000 - 0x7FFF) and a 32K contiguous region representing all of the destination threads (0x8000 - 0x8FFF).

Source threads are responsible for sending request transactions, accepting response transactions, and sending data transfer transactions. Destination threads are responsible for accepting request transactions, sending response transactions, and accepting data transfer transactions. A given thread generally performs only a single class of transactions (see PSI-L Defined Threads). Both source and destination threads may act as initiators for transfers on one of the PSI-L write sub-interfaces but in this case source threads are actually initiating data transfers while destination threads are only responding to a previous request issued by a source request. Source threads only transfer to destination threads and destination threads only transfer to source threads. Transfers between same thread types do not occur.

The following types of message transfers always originate from source threads and terminate in destination threads:

- Configuration write message
- Configuration read message
- QM push message
- QM pop message
- QM divert message
- Direct read operation message
- Direct write operation message
- DMA transfer request message
- DMA data message

The following types of message transfers always originate in destination threads and terminate in source threads:

- Configuration write response message
- Configuration read response message
- QM push response message
- QM pop response message
- QM divert response message
- DMA transfer response message

11.2.5.2.5 Arbitration Protocol

On each cycle, an arbiter built into the initiator side of each interface decides on which thread to transfer data in the next clock cycle. Both forward (from source threads) and reverse (from destination threads) direction

Copyright © 2026 Texas Instruments Incorporated

transfers can occur across the same physical interface so the arbiter must ensure that sustained blocking of any thread can occur.

The decision about which thread to allow using the interface is based on the following factors:

1. Whether data is ready at the source to be sent
2. Whether credit exists for the thread such that it is guaranteed there is a place for the data to land in the destination endpoint. For reverse direction transfers (response messages), it is assumed that credit always exists.
3. The relative priority of the traffic that the thread is carrying for the given packet duration
4. Whether the transfer is for a forward or reverse direction thread (requests versus responses).

No thread can be considered for inclusion in arbitration unless both data is available and credit exists in the destination endpoint. For each thread which can be considered in the current arbitration cycle, the arbiter should then pick the thread with the highest priority. Reverse transfers should be considered higher priority than forward transfers.

11.2.5.2.6 Thread Configuration

Each source thread has programmable registers for configuring the pairing with a corresponding destination thread and for enabling data flow. Each destination thread has programmable registers for reporting the buffering capability of the thread and for enabling data flow. These registers are accessed using configuration read and write messages from a configuration host.

Each configuration write message sent from a configuration host is acknowledged with a configuration write response message from the addressed endpoint. Each configuration read message sent from a configuration host is acknowledged with a configuration read response message from the addressed endpoint.

11.2.5.2.6.1 Thread Pairing

Depending on the type of traffic that each source thread carries it may or may not have a fixed pairing with a destination thread.

11.2.5.2.6.1.1 Configuration Transaction Pairing

Configuration write and read transactions are only initiated by a special module called configuration proxy (PSILCFG_PROXY). Each configuration proxy in the system uses a single source thread which can initiate configuration write and read transactions to the configuration registers of every other source and destination thread in the PSI-L system. The PSILCFG_PROXY registers are described in *DMASS_PSILCFG_0 Registers*, located in *DMA Controller Registers*.

Each source thread routes to a single target thread and this pairing is specified by programming a required pairing register set for each thread. A source has no ability to change what target thread it talks to. If a source must be able to talk to different targets (or target threads), then it can either have multiple threads or a streaming switch that can be programmed to route the transfers appropriately based on specific packet fields.

11.2.5.2.6.2 Configuration Registers Region

The configuration registers region contains static configuration information including the settings for linking a thread source to a thread destination. These registers are described in *DMA Controller Registers*. The thread ID mapping is shown in *PSI-L System Thread Map*, located in *Module Integration*.

Copyright © 2026 Texas Instruments Incorporated

11.3 Peripheral DMA (PDMA)

This chapter describes the PDMA architecture in the device.

11.3.1 PDMA Controller

11.3.1.1 PDMA Overview

The Peripheral DMA is a simple DMA which has been architected to specifically meet the data transfer needs of peripherals, which perform data transfers using memory mapped registers (MMRs) accessed via a standard non-coherent bus fabric. The PDMA module is located close to one or more peripherals which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only statically configured transfer request (TR) operations.

The PDMA is only responsible for performing the data movement transactions which interact with the peripherals themselves. Data which is read from a given peripheral is packed by a PDMA source channel into a PSI-L data stream which is then sent to a remote peer DMSS destination channel which then performs the movement of the data into memory. Likewise, a remote DMSS source channel fetches data from memory and transfers it to a peer PDMA destination channel over PSI-L which then performs the writes to the peripheral.

The PDMA architecture is intentionally heterogeneous (DMSS + PDMA) to right size the data transfer complexity at each point in the system to match the requirements of whatever is being transferred to or from. Peripherals are typically FIFO based and do not require multi-dimensional transfers beyond their FIFO dimensioning requirements, so the PDMA transfer engines are kept simple with only a few dimensions (typically for sample size and FIFO depth), hardcoded address maps, and simple triggering capabilities.

Multiple source and destination channels are provided within the PDMA which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs round-robin scheduling between channels in order to share the underlying DMA hardware.

The PDMA is linked to the DMSS through a direct PSI-L connection. For more details, refer to *Packet Streaming Interface Link (PSI-L)*.

11.3.1.1.1 PDMA Features

11.3.1.1.1.1 PDMA0 - SPI Features

The PDMA0 - SPI module supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit(s)
  - Each unit supports write bursts up to 64 bytes (on selected channel types)
  - Write Unit 0
    - Provides a 32-bit wide VBUSP write-only initiator interface for peripheral accesses
- Provides 1 memory read access unit(s)
  - Supports 1 outstanding read per interface (VBUSP)
  - Supports read burst up to 64 bytes (on selected channel types)
  - Read Unit 0
    - Provides a 32-bit wide VBUSP read-only initiator interface for peripheral accesses
- Supports up to 15 simultaneous destination (Tx) channels
- Supports negative credit monitoring on Tx channels
- Supports up to 15 simultaneous source (Rx) channels
- Supports Static Transfer Requests Only
- Supports X-Y, MCAN, and AASRC transfer modes (as per configuration)
- Provides per-channel buffering:
  - Provides 8 128-bit word deep data FIFO for each destination channel
  - Provides 8 128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to and from DMSS endpoints and remote peripherals

Copyright © 2026 Texas Instruments Incorporated

11.3.1.1.1.2 PDMA1 - UART Features

The PDMA1 - UART module supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit(s)
  - Each unit supports write bursts up to 64 bytes (on selected channel types)
  - Write Unit 0
    - Provides a 32-bit wide VBUSP write-only initiator interface for peripheral accesses
- Provides 1 memory read access unit(s)
  - Supports 1 outstanding read per interface (VBUSP)
  - Supports read burst up to 64 bytes (on selected channel types)
  - Read Unit 0
    - Provides a 32-bit wide VBUSP read-only initiator interface for peripheral accesses
- Supports up to 7 simultaneous destination (Tx) channels
- Supports negative credit monitoring on Tx channels
- Supports up to 7 simultaneous source (Rx) channels
- Supports Static Transfer Requests Only
- Supports X-Y, MCAN, and AASRC transfer modes (as per configuration)
- Provides per-channel buffering:
  - Provides 8 128-bit word deep data FIFO for each destination channel
  - Provides 8 128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to and from DMSS endpoints and remote peripherals

11.3.1.1.1.3 PDMA2 - McASP Features

The PDMA2 - McASP module supports the following features:

- Implements CPPI 5.0 compliant third-party Unified Transfer Controller (UTC)
- Provides 1 memory write access unit(s)
  - Each unit supports write bursts up to 64 bytes (X-Y FIFO transfer mode only)
  - Write Unit 0
    - Provides a 32-bit wide VBUSP write-only initiator interface for peripheral accesses
- Provides 1 memory read access unit(s)
  - Supports 1 outstanding read per interface (VBUSP)
  - Supports read burst up to 64 bytes (X-Y FIFO transfer mode only)
  - Read Unit 0
    - Provides a 32-bit wide VBUSP read-only initiator interface for peripheral accesses
- Supports up to 3 simultaneous destination (Tx) channels
- Supports up to 3 simultaneous source (Rx) channels
- Supports Static Transfer Requests Only
- Supports X-Y, MCAN, and AASRC transter modes (as per configuration)
- Provides per-channel buffering:
  - Provides 8 128-bit word deep data FIFO for each destination channel
  - Provides 8 128-bit word deep data FIFO for each source channel
- Provides 128-bit wide PSI-L compliant data interface to and from DMSS endpoints and remote peripherals

Note

Some features may not be available. See *Module Integration* for more information.

11.3.1.2 Functional Description - SPI

11.3.1.2.1 Compliance to Standards

The PDMA complies fully to all standards listed in the previous section.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2 Functional Operation

The Peripheral DMA is a simple, low cost implementation of the CPPI 5.0 Unified Transfer Controller. The PDMA module is required to be located close to one or more peripherals (both peripherals and PDMA direct connected to the SCR) which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only the static Transfer Request subset of UTC features which are useful for peripheral type transactions. Multiple source and destination channels are provided within the PDMA which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs time division multiplexing between channels in order to share the underlying DMA hardware. A scheduler is provided to control the ordering and rate at which this multiplexing occurs. A block diagram of the PDMA Controller is shown below.

11.3.1.2.2.1 Submodule Descriptions

11.3.1.2.2.1.1 Scheduler

The Scheduler block is responsible for monitoring the fullness level of the various FIFOs, monitoring input DMA triggering events, maintaining channel state information, and issuing credits to the Tx and Rx DMA units when it is time to perform each low level read or write operation.

11.3.1.2.2.1.2 Tx Per Channel Buffers

The Tx Per Channel Buffers implement channelized FIFOs for each DMA channel. A single logical data FIFO is provided for each destination DMA channel that is used for buffering payload data that has been pushed into the PDMA from the Tx PSI-L interface. The Tx Per Channel Buffers are always written with full Tx PSI-L wide words (except for EOL/EOP data phases) but are read on byte granularity from the Tx DMA units.

11.3.1.2.2.1.3 Tx DMA Unit

The Tx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC destination channels. The Tx DMA unit waits until it is triggered by an incoming dma event and then writes data from the Tx Per Channel Data FIFO associated with the channel to an external memory mapped target via a VBUSP controller interface. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.2.2.1.4 Rx Per Channel Buffers

Rx Per Channel Buffers implement a logical FIFO for each source DMA channel that is used for buffering payload data that has been fetched by the Rx DMA units. The buffers are byte oriented on write so that the data from the Rx DMA units which may not be full words can be packed properly. The buffers are word oriented on read in accordance with the transport mechanism outlined in the PSI-L Interface specification. Each channel in the Rx DMA controller maps directly onto a thread in the Rx PSI-L interface. The Rx Per Channel Buffer block outputs queue fullness information to the scheduler block which it then uses to determine when it should initiate DMA opportunities to backfill the buffers. The Rx Per Channel Buffer will initiate transfers to the remote paired thread whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block will simultaneously monitor the status of all of the threads and will perform a round robin arbitration between the different threads for the use of the Rx PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer will be included in the arbitration.

11.3.1.2.2.1.5 Rx DMA Unit

The Rx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC source channels. The Rx DMA unit waits until it is triggered by an incoming DMA event and then reads data from an external memory mapped source via a VBUSP controller read interface into the Rx Per Channel Data FIFO associated with the channel. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.2.2.2 General Functionality (Applicable to All Functions/Modes)

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.2.1 Operational States

At any given time, the PDMA can be in one of three different states, as shown in Table 11-76.

Table 11-76. Operational States

Operational State	Description
Init	This is the initial state of the machine during and immediately after reset. During this state, all of the RAMs inside the PDMA will be initialized to known values including the ECC redundant parity bits. While in the init state, the DMA will de-assert all ready signals on all applicable target interfaces and will de-assert all request signals on all applicable controller interfaces. The PDMA will automatically transition out of the init state into the idle state when all of the RAM initialization has been completed.
Idle	Once the PDMA leaves the Init state, it enters the Idle state whenever no outstanding transactions are pending on any of the PDMA interfaces (controller or target). The Idle state is generally a transient state and is used by the PDMA to determine when it is appropriate to allow the SoC power management complex to turn off the clock. For clock stop purposes, the module will only report idle if all DMA channels are disabled using the enable bit in PSIL register 2 for each channel.
Active	The PDMA enters the active state as soon as it issues a transaction or receives a transaction on any interface that uses a split protocol (expects a later response for a request). When all transactions have been accounted for (responses have all been either received or sent) the PDMA transitions to the Idle state.

11.3.1.2.2.2.2 Clock Stop

The clock stop interface is a request/acknowledge interface used to coordinate the handshaking of properly stopping the main clock to the IP. Before attempting to perform a clock stop operation, software is required to teardown all active channels (via UDMAP 'real time' registers in the UDMAP, or PSIL register 0x408 in PSIL based peripherals), and after this is complete, also clear the global enable bit for all channels (via PSIL register 0x2 in both the UDMAP and PSIL based peripherals). Attempting a clock stop on the IP without first performing the channel teardowns or clearing of global enable bits will result in IP specific behavior that may be UNDEFINED.

The PDMA will not drive clock stop IDLE or clock stop ACK while the global enable (PSIL register 0x2) is set for any channel.

11.3.1.2.2.2.3 Emulation Control

The emulation control input (susp_emususp) and the per channel emulation control 'free' bit allow DMA operation to be suspended on a per channel basis. When the emulation suspend state is entered, the DMA will stop processing receive and transmit TRs for each channel at the next TR boundary. Any TR currently in reception or transmission will be halted at its next FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. Emulation control is implemented for compatibility with other peripherals. Each source and destination channel can be configured to either honor the suspend signal, or to 'free run' without regard to suspend. This is controlled via the 'free' bit in the channel Enable Register. Note the real time emulation suspend request signal is not supported.

11.3.1.2.2.2.4 Dynamic Clock Gating

The PDMA includes up to two mid level clock gates using ksdw_pm_clkgate(). One is on the main clock of the PDMA while the other is on the interal ecc_aggr (when RAM is used). These clock gates turn off the clock to their corresponding IP under idle conditions, and then turn the clock back on when work is to be done. Detecting work is done via internal state, or activity on the PSIL or DMA triggers. The DMA triggers are thus always monitored, even when the clocking to the rest of the IP is disabled.

The dynamic clock gating can be disabled through an external port called pwr_disable_nogate. No additional signaling is needed to support the dynamic clock gate, although the PDMA supports the 'early wake' pins on PSIL and VBUSP.

11.3.1.2.2.3 Events and Flow Control

The following sections describe the simplified mechanism that is provided on the Peripheral DMA for triggering.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.3.1 Channel Triggering

Channels must be triggered in order for them to perform work. A local event input bus is provided on the peripheral DMA and each bit in the input bus corresponds to the trigger for the channel with the same channel index as the bit index in the bus (bit 0 triggers channel 0, bit 1 triggers channel 1, etc.). The event inputs for XY mode are triggered either via pulse, or by a clock synchronous rising edge. The counting mode is a design time configuration parameter. In MCAN mode, the inputs are expected to be single cycle pulses, synchronous with the PDMA clock.

- Event signals that are used in pulse mode must be sourced by the same clock as the PDMA.
- Event signals that are used in edge mode must be sourced by pseudo-synchronous sources (and be a slower clock multiple) to the PDMA clock, and they must remain high for at least 1 'slow' clock cycle.

The PDMA provides a 2 bit counter per event input to accommodate startup latency in the channel. (In AASRC mode, the signals are not counted, but latched whenever asserted.)

11.3.1.2.2.3.2 Completion Events

All transfers on PDMA are split TR in that they have a UDMA-P half and a (static) PDMA half. Completion events are designed to be triggered from the UDMA-P half of the split TR.

11.3.1.2.2.3.3 Channel Types

Each channel in the Peripheral DMA is configured at design time to be either standard X-Y FIFO mode or MCAN channel. The channel type is configured at design time, and can not be changed by software.

11.3.1.2.2.3.3.1 X-Y FIFO Mode

Most peripherals which are serviced by the Peripheral DMA follow the pattern of transferring X bytes from a fixed, non-changing address in a loop Y times for each triggering event that is received. The X parameter is typically 1-16 bytes and the Y parameter can be any integer up to 2048. If a peripheral can be serviced by this basic functionality then it is recommended to use this mode for the peripheral.

11.3.1.2.2.3.3.2 MCAN Mode

The MCANSS subsystem was found to require a few minor differences than standard X-Y mode. First, buffer ownership on transmit is different in that the DMA will start out owning the TX buffer space (instead of waiting for an initial DMA event from the MCAN). There is also a requirement to perform a write of a fixed value to a fixed address after each DMA event is serviced to pass ownership of a buffer back to the MCAN. Finally, MCAN packet fragmentation requirement to place an 8 byte header on each 64 byte chunk that is transmitted and to remove the header from each 64 byte chunk (other than the first block) that is received.

11.3.1.2.2.3.3.3 AASRC Mode

The AASRC mode of the PDMA is similar to the X-Y mode in that X specifies the sample width, any Y specifies the number of FIFO samples to transfer per DMA request, but in addition to this, a programmable FIFO list is supplied that allows a single PDMA channel to transfer data to multiple FIFOs. The FIFO list specifies which FIFOS to access, and in which order to access them. Both AASRC 'stream' and 'group' modes are supported in terms of DMA signalling and the FIFO memory map. When in AASRC mode, the entire PDMA operates exclusively in this mode. AASRC channels to not share the same PDMA as do XY and MCAN channels.

The following addition details apply to the PDMA in AASRC mode:

Main Features:

- Up to 16 independent RX channels and 16 TX channels
  - Each channel represents a different data stream to either the UDMA-P-P or a McASP PDMA
- Each channel can be configured to read or write any of the AASRC FIFOs in any order
- Packing support for 1, 2, 3, and 4 byte samples
  - Can directly talk to a McASP using any McASP data format
  - Packing sample size is fixed for a given channel
- Supports both Group and Stream AASRC signaling and FIFO memory maps

Copyright © 2026 Texas Instruments Incorporated

- There is a single stream/group mode setting for each PDMA channel

Additional details:

- Each channel has a 'mask' of which DMA requests must fire to activate the channel
  - In stream mode, the mask specifies the all the FIFO DMA requests that must fire
  - In group mode, the mask specifies the one group DMA request that must fire
- Each channel specifies an ordered list of how FIFOs should be accessed
  - All channels use a common list, specifying a range of entry indices within that list. This allows each channel to use a varying number of list elements.
  - The order list can be processed multiple times per DMA request. The number of samples to be transferred per request must match the configured threshold for the DMA request in the AASRC.
- All FIFOs are specified by index only
  - In the TX ordering table, the indices are assumed to be TX FIFOs
  - In the RX ordering table, the indices are assumed to be RX FIFOs
  - If a DMA channel is in 'Group Mode', the PDMA assumes the indices referenced by that channel represent Group Mode FIFOs; otherwise, it assumes Stream Mode FIFOs.
- The PDMA has configured base addressed and inter-FIFO spans such that is can always convert from a FIFO index to the proper Stream or Group mode FIFO address.

11.3.1.2.2.4 Transmit Operation

11.3.1.2.2.4.1 Destination (Tx) Channel Allocation

A total of 15 destination channels are provided within the DMA for concurrent transfers from Tx per channel buffers to the various attached peripherals. Each Tx channel requires a single PSI-L thread. The Tx channels are allocated as in the following table.

Table 11-77. Tx Channel Allocation

Tx DMA Channel	Function	Channel Type	Trigger Mode	Data FIFO Address	Strobe MMR Address	Control FIFO Address
8000	SPI 0 Tx Ch 0	XY	edge	000020100138	000000000000	000000000000
8001	SPI 0 Tx Ch 1	XY	edge	00002010014C	000000000000	000000000000
8002	SPI 0 Tx Ch 2	XY	edge	000020100160	000000000000	000000000000
8003	SPI 0 Tx Ch 3	XY	edge	000020100174	000000000000	000000000000
8004	SPI 1 Tx Ch 0	XY	edge	000020110138	000000000000	000000000000
8005	SPI 1 Tx Ch 1	XY	edge	00002011014C	000000000000	000000000000
8006	SPI 1 Tx Ch 2	XY	edge	000020110160	000000000000	000000000000
8007	SPI 1 Tx Ch 3	XY	edge	000020110174	000000000000	000000000000
8008	SPI 2 Tx Ch 0	XY	edge	000020120138	000000000000	000000000000
8009	SPI 2 Tx Ch 1	XY	edge	00002012014C	000000000000	000000000000
800a	SPI 2 Tx Ch 2	XY	edge	000020120160	000000000000	000000000000
800b	SPI 2 Tx Ch 3	XY	edge	000020120174	000000000000	000000000000
800c	MCAN 0 Tx Ch 0	XY	pulse	000002708000	0000027010D0	000000000000
800d	MCAN 0 Tx Ch 1	XY	pulse	000002708048	0000027010D0	000000000000
800e	MCAN 0 Tx Ch 2	XY	pulse	000002708090	0000027010D0	000000000000

11.3.1.2.2.4.2 Destination (Tx) Channel Out of Band Signals

The following table shows how PSIL signals are handled for destination channels.

Table 11-78. Tx Channel Out of Band Signals

PSI-L Signal	XY/AASRC Mode	MCAN Mode
sop	ignored	ignored
eop	ignored	closes current MCAN packet

Copyright © 2026 Texas Instruments Incorporated

Table 11-78. Tx Channel Out of Band Signals (continued)

PSI-L Signal	XY/AASRC Mode	MCAN Mode
sol	ignored	ignored
eol	ignored	ignored
drop	ignored	ignored
pkt_error	ignored	ignored
tdown	reflect in status, teardown when data marker reached	reflect in status, teardown when data marker reached

11.3.1.2.2.4.3 Destination Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will need to first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.2.2.4.3.1 PSI-L Destination Thread Pairing Registers

11.3.1.2.2.4.3.1.1 Enable Register (PSIL Address 0x002)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the TX channel is disabled. When disabled, the channel discards all ingress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.2.2.4.3.1.2 Local Capabilities Register (PSIL Address 0x040)

Bits	Field	Reset	Description
31:29	-	0x0	Reserved.
28:24	LocalThreadWidth	0x2	Read only element width for the local thread used for pairing purposes. This field is encoded as follows: 0 = 4 bytes, 1 = 8 bytes, 2 = 16 bytes.
23:8	-	0x0	Reserved.
7:0	LocalCreditCnt	0x7	Read only local thread free entry count used for pairing purposes.

11.3.1.2.2.4.3.2 PSI-L Destination Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host will pair each destination PDMA channel with (typically) a corresponding source channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the UDMA-P will send all their data to the destination channel in the PDMA and the destination channel in the PDMA will never see any data other than data from the source channel in the UDMA-P.

11.3.1.2.2.4.3.3 PSI-L Destination Thread Realtime Enable/Count Registers

11.3.1.2.2.4.3.3.1 RT Enable Register (PSIL Address 0x408)

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the TX channel is disabled. When disabled, it discards all held data. It clears DMA event counts and ignores all future DMA events from the peripheral. It maintains data exchange with the UDMA-P so that credit handshake is not disrupted. A 'hard teardown' can be performed by directly clearing this bit. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.
30	tdown	0x0	When set, the channel will commence a TX channel teardown procedure. To perform a TX teardown, the teardown bit should be set in the UDMA-P and it will automatically propagate to this register bit with the normal flow of peripheral data. Once the channel is fully stopped and ready to be reused (including returning all credits), the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume.
28	flush	0x0	When set, causes all TX channel data to be discarded instead of being written to the peripheral. It essentially allows the TX engine to 'free run' without DMA requests from the peripheral (it will also override 'pause'). This bit should be set only when a channel fails to complete its teardown procedure normally, because a peripheral is no longer functioning or because data flow was halted on a boundary that is not compatible with the static TR configuration.
27:3	-	0x0	Reserved.
2	error	0x0	When set, the channel has encountered a PSIL protocol violation. The error bit can only be set by hardware, and can only be cleared by software. Once this bit is set, the channel should be fully reset and re-initialized via the PSIL pairing registers.
1	idle	0x0	This is a read-only bit that signifies that the disabled channel is also idle. This bit is read only and can only become set if the enable bit in the RT enable register is cleared.
0	free	0x0	When cleared, the channel honors the emudbg suspend signal. When set, the channel will free run, regardless of the value of emudbg suspend.

11.3.1.2.2.4.3.3.2 Destination Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been written to the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

11.3.1.2.2.4.3.4 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will send configuration transactions across PSI-L and will set up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies somewhat.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.4.3.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
Z	0x401	23:0	Not used

11.3.1.2.2.4.3.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the MCAN burst description for more information.
Acc32	0x400	30	Not used
X	0x400	26:24	Not used
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be written to an MCAN TX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. The PDMA will break up the source packet into fragments of this buffer size, copying the 8 byte MCAN header for the initial fragment, and then skipping it for each additional fragment and thus reusing the header from the first fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
Z	0x401	23:0	Not used

11.3.1.2.2.4.3.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer to each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
Z	0x401	23:0	Not used

11.3.1.2.2.4.3.4.4 AASRC TxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this TX channel.

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each TX FIFO. The flags corresponding to all TX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

11.3.1.2.2.4.3.4.5 AASRC TxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	TX FIFO Index for slot 7
27:24	Entry6	6	TX FIFO Index for slot 6
23:20	Entry5	5	TX FIFO Index for slot 5
19:16	Entry4	4	TX FIFO Index for slot 4

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
15:12	Entry3	3	TX FIFO Index for slot 3
11:8	Entry2	2	TX FIFO Index for slot 2
7:4	Entry1	1	TX FIFO Index for slot 1
3:0	Entry0	0	TX FIFO Index for slot 0

11.3.1.2.2.4.3.4.6 AASRC TxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	TX FIFO Index for slot 15
27:24	Entry14	14	TX FIFO Index for slot 14
23:20	Entry13	13	TX FIFO Index for slot 13
19:16	Entry12	12	TX FIFO Index for slot 12
15:12	Entry11	11	TX FIFO Index for slot 11
11:8	Entry10	10	TX FIFO Index for slot 10
7:4	Entry9	9	TX FIFO Index for slot 9
3:0	Entry8	8	TX FIFO Index for slot 8

11.3.1.2.2.4.3.5 PSI-L Destination Thread Enables

PSI-L destination threads must first be enabled in order to accept data. Threads are enabled or disabled by setting or clearing the enable bit in the PSI-L pairing registers for the thread. When a thread is disabled, it must drop any data phases which are sent but properly return the credits for the data phases which are dropped.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of write transactions.

11.3.1.2.2.4.4 Data Transfer

Packet transmission is accomplished within the PDMA by unpacking and moving data from the Tx Per Channel FIFOs which were filled via the Transmit PSI-L Interface to specified memory mapped address ranges via the VBUSP controller interfaces. On the Tx side of the PDMA, these transfers are always writes. Each write transfer which is performed by the Tx DMA Unit is to a destination address that is hardcoded in the channel at design time and of a size as specified in the static Transfer Request.

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during transmit is dependent on the channel type. The following sections describe what will happen for the different channel types.

11.3.1.2.2.4.4.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter writes of 'X' parameter bytes to the data_address specified in the tchan_info parameter for the channel. Each write that the DMA performs will be a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The write transfers that are performed will be accomplished as quickly as possible given availability of data in the Tx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.2.2.4.4.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

Copyright © 2026 Texas Instruments Incorporated

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts will be sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA will 'gap' the byte enables on writes as needed

11.3.1.2.2.4.4.2 MCAN Mode Channel

The PDMA channel will remain idle until data is received from the UDMA-P over the PSIL interface. At this time, the PDMA will immediately start copying packet bytes into the MCAN TX buffer corresponding the PDMA/MCAN channel. The number of bytes copied is smaller of the remaining bytes in the packet, or the value of the 'Y' parameter. Once a TX buffer has been filled, the PDMA will transfer ownership of the buffer to MCAN by performing an MCAN register write. The PDMA will then wait to regain ownership of the TX buffer by waiting for the corresponding MCAN TX event. At this time, the PDMA will continue with refilling the TX buffer for the next packet fragment. On subsequent fills (until the end of packet is reached), the PDMA will skip over the 8 byte MCAN header, leaving the original contents from the initial fragment copy in place.

11.3.1.2.2.4.4.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

11.3.1.2.2.4.4.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's TxFifoConfig register. This register holds 3 basic pieces of information:

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the TxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Once the channel activates, it will start reading FIFO index values from the TxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs written for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width written to the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The write transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.4.5 Transmit PSI-L Interface Transactions

The PDMA will accept data across the Tx PSI-L Interface in accordance with the physical handshaking protocol as defined in the PSI-L Interface specification. The value sources for the output pins on the PSI-L interface are described in the following table.

Pin	Output Value / Source From PDMA
strm_i_link	This signal is asserted high to indicate the PSI-L link is valid.
strm_i_ready	This signal is asserted high anytime there is space in the PSI-L ingress fifo.
strm_i_sreq	This signal is asserted high anytime there is credit return data waiting in the PSI-L status fifo.
strm_i_sthreadid	This is set to the local destination thread that is returning credit.
strm_i_sccnt	This is set to the number of credits being returned, or 0 on a TX teardown acknowledgement.

11.3.1.2.2.4.6 Tx Pause

The Host initiates a channel pause by setting the pause bit in the RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.2.2.4.7 Tx Teardown

The Host initiates a channel teardown by setting the tdown bit in the UDMA-P channel that is paired with the PDMA. The UDMA-P communicates the teardown state through the PSI-L data channel, to ensure that the teardown is not seen by the PDMA until all the previous UDMA-P data for the channel has been flushed. At this time, the teardown state will be reflected in the PSI-L RT Enable register of the PDMA. Note that a non-synchronized teardown can also be initiated by directly clearing the enable bit in the PSI-L RT Enable register of the PDMA.

Once all data has been flushed from the PDMA to the peripheral, the enable state of the PDMA channel will be cleared in the PSI-L RT enable register, but the teardown bit will remain high. A teardown completion indication is sent back across the status pins of the PSI-L in the form of a credit response with sccnt=0. Upon completion, no further packet processing will occur until the Host re-configures the channel. If the channel fails to teardown because the peripheral has stopped responding, or if the UDMA-P transmission stops on a data boundary that is not compatible with the static TR configuration, the flush bit in the RT enable register can be set to guarantee that all data can be properly flushed from the internal pipe.

11.3.1.2.2.4.8 Tx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully, even with flush enabled. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L Enable register. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.2.2.4.9 Tx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For transmit, they are defined as in the following table.

Name	PSIL Addr	Field	Description
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to write to the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next write offset to use when writing to the CAN TX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.

Copyright © 2026 Texas Instruments Incorporated

Name	PSIL Addr	Field	Description
Flush	0x403	30	When set, the PDMA is processing in a flushing state, where it runs without waiting for DMA requests and without writing data to the peripheral. It is only operating its internal state machine to allow internal data pipes to drain properly.
Pause	0x403	29	When set, the PDMA waiting in a paused state. This bit will clear when data starts flowing again from the UDMA-P.
Data	0x403	28	When set, there is a non-zero amount of data still waiting to be written to the peripheral.
XData	0x403	27	When set, there is enough data still waiting to be written to the peripheral to start servicing a peripheral DMA event.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.

11.3.1.2.2.5 Receive Operation

11.3.1.2.2.5.1 Source (Rx) Channel Allocation

A total of 15 source channels are provided within the DMA for concurrent transfers from the various attached peripherals into the Rx per channel buffers and on to the PSI-L Rx Interface. Each Rx channel requires a single PSI-L thread. The Rx channels are allocated as in Table 11-79.

Table 11-79. Rx Channel Allocation

Rx DMA Channel	Function	Channel Type	Trigger Type	Data FIFO Address	Strobe MMR Address	Control FIFO Address
0	SPI 0 Rx Ch 0	XY	edge	00002010013C	000000000000	000000000000
1	SPI 0 Rx Ch 1	XY	edge	000020100150	000000000000	000000000000
2	SPI 0 Rx Ch 2	XY	edge	000020100164	000000000000	000000000000
3	SPI 0 Rx Ch 3	XY	edge	000020100178	000000000000	000000000000
4	SPI 1 Rx Ch 0	XY	edge	00002011013C	000000000000	000000000000
5	SPI 1 Rx Ch 1	XY	edge	000020110150	000000000000	000000000000
6	SPI 1 Rx Ch 2	XY	edge	000020110164	000000000000	000000000000
7	SPI 1 Rx Ch 3	XY	edge	000020110178	000000000000	000000000000
8	SPI 2 Rx Ch 0	XY	edge	00002012013C	000000000000	000000000000
9	SPI 2 Rx Ch 1	XY	edge	000020120150	000000000000	000000000000
10	SPI 2 Rx Ch 2	XY	edge	000020120164	000000000000	000000000000
11	SPI 2 Rx Ch 3	XY	edge	000020120178	000000000000	000000000000
12	MCAN 0 Rx Ch 0	XY	pulse	000002708900	000002701098	000000000000
13	MCAN 0 Rx Ch 1	XY	pulse	000002708990	000002701098	000000000000
14	MCAN 0 Rx Ch 2	XY	pulse	000002708A20	000002701098	000000000000

11.3.1.2.2.5.2 Source Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will need to first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.2.2.5.2.1 PSI-L Source Thread Pairing Registers

11.3.1.2.2.5.2.1.1 Peer Thread ID Register (PSIL Address 0x000)

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31:29	ThreadPriority	0x0	This field is hard coded to 0x0 and is not writable
28:24	PeerThreadWidth	0x0	Datapath width of paired peer destination thread. This field is encoded as follows: 0 = 32-bit, 1 = 64-bit, 2 = 128 bit, 3 = 256 bit, 4 = 512 bit
23:16	-	0x0	Reserved.
15:0	Peer ThreadID	0x0	Thread ID to which all non-Transfer Response, non-Config messages from this thread will be sent.

11.3.1.2.2.5.2.1.2 Peer Credit Register (PSIL Address 0x001)

Bits	Field	Reset	Description
31:8	-	0x0	Reserved.
7:0	CreditCnt	0x0	Free entries in destination thread. Legal range is 0 - 128 in number of elements.

11.3.1.2.2.5.2.1.3 Enable Register (PSIL Address 0x002)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all credit returns, and will not generate egress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.2.2.5.2.2 PSI-L Source Thread Realtime Enable/Count Registers

11.3.1.2.2.5.2.2.1 RT Enable Register (PSIL Address 0x408)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all DMA events from the peripheral. It maintains proper data transfers with the UDMA-P such that the credit handshake is not disrupted. However, unlike teardown, it does not close out the current open packet. Thus the teardown bit should always be used to disable an RX channel instead of manually clearing this bit. Failing to use teardown may result in stale data remaining in the internal RX FIFO, which would also prevent the channel from going idle without a channel reset. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
30	tdown	0x0	When set, the channel will commence a RX channel teardown procedure. It will stop on the next FIFO boundary. It then clears the DMA event count and ignores all future DMA events from the peripheral. After stopping peripheral reads, the PDMA sends a teardown message to the UDMA-P that also closes the current packet with an EOP. If no packet is open at the time of the teardown, the message also includes SOP. The EOP teardown message may or may not contain final packet data. Once the channel teardown is complete and ready to be reused, the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume. Pause will not stop teardown from completing.
28:2	-	0x0	Reserved.
1	idle	0x0	This is a read-only bit that signifies that the Paused or Disabled channel is also idle. This bit is read only and can only become set if pause is set or enable is cleared.
0	free	0x0	Free Run: 0 = channel honors the emudbg suspend signal, 1 = channel will free run, regardless of the value of emudbg suspend.

11.3.1.2.2.5.2.2.2 Source Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been read from the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

11.3.1.2.2.5.2.3 PSI-L Source Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host will pair each source PDMA channel with (typically) a corresponding destination channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the PDMA will send all their data to the source channel in the UDMA-P and the destination channel in the UDMA-P will never see any data other than data from the source channel in the PDMA.

11.3.1.2.2.5.2.4 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will send configuration transactions across PSI-L and will set up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies.

11.3.1.2.2.5.2.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information.

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full FIFO operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.2.2.5.2.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the MCAN burst description for more information.
Acc32	0x400	30	Not used
X	0x400	26:24	Not used
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be read from an MCAN RX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	Buffer Count. This field specifies how many MCAN RX buffers should be read before closing the CPPI packet with an 'EOP' indication. When this count is greater than 1, multiple MCAN RX buffers will be read into a single CPPI packet buffer. The 8 byte MCAN header will be skipped on subsequent MCAN buffer reads. Setting this field to NULL will suppress all packet delineation, and should be avoided.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.5.2.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each read which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer from each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full DMA request operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.2.2.5.2.4.4 AASRC RxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this RX channel.

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each RX FIFO. The flags corresponding to all RX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

11.3.1.2.2.5.2.4.5 AASRC RxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	RX FIFO Index for slot 7
27:24	Entry6	6	RX FIFO Index for slot 6
23:20	Entry5	5	RX FIFO Index for slot 5
19:16	Entry4	4	RX FIFO Index for slot 4
15:12	Entry3	3	RX FIFO Index for slot 3
11:8	Entry2	2	RX FIFO Index for slot 2
7:4	Entry1	1	RX FIFO Index for slot 1
3:0	Entry0	0	RX FIFO Index for slot 0

11.3.1.2.2.5.2.4.6 AASRC RxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	RX FIFO Index for slot 15
27:24	Entry14	14	RX FIFO Index for slot 14
23:20	Entry13	13	RX FIFO Index for slot 13
19:16	Entry12	12	RX FIFO Index for slot 12
15:12	Entry11	11	RX FIFO Index for slot 11
11:8	Entry10	10	RX FIFO Index for slot 10
7:4	Entry9	9	RX FIFO Index for slot 9
3:0	Entry8	8	RX FIFO Index for slot 8

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.5.2.5 PSI-L Source Thread Enables

PSI-L source threads must be enabled in order to process peripheral DMA requests and send data. When disabled, DMA requests are ignored.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of read transactions.

11.3.1.2.2.5.3 Data Transfer

Packet reception is accomplished within the PDMA by moving data from structures that are located in memory accessible via the VBUSP Memory Interface(s) onto the Receive PSI-L Interface. On the Rx side of the PDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx Per Channel Buffer for that channel. At a later time, the data is moved from the Rx Per Channel FIFO to a remote peer DMA entity via the Rx PSI-L interface

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during receive is dependent on the channel type. The following sections describe what will happen for the different channel types.

11.3.1.2.2.5.3.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter reads of 'X' parameter bytes from the data_address specified in the tchan_info parameter for the channel. Each read that the DMA performs will be a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the Rx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same read unit.

11.3.1.2.2.5.3.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts will be sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA will 'gap' the byte enables on writes as needed

11.3.1.2.2.5.3.2 MCAN Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. The DMA event pins are mapped from the CAN filter event bits. The event bits determine the channel and one of two CAN buffers to use for the RX operation. This allows the RX buffers on the CAN to be configured in a ping/pong fashion, preventing the loss of CAN packet data. When an event is received, the PDMA will start copying bytes out of the corresponding CAN buffer. The number of bytes copied is equal to the value of the 'Y' parameter in the channel configuration. Once all data has been copied from the buffer, the PDMA transfers ownership of the buffer back to the CAN by writing a CAN register. It then waits for another RX DMA event. The PDMA can copy multiple CAN RX buffers to the same CPPI packet. It will not close out the CPPI packet until it has copied out a number of RX buffers equal to the 'Z' parameter in the channel configuration. On subsequent RX buffer copies, the MCAN will skip the first 8 bytes of the RX buffer, which is the MCAN packet header.

Copyright © 2026 Texas Instruments Incorporated

The mapping of MCAN filter events to MCAN channels and buffer is important for properly configuring the DMA. The mapping is defined as follows.

CAN FE2	CAN FE1	CAN FE0	Description
0	0	0	Not used
0	0	1	Not used
0	1	0	PDMA CAN channel offset 0, RX buffer 0
0	1	1	PDMA CAN channel offset 0, RX buffer 1
1	0	0	PDMA CAN channel offset 1, RX buffer 2
1	0	1	PDMA CAN channel offset 1, RX buffer 3
1	1	0	PDMA CAN channel offset 2, RX buffer 4
1	1	1	PDMA CAN channel offset 2, RX buffer 5

By using two filter entries in the CAN, software can setup a ping/pong buffer for a particular RX packet ID. For example the following two filter entries will setup a ping/pong using RX buffers 0 and 1 for packet ID 5 on CAN RX channel 0:

Filter Entry	Packet Match ID	Filter Event Bits
0	5	0x2
1	5	0x3

11.3.1.2.2.5.3.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

11.3.1.2.2.5.3.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's RxFifoConfig register. This register holds 3 basic pieces of information:

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the RxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Once the channel activates, it will start reading FIFO index values from the RxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs read for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width read from the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.2.2.5.4 Receive PSI-L Interface Transactions

When an entire word has been packed into the Rx Per Channel Buffer for a given channel, a one word data phase transfer will be initiated for that channel/thread on the Rx PSI-L interface and the data will be popped from the Rx Per Channel FIFO. The value sources for the output pins on the PSI-L interface are described in the following table.

Pin	Output Value / Source From PDMA
strm_o_req	This signal is asserted when the PDMA determines that there is at least 1 word of data in it's TP-CC output FIFO
strm_o_sthread_id	Set equal to the TP-CC DMA channel number which is on the interface for this cycle
strm_o_dthread_id	Set equal to the target thread ID value given in the PSI-L pairing configuration registers for this DMA channel
strm_o_data_type	Will be set to the proper PSI-L data type. The PDMA can send config response, PSI info word 0, and PSI data word.
strm_o_wnum	Set equal to the packing lane for the current word within a contiguous transfer. This value will start at 0 for each new TR and will increment for each subsequent data phase in the TR.
strm_o_lastw	This signal will be asserted coincident with eop.
strm_o_xcnt	The xcnt will be equal to the data path width in bytes.
strm_o_worden	The worden will be modulated to indicate which 32-bit words are valid within each control type data phase.
strm_o_data	The data will be packed/left justified to comply to the big endian data ordering as required in the PSI-L I/F specification.
strm_o_sop	This signal is asserted for 1 data phase at the beginning of each new TR or data transfer packet.
strm_o_eop	This signal is asserted for 1 data phase at the end of each TR or data transfer packet.
strm_o_sol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the start of a new set of transactions designated by the count set in the Z field of PSI-L register 0x401.
strm_o_eol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the end of a set of transactions designated by the count set in the Z field of PSI-L register 0x401.
strm_o_priv	Set to zero.
strm_o_privid	Set to zero.
strm_o_virtid	Set to zero.
strm_o_secure	Set to zero.
strm_o_interest	Set to zero.
strm_o_sready	Always asserted as the PDMA is always able to accept credit returns.

11.3.1.2.2.5.5 Rx Pause

The Host initiates a channel pause by setting the pause bit in RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.2.2.5.6 Rx Teardown

The Host initiates a RX channel teardown by setting the tdown bit in the RT enable register for the target RX channel. The PDMA communicates the teardown state to the UDMA-P through the PSI-L data channel, to ensure that the teardown is not seen by the UDMA-P until all the previous PDMA data for the channel has been flushed.

The PDMA will not stop reading peripheral data until it reaches a FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. It will always attempt to complete the 'Y' count for the current event being processed. Upon reaching a stopping point, the PDMA will then clear the enable bit in the pairing register,

Copyright © 2026 Texas Instruments Incorporated

however the teardown bit will remain set. No further packet processing will occur until the Host re-configures the channel. The teardown process will propagate to the UDMA-P and its final status can be checked there.

11.3.1.2.2.5.7 Rx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L enable register. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.2.2.5.8 Rx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For receive, they are defined as follows.

Name	PSIL Addr	Field	Description
Z*	0x402	31:16	This field holds the lower 12 bits of the current Z count for legacy pursposes. See register 0x40F below for the full width version of Z.
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to be read from the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next read offset to use when read to the CAN RX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.
Tdown	0x403	30	When set, the PDMA is processing a teardown operation. This bit is set simultaneously with the teardown bit in the source (RX) RT enable register. This bit will clear when the teardown is complete, regardless as to if the teardown bit in the pairing register is cleared or not. The teardown will propagate to the UDMA-P and its full completion status can be checked there.
Pause	0x403	29	When set, the PDMA is stopped in a paused state. This bit will clear if the channel is un-paused or disabled.
Space	0x403	28	When set, there is a non-zero amount of internal FIFO space available to hold new read data.
XSpace	0x403	27	When set, there is a enough internal FIFO space available to start servicing a peripheral DMA event.
Buffer	0x403	26	This is the current RX buffer (0/1) for the current MCAN receive operation.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.
Z	0x40F	31:0	This field holds the full width value of the current Z count. In X-Y FIFO mode, this field holds the 1 based FIFO count of the FIFO being currently read, or the number of FIFO completions when the current operation completes. In MCAN mode, this field holds the zero based buffer index of the buffer currently being read, or the number of previously completed buffers.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3 Functional Description - UART

11.3.1.3.1 Compliance to Standards

The PDMA complies fully to all standards listed in the previous section.

11.3.1.3.2 Functional Operation

The Peripheral DMA is a simple, low cost implementation of the CPPI 5.0 Unified Transfer Controller. The PDMA module is required to be located close to one or more peripherals (both peripherals and PDMA direct connected to the SCR) which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only the static Transfer Request subset of UTC features which are useful for peripheral type transactions. Multiple source and destination channels are provided within the PDMA which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs time division multiplexing between channels in order to share the underlying DMA hardware. A scheduler is provided to control the ordering and rate at which this multiplexing occurs. A block diagram of the PDMA Controller is shown below.

11.3.1.3.2.1 Submodule Descriptions

11.3.1.3.2.1.1 Scheduler

The Scheduler block is responsible for monitoring the fullness level of the various FIFOs, monitoring input DMA triggering events, maintaining channel state information, and issuing credits to the Tx and Rx DMA units when it is time to perform each low level read or write operation.

11.3.1.3.2.1.2 Tx Per Channel Buffers

The Tx Per Channel Buffers implement channelized FIFOs for each DMA channel. A single logical data FIFO is provided for each destination DMA channel that is used for buffering payload data that has been pushed into the PDMA from the Tx PSI-L interface. The Tx Per Channel Buffers are always written with full Tx PSI-L wide words (except for EOL/EOP data phases) but are read on byte granularity from the Tx DMA units.

11.3.1.3.2.1.3 Tx DMA Unit

The Tx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC destination channels. The Tx DMA unit waits until it is triggered by an incoming dma event and then writes data from the Tx Per Channel Data FIFO associated with the channel to an external memory mapped target via a VBUSP controller interface. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.3.2.1.4 Rx Per Channel Buffers

Rx Per Channel Buffers implement a logical FIFO for each source DMA channel that is used for buffering payload data that has been fetched by the Rx DMA units. The buffers are byte oriented on write so that the data from the Rx DMA units which may not be full words can be packed properly. The buffers are word oriented on read in accordance with the transport mechanism outlined in the PSI-L Interface specification. Each channel in the Rx DMA controller maps directly onto a thread in the Rx PSI-L interface. The Rx Per Channel Buffer block outputs queue fullness information to the scheduler block which it then uses to determine when it should initiate DMA opportunities to backfill the buffers. The Rx Per Channel Buffer will initiate transfers to the remote paired thread whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block will simultaneously monitor the status of all of the threads and will perform a round robin arbitration between the different threads for the use of the Rx PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer will be included in the arbitration.

11.3.1.3.2.1.5 Rx DMA Unit

The Rx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC source channels. The Rx DMA unit waits until it is triggered by an incoming DMA event and then reads data from an external memory mapped source via a VBUSP controller read interface into the Rx Per Channel Data

Copyright © 2026 Texas Instruments Incorporated

FIFO associated with the channel. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.3.2.2 General Functionality (Applicable to All Functions/Modes)

11.3.1.3.2.2.1 Operational States

At any given time, the PDMA can be in one of three different states, as shown in Table 11-80.

Table 11-80. Operational States

Operational State	Description
Init	This is the initial state of the machine during and immediately after reset. During this state, all of the RAMs inside the PDMA will be initialized to known values including the ECC redundant parity bits. While in the init state, the DMA will de-assert all ready signals on all applicable target interfaces and will de-assert all request signals on all applicable controller interfaces. The PDMA will automatically transition out of the init state into the idle state when all of the RAM initialization has been completed.
Idle	Once the PDMA leaves the Init state, it enters the Idle state whenever no outstanding transactions are pending on any of the PDMA interfaces (controller or target). The Idle state is generally a transient state and is used by the PDMA to determine when it is appropriate to allow the SoC power management complex to turn off the clock. For clock stop purposes, the module will only report idle if all DMA channels are disabled using the enable bit in PSIL register 2 for each channel.
Active	The PDMA enters the active state as soon as it issues a transaction or receives a transaction on any interface that uses a split protocol (expects a later response for a request). When all transactions have been accounted for (responses have all been either received or sent) the PDMA transitions to the Idle state.

11.3.1.3.2.2.2 Clock Stop

The clock stop interface is a request/acknowledge interface used to coordinate the handshaking of properly stopping the main clock to the IP. Before attempting to perform a clock stop operation, software is required to teardown all active channels (via UDMAP 'real time' registers in the UDMAP, or PSIL register 0x408 in PSIL based peripherals), and after this is complete, also clear the global enable bit for all channels (via PSIL register 0x2 in both the UDMAP and PSIL based peripherals). Attempting a clock stop on the IP without first performing the channel teardowns or clearing of global enable bits will result in IP specific behavior that may be UNDEFINED.

The PDMA will not drive clock stop IDLE or clock stop ACK while the global enable (PSIL register 0x2) is set for any channel.

11.3.1.3.2.2.3 Emulation Control

The emulation control input (susp_emususp) and the per channel emulation control 'free' bit allow DMA operation to be suspended on a per channel basis. When the emulation suspend state is entered, the DMA will stop processing receive and transmit TRs for each channel at the next TR boundary. Any TR currently in reception or transmission will be halted at its next FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. Emulation control is implemented for compatibility with other peripherals. Each source and destination channel can be configured to either honor the suspend signal, or to 'free run' without regard to suspend. This is controlled via the 'free' bit in the channel Enable Register. Note the real time emulation suspend request signal is not supported.

11.3.1.3.2.2.4 Dynamic Clock Gating

The PDMA includes up to two mid level clock gates using ksdw_pm_clkgate(). One is on the main clock of the PDMA while the other is on the interal ecc_aggr (when RAM is used). These clock gates turn off the clock to their corresponding IP under idle conditions, and then turn the clock back on when work is to be done. Detecting work is done via internal state, or activity on the PSIL or DMA triggers. The DMA triggers are thus always monitored, even when the clocking to the rest of the IP is disabled.

Copyright © 2026 Texas Instruments Incorporated

The dynamic clock gating can be disabled through an external port called pwr_disable_nogate. No additional signaling is needed to support the dynamic clock gate, although the PDMA supports the 'early wake' pins on PSIL and VBUSP.

11.3.1.3.2.3 Events and Flow Control

The following sections describe the simplified mechanism that is provided on the Peripheral DMA for triggering.

11.3.1.3.2.3.1 Channel Triggering

Channels must be triggered in order for them to perform work. A local event input bus is provided on the peripheral DMA and each bit in the input bus corresponds to the trigger for the channel with the same channel index as the bit index in the bus (bit 0 triggers channel 0, bit 1 triggers channel 1, etc.). The event inputs for XY mode are triggered either via pulse, or by a clock synchronous rising edge. The counting mode is a design time configuration parameter. In MCAN mode, the inputs are expected to be single cycle pulses, synchronous with the PDMA clock.

- Event signals that are used in pulse mode must be sourced by the same clock as the PDMA.
- Event signals that are used in edge mode must be sourced by pseudo-synchronous sources (and be a slower clock multiple) to the PDMA clock, and they must remain high for at least 1 'slow' clock cycle.

The PDMA provides a 2 bit counter per event input to accommodate startup latency in the channel. (In AASRC mode, the signals are not counted, but latched whenever asserted.)

11.3.1.3.2.3.2 Completion Events

All transfers on PDMA are split TR in that they have a UDMA-P half and a (static) PDMA half. Completion events are designed to be triggered from the UDMA-P half of the split TR.

11.3.1.3.2.3.3 Channel Types

Each channel in the Peripheral DMA is configured at design time to be either standard X-Y FIFO mode or MCAN channel. The channel type is configured at design time, and can not be changed by software.

11.3.1.3.2.3.3.1 X-Y FIFO Mode

Most peripherals which are serviced by the Peripheral DMA follow the pattern of transferring X bytes from a fixed, non-changing address in a loop Y times for each triggering event that is received. The X parameter is typically 1-16 bytes and the Y parameter can be any integer up to 2048. If a peripheral can be serviced by this basic functionality then it is recommended to use this mode for the peripheral.

11.3.1.3.2.3.3.2 MCAN Mode

The MCANSS subsystem was found to require a few minor differences than standard X-Y mode. First, buffer ownership on transmit is different in that the DMA will start out owning the TX buffer space (instead of waiting for an initial DMA event from the MCAN). There is also a requirement to perform a write of a fixed value to a fixed address after each DMA event is serviced to pass ownership of a buffer back to the MCAN. Finally, MCAN packet fragmentation requirement to place an 8 byte header on each 64 byte chunk that is transmitted and to remove the header from each 64 byte chunk (other than the first block) that is received.

11.3.1.3.2.3.3.3 AASRC Mode

The AASRC mode of the PDMA is similar to the X-Y mode in that X specifies the sample width, any Y specifies the number of FIFO samples to transfer per DMA request, but in addition to this, a programmable FIFO list is supplied that allows a single PDMA channel to transfer data to multiple FIFOs. The FIFO list specifies which FIFOS to access, and in which order to access them. Both AASRC 'stream' and 'group' modes are supported in terms of DMA signalling and the FIFO memory map. When in AASRC mode, the entire PDMA operates exclusively in this mode. AASRC channels to not share the same PDMA as do XY and MCAN channels.

The following addition details apply to the PDMA in AASRC mode:

Main Features:

- Up to 16 independent RX channels and 16 TX channels

Copyright © 2026 Texas Instruments Incorporated

  - Each channel represents a different data stream to either the UDMA-P-P or a McASP PDMA
- Each channel can be configured to read or write any of the AASRC FIFOs in any order
- Packing support for 1, 2, 3, and 4 byte samples
  - Can directly talk to a McASP using any McASP data format
  - Packing sample size is fixed for a given channel
- Supports both Group and Stream AASRC signaling and FIFO memory maps
  - There is a single stream/group mode setting for each PDMA channel

Additional details:

- Each channel has a 'mask' of which DMA requests must fire to activate the channel
  - In stream mode, the mask specifies the all the FIFO DMA requests that must fire
  - In group mode, the mask specifies the one group DMA request that must fire
- Each channel specifies an ordered list of how FIFOs should be accessed
  - All channels use a common list, specifying a range of entry indices within that list. This allows each channel to use a varying number of list elements.
  - The order list can be processed multiple times per DMA request. The number of samples to be transferred per request must match the configured threshold for the DMA request in the AASRC.
- All FIFOs are specified by index only
  - In the TX ordering table, the indices are assumed to be TX FIFOs
  - In the RX ordering table, the indices are assumed to be RX FIFOs
  - If a DMA channel is in 'Group Mode', the PDMA assumes the indices referenced by that channel represent Group Mode FIFOs; otherwise, it assumes Stream Mode FIFOs.
- The PDMA has configured base addressed and inter-FIFO spans such that is can always convert from a FIFO index to the proper Stream or Group mode FIFO address.

11.3.1.3.2.4 Transmit Operation

11.3.1.3.2.4.1 Destination (Tx) Channel Allocation

A total of 7 destination channels are provided within the DMA for concurrent transfers from Tx per channel buffers to the various attached peripherals. Each Tx channel requires a single PSI-L thread. The Tx channels are allocated as shown in Table 11-81.

Table 11-81. Tx Channel Allocation

Tx DMA Channel	Function	Channel Type	Trigger Mode	Data FIFO Address	Strobe MMR Address	Control FIFO Address
8000	USART 0 Tx Ch 0	XY	edge	000002800000	000000000000	000000000000
8001	USART 1 Tx Ch 0	XY	edge	000002810000	000000000000	000000000000
8002	USART 2 Tx Ch 0	XY	edge	000002820000	000000000000	000000000000
8003	USART 3 Tx Ch 0	XY	edge	000002830000	000000000000	000000000000
8004	USART 4 Tx Ch 0	XY	edge	000002840000	000000000000	000000000000
8005	USART 5 Tx Ch 0	XY	edge	000002850000	000000000000	000000000000
8006	USART 6 Tx Ch 0	XY	edge	000002860000	000000000000	000000000000

11.3.1.3.2.4.2 Destination (Tx) Channel Out of Band Signals

The following table shows how PSIL signals are handled for destination channels.

Table 11-82. Tx Channel Out of Band Signals

PSI-L Signal	XY/AASRC Mode	MCAN Mode
sop	ignored	ignored
eop	ignored	closes current MCAN packet
sol	ignored	ignored
eol	ignored	ignored

Copyright © 2026 Texas Instruments Incorporated

Table 11-82. Tx Channel Out of Band Signals (continued)

PSI-L Signal	XY/AASRC Mode	MCAN Mode
drop	ignored	ignored
pkt_error	ignored	ignored
tdown	reflect in status, teardown when data marker reached	reflect in status, teardown when data marker reached

11.3.1.3.2.4.3 Destination Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will need to first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.3.2.4.3.1 PSI-L Destination Thread Pairing Registers

11.3.1.3.2.4.3.1.1 Enable Register (PSIL Address 0x002)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all credit returns, and will not generate egress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.3.2.4.3.1.2 Local Capabilities Register (PSIL Address 0x040)

Bits	Field	Reset	Description
31:29	-	0x0	Reserved.
28:24	LocalThreadWidth	0x2	Read only element width for the local thread used for pairing purposes. This field is encoded as follows: 0 = 4 bytes, 1 = 8 bytes, 2 = 16 bytes.
23:8	-	0x0	Reserved.
7:0	LocalCreditCnt	0x7	Read only local thread free entry count used for pairing purposes.

11.3.1.3.2.4.3.2 PSI-L Destination Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host will pair each destination PDMA channel with (typically) a corresponding source channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the UDMA-P will send all their data to the destination channel in the PDMA and the destination channel in the PDMA will never see any data other than data from the source channel in the UDMA-P.

11.3.1.3.2.4.3.3 PSI-L Destination Thread Realtime Enable/Count Registers

11.3.1.3.2.4.3.3.1 RT Enable Register (PSIL Address 0x408)

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the TX channel is disabled. When disabled, it discards all held data. It clears DMA event counts and ignores all future DMA events from the peripheral. It maintains data exchange with the UDMA-P so that credit handshake is not disrupted. A 'hard teardown' can be performed by directly clearing this bit. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.
30	tdown	0x0	When set, the channel will commence a TX channel teardown procedure. To perform a TX teardown, the teardown bit should be set in the UDMA-P and it will automatically propagate to this register bit with the normal flow of peripheral data. Once the channel is fully stopped and ready to be reused (including returning all credits), the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume.
28	flush	0x0	When set, causes all TX channel data to be discarded instead of being written to the peripheral. It essentially allows the TX engine to 'free run' without DMA requests from the peripheral (it will also override 'pause'). This bit should be set only when a channel fails to complete its teardown procedure normally, because a peripheral is no longer functioning or because data flow was halted on a boundary that is not compatible with the static TR configuration.
27:3	-	0x0	Reserved.
2	error	0x0	When set, the channel has encountered a PSIL protocol violation. The error bit can only be set by hardware, and can only be cleared by software. Once this bit is set, the channel should be fully reset and re-initialized via the PSIL pairing registers.
1	idle	0x0	This is a read-only bit that signifies that the disabled channel is also idle. This bit is read only and can only become set if the enable bit in the RT enable register is cleared.
0	free	0x0	When cleared, the channel honors the emudbg suspend signal. When set, the channel will free run, regardless of the value of emudbg suspend.

11.3.1.3.2.4.3.3.2 Destination Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been written to the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

11.3.1.3.2.4.3.4 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will send configuration transactions across PSI-L and will set up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3.2.4.3.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
Z	0x401	23:0	Not used

11.3.1.3.2.4.3.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the MCAN burst description for more information.
Acc32	0x400	30	Not used
X	0x400	26:24	Not used
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be written to an MCAN TX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. The PDMA will break up the source packet into fragments of this buffer size, copying the 8 byte MCAN header for the initial fragment, and then skipping it for each additional fragment and thus reusing the header from the first fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
Z	0x401	23:0	Not used

11.3.1.3.2.4.3.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer to each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
Z	0x401	23:0	Not used

11.3.1.3.2.4.3.4.4 AASRC TxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this TX channel.

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each TX FIFO. The flags corresponding to all TX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

11.3.1.3.2.4.3.4.5 AASRC TxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	TX FIFO Index for slot 7
27:24	Entry6	6	TX FIFO Index for slot 6
23:20	Entry5	5	TX FIFO Index for slot 5
19:16	Entry4	4	TX FIFO Index for slot 4

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
15:12	Entry3	3	TX FIFO Index for slot 3
11:8	Entry2	2	TX FIFO Index for slot 2
7:4	Entry1	1	TX FIFO Index for slot 1
3:0	Entry0	0	TX FIFO Index for slot 0

11.3.1.3.2.4.3.4.6 AASRC TxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	TX FIFO Index for slot 15
27:24	Entry14	14	TX FIFO Index for slot 14
23:20	Entry13	13	TX FIFO Index for slot 13
19:16	Entry12	12	TX FIFO Index for slot 12
15:12	Entry11	11	TX FIFO Index for slot 11
11:8	Entry10	10	TX FIFO Index for slot 10
7:4	Entry9	9	TX FIFO Index for slot 9
3:0	Entry8	8	TX FIFO Index for slot 8

11.3.1.3.2.4.3.5 PSI-L Destination Thread Enables

PSI-L destination threads must first be enabled in order to accept data. Threads are enabled or disabled by setting or clearing the enable bit in the PSI-L pairing registers for the thread. When a thread is disabled, it must drop any data phases which are sent but properly return the credits for the data phases which are dropped.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of write transactions.

11.3.1.3.2.4.4 Data Transfer

Packet reception is accomplished within the PDMA by moving data from structures that are located in memory accessible via the VBUSP Memory Interface(s) onto the Receive PSI-L Interface. On the Rx side of the PDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx Per Channel Buffer for that channel. At a later time, the data is moved from the Rx Per Channel FIFO to a remote peer DMA entity via the Rx PSI-L interface

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during receive is dependent on the channel type. The following sections describe what will happen for the different channel types.

11.3.1.3.2.4.4.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter reads of 'X' parameter bytes from the data_address specified in the tchan_info parameter for the channel. Each read that the DMA performs will be a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the Rx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same read unit.

11.3.1.3.2.4.4.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

Copyright © 2026 Texas Instruments Incorporated

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts will be sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA will 'gap' the byte enables on writes as needed

11.3.1.3.2.4.4.2 MCAN Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. The DMA event pins are mapped from the CAN filter event bits. The event bits determine the channel and one of two CAN buffers to use for the RX operation. This allows the RX buffers on the CAN to be configured in a ping/pong fashion, preventing the loss of CAN packet data. When an event is received, the PDMA will start copying bytes out of the corresponding CAN buffer. The number of bytes copied is equal to the value of the 'Y' parameter in the channel configuration. Once all data has been copied from the buffer, the PDMA transfers ownership of the buffer back to the CAN by writing a CAN register. It then waits for another RX DMA event. The PDMA can copy multiple CAN RX buffers to the same CPPI packet. It will not close out the CPPI packet until it has copied out a number of RX buffers equal to the 'Z' parameter in the channel configuration. On subsequent RX buffer copies, the MCAN will skip the first 8 bytes of the RX buffer, which is the MCAN packet header.

The mapping of MCAN filter events to MCAN channels and buffer is important for properly configuring the DMA. The mapping is defined as follows.

CAN FE2	CAN FE1	CAN FE0	Description
0	0	0	Not used
0	0	1	Not used
0	1	0	PDMA CAN channel offset 0, RX buffer 0
0	1	1	PDMA CAN channel offset 0, RX buffer 1
1	0	0	PDMA CAN channel offset 1, RX buffer 2
1	0	1	PDMA CAN channel offset 1, RX buffer 3
1	1	0	PDMA CAN channel offset 2, RX buffer 4
1	1	1	PDMA CAN channel offset 2, RX buffer 5

By using two filter entries in the CAN, software can setup a ping/pong buffer for a particular RX packet ID. For example the following two filter entries will setup a ping/pong using RX buffers 0 and 1 for packet ID 5 on CAN RX channel 0:

Filter Entry	Packet Match ID	Filter Event Bits
0	5	0x2
1	5	0x3

11.3.1.3.2.4.4.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

11.3.1.3.2.4.4.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's RxFifoConfig register. This register holds 3 basic pieces of information:

Copyright © 2026 Texas Instruments Incorporated

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the RxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Once the channel activates, it will start reading FIFO index values from the RxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs read for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width read from the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.3.2.4.5 Transmit PSI-L Interface Transactions

The PDMA will accept data across the Tx PSI-L Interface in accordance with the physical handshaking protocol as defined in the PSI-L Interface specification. The value sources for the output pins on the PSI-L interface are described in the following table.

Pin	Output Value / Source From PDMA
strm_i_link	This signal is asserted high to indicate the PSI-L link is valid.
strm_i_ready	This signal is asserted high anytime there is space in the PSI-L ingress fifo.
strm_i_sreq	This signal is asserted high anytime there is credit return data waiting in the PSI-L status fifo.
strm_i_sthreadid	This is set to the local destination thread that is returning credit.
strm_i_sccnt	This is set to the number of credits being returned, or 0 on a TX teardown acknowledgement.

11.3.1.3.2.4.6 Tx Pause

The Host initiates a channel pause by setting the pause bit in the RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.3.2.4.7 Tx Teardown

The Host initiates a channel teardown by setting the tdown bit in the UDMA-P channel that is paired with the PDMA. The UDMA-P communicates the teardown state through the PSI-L data channel, to ensure that the teardown is not seen by the PDMA until all the previous UDMA-P data for the channel has been flushed. At this time, the teardown state will be reflected in the PSI-L RT Enable register of the PDMA. Note that a non-synchronized teardown can also be initiated by directly clearing the enable bit in the PSI-L RT Enable register of the PDMA.

Once all data has been flushed from the PDMA to the peripheral, the enable state of the PDMA channel will be cleared in the PSI-L RT enable register, but the teardown bit will remain high. A teardown completion indication is sent back across the status pins of the PSI-L in the form of a credit response with sccnt=0. Upon completion, no further packet processing will occur until the Host re-configures the channel. If the channel fails to teardown because the peripheral has stopped responding, or if the UDMA-P transmission stops on a data boundary that is not compatible with the static TR configuration, the flush bit in the RT enable register can be set to guarantee that all data can be properly flushed from the internal pipe.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3.2.4.8 Tx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully, even with flush enabled. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L Enable register. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.3.2.4.9 Tx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For transmit, they are defined as in the following table.

Name	PSIL Addr	Field	Description
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to write to the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next write offset to use when writing to the CAN TX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.
Flush	0x403	30	When set, the PDMA is processing in a flushing state, where it runs without waiting for DMA requests and without writing data to the peripheral. It is only operating its internal state machine to allow internal data pipes to drain properly.
Pause	0x403	29	When set, the PDMA waiting in a paused state. This bit will clear when data starts flowing again from the UDMA-P.
Data	0x403	28	When set, there is a non-zero amount of data still waiting to be written to the peripheral.
XData	0x403	27	When set, there is enough data still waiting to be written to the peripheral to start servicing a peripheral DMA event.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.

11.3.1.3.2.5 Receive Operation

11.3.1.3.2.5.1 Source (Rx) Channel Allocation

A total of 7 source channels are provided within the DMA for concurrent transfers from the various attached peripherals into the Rx per channel buffers and on to the PSI-L Rx Interface. Each Rx channel requires a single PSI-L thread. The Rx channels are allocated as shown in Table 11-83.

Table 11-83. Rx Channel Allocation

Rx DMA Channel	Function	Channel Type	Trigger Type	Data FIFO Address	Strobe MMR Address	Control FIFO Address
0	USART 0 Rx Ch 0	XY	edge	000002800000	000000000000	000000000000
1	USART 1 Rx Ch 0	XY	edge	000002810000	000000000000	000000000000
2	USART 2 Rx Ch 0	XY	edge	000002820000	000000000000	000000000000
3	USART 3 Rx Ch 0	XY	edge	000002830000	000000000000	000000000000
4	USART 4 Rx Ch 0	XY	edge	000002840000	000000000000	000000000000
5	USART 5 Rx Ch 0	XY	edge	000002850000	000000000000	000000000000
6	USART 6 Rx Ch 0	XY	edge	000002860000	000000000000	000000000000

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3.2.5.2 Source Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will need to first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.3.2.5.2.1 PSI-L Source Thread Pairing Registers

11.3.1.3.2.5.2.1.1 Peer Thread ID Register (PSIL Address 0x000)

Bits	Field	Reset	Description
31:29	ThreadPriority	0x0	This field is hard coded to 0x0 and is not writable
28:24	PeerThreadWidth	0x0	Datapath width of paired peer destination thread. This field is encoded as follows: 0 = 32-bit, 1 = 64-bit, 2 = 128 bit, 3 = 256 bit, 4 = 512 bit
23:16	-	0x0	Reserved.
15:0	Peer ThreadID	0x0	Thread ID to which all non-Transfer Response, non-Config messages from this thread will be sent.

11.3.1.3.2.5.2.1.2 Peer Credit Register (PSIL Address 0x001)

Bits	Field	Reset	Description
31:8	-	0x0	Reserved.
7:0	CreditCnt	0x0	Free entries in destination thread. Legal range is 0 - 128 in number of elements.

11.3.1.3.2.5.2.1.3 Enable Register (PSIL Address 0x002)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all credit returns, and will not generate egress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.3.2.5.2.2 PSI-L Source Thread Realtime Enable/Count Registers

11.3.1.3.2.5.2.2.1 RT Enable Register (PSIL Address 0x408)

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all DMA events from the peripheral. It maintains proper data transfers with the UDMA-P such that the credit handshake is not disrupted. However, unlike teardown, it does not close out the current open packet. Thus the teardown bit should always be used to disable an RX channel instead of manually clearing this bit. Failing to use teardown may result in stale data remaining in the internal RX FIFO, which would also prevent the channel from going idle without a channel reset. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.
30	tdown	0x0	When set, the channel will commence a RX channel teardown procedure. It will stop on the next FIFO boundary. It then clears the DMA event count and ignores all future DMA events from the peripheral. After stopping peripheral reads, the PDMA sends a teardown message to the UDMA-P that also closes the current packet with an EOP. If no packet is open at the time of the teardown, the message also includes SOP. The EOP teardown message may or may not contain final packet data. Once the channel teardown is complete and ready to be reused, the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume. Pause will not stop teardown from completing.
28:2	-	0x0	Reserved.
1	idle	0x0	This is a read-only bit that signifies that the Paused or Disabled channel is also idle. This bit is read only and can only become set if pause is set or enable is cleared.
0	free	0x0	Free Run: 0 = channel honors the emudbg suspend signal, 1 = channel will free run, regardless of the value of emudbg suspend.

11.3.1.3.2.5.2.2.2 Source Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been read from the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3.2.5.2.3 PSI-L Source Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host will pair each source PDMA channel with (typically) a corresponding destination channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the PDMA will send all their data to the source channel in the UDMA-P and the destination channel in the UDMA-P will never see any data other than data from the source channel in the PDMA.

11.3.1.3.2.5.2.4 Static Transfer Request Setup

After reset, all channels in the PDMA will be idle and waiting for work to be assigned to them. In order to initiate PDMA operation, the Host will send configuration transactions across PSI-L and will set up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies.

11.3.1.3.2.5.2.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full FIFO operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.3.2.5.2.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the MCAN burst description for more information.
Acc32	0x400	30	Not used
X	0x400	26:24	Not used

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be read from an MCAN RX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	Buffer Count. This field specifies how many MCAN RX buffers should be read before closing the CPPI packet with an 'EOP' indication. When this count is greater than 1, multiple MCAN RX buffers will be read into a single CPPI packet buffer. The 8 byte MCAN header will be skipped on subsequent MCAN buffer reads. Setting this field to NULL will suppress all packet delineation, and should be avoided.

11.3.1.3.2.5.2.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each read which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer from each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full DMA request operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.3.2.5.2.4.4 AASRC RxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this RX channel.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each RX FIFO. The flags corresponding to all RX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

11.3.1.3.2.5.2.4.5 AASRC RxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	RX FIFO Index for slot 7
27:24	Entry6	6	RX FIFO Index for slot 6
23:20	Entry5	5	RX FIFO Index for slot 5
19:16	Entry4	4	RX FIFO Index for slot 4
15:12	Entry3	3	RX FIFO Index for slot 3
11:8	Entry2	2	RX FIFO Index for slot 2
7:4	Entry1	1	RX FIFO Index for slot 1
3:0	Entry0	0	RX FIFO Index for slot 0

Copyright © 2026 Texas Instruments Incorporated

11.3.1.3.2.5.2.4.6 AASRC RxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	RX FIFO Index for slot 15
27:24	Entry14	14	RX FIFO Index for slot 14
23:20	Entry13	13	RX FIFO Index for slot 13
19:16	Entry12	12	RX FIFO Index for slot 12
15:12	Entry11	11	RX FIFO Index for slot 11
11:8	Entry10	10	RX FIFO Index for slot 10
7:4	Entry9	9	RX FIFO Index for slot 9
3:0	Entry8	8	RX FIFO Index for slot 8

11.3.1.3.2.5.2.5 PSI-L Source Thread Enables

PSI-L source threads must be enabled in order to process peripheral DMA requests and send data. When disabled, DMA requests are ignored.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of read transactions.

11.3.1.3.2.5.3 Data Transfer

Packet reception is accomplished within the PDMA by moving data from structures that are located in memory accessible via the VBUSP Memory Interface(s) onto the Receive PSI-L Interface. On the Rx side of the PDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx Per Channel Buffer for that channel. At a later time, the data is moved from the Rx Per Channel FIFO to a remote peer DMA entity via the Rx PSI-L interface

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during receive is dependent on the channel type. The following sections describe what will happen for the different channel types.

11.3.1.3.2.5.3.1 X-Y FIFO Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA will sequentially issue a total of 'Y' parameter reads of 'X' parameter bytes from the data_address specified in the tchan_info parameter for the channel. Each read that the DMA performs will be a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the Rx channelized FIFO and given the arbitration that may occur as a result of other channels also using the same read unit.

11.3.1.3.2.5.3.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts will be sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase

Copyright © 2026 Texas Instruments Incorporated

  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA will 'gap' the byte enables on writes as needed

11.3.1.3.2.5.3.2 MCAN Mode Channel

The PDMA channel will remain idle until a pulse is detected on the associated input DMA request event pin. The DMA event pins are mapped from the CAN filter event bits. The event bits determine the channel and one of two CAN buffers to use for the RX operation. This allows the RX buffers on the CAN to be configured in a ping/pong fashion, preventing the loss of CAN packet data. When an event is received, the PDMA will start copying bytes out of the corresponding CAN buffer. The number of bytes copied is equal to the value of the 'Y' parameter in the channel configuration. Once all data has been copied from the buffer, the PDMA transfers ownership of the buffer back to the CAN by writing a CAN register. It then waits for another RX DMA event. The PDMA can copy multiple CAN RX buffers to the same CPPI packet. It will not close out the CPPI packet until it has copied out a number of RX buffers equal to the 'Z' parameter in the channel configuration. On subsequent RX buffer copies, the MCAN will skip the first 8 bytes of the RX buffer, which is the MCAN packet header.

The mapping of MCAN filter events to MCAN channels and buffer is important for properly configuring the DMA. The mapping is defined as follows.

CAN FE2	CAN FE1	CAN FE0	Description
0	0	0	Not used
0	0	1	Not used
0	1	0	PDMA CAN channel offset 0, RX buffer 0
0	1	1	PDMA CAN channel offset 0, RX buffer 1
1	0	0	PDMA CAN channel offset 1, RX buffer 2
1	0	1	PDMA CAN channel offset 1, RX buffer 3
1	1	0	PDMA CAN channel offset 2, RX buffer 4
1	1	1	PDMA CAN channel offset 2, RX buffer 5

By using two filter entries in the CAN, software can setup a ping/pong buffer for a particular RX packet ID. For example the following two filter entries will setup a ping/pong using RX buffers 0 and 1 for packet ID 5 on CAN RX channel 0:

Filter Entry	Packet Match ID	Filter Event Bits
0	5	0x2
1	5	0x3

11.3.1.3.2.5.3.2.1 MCAN Burst Mode

Since MCAN buffers are stored in linear memory, the burst mode for MCAN is a simple linear burst across the transfer window. The max burst size is set to the 72 byte size of the MCAN buffer. This will allow a full MCAN packet to be read out as a single burst.

11.3.1.3.2.5.3.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's RxFifoConfig register. This register holds 3 basic pieces of information:

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel will remain idle until a pulse is detected on ALL associated input DMA request event pins required by the RxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Copyright © 2026 Texas Instruments Incorporated

Once the channel activates, it will start reading FIFO index values from the RxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs read for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width read from the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed the channel will return to an idle state and wait until it is triggered again. The read transfers that are performed will be accomplished as quickly as possible given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.3.2.5.4 Receive PSI-L Interface Transactions

When an entire word has been packed into the Rx Per Channel Buffer for a given channel, a one word data phase transfer will be initiated for that channel/thread on the Rx PSI-L interface and the data will be popped from the Rx Per Channel FIFO. The value sources for the output pins on the PSI-L interface are described in the following table.

Pin	Output Value / Source From PDMA
strm_o_req	This signal is asserted when the PDMA determines that there is at least 1 word of data in it's TP-CC output FIFO
strm_o_sthread_id	Set equal to the TP-CC DMA channel number which is on the interface for this cycle
strm_o_dthread_id	Set equal to the target thread ID value given in the PSI-L pairing configuration registers for this DMA channel
strm_o_data_type	Will be set to the proper PSI-L data type. The PDMA can send config response, PSI info word 0, and PSI data word.
strm_o_wnum	Set equal to the packing lane for the current word within a contiguous transfer. This value will start at 0 for each new TR and will increment for each subsequent data phase in the TR.
strm_o_lastw	This signal will be asserted coincident with eop.
strm_o_xcnt	The xcnt will be equal to the data path width in bytes.
strm_o_worden	The worden will be modulated to indicate which 32-bit words are valid within each control type data phase.
strm_o_data	The data will be packed/left justified to comply to the big endian data ordering as required in the PSI-L I/F specification.
strm_o_sop	This signal is asserted for 1 data phase at the beginning of each new TR or data transfer packet.
strm_o_eop	This signal is asserted for 1 data phase at the end of each TR or data transfer packet.
strm_o_sol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the start of a new set of transactions designated by the count set in the Z field of PSI-L register 0x401.
strm_o_eol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the end of a set of transactions designated by the count set in the Z field of PSI-L register 0x401.
strm_o_priv	Set to zero.
strm_o_privid	Set to zero.
strm_o_virtid	Set to zero.
strm_o_secure	Set to zero.
strm_o_interest	Set to zero.

Copyright © 2026 Texas Instruments Incorporated

Pin	Output Value / Source From PDMA
strm_o_sready	Always asserted as the PDMA is always able to accept credit returns.

11.3.1.3.2.5.5 Rx Pause

The Host initiates a channel pause by setting the pause bit in RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.3.2.5.6 Rx Teardown

The Host initiates a RX channel teardown by setting the tdown bit in the RT enable register for the target RX channel. The PDMA communicates the teardown state to the UDMA-P through the PSI-L data channel, to ensure that the teardown is not seen by the UDMA-P until all the previous PDMA data for the channel has been flushed.

The PDMA will not stop reading peripheral data until it reaches a FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. It will always attempt to complete the 'Y' count for the current event being processed. Upon reaching a stopping point, the PDMA will then clear the enable bit in the pairing register, however the teardown bit will remain set. No further packet processing will occur until the Host re-configures the channel. The teardown process will propagate to the UDMA-P and its final status can be checked there.

11.3.1.3.2.5.7 Rx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L enable register. This will cause a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.3.2.5.8 Rx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For receive, they are defined as follows.

Name	PSIL Addr	Field	Description
Z*	0x402	31:16	This field holds the lower 12 bits of the current Z count for legacy pursposes. See register 0x40F below for the full width version of Z.
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to be read from the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next read offset to use when read to the CAN RX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.
Tdown	0x403	30	When set, the PDMA is processing a teardown operation. This bit is set simultaneously with the teardown bit in the source (RX) RT enable register. This bit will clear when the teardown is complete, regardless as to if the teardown bit in the pairing register is cleared or not. The teardown will propagate to the UDMA-P and its full completion status can be checked there.
Pause	0x403	29	When set, the PDMA is stopped in a paused state. This bit will clear if the channel is un-paused or disabled.
Space	0x403	28	When set, there is a non-zero amount of internal FIFO space available to hold new read data.
XSpace	0x403	27	When set, there is a enough internal FIFO space available to start servicing a peripheral DMA event.

Copyright © 2026 Texas Instruments Incorporated

Name	PSIL Addr	Field	Description
Buffer	0x403	26	This is the current RX buffer (0/1) for the current MCAN receive operation.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.
Z	0x40F	31:0	This field holds the full width value of the current Z count. In X-Y FIFO mode, this field holds the 1 based FIFO count of the FIFO being currently read, or the number of FIFO completions when the current operation completes. In MCAN mode, this field holds the zero based buffer index of the buffer currently being read, or the number of previously completed buffers.

11.3.1.4 Functional Description - McASP

11.3.1.4.1 Compliance to Standards

The PDMA complies fully to all standards listed in the previous section.

11.3.1.4.2 Functional Operation

The Peripheral DMA is a simple, low cost implementation of the CPPI 5.0 Unified Transfer Controller. The PDMA module is required to be located close to one or more peripherals (both peripherals and PDMA direct connected to the SCR) which require an external DMA for data movement and is architected to reduce cost by using VBUSP interfaces and supporting only the static Transfer Request subset of UTC features which are useful for peripheral type transactions. Multiple source and destination channels are provided within the PDMA, which allow multiple simultaneous transfer operations to be ongoing. The DMA controller maintains state information for each of the channels and employs time division multiplexing between channels to share the underlying DMA hardware. A scheduler is provided to control the ordering and rate at which this multiplexing occurs. A block diagram of the PDMA Controller is shown below:

11.3.1.4.2.1 Submodule Descriptions

11.3.1.4.2.1.1 Scheduler

The Scheduler block is responsible for monitoring the fullness level of the various FIFOs, monitoring input DMA triggering events, maintaining channel state information, and issuing credits to the Tx and Rx DMA units when it is time to perform each low level read or write operation.

11.3.1.4.2.1.2 Tx Per Channel Buffers

The Tx Per Channel Buffers implement channelized FIFOs for each DMA channel. A single logical data FIFO is provided for each destination DMA channel that is used for buffering payload data that has been pushed into the PDMA from the Tx PSI-L interface. The Tx Per Channel Buffers are always written with full Tx PSI-L wide words (except for EOL/EOP data phases) but are read on byte granularity from the Tx DMA units.

11.3.1.4.2.1.3 Tx DMA Unit

The Tx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC destination channels. The Tx DMA unit waits until it is triggered by an incoming dma event and then writes data from the Tx Per Channel Data FIFO associated with the channel to an external memory mapped target via a VBUSP controller interface. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.4.2.1.4 Rx Per Channel Buffers

Rx Per Channel Buffers implement a logical FIFO for each source DMA channel that is used for buffering payload data that has been fetched by the Rx DMA units. The buffers are byte oriented on write so that the data

Copyright © 2026 Texas Instruments Incorporated

from the Rx DMA units which may not be full words can be packed properly. The buffers are word oriented on read in accordance with the transport mechanism outlined in the PSI-L Interface specification. Each channel in the Rx DMA controller maps directly onto a thread in the Rx PSI-L interface. The Rx Per Channel Buffer block outputs queue fullness information to the scheduler block, which it then uses to determine when it should initiate DMA opportunities to backfill the buffers. The Rx Per Channel Buffer initiates transfers to the remote paired thread whenever any data is available in each channel buffer and credits are available in the corresponding thread. The block simultaneously monitors the status of all of the threads and performs a round robin arbitration between the different threads for the use of the Rx PSI-L interface. Each thread for which the target is indicating it can accept data and which currently has data available in the channel buffer is included in the arbitration.

11.3.1.4.2.1.5 Rx DMA Unit

The Rx DMA Unit block implements all of the state machine functionality necessary to implement static TR type UTC source channels. The Rx DMA unit waits until it is triggered by an incoming DMA event and then reads data from an external memory mapped source via a VBUSP controller read interface into the Rx Per Channel Data FIFO associated with the channel. The number and width of the writes that are performed is in accordance with the parameters which were programmed via PSI-L into the static TR for the channel.

11.3.1.4.2.2 General Functionality (Applicable to All Functions/Modes)

11.3.1.4.2.2.1 Operational States

At any given time, the PDMA can be in one of three different states, as shown in Table 11-84.

Table 11-84. PDMA Operational States

Operational State	Description
Init	This is the initial state of the machine during and immediately after reset. During this state, all of the RAMs inside the PDMA will be initialized to known values including the ECC redundant parity bits. While in the init state, the DMA will de-assert all ready signals on all applicable target interfaces and will de-assert all request signals on all applicable controller interfaces. The PDMA will automatically transition out of the init state into the idle state when all of the RAM initialization has been completed.
Idle	Once the PDMA leaves the Init state, it enters the Idle state whenever no outstanding transactions are pending on any of the PDMA interfaces (controller or target). The Idle state is generally a transient state and is used by the PDMA to determine when it is appropriate to allow the SoC power management complex to turn off the clock. For clock stop purposes, the module will only report idle if all DMA channels are disabled using the enable bit in PSIL register 2 for each channel.
Active	The PDMA enters the active state as soon as it issues a transaction or receives a transaction on any interface that uses a split protocol (expects a later response for a request). When all transactions have been accounted for (responses have all been either received or sent) the PDMA transitions to the Idle state.

11.3.1.4.2.2.2 Clock Stop

The clock stop interface is used to instruct the PDMA to stop issuing commands on its external interfaces. Note that the PDMA will report a clock stop idle if all its DMA channels have been previously disabled by software using the enable bit in PSIL register 2 for each channel.

11.3.1.4.2.2.3 Emulation Control

The emulation control input (susp_emususp) and the per channel emulation control 'free' bit allow DMA operation to be suspended on a per channel basis. When the emulation suspend state is entered, the DMA will stop processing receive and transmit TRs for each channel at the next TR boundary. Any TR currently in reception or transmission will be halted at its next FIFO boundary as configured through the 'X' and 'Y' parameters in the static TR. Emulation control is implemented for compatibility with other peripherals. Each source and destination channel can be configured to either honor the suspend signal, or to 'free run' without regard to suspend. This is controlled via the 'free' bit in the channel Enable Register. Note the real time emulation suspend request signal is not supported.

11.3.1.4.2.3 Events and Flow Control

The following sections describe the simplified mechanism provided on the Peripheral DMA for triggering.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.4.2.3.1 Channel Triggering

Channels must be triggered for them to perform work. A local event input bus is provided on the peripheral DMA and each bit in the input bus corresponds to the trigger for the channel with the same channel index as the bit index in the bus (bit 0 triggers channel 0, bit 1 triggers channel 1, and so forth). The event inputs for XY mode are triggered either via pulse, or by a clock synchronous rising edge. The counting mode is a design time configuration parameter. In MCAN mode, the inputs are expected to be single cycle pulses, synchronous with the PDMA clock.

- Event signals that are used in pulse mode must be sourced by the same clock as the PDMA.
- Event signals that are used in edge mode must be sourced by pseudo-synchronous sources (and be a slower clock multiple) to the PDMA clock, and they must remain high for at least 1 'slow' clock cycle.

The PDMA provides a 2-bit counter per event input to accommodate startup latency in the channel. (In AASRC mode, the signals are not counted, but latched whenever asserted.)

11.3.1.4.2.3.2 Completion Events

All transfers on PDMA are split TR in that they have a UDMA-P half and a (static) PDMA half. Completion events are designed to be triggered from the UDMA-P half of the split TR.

11.3.1.4.2.3.3 Channel Types

Each channel in the Peripheral DMA is configured at design time to be either standard X-Y FIFO mode or MCAN channel. The channel type is configured at design time, and can not be changed by software.

11.3.1.4.2.3.3.1 X-Y FIFO Mode

Most peripherals which are serviced by the Peripheral DMA follow the pattern of transferring X bytes from a fixed, non-changing address in a loop Y times for each triggering event that is received. The X parameter is typically 1-16 bytes and the Y parameter can be any integer up to 2048. If a peripheral can be serviced by this basic functionality then it is recommended to use this mode for the peripheral.

11.3.1.4.2.3.3.2 MCAN Mode

The MCANSS subsystem was found to require a few minor differences than standard X-Y mode. First, buffer ownership on transmit is different in that the DMA starts out owning the TX buffer space (instead of waiting for an initial DMA event from the MCAN). There is also a requirement to perform a write of a fixed value to a fixed address after each DMA event is serviced to pass ownership of a buffer back to the MCAN. Finally, MCAN packet fragmentation requirement to place an 8-byte header on each 64-byte chunk that is transmitted and to remove the header from each 64-byte chunk (other than the first block) that is received.

11.3.1.4.2.3.3.3 AASRC Mode

The AASRC mode of the PDMA is similar to the X-Y mode in that X specifies the sample width, any Y specifies the number of FIFO samples to transfer per DMA request, but in addition to this, a programmable FIFO list is supplied that allows a single PDMA channel to transfer data to multiple FIFOs. The FIFO list specifies which FIFOS to access, and in which order to access them. Both AASRC 'stream' and 'group' modes are supported in terms of DMA signalling and the FIFO memory map. When in AASRC mode, the entire PDMA operates exclusively in this mode. AASRC channels to not share the same PDMA as do XY and MCAN channels.

The following additional details apply to the PDMA in AASRC mode.

Main Features:

- Up to 16 independent RX channels and 16 TX channels
  - Each channel represents a different data stream to either the UDMA-P-P or a McASP PDMA
- Each channel can be configured to read or write any of the AASRC FIFOs in any order
- Packing support for 1, 2, 3, and 4 byte samples
  - Can directly talk to a McASP using any McASP data format
  - Packing sample size is fixed for a given channel
- Supports both Group and Stream AASRC signaling and FIFO memory maps

Copyright © 2026 Texas Instruments Incorporated

  - There is a single stream/group mode setting for each PDMA channel

Additional details:

- Each channel has a 'mask' of which DMA requests must fire to activate the channel
  - In stream mode, the mask specifies the all the FIFO DMA requests that must fire
  - In group mode, the mask specifies the one group DMA request that must fire
- Each channel specifies an ordered list of how FIFOs should be accessed
  - All channels use a common list, specifying a range of entry indices within that list. This lets each channel use a varying number of list elements.
  - The order list can be processed multiple times per DMA request. The number of samples to be transferred per request must match the configured threshold for the DMA request in the AASRC.
- All FIFOs are specified by index only
  - In the TX ordering table, the indices are assumed to be TX FIFOs
  - In the RX ordering table, the indices are assumed to be RX FIFOs
  - If a DMA channel is in 'Group Mode', the PDMA assumes the indices referenced by that channel represent Group Mode FIFOs; otherwise, it assumes Stream Mode FIFOs.
- The PDMA has configured base addressed and inter-FIFO spans such that is can always convert from a FIFO index to the proper Stream or Group mode FIFO address

11.3.1.4.2.4 Transmit Operation

11.3.1.4.2.4.1 Destination (Tx) Channel Allocation

A total of 3 destination channels are provided within the DMA for concurrent transfers from Tx per channel buffers to the various attached peripherals. Each Tx channel requires a single PSI-L thread. The Tx channels are allocated as follows:

Tx DMA Channel	Function	Channel Type	Trigger Mode	Data FIFO Address	Strobe MMR Address	Control FIFO Address
8000	McASP 0 Tx Ch 0	XY	edge	000002B08000	000000000000	000000000000
8001	McASP 1 Tx Ch 0	XY	edge	000002B18000	000000000000	000000000000
8002	McASP 2 Tx Ch 0	XY	edge	000002B28000	000000000000	000000000000

11.3.1.4.2.4.2 Destination (Tx) Channel Out of Band Signals

The following table shows how PSIL signals are handled for destination channels.

PSI-L Signal	XY/AASRC Mode	MCAN Mode
sop	ignored	ignored
eop	ignored	closes current MCAN packet
sol	ignored	ignored
eol	ignored	ignored
drop	ignored	ignored
pkt_error	ignored	ignored
tdown	reflect in status, teardown when data marker reached	reflect in status, teardown when data marker reached

11.3.1.4.2.4.3 Destination Channel Initialization

After reset, all threads/channels in the PDMA is idle and waiting for work to be assigned to them. To initiate PDMA operation, the Host must first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.4.2.4.3.1 PSI-L Destination Thread Pairing Registers

11.3.1.4.2.4.3.1.1 Enable Register (PSIL Address 0x002)

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the TX channel is disabled. When disabled, the channel discards all ingress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.4.2.4.3.1.2 Peer Credit Register (PSIL Address 0x040)

Bits	Field	Reset	Description
31:29	-	0x0	Reserved.
28:24	LocalThreadWidth	0x2	Read only element width for the local thread used for pairing purposes. This field is encoded as follows: 0 = 4 bytes, 1 = 8 bytes, 2 = 16 bytes.
23:8	-	0x0	Reserved.
7:0	LocalCreditCnt	0x8	Read only local thread free entry count used for pairing purposes.

11.3.1.4.2.4.3.2 PSI-L Destination Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host pairs each destination PDMA channel with (typically) a corresponding source channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread, and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the UDMA-P send all their data to the destination channel in the PDMA, and the destination channel in the PDMA never sees any data other than data from the source channel in the UDMA-P.

11.3.1.4.2.4.3.3 PSI-L Destination Thread Realtime Enable/Count Registers

11.3.1.4.2.4.3.3.1 RT Enable Register (PSIL Address 0x408)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the TX channel is disabled. When disabled, it discards all held data. It clears DMA event counts and ignores all future DMA events from the peripheral. It maintains data exchange with the UDMA-P so that credit handshake is not disrupted. A 'hard teardown' can be performed by directly clearing this bit. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.
30	tdown	0x0	When set, the channel will commence a TX channel teardown procedure. To perform a TX teardown, the teardown bit should be set in the UDMA-P and it will automatically propagate to this register bit with the normal flow of peripheral data. Once the channel is fully stopped and ready to be reused (including returning all credits), the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
28	flush	0x0	When set, causes all TX channel data to be discarded instead of being written to the peripheral. It essentially allows the TX engine to 'free run' without DMA requests from the peripheral (it will also override 'pause'). This bit should be set only when a channel fails to complete its teardown procedure normally, because a peripheral is no longer functioning or because data flow was halted on a boundary that is not compatible with the static TR configuration.
27:3	-	0x0	Reserved.
2	error	0x0	When set, the channel has encountered a PSIL protocol violation. The error bit can only be set by hardware, and can only be cleared by software. Once this bit is set, the channel should be fully reset and re-initialized via the PSIL pairing registers.
1	idle	0x0	This is a read-only bit that signifies that the disabled channel is also idle. This bit is read only and can only become set if the enable bit in the RT enable register is cleared.
0	free	0x0	When cleared, the channel honors the emudbg suspend signal. When set, the channel will free run, regardless of the value of emudbg suspend.

11.3.1.4.2.4.3.3.2 Destination Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been written to the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

11.3.1.4.2.4.3.4 Static Transfer Request Setup

After reset, all channels in the PDMA are idle and waiting for work to be assigned to them. To initiate PDMA operation, the Host sends configuration transactions across PSI-L and sets up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies somewhat.

11.3.1.4.2.4.3.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
Z	0x401	23:0	Not used

Copyright © 2026 Texas Instruments Incorporated

11.3.1.4.2.4.3.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Not used
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be written to an MCAN TX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. The PDMA will break up the source packet into fragments of this buffer size, copying the 8 byte MCAN header for the initial fragment, and then skipping it for each additional fragment and thus reusing the header from the first fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
Z	0x401	23:0	Not used

11.3.1.4.2.4.3.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information:

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer to each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
Z	0x401	23:0	Not used

11.3.1.4.2.4.3.4.4 AASRC TxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this TX channel.

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the TX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each TX FIFO. The flags corresponding to all TX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

11.3.1.4.2.4.3.4.5 AASRC TxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	TX FIFO Index for slot 7
27:24	Entry6	6	TX FIFO Index for slot 6
23:20	Entry5	5	TX FIFO Index for slot 5
19:16	Entry4	4	TX FIFO Index for slot 4
15:12	Entry3	3	TX FIFO Index for slot 3
11:8	Entry2	2	TX FIFO Index for slot 2
7:4	Entry1	1	TX FIFO Index for slot 1
3:0	Entry0	0	TX FIFO Index for slot 0

11.3.1.4.2.4.3.4.6 AASRC TxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. TX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all TX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	TX FIFO Index for slot 15
27:24	Entry14	14	TX FIFO Index for slot 14
23:20	Entry13	13	TX FIFO Index for slot 13
19:16	Entry12	12	TX FIFO Index for slot 12
15:12	Entry11	11	TX FIFO Index for slot 11
11:8	Entry10	10	TX FIFO Index for slot 10
7:4	Entry9	9	TX FIFO Index for slot 9

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
3:0	Entry8	8	TX FIFO Index for slot 8

11.3.1.4.2.4.3.5 PSI-L Destination Thread Enables

PSI-L destination threads must first be enabled to accept data. Threads are enabled or disabled by setting or clearing the enable bit in the PSI-L pairing registers for the thread. When a thread is disabled, it must drop any data phases which are sent but properly return the credits for the data phases which are dropped.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of write transactions.

11.3.1.4.2.4.4 Data Transfer

Packet transmission is accomplished within the PDMA by unpacking and moving data from the Tx Per Channel FIFOs which were filled via the Transmit PSI-L Interface to specified memory mapped address ranges via the VBUSP controller interfaces. On the Tx side of the PDMA, these transfers are always writes. Each write transfer which is performed by the Tx DMA Unit is to a destination address that is hardcoded in the channel at design time and of a size as specified in the static Transfer Request.

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during transmit is dependent on the channel type. The following sections describe what will happen for the different channel types.

11.3.1.4.2.4.4.1 X-Y FIFO Mode Channel

The PDMA channel remains idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA sequentially issues a total of 'Y' parameter writes of 'X' parameter bytes to the data_address specified in the tchan_info parameter for the channel. Each write that the DMA performs is a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed, the channel returns to an idle state and waits until it is triggered again. The write transfers that are performed are accomplished as quickly as possible given availability of data in the Tx channelized FIFO, and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.4.2.4.4.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. So for example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address increments throughout the burst
- Bursts are sub-divided to fit into a 64 byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA 'gaps' the byte enables on writes as needed

11.3.1.4.2.4.4.2 MCAN Mode Channel

The PDMA channel remains idle until data is received from the UDMA-P over the PSIL interface. At this time, the PDMA immediately starts copying packet bytes into the MCAN TX buffer corresponding the PDMA/MCAN channel. The number of bytes copied is smaller of the remaining bytes in the packet, or the value of the 'Y' parameter. Once a TX buffer has been filled, the PDMA transfers ownership of the buffer to MCAN by performing an MCAN register write. The PDMA then waits to regain ownership of the TX buffer by waiting for the corresponding MCAN TX event. At this time, the PDMA continues with refilling the TX buffer for the next

Copyright © 2026 Texas Instruments Incorporated

packet fragment. On subsequent fills (until the end of packet is reached), the PDMA skips over the 8 byte MCAN header, leaving the original contents from the initial fragment copy in place.

11.3.1.4.2.4.4.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's TxFifoConfig register. This register holds 3 basic pieces of information:

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel remains idle until a pulse is detected on ALL associated input DMA request event pins required by the TxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Once the channel activates, it starts reading FIFO index values from the TxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs written for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width written to the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed, the channel returns to an idle state and waits until it is triggered again. The write transfers that are performed are accomplished as quickly as possible, given availability of data in the TX channelized FIFO and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.4.2.4.5 Transmit PSI-L Interface Transactions

The PDMA accepts data across the Tx PSI-L Interface in accordance with the physical handshaking protocol, as defined in the PSI-L Interface specification. The value sources for the output pins on the PSI-L interface are described in the following table.

Pin	Output Value / Source From PDMA
strm_i_link	This signal is asserted high to indicate the PSI-L link is valid.
strm_i_ready	This signal is asserted high anytime there is space in the PSI-L ingress fifo.
strm_i_sreq	This signal is asserted high anytime there is credit return data waiting in the PSI-L status fifo.
strm_i_sthreadid	This is set to the local destination thread that is returning credit.
strm_i_sccnt	This is set to the number of credits being returned, or 0 on a TX teardown acknowledgement.

11.3.1.4.2.4.6 Tx Pause

The Host initiates a channel pause by setting the pause bit in the RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.4.2.4.7 Tx Teardown

The Host initiates a channel teardown by setting the tdown bit in the UDMA-P channel paired with the PDMA. The UDMA-P communicates the teardown state through the PSI-L data channel, to ensure that the teardown is not seen by the PDMA until all the previous UDMA-P data for the channel has been flushed. At this time, the teardown state is reflected in the PSI-L RT Enable register of the PDMA. Note that a non-synchronized teardown can also be initiated by directly clearing the enable bit in the PSI-L RT Enable register of the PDMA.

Once all data has been flushed from the PDMA to the peripheral, the enable state of the PDMA channel is cleared in the PSI-L RT enable register, but the teardown bit remains high. A teardown completion indication is

Copyright © 2026 Texas Instruments Incorporated

sent back across the status pins of the PSI-L in the form of a credit response with sccnt=0. Upon completion, no further packet processing occurs until the Host re-configures the channel. If the channel fails to teardown because the peripheral has stopped responding, or if the UDMA-P transmission stops on a data boundary that is not compatible with the static TR configuration, the flush bit in the RT enable register can be set to ensure that all data can be properly flushed from the internal pipe.

11.3.1.4.2.4.8 Tx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully, even with flush enabled. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L Enable register. This causes a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.4.2.4.9 Tx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For transmit, they are defined as in Table 11-85.

Table 11-85. Tx Debug/State Register

Name	PSIL Addr	Field	Description
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to write to the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next write offset to use when writing to the CAN TX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.
Flush	0x403	30	When set, the PDMA is processing in a flushing state, where it runs without waiting for DMA requests and without writing data to the peripheral. It is only operating its internal state machine to allow internal data pipes to drain properly.
Pause	0x403	29	When set, the PDMA waiting in a paused state. This bit will clear when data starts flowing again from the UDMA-P.
Data	0x403	28	When set, there is a non-zero amount of data still waiting to be written to the peripheral.
XData	0x403	27	When set, there is enough data still waiting to be written to the peripheral to start servicing a peripheral DMA event.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.

11.3.1.4.2.5 Receive Operation

11.3.1.4.2.5.1 Source (Rx) Channel Allocation

A total of 3 source channels are provided within the DMA for concurrent transfers from the various attached peripherals into the Rx per channel buffers and on to the PSI-L Rx Interface. Each Rx channel requires a single PSI-L thread. The Rx channels are allocated as follows.

Rx DMA Channel	Function	Channel Type	Trigger Type	Data FIFO Address	Strobe MMR Address	Control FIFO Address
0	McASP 0 Rx Ch 0	XY	edge	000002B08000	000000000000	000000000000
1	McASP 1 Rx Ch 0	XY	edge	000002B18000	000000000000	000000000000
2	McASP 2 Rx Ch 0	XY	edge	000002B28000	000000000000	000000000000

Copyright © 2026 Texas Instruments Incorporated

11.3.1.4.2.5.2 Source Channel Initialization

After reset, all threads/channels in the PDMA will be idle and waiting for work to be assigned to them. To initiate PDMA operation, the Host must first pair the channel with a remote data source (normally a UDMA-P channel), set up the static TR for the channel, and enable the thread.

11.3.1.4.2.5.2.1 PSI-L Source Thread Pairing Registers

11.3.1.4.2.5.2.1.1 Peer Thread ID Register (PSIL Address 0x000)

Bits	Field	Reset	Description
31:29	ThreadPriority	0x0	This field is hard coded to 0x0 and is not writable
28:24	PeerThreadWidth	0x0	Datapath width of paired peer destination thread. This field is encoded as follows: 0 = 32-bit, 1 = 64-bit, 2 = 128 bit, 3 = 256 bit, 4 = 512 bit
23:16	-	0x0	Reserved.
15:0	Peer ThreadID	0x0	Thread ID to which all non-Transfer Response, non-Config messages from this thread will be sent.

11.3.1.4.2.5.2.1.2 Peer Credit Register (PSIL Address 0x001)

Bits	Field	Reset	Description
31:8	-	0x0	Reserved.
7:0	CreditCnt	0x0	Free entries in destination thread. Legal range is 0 - 128 in number of elements.

11.3.1.4.2.5.2.1.3 Enable Register (PSIL Address 0x002)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all credit returns, and will not generate egress data. A one to zero transition on this bit fully resets the channel.
30:0	-	0x0	Reserved.

11.3.1.4.2.5.2.2 PSI-L Source Thread Realtime Enable/Count Registers

11.3.1.4.2.5.2.2.1 RT Enable Register (PSIL Address 0x408)

Bits	Field	Reset	Description
31	enable	0x0	When set, the channel is enabled. When cleared, the RX channel is disabled. When disabled, the channel ignores all DMA events from the peripheral. It maintains proper data transfers with the UDMA-P such that the credit handshake is not disrupted. However, unlike teardown, it does not close out the current open packet. Thus the teardown bit should always be used to disable an RX channel instead of manually clearing this bit. Failing to use teardown may result in stale data remaining in the internal RX FIFO, which would also prevent the channel from going idle without a channel reset. Note that this bit cannot be changed from 0 to 1 if the global enable bit in the peer enable register is 0.

Copyright © 2026 Texas Instruments Incorporated

Bits	Field	Reset	Description
30	tdown	0x0	When set, the channel will commence a RX channel teardown procedure. It will stop on the next FIFO boundary. It then clears the DMA event count and ignores all future DMA events from the peripheral. After stopping peripheral reads, the PDMA sends a teardown message to the UDMA-P that also closes the current packet with an EOP. If no packet is open at the time of the teardown, the message also includes SOP. The EOP teardown message may or may not contain final packet data. Once the channel teardown is complete and ready to be reused, the enable bit is cleared.
29	pause	0x0	When set, the channel is in a paused state. It will stop on the next FIFO boundary. It continues to accept and count DMA events from the peripherals, but will not act on them. The pause bit can be cleared and data will resume. Pause will not stop teardown from completing.
28:2	-	0x0	Reserved.
1	idle	0x0	This is a read-only bit that signifies that the Paused or Disabled channel is also idle. This bit is read only and can only become set if pause is set or enable is cleared.
0	free	0x0	Free Run: 0 = channel honors the emudbg suspend signal, 1 = channel will free run, regardless of the value of emudbg suspend.

11.3.1.4.2.5.2.2.2 Source Thread Byte Count Register (PSIL Address 0x404)

Bits	Field	Reset	Description
31:0	bytes	0x0	This register contains the number of bytes that have been read from the VBUSP mapped peripheral. The register is write to decrement, so running counts can be tracked by reading register and then writing back the value that was read. It will wrap on overflow. It will reset to zero on a channel reset.

11.3.1.4.2.5.2.3 PSI-L Source Thread Pairing

After reset, all threads on PSI-L are uninitialized and unpaired. The Host pairs each source PDMA channel with (typically) a corresponding destination channel in the UDMA-P. The PDMA thread is configured to point to the UDMA-P thread, and the UDMA-P thread is configured to point to the PDMA thread. Once paired, the source channels in the PDMA send all their data to the source channel in the UDMA-P, and the destination channel in the UDMA-P never sees any data other than data from the source channel in the PDMA.

11.3.1.4.2.5.2.4 Static Transfer Request Setup

After reset, all channels in the PDMA are idle and waiting for work to be assigned to them. To initiate PDMA operation, the Host sends configuration transactions across PSI-L and sets up the static TR for each destination channel. The format of the TR is identical for each DMA mode, consisting of an X, Y, and Z parameter, but the field interpretation varies.

11.3.1.4.2.5.2.4.1 X-Y FIFO Mode Static TR

In X-Y FIFO mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	When set, enables VBUSP burst mode on this channel. See the XY burst description for more information.

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Element size. This field specifies how much data is transferred in each write which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, 4 = 64 bits, and 5-7 = RESERVED
Y	0x400	11:0	Element count. This field specifies how many elements to transfer each time a trigger is received on the channel.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full FIFO operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.4.2.5.2.4.2 MCAN Mode Static TR

In MCAN mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.
X	0x400	26:24	Not used
Y	0x400	11:0	Buffer Size. This field specifies how many bytes should be read from an MCAN RX buffer. This field includes the 8 byte MCAN header on the initial packet fragment. A buffer size less than 16 is treated as 16, and a buffer size greater than 72 is treated as 72.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	Buffer Count. This field specifies how many MCAN RX buffers should be read before closing the CPPI packet with an 'EOP' indication. When this count is greater than 1, multiple MCAN RX buffers will be read into a single CPPI packet buffer. The 8 byte MCAN header will be skipped on subsequent MCAN buffer reads. Setting this field to NULL will suppress all packet delineation, and should be avoided.

11.3.1.4.2.5.2.4.3 AASRC Mode Static TR

AASRC mode is similar to X-Y FIFO mode, the static TR consists of the following information.

Param	PSIL Addr	Field	Description
Burst	0x400	31	Not used
Acc32	0x400	30	When set, enables 32-bit access mode. On a 32-bit PDMA, all accesses will have XCNT=4 to support legacy IP that is not fully VBUSP compliant. This bit is ignored if the PDMA VBUSP port is not 32 bits wide.

Copyright © 2026 Texas Instruments Incorporated

Param	PSIL Addr	Field	Description
X	0x400	26:24	Element size. This field specifies how much data is transferred in each read which is performed by the DMA. This field is encoded as follows: 0 = 8 bits, 1 = 16 bits, 2 = 24 bits, 3 = 32 bits, and 4-7 = RESERVED
Y	0x400	11:0	FIFO element count. In AASRC mode, a channel can service multiple FIFOs using a list supplied in its FIFO configuration. This field specifies how many times to service the FIFO list, hence how many elements to transfer from each FIFO, each time a trigger is received on the channel. Note for each loop specified by the value of Y, the entire list is processed. For example, if the FIFO list is 0, 1, 2, 3, and Y is set to 2 loops, the FIFOs are serviced in the order: 0, 1, 2, 3, 0, 1, 2, 3.
EOL	0x401	31	EOL Mode. Normally, when the Z count of FIFO operations has been reached, the PDMA will close the packet with an 'EOP' indication. When this flag is set, the PDMA will instead trigger an EOL at the completion of Z.
Z	0x401	23:0	FIFO count. This field specifies how many full DMA request operations comprise a complete packet. When the count has been reached, the PDMA will close the packet with an 'EOP' indication. If this parameter is set to NULL, then no packet delineation is supplied by the PDMA and all framing is controlled via the UDMA-P TR.

11.3.1.4.2.5.2.4.4 AASRC RxFifoConfig (PSIL Address 0x405)

This register describes the FIFO list to be processed for this RX channel.

Bits	Field	Reset	Description
31	GroupMode	0x0	When set, the channel is in 'Group Mode'. It will look for Group Mode DMA requests, and access the Group Mode FIFOs. When clear, the channel is 'Stream Mode'. It will look for Stream FIFO DMA requests, and access the Stream Mode FIFOs.
30	DmaReqReset	0x0	When set, resets any latched DMA request using the DmaReqMask. This bit is self-clearing. It should be used to synchronize the AASRC event status with the PDMA in the event that the AASRC has been previously used and it is not known if the PDMA may have latched, and is holding, previous DMA requests.
23:20	LastSlot	0x0	This is the index (0-15) of the last slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
19:16	FirstSlot	0x0	This is the index (0-15) of the first slot in the RX FIFO ordering table used by this channel. The ordering table is read to get the FIFO index to access for each slot, starting with the first and ending with the last.
15:0	DmaReqMask	0x0	This field holds a set of flags indicating which AASRC DMA requests must fire in order for this channel to activate. In Steam Mode, these 16 flags correspond to the 16 DMA requests for each RX FIFO. The flags corresponding to all RX FIFOs involved with the channel should be set to 1. In Group Mode, these flags indicate which Group Mode DMA requests must fire. In this case, only bits 3:0 are relevant and only one bit should be set to 1 as a DMA channel only services a single group.

Copyright © 2026 Texas Instruments Incorporated

11.3.1.4.2.5.2.4.5 AASRC RxOrderTable0 (PSIL Address 0x406)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry7	7	RX FIFO Index for slot 7
27:24	Entry6	6	RX FIFO Index for slot 6
23:20	Entry5	5	RX FIFO Index for slot 5
19:16	Entry4	4	RX FIFO Index for slot 4
15:12	Entry3	3	RX FIFO Index for slot 3
11:8	Entry2	2	RX FIFO Index for slot 2
7:4	Entry1	1	RX FIFO Index for slot 1
3:0	Entry0	0	RX FIFO Index for slot 0

11.3.1.4.2.5.2.4.6 AASRC RxOrderTable1 (PSIL Address 0x407)

This table is filled with FIFO index values. RX channels reference a starting and ending 'slot' in this table for each transfer. The actual FIFO accessed by the DMA for each slot is determined by the FIFO index stored in this table. Note: This is a single register that is shared by all RX threads.

Bits	Field	Reset	Description
31:28	Entry15	15	RX FIFO Index for slot 15
27:24	Entry14	14	RX FIFO Index for slot 14
23:20	Entry13	13	RX FIFO Index for slot 13
19:16	Entry12	12	RX FIFO Index for slot 12
15:12	Entry11	11	RX FIFO Index for slot 11
11:8	Entry10	10	RX FIFO Index for slot 10
7:4	Entry9	9	RX FIFO Index for slot 9
3:0	Entry8	8	RX FIFO Index for slot 8

11.3.1.4.2.5.2.5 PSI-L Source Thread Enables

PSI-L source threads must be enabled in order to process peripheral DMA requests and send data. When disabled, DMA requests are ignored.

Once a thread is paired, the TR has been set up and the enable is asserted, the channel is armed and ready to be triggered to perform the specified set of read transactions.

11.3.1.4.2.5.3 Data Transfer

Packet reception is accomplished within the PDMA by moving data from structures that are located in memory accessible via the VBUSP Memory Interface(s) onto the Receive PSI-L Interface. On the Rx side of the PDMA, these transfers are always reads. Data is read from an attached memory mapped space and packed into the Rx Per Channel Buffer for that channel. At a later time, the data is moved from the Rx Per Channel FIFO to a remote peer DMA entity via the Rx PSI-L interface

The sequences of logical transactions that are performed by the PDMA on the Memory I/F during receive is dependent on the channel type. The following sections describe what happens for the different channel types.

11.3.1.4.2.5.3.1 X-Y FIFO Mode Channel

The PDMA channel remains idle until a pulse is detected on the associated input DMA request event pin. Once the pulse is detected, the DMA sequentially issues a total of 'Y' parameter reads of 'X' parameter bytes from the data_address specified in the tchan_info parameter for the channel. Each read that the DMA performs is a single 'X' element in size (no large bursts). Once the total specified number of transactions has been completed,

Copyright © 2026 Texas Instruments Incorporated

the channel returns to an idle state and waits until it is triggered again. The read transfers that are performed are accomplished as quickly as possible given availability of data in the Rx channelized FIFO, and given the arbitration that may occur as a result of other channels also using the same read unit.

11.3.1.4.2.5.3.1.1 X-Y FIFO Burst Mode

The burst mode for XY is designed to allow the PDMA to burst across a FIFO region, while extracting out the 'sample size' from each data phase. For example, it can read a "burst" of multiple 32-bit words from a McASP, and grab only 24-bits from each data phase.

It uses the same registers as standard XY mode, where X is the encoded sample size, and Y is the number of samples to read/write per DMA request. Setting bit 31 of the XY register enables burst.

The following are the rules for enabling burst mode. Incompatible peripherals do not need to enable burst, or can enable it on RX without enabling it on TX.

- The VBUSP address will increment throughout the burst
- Bursts are sub-divided to fit into a 64-byte transfer window
- One sample is transferred for every bus data phase
  - Cannot burst one byte samples from a 16-bit peripheral on a 32-bit bus
  - Cannot burst 64-bit samples on a 32-bit bus (configure for 2 32-bit samples instead)
- PDMA 'gaps' the byte enables on writes as needed

11.3.1.4.2.5.3.2 MCAN Mode Channel

The PDMA channel remains idle until a pulse is detected on the associated input DMA request event pin. The DMA event pins are mapped from the CAN filter event bits. The event bits determine the channel and one of two CAN buffers to use for the RX operation. This allows the RX buffers on the CAN to be configured in a ping/pong fashion, preventing the loss of CAN packet data. When an event is received, the PDMA starts copying bytes out of the corresponding CAN buffer. The number of bytes copied is equal to the value of the 'Y' parameter in the channel configuration. Once all data has been copied from the buffer, the PDMA transfers ownership of the buffer back to the CAN by writing a CAN register. It then waits for another RX DMA event. The PDMA can copy multiple CAN RX buffers to the same CPPI packet. It will not close out the CPPI packet until it has copied out a number of RX buffers equal to the 'Z' parameter in the channel configuration. On subsequent RX buffer copies, the MCAN skips the first 8 bytes of the RX buffer, which is the MCAN packet header.

The mapping of MCAN filter events to MCAN channels and buffer is important for properly configuring the DMA. The mapping is defined as follows.

CAN FE2	CAN FE1	CAN FE0	Description
0	0	0	Not used
0	0	1	Not used
0	1	0	PDMA CAN channel offset 0, RX buffer 0
0	1	1	PDMA CAN channel offset 0, RX buffer 1
1	0	0	PDMA CAN channel offset 1, RX buffer 2
1	0	1	PDMA CAN channel offset 1, RX buffer 3
1	1	0	PDMA CAN channel offset 2, RX buffer 4
1	1	1	PDMA CAN channel offset 2, RX buffer 5

By using two filter entries in the CAN, software can setup a ping/pong buffer for a particular RX packet ID. For example, the following two filter entries set up a ping/pong using RX buffers 0 and 1 for packet ID 5 on CAN RX channel 0.

Filter Entry	Packet Match ID	Filter Event Bits
0	5	0x2
1	5	0x3

Copyright © 2026 Texas Instruments Incorporated

11.3.1.4.2.5.3.3 AASRC Channel

The AASRC channel is controlled primarily via the channel's RxFifoConfig register. This register holds 3 basic pieces of information:

- Whether the channel uses AASRC Stream Mode or Group Mode
- The slot range in the order table used (from designated first slot to designated last slot)
- The AASRC DMA request event(s) that must fire before the channel becomes active

The channel remains idle until a pulse is detected on ALL associated input DMA request event pins required by the RxFifoConfig setting. The pulses for the individual events are latched and held by the PDMA until they all arrive.

Once the channel activates, it starts reading FIFO index values from the RxOrderTable, starting at the configured 'First Slot' and ending with the 'Last Slot'. The actual FIFO indices used are obtained from the ordering table. For example, say FirstSlot=3 and LastSlot=5. If the first 6 slots of the ordering table were: 0, 2, 4, 6, 8, 10, the FIFOs read for the event would be 6, 8, and 10, because 'slot 3' of the table contains 6, and it would proceed through 'slot 5' of the table which contains 10.

The X and Y registers are still used as they are in a normal X-Y mode. When accessing each FIFO, the setting of X determines the sample byte width read from the FIFO. The value of 'Y' determines how many times the entire FIFO list is processed for each activation of the channel.

Once the total specified number of transactions has been completed, the channel returns to an idle state and waits until it is triggered again. The read transfers that are performed are accomplished as quickly as possible given availability of data in the TX channelized FIFO, and given the arbitration that may occur as a result of other channels also using the same write unit.

11.3.1.4.2.5.4 Receive PSI-L Interface Transactions

When an entire word has been packed into the Rx Per Channel Buffer for a given channel, a one word data phase transfer is initiated for that channel/thread on the Rx PSI-L interface, and the data is popped from the Rx Per Channel FIFO. The value sources for the output pins on the PSI-L interface are described in Table 11-86.

Table 11-86. PSI-L Interface Output Pins

Pin	Output Value / Source From PDMA
strm_o_req	This signal is asserted when the PDMA determines that there is at least 1 word of data in it's TP-CC output FIFO
strm_o_sthread_id	Set equal to the TP-CC DMA channel number which is on the interface for this cycle
strm_o_dthread_id	Set equal to the target thread ID value given in the PSI-L pairing configuration registers for this DMA channel
strm_o_data_type	Will be set to the proper PSI-L data type. The PDMA can send config response, PSI info word 0, and PSI data word.
strm_o_wnum	Set equal to the packing lane for the current word within a contiguous transfer. This value will start at 0 for each new TR and will increment for each subsequent data phase in the TR.
strm_o_lastw	This signal will be asserted coincident with eop.
strm_o_xcnt	The xcnt will be equal to the data path width in bytes.
strm_o_worden	The worden will be modulated to indicate which 32-bit words are valid within each control type data phase.
strm_o_data	The data will be packed/left justified to comply to the big endian data ordering as required in the PSI-L I/F specification.
strm_o_sop	This signal is asserted for 1 data phase at the beginning of each new TR or data transfer packet.
strm_o_eop	This signal is asserted for 1 data phase at the end of each TR or data transfer packet.
strm_o_sol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the start of a new set of transactions designated by the count set in the Z field of PSI-L register 0x401.

Copyright © 2026 Texas Instruments Incorporated

Table 11-86. PSI-L Interface Output Pins (continued)

Pin	Output Value / Source From PDMA
strm_o_eol	This signal is set to zero unless the EOL bit is set in PSI-L register 0x401, in which case, it is set for 1 data phase at the end of a set of transactions designated by the count set in the Z field of PSI-L register 0x401.
strm_o_priv	Set to zero.
strm_o_privid	Set to zero.
strm_o_virtid	Set to zero.
strm_o_secure	Set to zero.
strm_o_interest	Set to zero.
strm_o_sready	Always asserted as the PDMA is always able to accept credit returns.

11.3.1.4.2.5.5 Rx Pause

The Host initiates a channel pause by setting the pause bit in RT enable register. The paused channel can be resumed by clearing the register bit.

11.3.1.4.2.5.6 Rx Teardown

The Host initiates a RX channel teardown by setting the tdown bit in the RT enable register for the target RX channel. The PDMA communicates the teardown state to the UDMA-P through the PSI-L data channel, to ensure that the teardown is not seen by the UDMA-P until all the previous PDMA data for the channel has been flushed.

The PDMA does not stop reading peripheral data until it reaches a FIFO boundary, as configured through the 'X' and 'Y' parameters in the static TR. It always attempts to complete the 'Y' count for the current event being processed. Upon reaching a stopping point, the PDMA then clears the enable bit in the pairing register; however, the teardown bit remains set. No further packet processing occurs until the Host re-configures the channel. The teardown process propagates to the UDMA-P and its final status can be checked there.

11.3.1.4.2.5.7 Rx Channel Reset

In the unlikely event that channel synchronization is corrupted, a channel may fail to teardown gracefully. If this occurs, the channel may be reset by clearing the enable bit in the PSI-L enable register. This causes a local reset of the entire channel, including TR and pairing registers. Note that it does not reset the UDMA-P peer. Resetting the UDMA-P peer is also required before re-initializing and re-pairing the channel.

11.3.1.4.2.5.8 Rx Debug/State Register

The debug/state registers are supplied to give software applications additional information about the PDMA than they would need in regular operation, but which may be useful in debug situations. The registers appear on the PSI-L bus, near the static TR registers. For receive, they are defined as in Table 11-87.

Table 11-87. Rx Debug/State Register

Name	PSIL Addr	Field	Description
Z*	0x402	31:16	This field holds the lower 12 bits of the current Z count for legacy pursposes. See register 0x40F below for the full width version of Z.
Y	0x402	15:0	This field holds the current Y count. In X-Y FIFO mode, this is the number of X sized samples yet to be read from the peripheral for the DMA event being serviced. In MCAN mode, this field holds the next read offset to use when read to the CAN RX buffer.
InEvent	0x403	31	When set, the PDMA is in the middle of processing a FIFO event.

Copyright © 2026 Texas Instruments Incorporated

Table 11-87. Rx Debug/State Register (continued)

Name	PSIL Addr	Field	Description
Tdown	0x403	30	When set, the PDMA is processing a teardown operation. This bit is set simultaneously with the teardown bit in the source (RX) RT enable register. This bit will clear when the teardown is complete, regardless as to if the teardown bit in the pairing register is cleared or not. The teardown will propagate to the UDMA-P and its full completion status can be checked there.
Pause	0x403	29	When set, the PDMA is stopped in a paused state. This bit will clear if the channel is un-paused or disabled.
Space	0x403	28	When set, there is a non-zero amount of internal FIFO space available to hold new read data.
XSpace	0x403	27	When set, there is a enough internal FIFO space available to start servicing a peripheral DMA event.
Buffer	0x403	26	This is the current RX buffer (0/1) for the current MCAN receive operation.
State	0x403	23:20	This code reflects the current state of the PDMA channel, and is specific to the current implementation.
EventCnt	0x403	19:16	This field holds the number of backlogged DMA events yet to be serviced.
Z	0x40F	31:0	This field holds the full width value of the current Z count. In X-Y FIFO mode, this field holds the 1 based FIFO count of the FIFO being currently read, or the number of FIFO completions when the current operation completes. In MCAN mode, this field holds the zero based buffer index of the buffer currently being read, or the number of previously completed buffers.

Copyright © 2026 Texas Instruments Incorporated

Chapter 12

Peripherals

12.1 General Connectivity Peripherals 1144
12.3 Memory Interfaces 1439
12.4 Industrial and Communication Interfaces 1666
12.5 Audio Peripherals 1820
12.6 Camera Peripherals 1896
12.7 Timer Modules 1944
12.8 Internal Diagnostics Modules 1989
12.9 Display Subsystem and Peripherals 2312

Copyright © 2026 Texas Instruments Incorporated

12.1 General Connectivity Peripherals

Copyright © 2026 Texas Instruments Incorporated

12.1.1 General-Purpose Interface (GPIO)

This chapter describes the General-Purpose Input/Output (GPIO) for the device.

12.1.1.1 GPIO Overview

The General-Purpose Input/Output (GPIO) peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, the user can write to an internal register to control the state driven on the output pin. When configured as an input, user can obtain the state of the input by reading the state of an internal register.

In addition, the GPIO peripheral can produce host CPU interrupts and DMA synchronization events in different interrupt/event generation modes.

The device has one or more instances of GPIO modules. The GPIO pins are grouped into banks (16 pins per bank and 9 banks per module), which means that each GPIO module provides up to 144 dedicated general-purpose pins with input and output capabilities; thus, the general-purpose interface supports up to 432 (3 instances × (9 banks × 16 pins)) pins. Not all GPIO pins are pinned out. See General Purpose Input/Output (GPIO) in the Module Integration section for more information.

Figure 12-1 presents the GPIO modules overview.

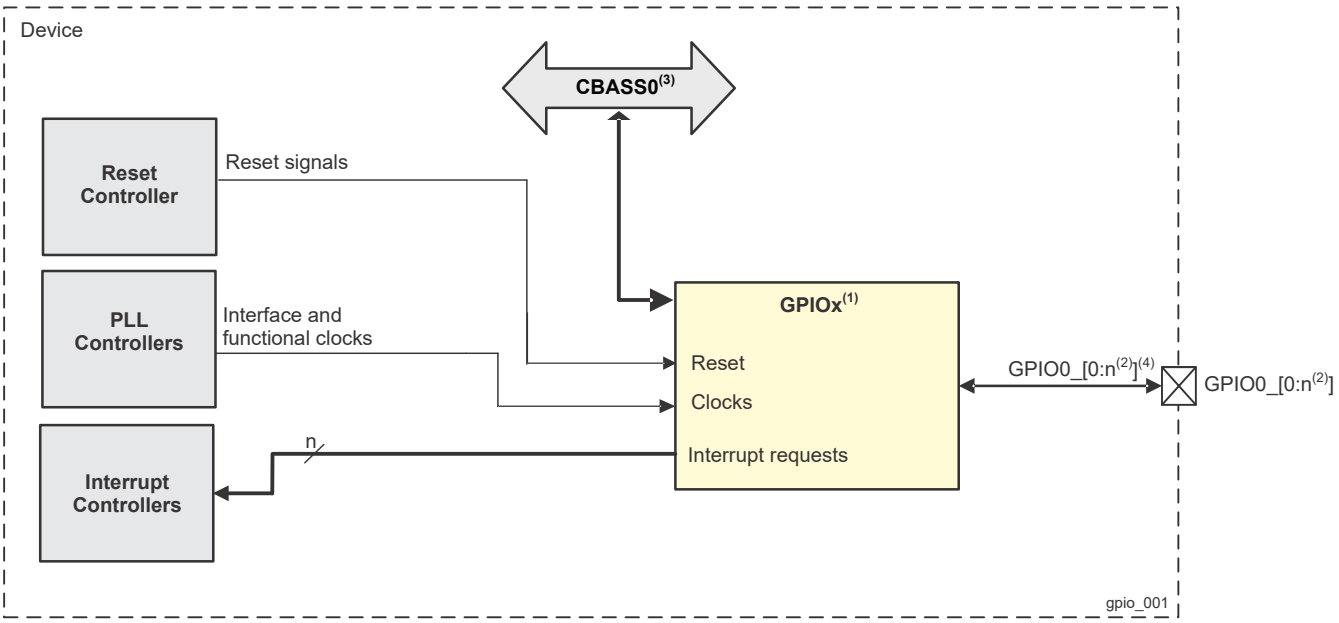


Figure 12-1. GPIO Modules Overview

A. (1): x represents a valid instance of GPIO in a domain.
B. (2): n represents the maximum number of GPIO signals -1.
C. (3): CBASS0 represents the connectivity in the domain of the IP.
D. (4): Some GPIO signals may be uni-directional. Consult the device datasheet for specifics.

12.1.1.1.1 GPIO Features

Each channel in the GPIO modules has the following features:

- Supports 9 banks of 16 GPIO signals
- Supports up to 9 banks of interrupt capable GPIOs
- Interrupts:
  - Can enable interrupts for each bank of 16 GPIO signals
  - Interrupts can be triggered by rising and/or falling edge, specified for each interrupt capable GPIO signal

Copyright © 2026 Texas Instruments Incorporated

- Set/clear functionality:
  - Firmware writes 1 to corresponding bit position(s) to set or to clear GPIO signal(s). This allows multiple firmware processes to toggle GPIO output signals without critical section protection (disable interrupts, program GPIO, re-enable interrupts, to prevent context switching to another process during GPIO programming).
- Separate Input/Output registers:
  - If preferred by firmware, some GPIO output signals can be toggled by direct write to the output register(s) in addition to set/clear.
  - Output register, when read in, reflects output drive status. This, in addition to the input register reflecting pin status, allows wired logic be implemented.

12.1.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.1.1.2 GPIO Environment

This section describes the GPIO external connections (environment).

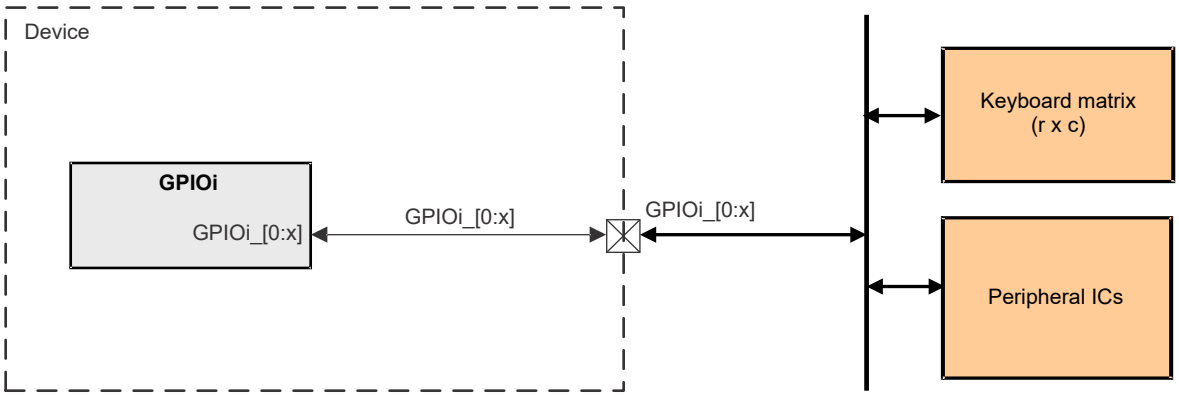
The general-purpose interface combines three GPIO modules for a flexible, user-programmable, general-purpose input/output (I/O) controller. The general-purpose interface implements functions that are not implemented with the dedicated controllers in the device and require simple input and/or output software-controlled signals. The GPIO allows a variety of custom connections and expands the I/O capabilities of the system to the real world.

The general-purpose interface can physically connect the device to a keyboard matrix and peripheral integrated circuits (ICs).

Figure 12-2 shows a typical application using the general-purpose interface.

12.1.1.2.1 GPIO Interface Signals

Figure 12-2 shows all of the GPIO interface signals.



Note

i represents a GPIO instance. See the device datasheet for available domains and GPIO instances.

Note

See Module Integration section for signal specifics in GPIO.

Figure 12-2. GPIO Interface Signals and Typical Application

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.1.3 Integration

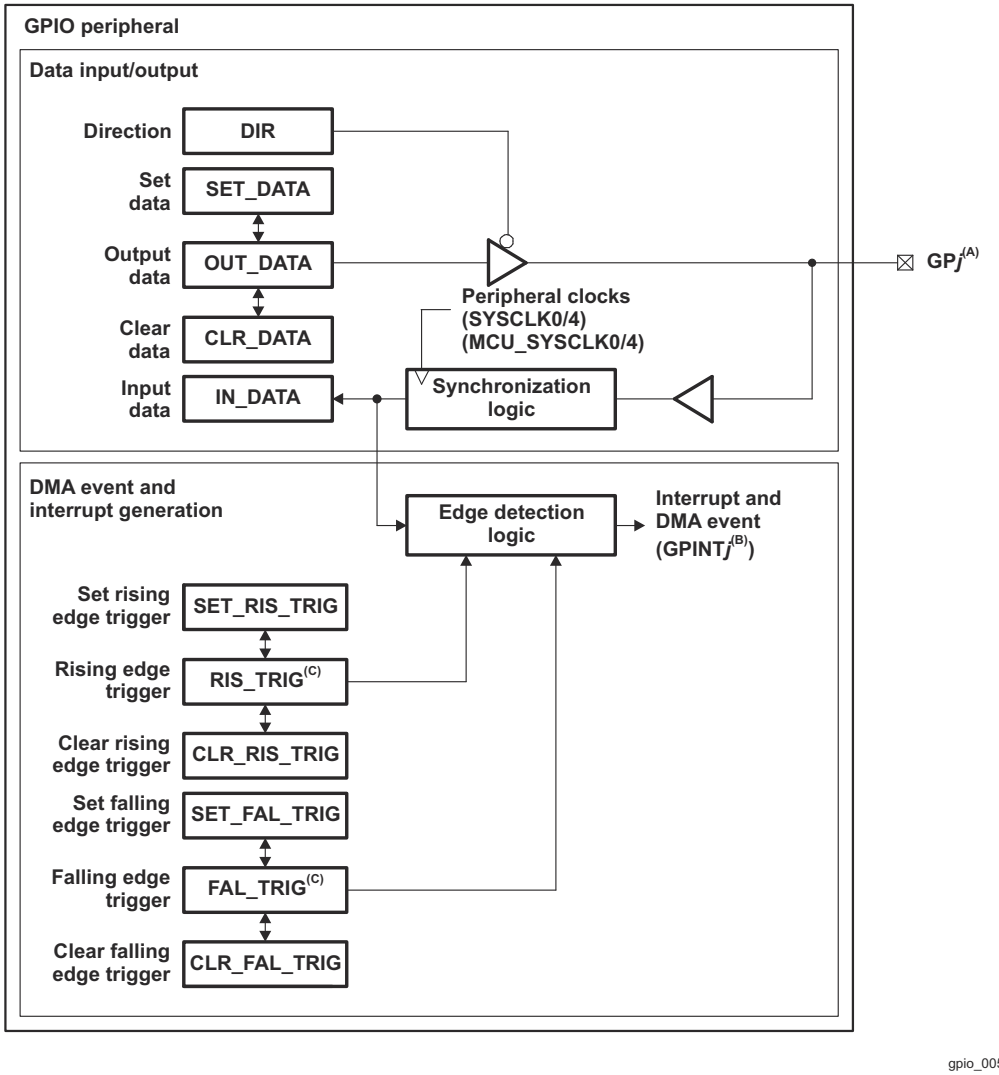
See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.1.1.4 GPIO Functional Description

12.1.1.4.1 GPIO Block Diagram

Figure 12-3 shows the general-purpose interface block diagram.



A. Some of the GPj pins are muxed with other device signals. For details, see the device-specific Datasheet.
B. All GPINTj can be used as host CPU interrupts and synchronization events to the DMA.
C. The RIS_TRIG and FAL_TRIG registers are internal to the GPIO module and are not visible to the host CPU.

Figure 12-3. GPIO Block Diagram

12.1.1.4.2 GPIO Function

Each GPIO pin (GPj) can be independently configured as either an input or an output using the GPIO direction registers. The GPIO direction register (DIR) specifies the direction of each GPIO signal. Logic 0 indicates the GPIO pin is configured as output, and logic 1 indicates input.

When configured as output, writing a 0x1 to a bit in the set data register drives the corresponding GPj to a logic-high state. Writing a 0x1 to a bit in the clear data register drives the corresponding GPj to a logic-low state. The output state of each GPj can also be directly controlled by writing to the output data register.

For example, to set GP8 to a logic-high state, the software can perform one of the following:

Copyright © 2026 Texas Instruments Incorporated

- Write 100h to the GPIO0_SET_DATA01 register.
- Write 0h to the GPIO_DIR01 register to configure as output pin.
- Read in GPIO_OUT_DATA01 register, change bit 8 to 0x1, and write the new value back to GPIO_OUT_DATA01.

To set GP8 to a logic-low state, the software can perform one of the following:

- Write 100h to the GPIO_CLR_DATA01 register.
- Write 0h to the GPIO_DIR01 register to configure as output pin.
- Read in GPIO_OUT_DATA01 register, change bit 8 to 0x0, and write the new value back to GPIO_OUT_DATA01.

Note that writing a 0x0 to bits in the set data and clear data registers does not affect the GPIO pin state.

Also, for GPIO pins configured as input, writing to the set data, clear data, or output data registers does not affect the pin state.

For a GPIO pin configured as input, reading the input data register (IN_DATA) will return the pin state. Reading the SET_DATA register or the CLR_DATA data register will return the value in OUT_DATA, not the actual pin state. The pin state is available by reading the input data register. Note that when the direction is configured as input, the output state is determined by software's programming set/clear/output registers, and may not agree with the pin state, which is driven by an external device.

12.1.1.4.3 Interrupt and Event Generation

Each GPIO pin (GPj) can be configured to generate a host CPU interrupt (GPINTj) or a synchronization event to the DMA (GPINTj). Configuration is on per-bank basis. Each bit of the BINT_EN parameter dictates YES/NO option for each bank. Bit 0 controls bank 0, bit 1 controls bank 1, and so on.

The interrupt and DMA event can be generated on the rising-edge, falling-edge, or on both edges of the GPIO signal. The edge detection logic is synchronized to the GPIO peripheral clock.

The direction of the GPIO pin does not need to be input when using the pin to generate the interrupt or DMA event. When the GPIO pin is configured as input, transitions on the pin trigger interrupts or DMA events. When the GPIO pin is configured as output, software can toggle the GPIO output register to change the pin state and in turn trigger the interrupt or DMA event.

Note that the direction of the pin need not be input for interrupt generation to work. When the GPINT pin is configured as input, transitions on the pin trigger interrupts. When the GPINT pin is configured as output, firmware can toggle the GPIO output register to change the pin state, and in turn trigger interrupts.

Each interrupt output of GPINT signal are available at the module boundary. Each group of 16 GPINT signals also has their masked interrupt outputs ORed together to generate a per bank interrupt, available at the module boundary. The idea is to either connect individual interrupts or per bank interrupts to the system interrupt controller.

12.1.1.4.3.1 Interrupt Enable (per Bank)

The GPIO_BINTEN register provides interrupt enable/disable feature for each bank of 16 GPINT signals.

12.1.1.4.3.2 Trigger Configuration (per Bit)

Two internal registers, RIS_TRIG and FAL_TRIG, specify which edge of the GPj signal generates an interrupt or DMA event. Each bit in these two registers corresponds to a GPj pin. Table 12-1 describes the host CPU interrupt and DMA event generation of GPj pin based on the bit settings of the RIS_TRIG and FAL_TRIG registers.

Copyright © 2026 Texas Instruments Incorporated

Table 12-1. GPIO Interrupt and DMA Event Configuration Options

Clarifying configuration of GPIO interrupt generation RIS_TRIG Bit n	FAL_TRIG Bit n	Host CPU Interrupt and DMA Event Generation
0	0	GPINTj interrupt and DMA event is disabled
0	1	GPINTj interrupt and DMA event is triggered on falling edge of GPj signal
1	0	GPINTj interrupt and DMA event is triggered on rising edge of GPj signal
1	1	GPINTj interrupt and DMA event is triggered on both rising and falling edge of GPj signal

The RIS_TRIG and FAL_TRIG registers are not directly accessible or visible to the host CPU. These registers are accessed indirectly through four registers: SET_RIS_TRIG, CLR_RIS_TRIG, SET_FAL_TRIG, and CLR_FAL_TRIG. Writing 1 to a bit on the SET_RIS_TRIG register sets the corresponding bit on the RIS_TRIG register. Writing 1 to a bit of the CLR_RIS_TRIG register clears the corresponding bit on the RIS_TRIG register. Writing to the SET_FAL_TRIG and CLR_FAL_TRIG registers works the same way on the FAL_TRIG register.

Reading the SET_RIS_TRIG or CLR_RIS_TRIG register returns the value of the RIS_TRIG register. Reading from the SET_FAL_TRIG and CLR_FAL_TRIG register returns the value of the FAL_TRIG register.

To use the GPIO pins as sources for host CPU interrupts and DMA events, the associated bank interrupt enable register bit in GPIO_BINTEN must also be set to 1. For example, to enable GPIO0_19 (which is in bank 1), GPIO_BINTEN[1] = 1 should be set to enable interrupts for bank 1.

12.1.1.4.3.3 Interrupt Status and Clear (per Bit)

The INTSTAT registers provide interrupt status upon reading, and interrupt clear feature upon writing 1 to the corresponding bit position(s). Upon receiving an interrupt, the ISR can examine the interrupt status and clear the processed interrupts.

Note

The GPIO module generates an interrupt pulse on the individual GPINT interrupt in response to each occurrence of the specified edge condition. Therefore, for GPINT signals having their interrupts routed directly to the interrupt controller, it is not necessary to clear the status bits in this module. The interrupt status and clear register is a facility for the per-bank interrupt connection.

12.1.1.4.4 GPIO Interrupt Connectivity

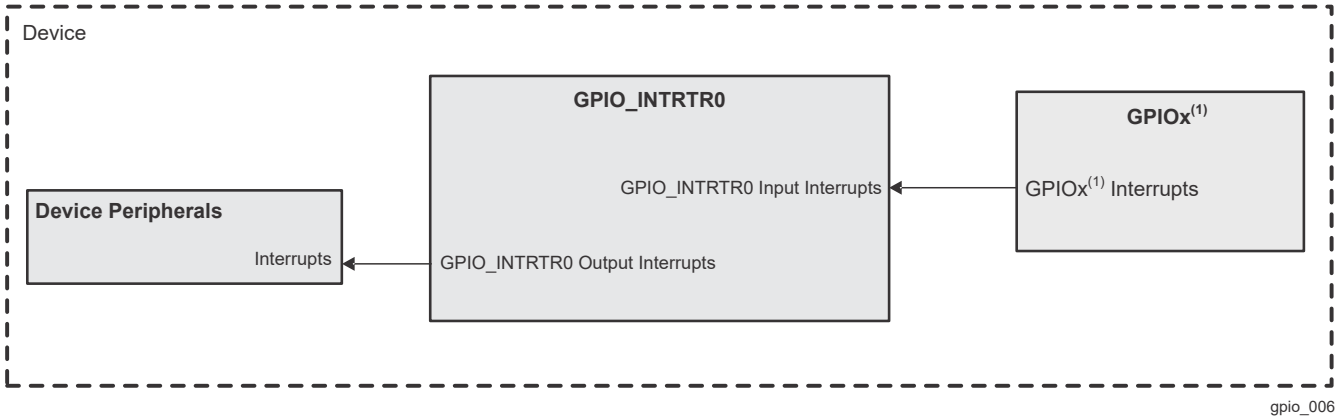
Because this device muxes GPIO signals with other functional signals, the availability of any particular GPIO and hence the usability of its associated interrupt will change based on the use case pin muxing. The large number of possible GPIO interrupt sources makes it impractical to route all interrupt events to each processing element. Since most applications do not typically require a large number of GPIO interrupts, the interrupt uncertainty is resolved by mapping all GPIO interrupts to a series of event muxes implemented using Interrupt Router (IntRouter) modules. These muxes allow any one of the available GPIO interrupts to be selected and passed on as an event to the various processor interrupt controllers and DMA controllers. Event selection is controlled through associated registers within each IntRouter.

The GPIO bank interrupts already represent a consolidation of the 16 GPIO interrupts associated with each bank and are routed directly to various interrupt controllers rather than through the GPIO IntRouters.

One of the GPIO pins has a potential use case as an Arm reset input and should therefore be routed as highest priority interrupt in the GIC and mapped to nFIQ in software.

Figure 12-4 shows the GPIO Interrupt Routers connectivity. Refer to *Interrupt Routers*, for more details on the GPIO Interrupt Routers connectivity.

Copyright © 2026 Texas Instruments Incorporated



A. (1): x represents a valid instance of GPIO in a domain.

Figure 12-4. GPIO Interrupt Router Connectivity

12.1.1.4.5 Emulation Halt Operation

The GPIO peripheral is not affected by emulation halts.

Copyright © 2026 Texas Instruments Incorporated

12.1.1.5 GPIO Programming Guide

12.1.1.5.1 GPIO Low-Level Programming Models

12.1.1.5.1.1 Global Initialization

12.1.1.5.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the general-purpose interface module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the environment and integration of the general-purpose interface. For more information, see *GPIO Environment* and *Module Integration*.

Table 12-2. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
LPSC0	Module reset must be enabled. For more information about the module configuration, see *Reset*.
PLLCTRL0	Interface and functional clocks must be enabled. For more information about the module configuration, see *Clocking*.
MCU_PLLCTRL0	Interface and functional clocks must be enabled. For more information about the module configuration, see *Clocking*.
GPIOMUX_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling GPIOMUX_INTRTR0 interrupts, see *Interrupts*.
MCU_GPIOMUX_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_GPIOMUX_INTRTR0 interrupts, see *Interrupts*.
Interconnects	For information about the MCU_CBASS0, and CBASS0 interconnects configuration, see *System Interconnect*.

12.1.1.5.1.1.2 GPIO Module Global Initialization

This procedure initializes the general-purpose Interface module after a power-on reset (POR) or software reset.

Table 12-3. GPIO Global Initialization

Step	Register/Bit Field/Programming Model	Value
Configure GPIO channels as input or output of the corresponding bank	DIR	-h
Interrupt requests configuration		
Configure detection events	SET_RIS_TRIG and/or SET_FAL_TRIG	-h
Clear interrupt status	INTSTAT	FFFFh
Enable interrupts for desired banks [0:8]	GPIO_BINTEN[8-0]	-h

12.1.1.5.1.2 GPIO Operational Modes Configuration

12.1.1.5.1.2.1 GPIO Read Input Register

Table 12-4. GPIO Read Input Register

Step	Register/Bit Field/Programming Model	Value
Read interrupt status of the corresponding bank	INTSTAT	-h
Read input register value	IN_DATA	-h
Clear interrupt status	INTSTAT	FFFF FFFFh

12.1.1.5.1.2.2 GPIO Set Bit Function

Table 12-5. GPIO Set Bit Function

Step	Register/Bit Field/Programming Model	Value
Write 1h to set desired bit(s) in SET_DATA register.	SET_DATA	-h

Copyright © 2026 Texas Instruments Incorporated

12.1.1.5.1.2.3 GPIO Clear Bit Function

Table 12-6. GPIO Clear Bit Function

Step	Register/Bit Field/Programming Model	Value
Write 1h to clear desired bit(s) in CLR_DATA register.	CLR_DATA	-h

Copyright © 2026 Texas Instruments Incorporated

12.1.2 Inter-Integrated Circuit (I2C) Interface

This section describes the Inter-Integrated Circuit (I2C) module in the device.

12.1.2.1 I2C Overview

The device contains multicontroller Inter-Integrated Circuit (I2C) controllers each of which provides an interface between a local host (LH), such as an Arm and any I^{2}C-bus-compatible device that connects via the I^{2}C serial bus. External components attached to the I^{2}C bus can serially transmit and receive up to 8 bits of data to and from the LH device through the 2-wire I^{2}C interface.

Each multicontroller I^{2}C module can be configured to act like a target or controller I^{2}C-compatible device.

I^{2}C instances may be implemented with dedicated, I^{2}C compliant, open-drain I/O buffers, or with standard LVCMOS I/O buffers. The I^{2}C instances associated with open-drain I/O buffers can support Hs-mode (up to 3.4 Mbps when the I/O buffers are operating at 1.8 V but limited to 400 kbps when the I/O buffers are operating at 3.3 V).

The I^{2}C instances associated with standard LVCMOS I/O buffers can support Fast-mode (up to 400 kbps). The LVCMOS I/O buffers being used on these ports are connected such they emulate open-drain outputs. This emulation is achieved by forcing a constant low output and disabling the output buffer to enter the Hi-Z state.

Refer to the device specific datasheet for details on which instances support which buffer type.

For the specific I/O timing characteristics of the different I^{2}C instances, see the device-specific Datasheet.

I2C Modules Overview shows the I2C modules overview.

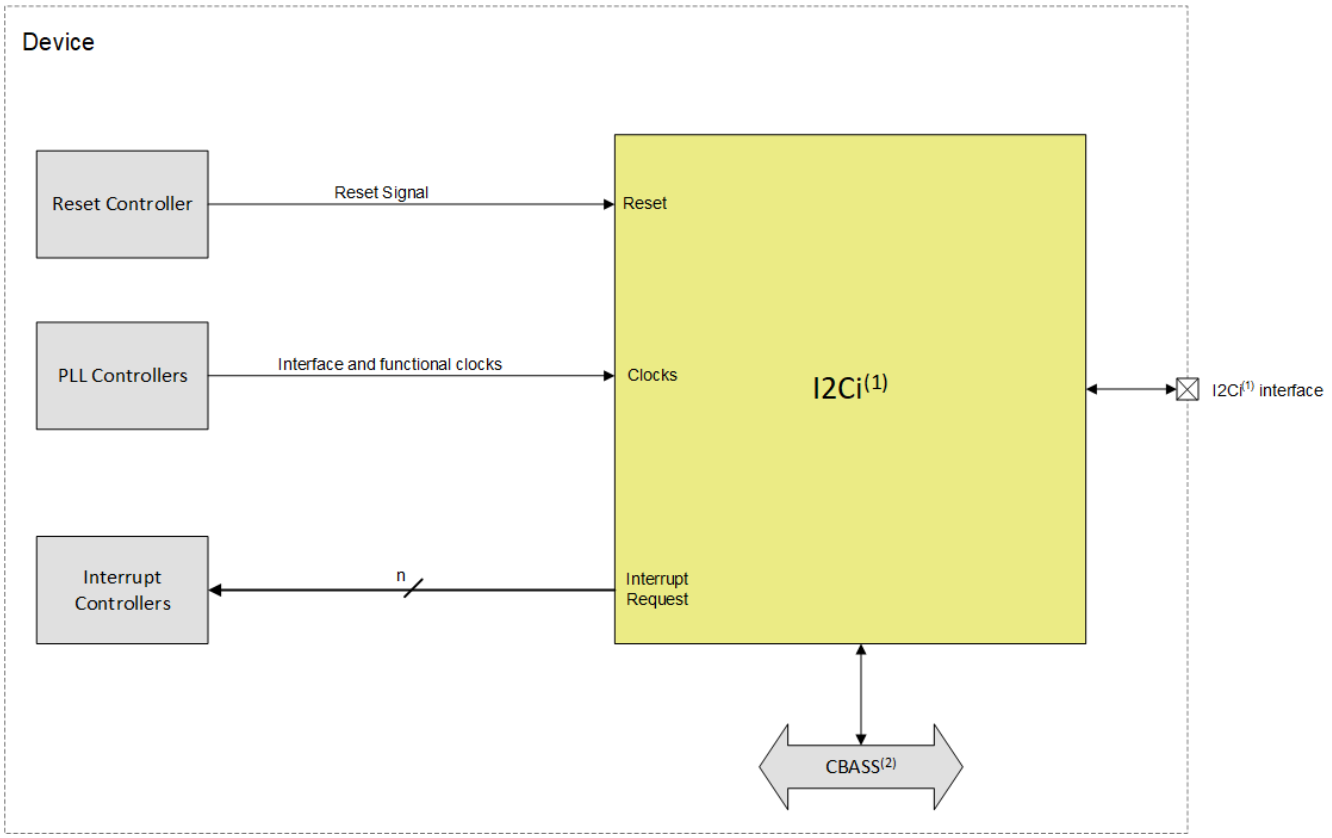


Figure 12-5. I2C Module Overview

Copyright © 2026 Texas Instruments Incorporated

Note

(1): i = 0 to (number of instances - 1) in the Domain

Note

(2): If MCU Domain is applicable, MCU_I2C instances are connected to MCU_CBASS

12.1.2.1.1 I2C Features

Each multicontroller I2C module has the following features:

- Compliant with Philips I^2C-bus specification version 2.1
- Supports a standard mode (up to 100 Kbps) and fast mode (up to 400 Kbps)
- Supports HS mode (up to 3.4 Mbps) only for instances with true open drain buffer and in 1.8 V mode
- 7-bit and 10-bit device addressing modes
- General call
- Start/Restart/Stop
- Multicontroller transmitter/target receiver mode
- Multicontroller receiver/target transmitter mode
- Combined controller transmit/receive and receive/transmit mode
- Built-in FIFO for buffered read
- Module enable/disable capability
- Programmable multitarget channel (responds to four separate addresses)
- Programmable clock generation
- 8-bit-wide data access
- Low power consumption
- Support Auto Idle mechanism
- Support Idle Request/Idle Acknowledge handshake mechanism
- Support for asynchronous wakeup mechanism
- Wide interrupt capability

12.1.2.1.2 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

12.1.2.1.3 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.2 I2C Environment

This section describes the I2C external connections (environment).

12.1.2.2.1 I2C Typical Application

Figure 12-6 shows the multicontroller I2C and their related connections with I²C-compliant devices.

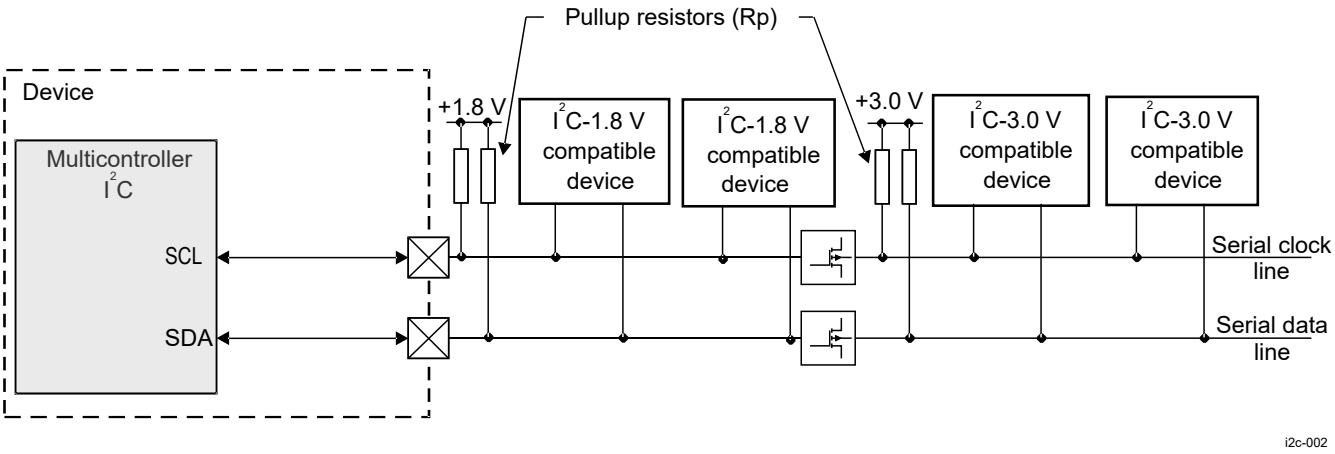
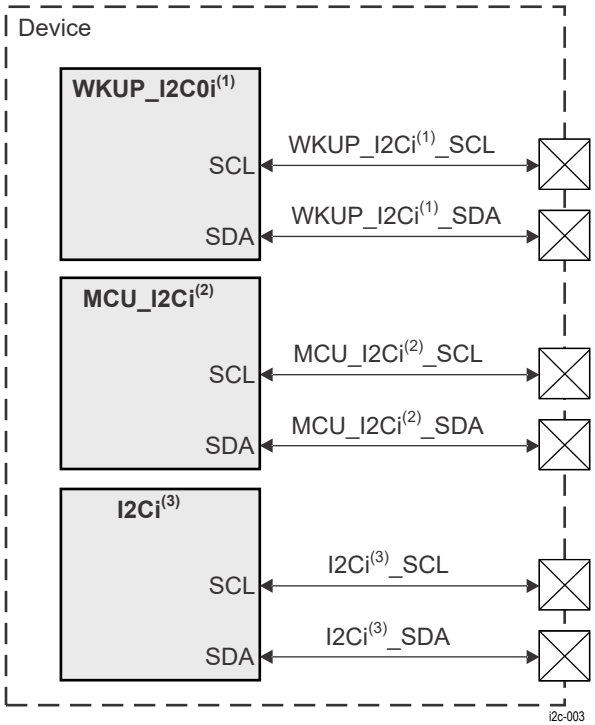


Figure 12-6. I2C and Typical Connections to I2C Devices

12.1.2.2.1.1 I2C Pins for Typical Connections in I2C Mode

Figure 12-7 shows the multicontroller I2C pins used for typical connections with I²C devices.



1. i = 0 to (number of devices - 1) in WKUP Domain, if applicable
2. i = 0 to (number of devices - 1) in MCU Domain, if applicable
3. i = 0 to (number of devices - 1) in MAIN Domain, if applicable

Figure 12-7. I2C Interface Signals

Copyright © 2026 Texas Instruments Incorporated

12.1.2.2.1.2 I2C Interface Typical Connections

Table 12-7 describes the I2C I/O signals.

Table 12-7. I2C I/O Signals

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value
			WKUP_I2Ci[2]	
SCL	WKUP_I2Ci[2]_SCL	I/O	I²C serial clock line.	1
SDA	WKUP_I2Ci[2]_SDA	I/O	I²C serial data line.	1
			MCU_I2Ci[2]	
SCL	MCU_I2Ci[2]_SCL	I/O	I²C serial clock line.	1
SDA	MCU_I2Ci[2]_SDA	I/O	I²C serial data line.	1
			I2Ci[2]	
SCL	I2Ci[2]_SCL	I/O	I²C serial clock line.	1
SDA	I2Ci[2]_SDA	I/O	I²C serial data line.	1

(1) I = Input; O = Output; I/O = Bidirectional
(2) i represents an I2C instance. See the device datasheet for available domains and I2C instances.

12.1.2.2.1.3

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.2.2 I2C Typical Connection Protocol and Data Format

12.1.2.2.1 I2C Serial Data Format

The I2C controller operates in 8-bit word data format (byte write access supported for the last access). Each byte transmitted or received on the serial data line is 8 bits long. The number of bytes that can be transmitted or received is not restricted. The data is transferred with the most-significant bit (MSB) first. In receiver mode, each byte is followed by an acknowledge bit from the I2C module.

Figure 12-8 shows a typical I²C communication format.

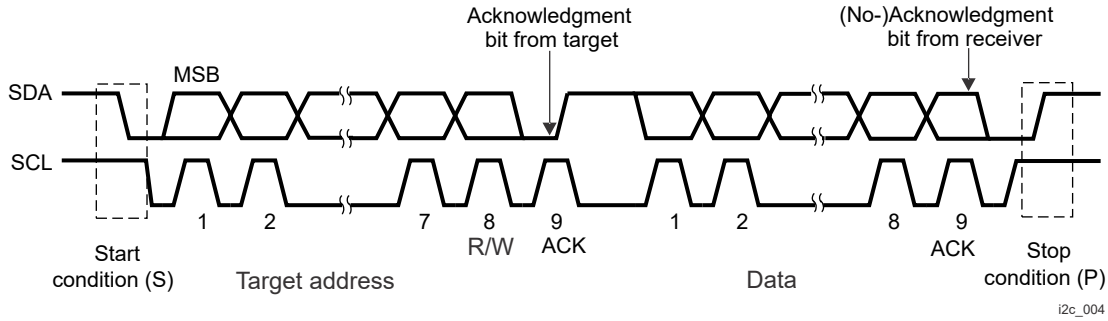


Figure 12-8. I2C Data Transfer

12.1.2.2.2 I2C Data Validity

The data on the serial data line (SDA) must be stable during the high period of the serial clock line. The high and low states of the data line can change only when the clock signal on the serial clock line (SCL) is low.

Figure 12-9 is an example of data validity requirements.

Copyright © 2026 Texas Instruments Incorporated

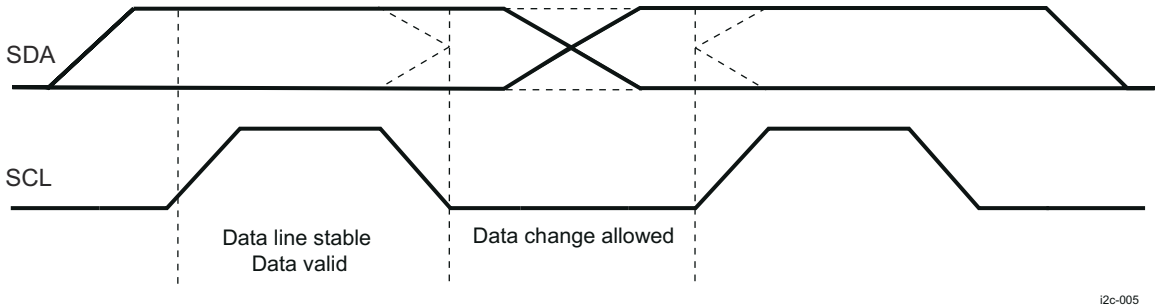


Figure 12-9. I2C Bit Transfer on the I2C Bus

12.1.2.2.2.3 I2C Start and Stop Conditions

The I2C module generates start (S) and stop (P) conditions when it is configured as a controller.

- An S condition is a high-to-low transition on the serial data line while serial clock line is high.
- A P condition is a low-to-high transition on the serial data line while serial clock line is high.

The bus is considered busy after the S condition (the I2C_IRQSTATUS_RAW [12] BB bit is 1 to indicate that the bus is busy) and free after the P condition (the I2C_IRQSTATUS_RAW [12] BB bit is 0 to indicate that the bus is free).

Figure 12-10 shows the waveforms that occur during an S and a P condition.

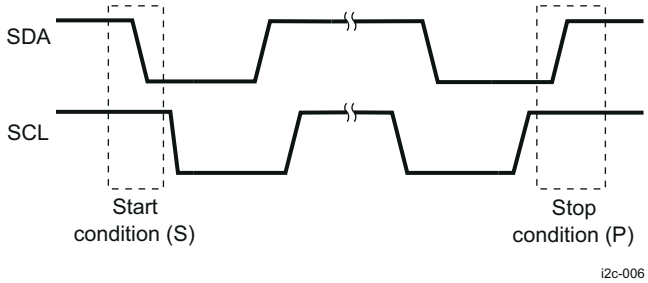


Figure 12-10. I2C S and P Condition Events

Note

I2C controller does not support messages non-compliant with I^2C standard. Void messages are non-standard I^2C messages and will lockup the controller. A void message is a START condition followed by a STOP condition, in other words, while the bus is free the SDA line is pulled low (START) and then released (STOP). This would result in a timeout (software) of the next controller transfer which would never complete. A soft reset of the controller is recommended for recovery.

12.1.2.2.2.4 I2C Addressing

The I2C module supports two data formats in fast/standard (F/S) and HS modes:

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start (Sr) condition

12.1.2.2.2.4.1 Data Transfer Formats in F/S Mode

Figure 12-11 shows the I2C data transfer formats in F/S mode.

Copyright © 2026 Texas Instruments Incorporated

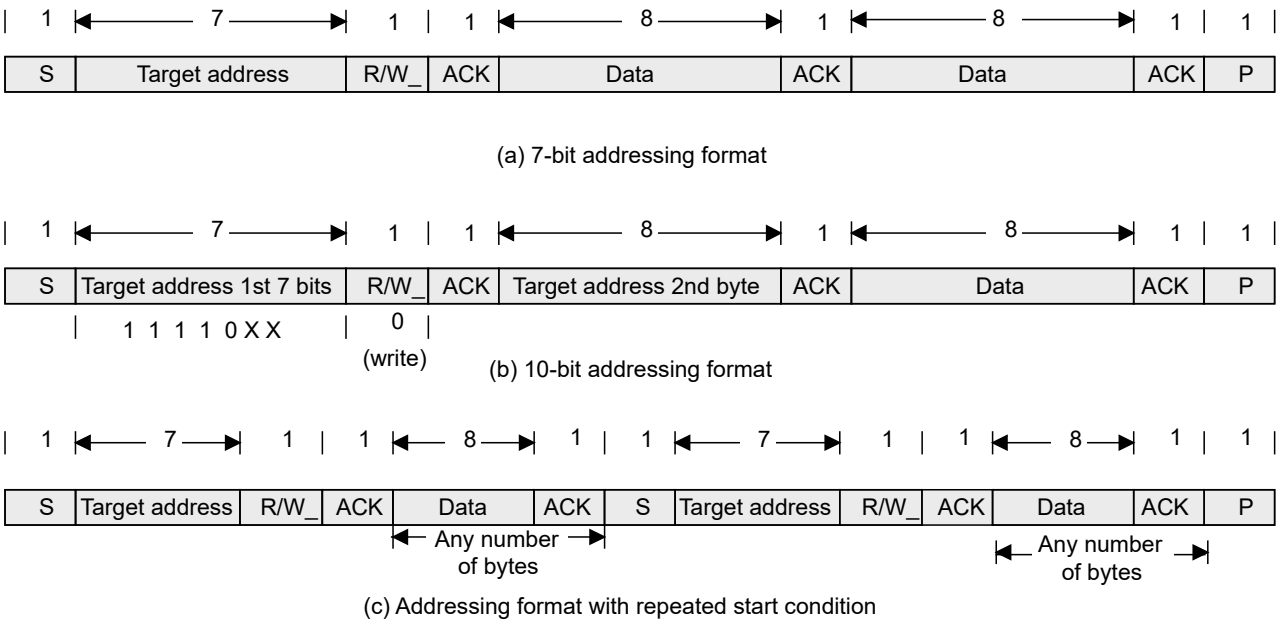


Figure 12-11. I2C Data Transfer Formats in F/S Mode

The first word after an S condition consists of 8 bits. In acknowledge mode, an extra dedicated acknowledgment bit is inserted after each byte.

In addressing formats with 7-bit addresses, the first byte is composed of 7 MSB target address bits and 1 least-significant bit (LSB) R/W_ bit.

The LSB R/W_ bit of the address byte indicates the transmission direction of the data bytes that follow it. If R/W_ is 0, the controller writes data to the selected target; if it is 1, the controller reads data from the target.

In addressing formats with 10-bit addresses, the structure of the first byte is 11110XXY, where XX is the two MSBs of the 10-bit addresses, and Y is the R/W_ bit. If the R/W_ bit is 0, the next byte contains the last 8 bits of the target address. If the R/W_ bit is 1, the next byte contains data transmitted from the target to the controller.

12.1.2.2.2.4.2 Data Transfer Format in HS Mode

Figure 12-12 shows the I2C data transfer format in HS mode.

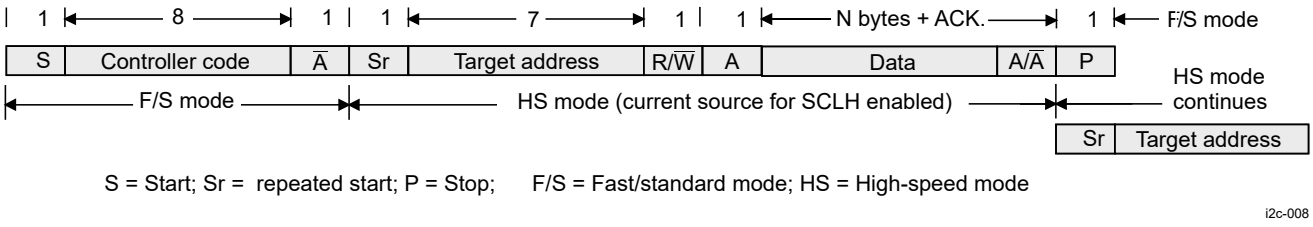


Figure 12-12. HS I2C Data Transfer in HS Mode

Each multicontroller I2C can also operate in HS mode. In this case, after the S condition, the module, which is in F/S mode, writes the controller code address (0x00001XXX, where XXX is the variable portion of the controller code) on the bus. No device connected on the same bus acknowledges this address. The module switches the clock to the HS clock and after an Sr condition, and sends the target address and the data, as shown in Figure 12-12.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.2.2.5 I2C Controller Transmitter

In controller transmitter mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in sync with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL is held low when the intervention of the processor is required (XUDF) after a byte is transmitted.

12.1.2.2.2.6 I2C Controller Receiver

Controller receiver mode can be entered only from controller transmitter mode. With any of the address formats (a), (b), or (c) (see Figure 12-11), if R/W_ is high, the module enters controller receiver mode after the target address byte and bit R/W_ are transmitted. Serial data bits received on bus line SDA are shifted in synchronization with the self-generated clock pulses on SCL.

12.1.2.2.2.7 I2C Target Transmitter

Target transmitter mode can be entered only from target receiver mode. With any of the address formats (a), (b), or (c) (see Figure 12-11), the target transmitter is entered if the target address byte is the same as its own address and bit R/W_ is transmitted, if R/W_ is high. The target transmitter shifts the serial data out on the data line SDA in sync with the clock pulses that are generated by the controller device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (XUDF).

12.1.2.2.2.8 I2C Target Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in sync with the clock pulses on SCL that are generated by the controller device. It does not generate the clock but it can hold clock line SCL low while intervention of the LH is required (ROVR) after a byte is received.

12.1.2.2.2.9 I2C Bus Arbitration

If two or more controller transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the target receiver mode, sets the arbitration lost (I2C_IRQSTATUS_RAW[0] AL) flag, and generates the arbitration lost interrupt. I2C Arbitration Between Controller Transmitters shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

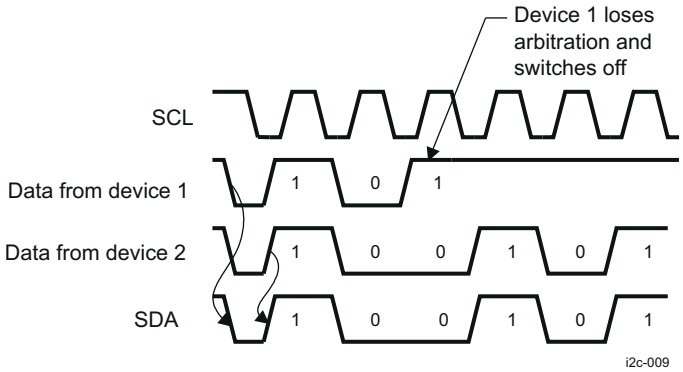


Figure 12-13. I2C Arbitration Between Controller Transmitters

12.1.2.2.2.10 I2C Clock Generation and Synchronization

Under normal conditions, only one controller device generates the clock signal - SCL. However, there are two or more controller devices during the arbitration procedure, and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device

Copyright © 2026 Texas Instruments Incorporated

with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period. If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT-state. In this way a target can slow down a fast controller and the slow device can create enough time to store a received byte or prepare a byte to be transmitted (clock stretching).

Note

In case the SCL or SDA lines are stuck low, a bus clearing operation is supported:

- If the clock line (SCL) is stuck low, the preferential procedure is to reset the bus using the hardware reset signal if the I^2C devices have hardware reset inputs. If the I^2C devices do not have hardware reset inputs, cycle power to the devices to activate the mandatory internal power-on reset (POR) circuit.
- If the data line (SDA) is stuck low, the controller should send nine clock pulses. The device that held the bus low should release it sometime within those nine clocks. If not, then use the hardware reset or cycle power to clear the bus.

Figure 12-14 shows clock synchronization.

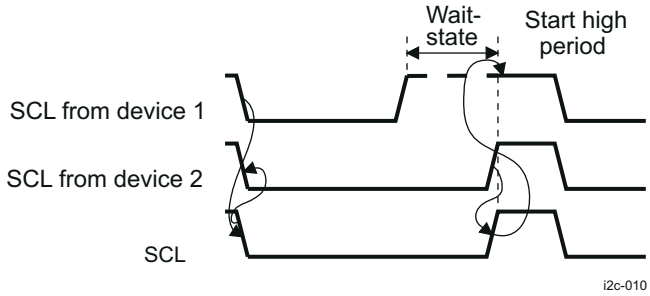


Figure 12-14. I2C Clock Generators Synchronization

Copyright © 2026 Texas Instruments Incorporated

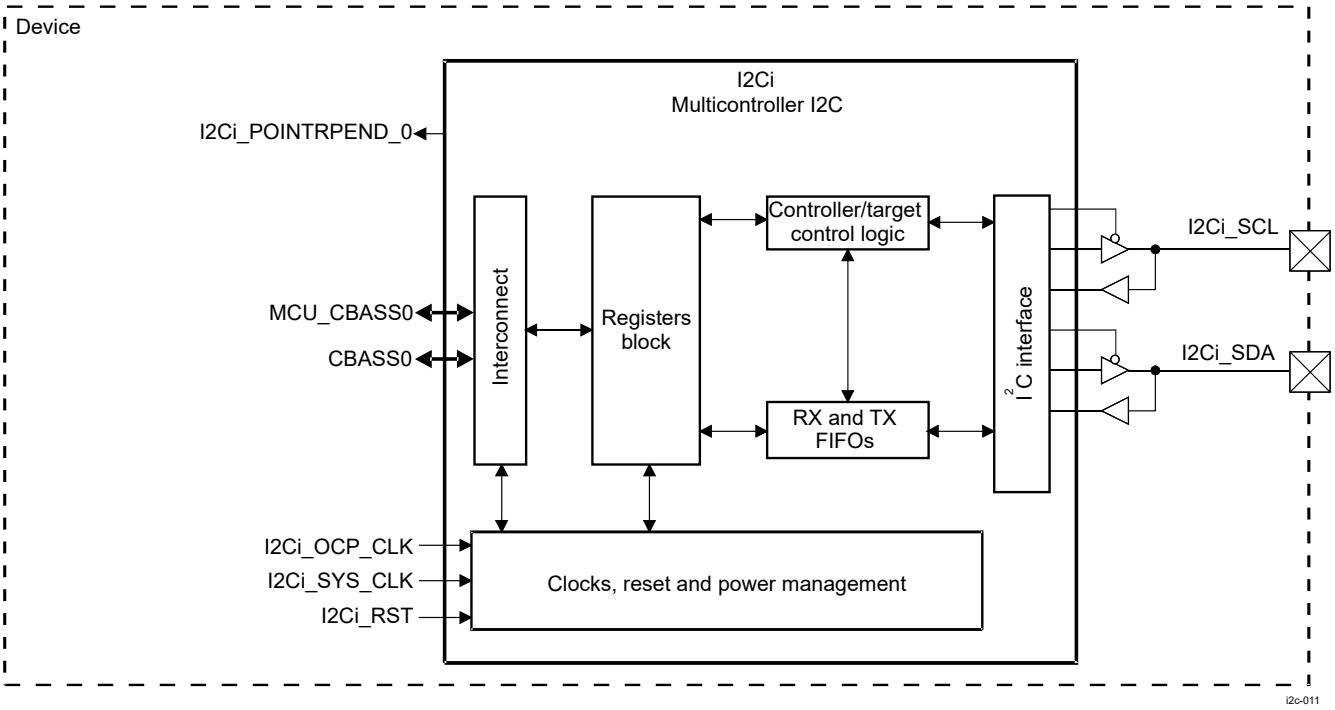
12.1.2.3 I2C Functional Description

12.1.2.3.1 I2C Block Diagram

Note

DMA mode and SCCB Protocol are not supported on this family of devices.

Figure 12-15 presents the multicontroller I2C block diagram.



A. i = 0 to number of devices in MCU domain -1, if applicable.
B. i = 0 to number of devices in MAIN domain -1, if applicable.

Figure 12-15. I2C Block Diagram

The ten multicontroller I2C can be configured in F/S I^2C mode or HS I^2C mode. The operation mode is selected by configuring the I2C_CON[13-12] OPMODE bit field.

Table 12-8 lists the available operation modes.

Table 12-8. I^2C Operation Mode Selection

Operation Mode	Value of I2C_CON[13-12] OPMODE
F/S I^2C	0x0
HS I^2C	0x1
SCCB	0x2
Reserved (not used)	0x3

12.1.2.3.2 I2C Clocks

12.1.2.3.2.1 I2C Clocking

Each multicontroller I2C uses the SYS_CLK functional clock in the Device Configuration section. The internal sampling clock INTERNAL_CLK is generated by dividing the functional clock by the I2C_PSC[7-0] PSC bit field value + 1 in F/S mode, or in the first phase of HS mode; or by directly using the functional clock in the second phase of HS mode (prescaler is bypassed).

Copyright © 2026 Texas Instruments Incorporated

The low time of the SCLL signal is determined by the I2C_SCLL[7-0] SCLL bit field in F/S mode and in the first phase of HS mode; or by the I2C_SCLL[15-8] HSSCLL bit field in the second phase of HS mode.

The high time of the SCLH signal is determined by the I2C_SCLH[7-0] SCLH bit field in F/S mode and in the first phase of HS mode; or by the I2C_SCLH[15-8] HSSCLH bit field in the second phase of HS mode.

Table 12-9 lists the t_{LOW} and t_{high} values in controller mode only (in target mode, the I2C controller does not generate the I^2C clock).

Table 12-9. I^2C t_{LOW} and t_{high} Values of the I^2C Clock

Mode	Clock	t_{LOW}	t_{high}
F/S or HS first phase	INTERNAL_CLK = SYS_CLK / (I2C_PSC[7-0] PSC bit field + 1)	(I2C_SCLL[7-0] SCLL bit field value + 7) × INTERNAL_CLK period	(I2C_SCLH[7-0] SCLH bit field value + 5) × I2Ci_INTERNAL_CLK period
HS second phase	SYS_CLK	(I2C_SCLL[15-8] HSSCLL bit field value + 7) × SYS_CLK period	(I2C_SCLL[15-8] HSSCLH bit field value + 5) × I2Ci_SYS_CLK period

Note

i = 0 to number of devices in MCU domain -1, if applicable.

Note

i = 0 to number of devices in MAIN domain -1, if applicable.

Note

For HS mode, the I2C_SCLL[15-8] HSSCLL and I2C_SCLL[7-0] SCLL bit fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

For HS mode, the I2C_SCLH[15-8] HSSCLH and I2C_SCLH[7-0] SCLH bit fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

Note

The equations in Table 12-9 give the SCLL timing values for SCLL/SCLH/HSSCLL/HSSCLH at I2C controller outputs. Actual t_{low} and thigh periods may vary depending on the board (the load capacitance on the SCLL signal). If necessary, any adjustments to the SCLL/SCLH/HSSCLL/HSSCLH values must be determined by measurements of actual SCL signal on the board.

CAUTION

During active mode (the I2C_CON[15] I2C_EN bit is set to 1), make no changes to the I2C_SCLL and I2C_SCLH registers. Changes may result in unpredictable behavior.

Table 12-10 lists the register values for obtaining the maximum I^2C bit rates and the maximum period of the filtered spikes in F/S mode and HS mode.

Table 12-10. I2C Register Values for Maximum I2C Bit Rates in I2C F/S, I2C HS Modes (1)

	I^2C Mode for			Description
	Standard Mode	Fast Mode	High-Speed Mode(2)	
SYS_CLK frequency (MHz)	96			
OCP_CLK frequency (MHz)	133			

Copyright © 2026 Texas Instruments Incorporated

Table 12-10. I2C Register Values for Maximum I2C Bit Rates in I2C F/S, I2C HS Modes (1) (continued)

	I²C Mode for			Description
I2C_PSC[7-0] PSC	**23**	**9**	**1**	Prescaler value for F/S and HS modes
INTERNAL_CLK frequency (MHz)	4	9.6	96	
I2C_SCLL[7-0] SCLL	**13**	**7**	**115**	Value for F/S mode and first phase of HS mode
I2C_SCLH[7-0] SCLH	**15**	**5**	**113**	Value for F/S mode and first phase of HS mode
Maximum bit rate (Mbps)	0.1	0.4	0.4	F/S mode and first phase in HS mode maximum bit rate
Maximum filter period (ns)	250	104.2	10	
I2C_SCLL[15-8] HSSCLL			**12**	Values for second phase of HS mode
I2C_SCLH[15-8] HSSCLH			**5**	Values for second phase of HS mode
HS mode maximum bit rate (Mbps)			3.31	HS mode maximum bit rate
Maximum filter period (ns)			10	

(1) Programmable fields are in bold.
(2) HS mode is not supported on this family of devices.

Note

This table presents informative values only for the configuration parameters and the I²C bus performance obtained according to these values. The delays added by the analog pads are not considered in these figures.

Note

For MCU_I2C[0-1]

For I2C[0-3]

I2Ci_INTERNAL_CLK freq = I2Ci_SYS_CLK / (PSC +1)

F/S filter period = 1 / I2Ci_INTERNAL_CLK

HS filter period = 1 / I2Ci_SYS_CLK freq

HS bit rate = I2Ci_SYS_CLK freq / (HSSCLL+ 7 + HSSCLH + 5)

FS bit rate = I2Ci_INTERNAL_CLK / (SCLL+ 7 + SCLH + 5)

12.1.2.3.2.2 I2C Automatic Blocking of the I2C Clock Feature

This feature offers the possibility for the LH to command the blocking of the I²C clock after the target addressing phase, when the I2C controller is addressed by an external controller device using a certain Own Address.

The release of the I²C clock can be performed independently for each Own Address (I2C_OA, and I2C_OAx registers, where x = 1, 2, 3) by deasserting the corresponding bit in the I2C_SBLOCK register.

12.1.2.3.3 I2C Software Reset

Each multicontroller I2C supports the software reset by accessing the I2C_SYSC[1] SRST bit (1: reset; 0: normal mode).

The software reset status can be checked by accessing the I2C_SYSS[0] RDONE bit (1: reset is done; 0: reset is ongoing).

Copyright © 2026 Texas Instruments Incorporated

To do a software reset, the following steps must be done:

1. Ensure that the module is disabled (clear the I2C_CON[15] I2C_EN bit to 0).
2. Set the I2C_SYSC[1] SRST bit to 1.
3. Enable the module by setting I2C_CON[15] I2C_EN bit to 1.
4. Check the I2C_SYSS[0] RDONE bit until it is set to 1 to indicate the software reset is complete.

Note

The I2C_CON[15] I2C_EN bit can hold the functional clock domain of the multicontroller I2C in reset after the device reset has been released. When the system bus reset is removed, this bit remains cleared. The functional part of the I2C controller is held in reset state while this bit is 0, and all configuration registers can be accessed.

The I2C_CON[15] I2C_EN bit must be set to 1 to enable the functional part of the I2C controller.

The I2C_SYSS[0] RDONE bit is asserted only after the module is enabled by setting the I2C_CON[15] I2C_EN bit to 1.

12.1.2.3.4 I2C Power Management

Table 12-11 describes power-management features available for the multicontroller I2C.

Note

For information about source clock gating, see *Power* in the *Device Configuration*.

Note

Some of the I2C features described in this section may not be supported on this family of devices. See the *Module Integration* section for more information about unsupported features..

Table 12-11. I2C Local Power-Management Features

Feature	Registers	Description
Clock auto gating	I2C_SYSC[0] AUTOIDLE	This bit allows a local power optimization inside the module.
Target idle modes	I2C_SYSC[4-3] IDLEMODE	Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available.
Clock activity	I2C_SYSC[9-8] CLOCKACTIVITY	For configuration details, see Table 12-12.
Global wake-up enable	I2C_SYSC[2] ENAWAKEUP	This bit enables the wake-up feature at module level.

Table 12-12. I2C Clock Activity Settings

I2C_SYSC[9-8] CLOCKACTIVITY	Clock State When Module is in IDLE State		Features Available/Unavailable When Module is in IDLE State
	OCP_CLK	SYS_CLK	
00	OFF	OFF	Both clocks are disabled.
10	OFF	ON	Interface clock is disabled; Functional clock is enabled
01	ON	OFF	Functional clock is disabled; Interface clock is enabled
11	ON	ON	Both clocks are enabled.

12.1.2.3.5 I2C Interrupt Requests

The I2C controller must allocate a minimum of five registers for interrupts.

- Interrupt Raw Status (I2C_IRQSTATUS_RAW)
- Interrupt Enabled Status (I2C_IRQSTATUS)
- Interrupt Enable Set (I2C_IRQENABLE_SET)
- Interrupt enable Clear (I2C_IRQENABLE_CLR)

Copyright © 2026 Texas Instruments Incorporated

- End of interrupt (I2C_EOI)

End of Interrupt (I2C_EOI) is used by software to trigger an interrupt service completion.

Table 12-13 lists the event flags, and their mask, that can cause module interrupts.

Table 12-13. I2C Events

Event Flag	Event Unmask	Event Mask	Description
I2C_IRQSTATUS[0] AL	I2C_IRQENABLE_SET[0] AL_IE	I2C_IRQENABLE_CLR[0] AL_IE	Arbitration lost. This bit is automatically set by the hardware when it loses the arbitration in controller transmit mode, an interrupt is signaled to the host.
I2C_IRQSTATUS[1] NACK	I2C_IRQENABLE_SET[1] NACK_IE	I2C_IRQENABLE_CLR[1] NACK_IE	No acknowledgement. Bit is set when No Acknowledge is received, an interrupt is signaled to the host.
I2C_IRQSTATUS[2] ARDY	I2C_IRQENABLE_SET[2] ARDY_IE	I2C_IRQENABLE_CLR[2] ARDY_IE	Register access ready. When set to 1 it indicates that previous access has been performed and registers are ready to be accessed again. An interrupt is signaled to the host.
I2C_IRQSTATUS[3] RRDY	I2C_IRQENABLE_SET[3] RRDY_IE	I2C_IRQENABLE_CLR[3] RRDY_IE	Receive data ready. Set to 1 by core when in receiver mode, a new data can be read. An interrupt is signaled to the host.
I2C_IRQSTATUS[4] XRDY	I2C_IRQENABLE_SET[4] XRDY_IE	I2C_IRQENABLE_CLR[4] XRDY_IE	Transmit data ready. Set to 1 by core when transmitter is ready for new data. An interrupt is signaled to the host.
I2C_IRQSTATUS[5] GC	I2C_IRQENABLE_SET[5] GC_IE	I2C_IRQENABLE_CLR[5] GC_IE	General call. Set to 1 by core when General Call address was detected. An interrupt is signaled to the host.
I2C_IRQSTATUS[6] STC	I2C_IRQENABLE_SET[6] STC_IE	I2C_IRQENABLE_CLR[6] STC_IE	Start condition detected. An interrupt is signaled to the host.
I2C_IRQSTATUS[7] AERR	I2C_IRQENABLE_SET[7] AERR_IE	I2C_IRQENABLE_CLR[7] AERR_IE	Bus Access Error. An interrupt is signaled to the host.
I2C_IRQSTATUS[8] BF	I2C_IRQENABLE_SET[8] BF_IE	I2C_IRQENABLE_CLR[8] BF_IE	Bus free. An interrupt is signaled to the host.
I2C_IRQSTATUS[9] AAS	I2C_IRQENABLE_SET[9] AAS_IE	I2C_IRQENABLE_CLR[9] AAS_IE	Address recognized as target. An interrupt is signaled to the host.
I2C_IRQSTATUS[10] XUDF	I2C_IRQENABLE_SET[10] XUDF_IE	I2C_IRQENABLE_CLR[10] XUDF_IE	Transmit underflow. An interrupt is signaled to the host.
I2C_IRQSTATUS[11] ROVR	I2C_IRQENABLE_SET[11] ROVR_IE	I2C_IRQENABLE_CLR[11] ROVR_IE	Receive overrun. An interrupt is signaled to the host.
I2C_IRQSTATUS[12] BB	N/A	N/A	Bus busy indicator
I2C_IRQSTATUS[13] RDR	I2C_IRQENABLE_SET[13] RDR_IE	I2C_IRQENABLE_CLR[13] RDR_IE	Receive draining. An interrupt is signaled to the host.
I2C_IRQSTATUS[14] XDR	I2C_IRQENABLE_SET[14] XDR_IE	I2C_IRQENABLE_CLR[14] XDR_IE	Transmit draining. An interrupt is signaled to the host.

12.1.2.3.6 I2C Programmable Multitarget Channel Feature

This feature allows each multicontroller I2C to be addressed using four separate Own Addresses configured in the I2C_OA and I2C_OAx registers (where x = 1, 2, 3). An additional register (I2C_ACTOA) is used to indicate to the LH which address is used by the external controller to communicate with the I2C controller.

Each Own Address can be independently configured in 7-bit or 10-bit mode by setting the corresponding bit (I2C_CON[7] XOA0, I2C_CON[6] XOA1, I2C_CON[5] XOA2, or I2C_CON[4] XOA3).

Copyright © 2026 Texas Instruments Incorporated

12.1.2.3.7 I2C FIFO Management

The depth of the RX and TX FIFOs can be checked by reading the I2C_BUFSTAT[15-14] FIFODEPTH bit field (0x0: 8 bytes, 0x1: 16 bytes, 0x2: 32 bytes, and 0x3: 64 bytes).

12.1.2.3.7.1 I2C FIFO Interrupt Mode

In FIFO interrupt mode (relevant interrupts enabled by the I2C_IRQENABLE_SET register), an interrupt signal informs the processor of the receiver and transmitter status. These interrupts are raised when the RX/TX FIFO thresholds (defined by the I2C_BUF[13-8] RXTRSH bit field value + 1 for the RX FIFO or the I2C_BUF[5-0] TXTRSH bit field value + 1 for the TX FIFO) are reached; the interrupt signals instruct the LH to transfer data to the destination (from the I2C controller in receive mode and/or from any source to the I2C controller FIFO in transmit mode).

Figure 12-16 and Figure 12-17 show receive and transmit operations, respectively, from a FIFO management point of view.

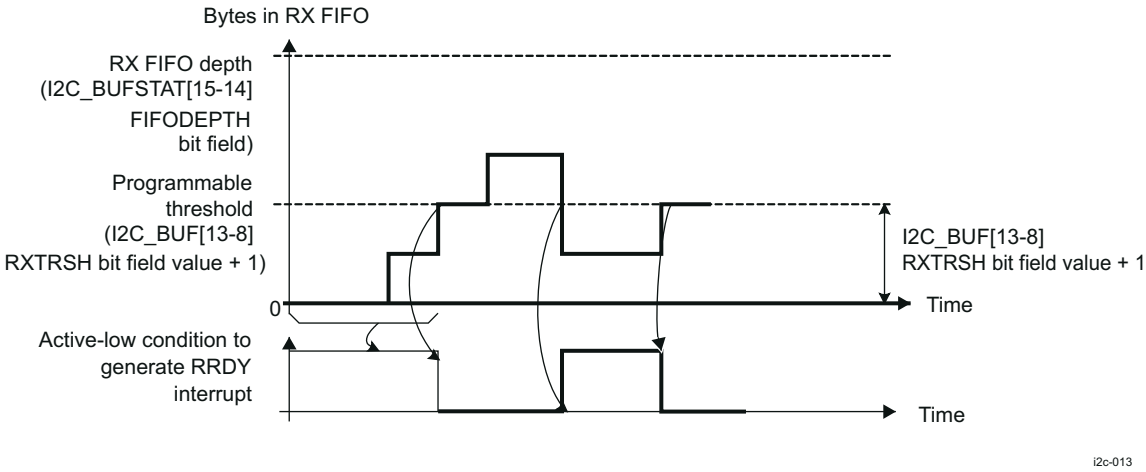


Figure 12-16. I2C Receive FIFO Interrupt Request Generation

In Figure 12-16 , the RRDY interrupt condition shows that the condition for generating an RRDY interrupt is achieved. The interrupt request is generated when this signal is active, and it can be cleared only by the LH by writing 1 in the corresponding bit. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In receive mode, an RRDY interrupt is generated as soon as the FIFO reaches its receive threshold (I2C_BUF[13-8] RXTRSH bit field value + 1). The interrupt can be deasserted only when the LH has handled enough bytes to make the number of bytes in the RX FIFO lower than the programmed threshold. For each interrupt, the LH can be configured to read a number of bytes equal to the value of the RX FIFO threshold.

Copyright © 2026 Texas Instruments Incorporated

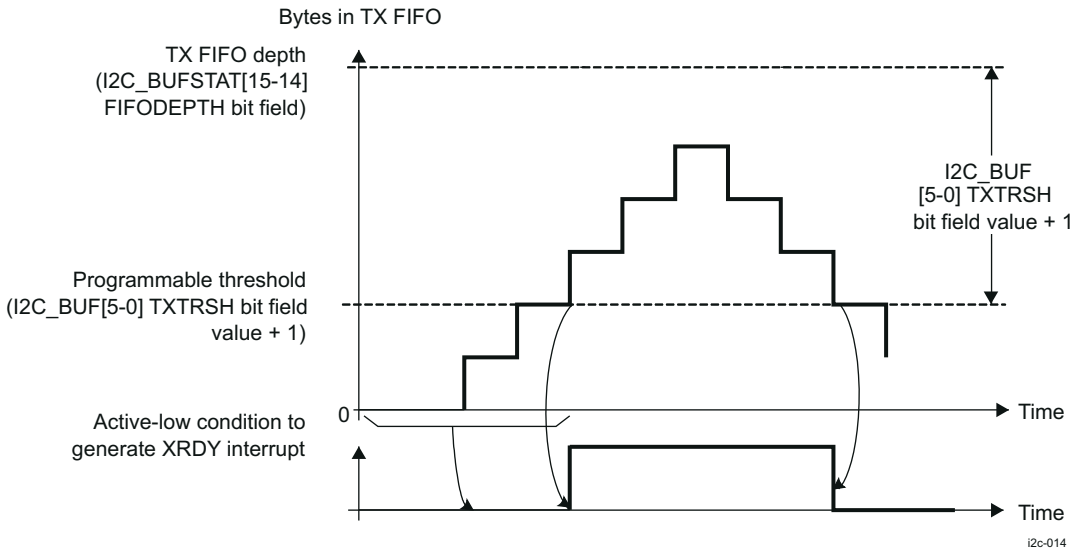


Figure 12-17. I2C Transmit FIFO Interrupt Request Generation

In Figure 12-17 , the XRDY interrupt condition shows that the condition for generating an XRDY interrupt is achieved. The interrupt request is generated when TX FIFO is empty or when the TX FIFO threshold is not reached, and the LH can clear the XRDY status bit by setting the I2C_IRQENABLE_CLR[4] XRDY_IE bit to 1 after transmitting the configured number of bytes. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In interrupt mode, the module offers two options for the LH application to handle the interrupts:

- When detecting an interrupt request (XRDY or RRDY type), the LH can write/read 1 data byte to/from the TX/RX FIFO and then clear the interrupt. The module reasserts the interrupt until the interrupt condition is not met.
- When detecting an interrupt request (XRDY or RRDY type), the LH can be programmed to write/read the amount of data bytes specified by the corresponding FIFO threshold (I2C_BUF[5-0] TXTRSH + 1 or I2C_BUF[5-0] RXTRSH + 1). In this case, the interrupt condition is cleared and the next interrupt is asserted again when the XRDY or RRDY condition is met again.

If the second-interrupt-serving approach is used, an additional mechanism (draining feature) is implemented for cases where the transfer length is not a multiple of the FIFO threshold value (see Section 12.1.2.3.7.3, *Draining Feature [I2C Mode Only]*).

Note

In target transmit mode (the I2C_CON[10] MST bit is cleared and the I2C_CON[9] TRX bit is set to 1), the draining feature must not be used, because the transfer length is not known at configuration time, and the external controller can end the transfer at any point by not acknowledging 1 data byte. If the draining feature is used in target transmit mode, data can remain in the TX FIFO without being transmitted over the I^2C bus. In this case, the TX FIFO must be cleared by setting the I2C_BUF[6] TXFIFO_CLR bit.

12.1.2.3.7.2 I2C FIFO Polling Mode

In FIFO polling mode (the I2C_IRQENABLE_SET[4] XRDY_IE and I2C_IRQENABLE_SET[3] RRDY_IE bits are disabled), the status of the module (receiver or transmitter) can be checked by polling the I2C_IRQSTATUS_RAW[4] XRDY and the I2C_IRQSTATUS_RAW[3] RRDY bits (the I2C_IRQSTATUS_RAW[13] RDR and I2C_IRQSTATUS_RAW[14] XDR bits can also be polled if the draining feature is enabled). The I2C_IRQSTATUS_RAW[4] XRDY and I2C_IRQSTATUS_RAW[3] RRDY bits accurately reflect the interrupt conditions described in the discussion of FIFO interrupt mode.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.3.7.3 I2C Draining Feature

Note

DMA mode and SCCB Protocol are not supported on this family of devices.

The draining feature is implemented to handle the end of a transfer whose length is not a multiple of the FIFO threshold values (the I2C_BUF[13-8] RXTRSH bit field value + 1 for the RX threshold and the I2C_BUF[5-0] TXTRSH field value + 1 for the TX threshold). It can also transfer the remaining number of bytes (because the threshold is not reached).

This feature prevents the LH or the DMA controller from trying more FIFO accesses than necessary (for example, to generate at the end of a transfer a DMA RX request having fewer bytes in the FIFO than the configured DMA transfer length). Otherwise, an AERR interrupt is generated by the I2C_IRQSTATUS_RAW[7] AERR bit.

The draining mechanism generates an interrupt using the I2C_IRQSTATUS_RAW[13] RDR or I2C_IRQSTATUS_RAW[14] XDR bit at the end of the transfer, informing the LH that it must check the amount of data left to be transferred (the I2C_BUFSTAT[13-8] RXSTAT or I2C_BUFSTAT[5-0] TXSTAT bit fields) and enable the draining feature of the DMA controller by reconfiguring the DMA transfer length according to this value (when the DMA mode is enabled) or perform only the required number of data accesses (when the DMA mode is disabled).

In receive mode (controller or target), if the RX FIFO threshold (the I2C_BUF[13-8] RXTRSH bit field value + 1) is not reached, but the transfer ends on the I^2C bus and data remains in the RX FIFO (less than the threshold), the receive draining interrupt (the I2C_IRQSTATUS_RAW[13] RDR bit) is asserted to inform the LH that it can read the amount of data in the RX FIFO (the I2C_BUFSTAT[13-8] RXSTAT bit field). The LH performs a number of data read accesses equal to the I2C_BUFSTAT[13-8] RXSTAT bit field (interrupt or polling mode), or reconfigures the DMA controller with the required value to drain the FIFO.

In controller transmit mode, if the TX FIFO threshold (the I2C_BUF[5-0] TXTRSH bit field value + 1) is not reached, but the amount of data remaining to be written in the TX FIFO is less than the threshold, the transmit draining interrupt (the I2C_IRQSTATUS_RAW[14] XDR bit) is asserted to inform the LH that it can read the amount of data remaining to be written in the TX FIFO (the I2C_BUFSTAT[5-0] TXSTAT bit field). The LH must write the required number of data bytes specified by the I2C_BUFSTAT[5-0] TXSTAT bit field value or reconfigure the DMA controller with the value required to transfer the last bytes to the FIFO.

In controller mode, the LH can alternately not check the values of the I2C_BUFSTAT[5-0] TXSTAT and I2C_BUFSTAT[13-8] RXSTAT bit fields, because it can obtain this information internally (by computing the I2C_CNT[15-0] DATACOUNT bit field value modulo I2C_BUF[13-8] RXTRSH or I2C_BUF[5-0] TXTRSH).

By default, the draining feature is disabled; it can be enabled using the I2C_IRQENABLE_SET[14] XDR_IE or I2C_IRQENABLE_SET[13] RDR_IE bits (default disabled) only for transfers with lengths not equal to the threshold values (I2C_BUF[5-0] TXTRSH bit field value + 1 for the TX threshold or the I2C_BUF[13-8] RXTRSH bit field value + 1 for the RX threshold).

12.1.2.3.8 I2C Noise Filter

The noise filter is used to suppress any noise that is 50 ns or less in case of F/S operation modes, and any noise that is 10 ns or less in case of HS mode operation. The noise filter is always one period of the INTERNAL_CLK clock. This way, for HS mode operation (prescaler bypassed), the filter suppresses spikes of less than 10.4 ns.

For standard mode (for example, the I2C_PSC[7-0] PSC bit field = 4), the maximum width of suppressed spikes is 46.1 ns.

To ensure correct filtering, the prescaler must be programmed accordingly by the I2C*i*.I2C_PSC[7-0] PSC bit field.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.3.9 I2C System Test Mode

A system test mode is available for multicontroller I2C module testing. This mode is enabled by setting the I2C_SYSTEST[15] ST_EN bit to 1. When this bit is cleared to 0, the I2C controller is configured in normal operation mode.

In system test mode, the I2C_SYSTEST[13-12] TMODE bit field selects the type of test. Table 12-14 lists the tests available for the multicontroller HS I2C.

Table 12-14. I2C List of Tests

I2C_SYSTEST[13-12] TMODE	Test	Description
00	Functional mode	Normal operation mode
01	Reserved (not used)	
10	Test of SCL serial clock line	The SCL line is driven with a permanent clock as if controlled with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers.
11	Loop-back mode + SCL/SDA I/O	In controll transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register through an internal path through the FIFO buffers. The interrupt request is normally generated if it is enabled. Moreover, the SCL and SDA are controlled with the I2C_SYSTEST[3-0] bits.

Note

When the I2C_SYSTEST[13-12] TMODE bit field is set to 11, the I2C controller must be configured in I^2C F/S (I2C_CON[13-12] OPMODE set to 00) or I^2C HS mode (I2C_CON[13-12] OPMODE set to 01).

Note

In normal operation mode (the I2C_SYSTEST[15] ST_EN bit cleared to 0), the I2C_SYSTEST[3-0] bits that control the SCL, SDA lines in system test mode are read-only bits.

In system test mode (the I2C_SYSTEST[15] ST_EN bit set to 1), the I2C_IRQSTATUS_RAW[4] XRDY, I2C_IRQSTATUS_RAW[3] RRDY, I2C_IRQSTATUS_RAW[10] XUDF, I2C_IRQSTATUS_RAW[11] ROVR, I2C_IRQSTATUS_RAW[2] ARDY and I2C_IRQSTATUS_RAW[1] NACK status bits can be set to 1 when the I2C_SYSTEST[11] SSB bit is set to 1. Clearing the I2C_SYSTEST[11] SSB bit to 0 does not clear the I2C_IRQSTATUS_RAW bits to 0. The I2C_IRQSTATUS_RAW bit field can be cleared to 0 only by writing 1 in the corresponding bits.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.4 I2C Programming Guide

12.1.2.4.1 I2C Low-Level Programming Models

12.1.2.4.1.1 I2C Programming Model

This section describes the programming model of the multicontroller I2C configured in I^2C mode.

12.1.2.4.1.1.1 Main Program

12.1.2.4.1.1.1.1 Configure the Module Before Enabling the I2C Controller

Before enabling the I2C controller, perform the following steps:

1. Enable the functional and interface clocks (see *Module Integration* for details).
2. Program the prescaler to obtain an approximately 12-MHz internal sampling clock by programming the corresponding value in the I2C_PSC[7-0] PSC bit field. This value depends on the frequency of the functional clock (SYS_CLK).
3. Program the I2C_SCLL[7-0] SCLL and I2C_SCLH[7-0] SCLH bit fields to obtain a bit rate of 100 kbps or 400 kbps. These values depend on the internal sampling clock frequency (see Table 12-9).
4. (Optional) Program the I2C_SCLL[15-8] HSSCLL and I2C_SCLH[15-8] HSSCLH bit fields to obtain a bit rate of 400 kbps or 3.4 Mbps (for the second phase of HS mode). These values depend on the internal sampling clock frequency (see Table 12-9).
5. Configure the Own Address of the I2C controller by storing it in the I2C_OA register. Up to four Own Addresses can be programmed in the I2C_OA and I2C_OAx registers (where x = 1, 2, 3) for each I2C controller.

Note

For a 10-bit address, set the corresponding expand Own Address bit in the I2C_CON register.

6. Set the TX threshold (in transmitter mode) and the RX threshold (in receiver mode) by setting the I2C_BUF[5-0] TXTRSH bit field to (TX threshold – 1) and the I2C_BUF[13-8] RXTRSH bit field to (RX threshold – 1), where the TX and RX thresholds are greater than or equal to 1.
7. Take the I2C controller out of reset by setting the I2C_CON[15] I2C_EN bit to 1.

12.1.2.4.1.1.1.2 Initialize the I2C Controller

To initialize the I2C controller, perform the following steps:

1. Configure the I2C_CON register:
   - For controller or target mode, set the I2C_CON[10] MST bit (0: target; 1: controller).
   - For transmitter or receiver mode, set the I2C_CON[9] TRX bit (0: receiver; 1: transmitter).
2. If using an interrupt to transmit and receive data, set the corresponding bit in the I2C_IRQENABLE_SET register to 1 (the I2C_IRQENABLE_SET[4] XRDY_IE bit for the transmit interrupt, the I2C_IRQENABLE_SET[3] RRDY bit for the receive interrupt).

12.1.2.4.1.1.1.3 Configure Target Address and the Data Control Register

In controller mode, configure the target address register by programming the I2C_SA[9-0] SA bit field and the number of data bytes (I^2C data payload) associated with the transfer by programming the I2C_CNT[15-0] DCOUNT bit field.

Note

For a 10-bit address, set the I2C_CON[8] XSA bit to 1.

12.1.2.4.1.1.1.4 Initiate a Transfer

Poll the I2C_IRQSTATUS_RAW[12] BB bit. If it is cleared to 0 (bus not busy), configure the I2C_CON[0] STT and I2C_CON[1] STP bits. To initiate a transfer, the I2C_CON[0] STT bit must be set to 1, and it is not mandatory to set the I2C_CON[1] STP bit to 1.

Copyright © 2026 Texas Instruments Incorporated

12.1.2.4.1.1.5 Receive Data

Poll the I2C_IRQSTATUS_RAW[3] RRDY bit, or use the RRDY interrupt (the I2C_IRQENABLE_SET[3] RRDY_IE bit must be set to 1) to read the receive data in the I2C_DATA register.

If the transfer length does not equal the RX FIFO threshold (the I2C_BUF[13-8] RTRSH bit field + 1), use the draining feature (enable the RDR interrupt by setting the I2C_IRQENABLE_SET[13] RDR_IE bit to 1).

Note

In receive mode only, the I2C_IRQSTATUS_RAW[11] ROVR (receive overrun) bit indicates whether the receiver has experienced overrun. An overrun condition occurs when the shift register and the RX FIFO are full. An overrun condition does not result in data loss; the I2C controller simply holds SCL to low to prevent other bytes from being received.

The I2C_IRQSTATUS_RAW[7] AERR bit is set to 1 when a read access is performed in the I2C_DATA register while the RX FIFO is empty. The corresponding interrupt can be enabled by setting the I2C_IRQENABLE_SET[7] AERR_IE bit to 1.

12.1.2.4.1.1.6 Transmit Data

Poll the I2C_IRQSTATUS_RAW[4] XRDY bit, or use the XRDY interrupt (the I2C_IRQENABLE_SET[4] XRDY_IE bit must be set to 1) to write data to the I2C_DATA register.

If the transfer length does not equal the TX FIFO threshold (the I2C_BUF[5-0] TXTRSH bit field + 1), use the draining feature (enable the XDR interrupt by setting the I2C_IRQENABLE_SET[14] XDR_IE bit to 1).

Note

In transmit mode only, the I2C_IRQSTATUS_RAW[10] XUDF bit indicates whether the transmitter has experienced underflow.

In controller transmit mode, underflow occurs when the shift register and the TX FIFO are empty and there are still some bytes to transmit (the value of the I2C_CNT[15-0] DCOUNT bit field is not 0).

In target transmit mode, underflow occurs when the shift register and the TX FIFO are empty and the external I^2C controller device still requests data bytes to be read.

The I2C_IRQSTATUS_RAW[7] AERR bit is set to 1 when a write access is performed in the I2C_DATA register while the TX FIFO is full. The corresponding interrupt can be enabled by setting the I2C_IRQENABLE_SET[7] AERR_IE bit to 1.

12.1.2.4.1.1.2 Interrupt Subroutine Sequence

1. Test for arbitration lost (the I2C_IRQSTATUS_RAW[0] AL bit) and resolve accordingly.
2. Test for no acknowledgment (the I2C_IRQSTATUS_RAW[1] NACK bit) and resolve accordingly.
3. Test for register access ready (the I2C_IRQSTATUS_RAW[2] ARDY bit) and resolve accordingly.
4. Test for receive data ready (the I2C_IRQSTATUS_RAW[3] RRDY bit) and resolve accordingly.
5. Test for transmit data ready (the I2C_IRQSTATUS_RAW[4] XRDY bit) and resolve accordingly.
6. Test for general call (the I2C_IRQSTATUS_RAW[5] GC bit) and resolve accordingly.
7. Test for start (S) condition (the I2C_IRQSTATUS_RAW[6] STC bit) and resolve accordingly. For this test, the functional clock must be inactive.
8. Test for access error (the I2C_IRQSTATUS_RAW[7] AERR bit) and resolve accordingly.
9. Test for bus free (the I2C_IRQSTATUS_RAW[8] BF bit) and resolve accordingly.

12.1.2.4.1.1.3 Programming Flow-Diagrams

Figure 12-18 through Figure 12-24 are procedure flow charts for programming the F/S and HS I^2C modes.

Copyright © 2026 Texas Instruments Incorporated

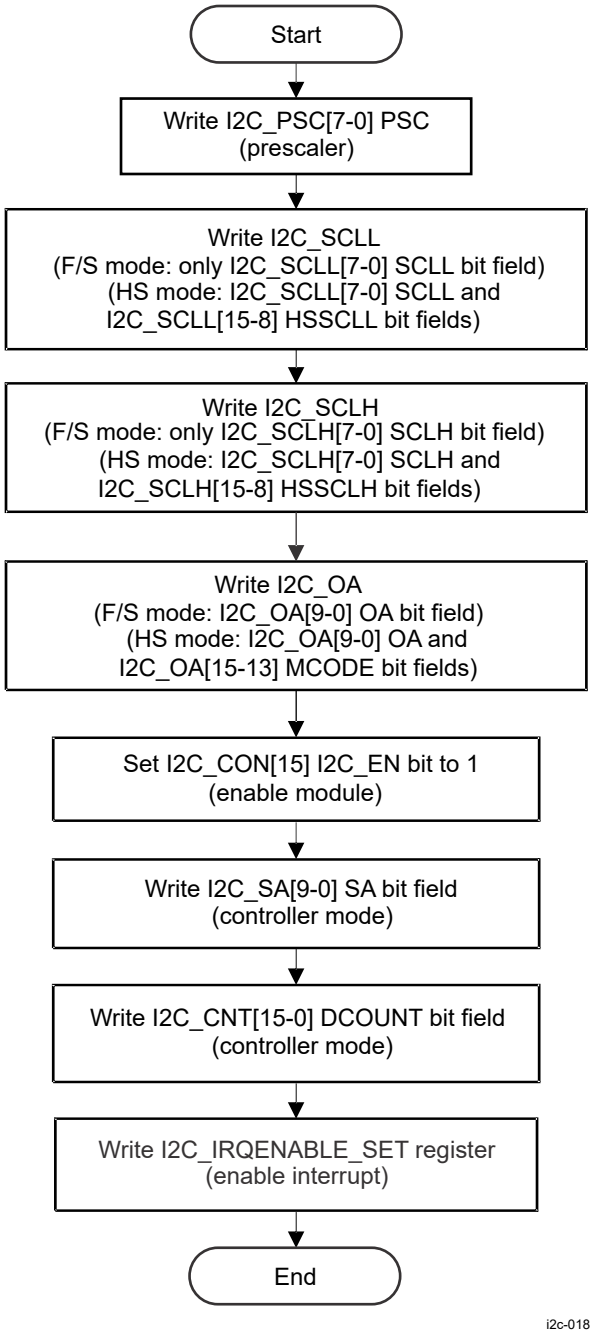
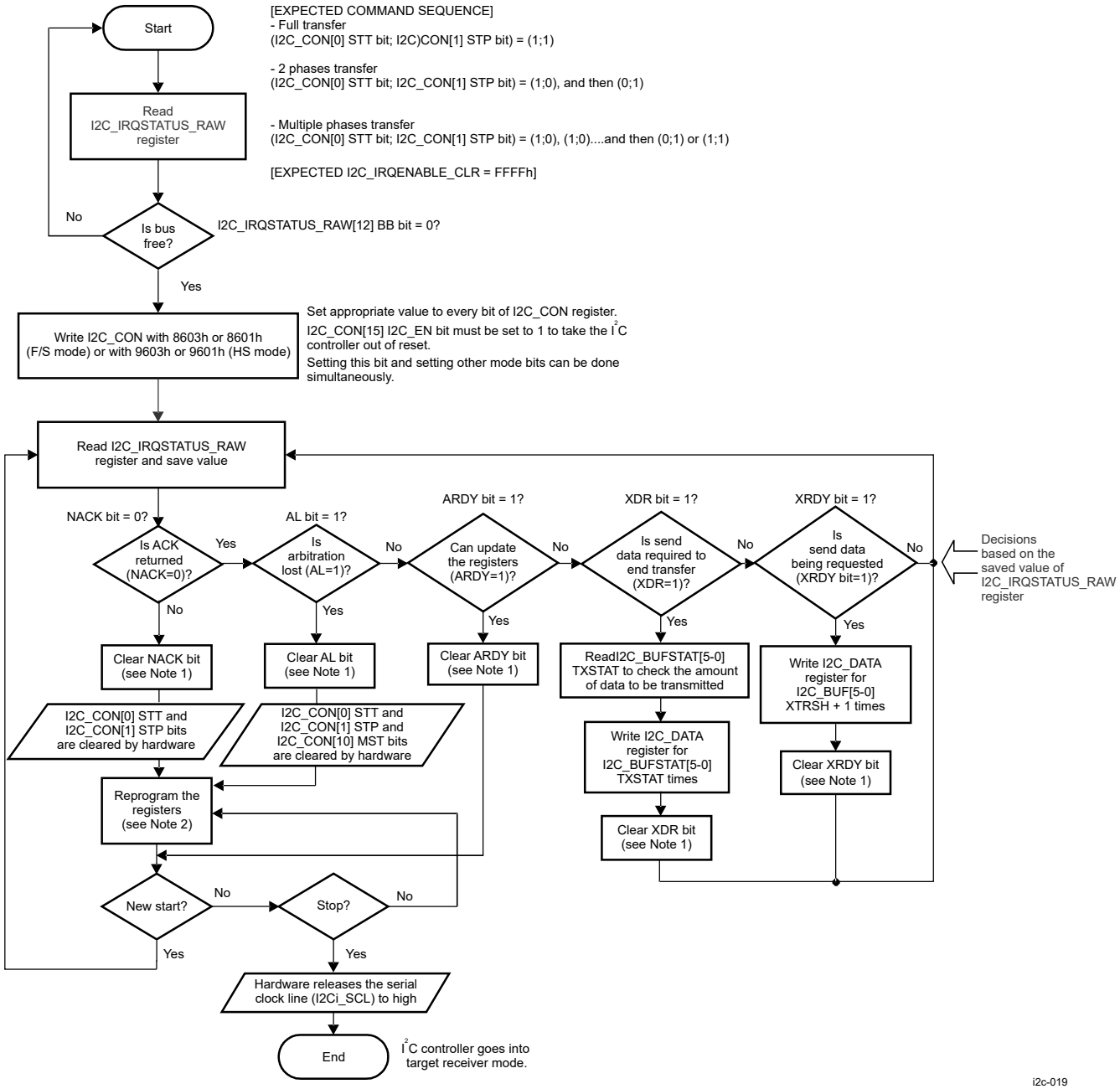


Figure 12-18. I2C Setup Procedure

Copyright © 2026 Texas Instruments Incorporated



A. The NACK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

B. Reprogram the registers means: I2C_CON[11] STB and/or I2C_CON[10] MST bit and/or I2C_SA[9-0] SA register and/or I2C_CNT[15-0] DCOUNT register and/or I2C_CON[0] STT bit and/or I2C_CON[1] STP bit.

Figure 12-19. I2C Controller Transmitter Mode, Polling Method, in F/S and HS Modes

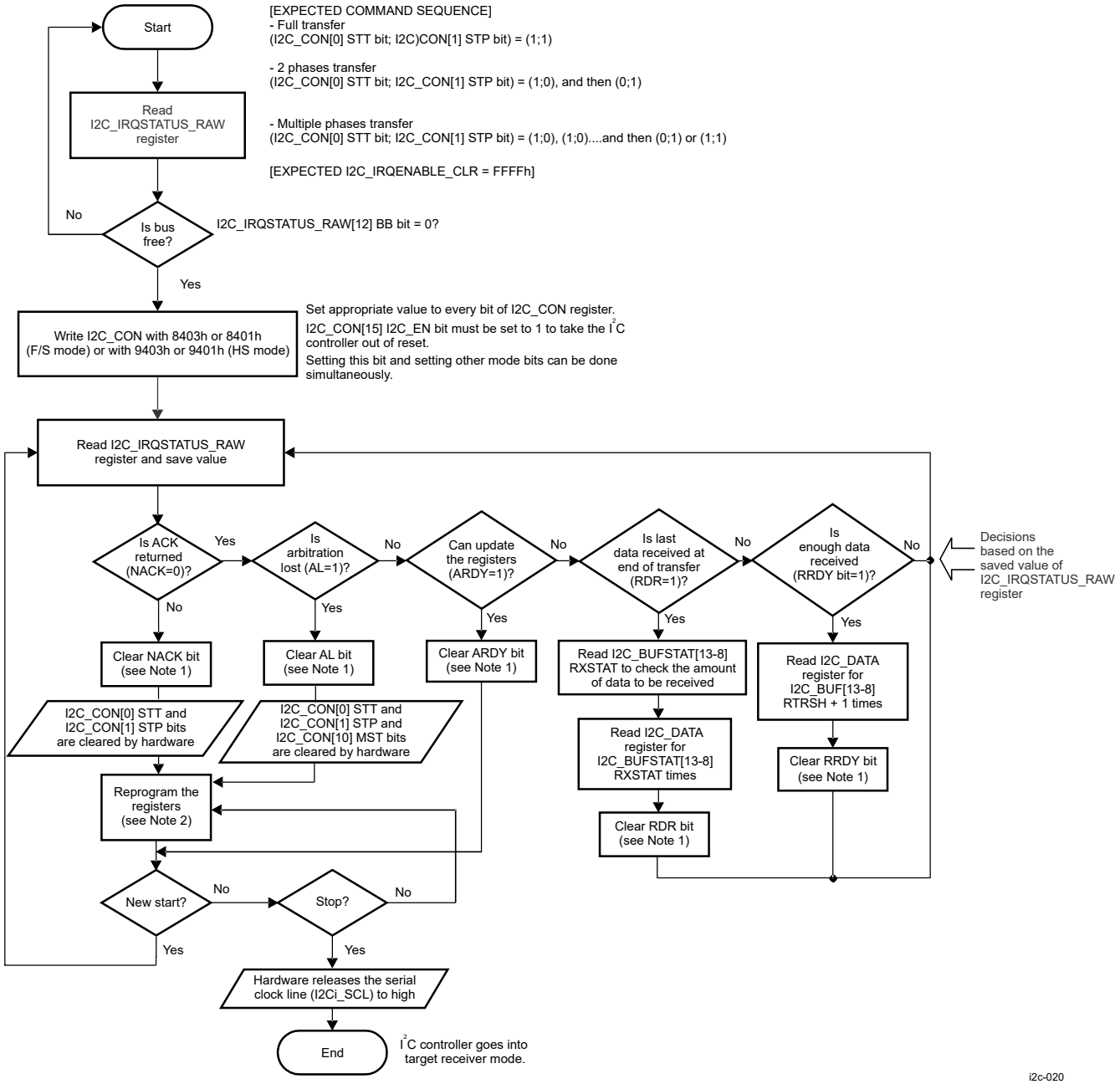
Note

The FIFO clearing can be made when the module is configured as transmitter, the receiver send a NACK in the middle of the transfer, and there is still data in the FIFO.

Copyright © 2026 Texas Instruments Incorporated

Note

In HS mode, the Sr condition and clock frequency switching are automatically generated by the multicontroller I2C.

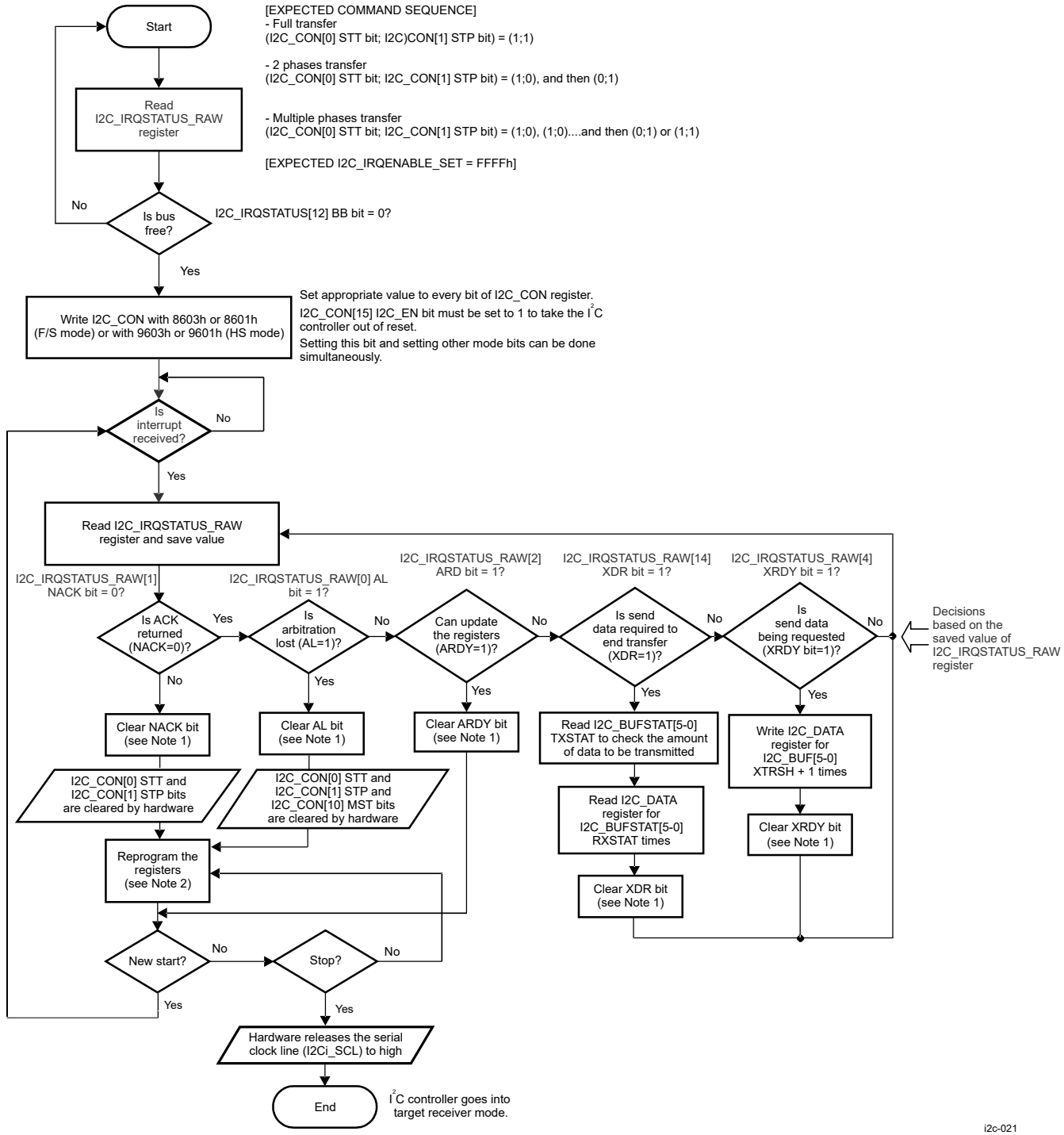


A. The NACK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

B. Reprogram registers means: I2C_CON[11] STB and/or I2C_CON[10] MST bit and/or I2C_SA[9-0] SA register and/or I2C_CNT[15-0] DCOUNT register and/or I2C_CON[0] STT bit and/or I2C_CON[1] STP bit.

Figure 12-20. I2C Controller Receiver Mode, Polling Method, in F/S and HS Modes

Copyright © 2026 Texas Instruments Incorporated



A. The NACK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

B. Reprogram registers means: I2C_CON[11] STB and/or I2C_CON[10] MST bit and/or I2C_SA[9-0] SA register and/or I2C_CNT[15-0] DCOUNT register and/or I2C_CON[0] STT bit and/or I2C_CON[1] STP bit.

Figure 12-21. I2C Controller Transmitter Mode, Interrupt Method, in F/S and HS Modes

Copyright © 2026 Texas Instruments Incorporated

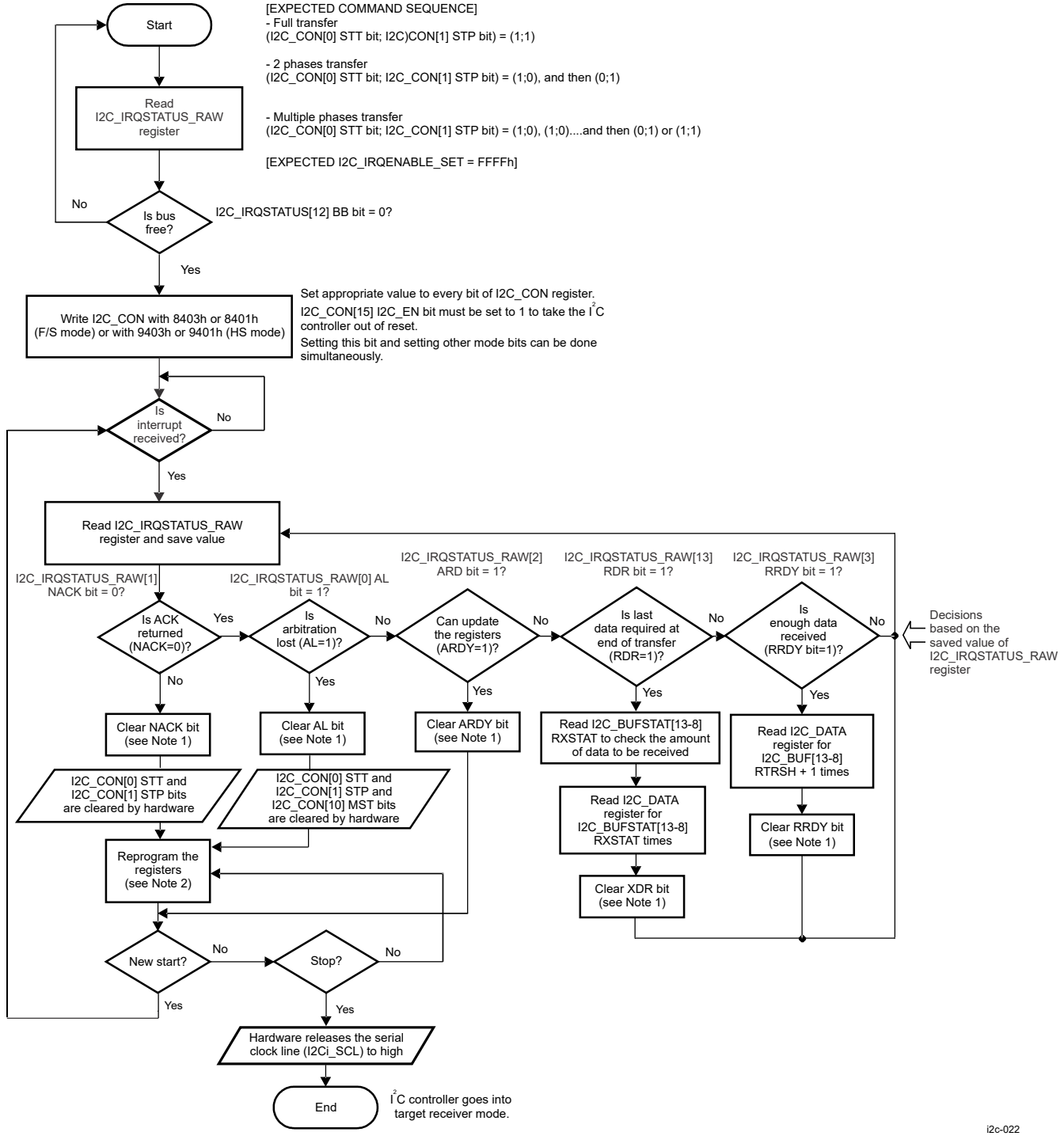
Note

The FIFO clearing can be made when the module is configured as transmitter, the receiver send a NACK in the middle of the transfer, and there is still data in the FIFO.

Note

In HS mode, the Sr condition and clock frequency switching are automatically generated by the multicontroller I2C.

Copyright © 2026 Texas Instruments Incorporated

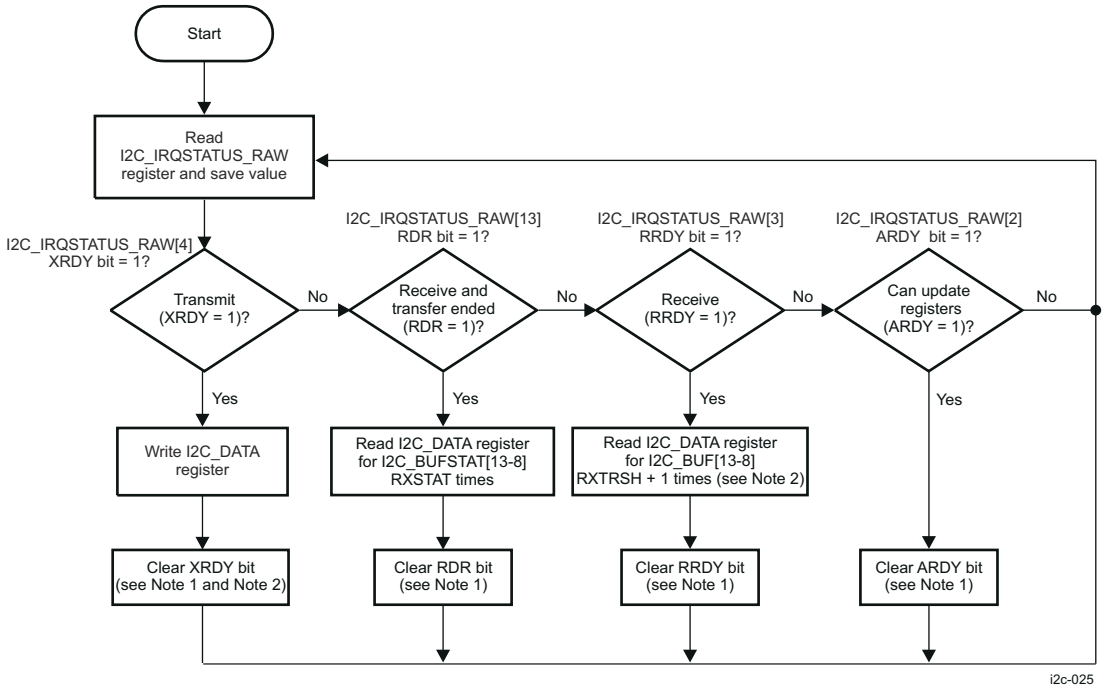


A. The NACK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

B. Reprogram registers means: I2C_CON[11] STB and/or I2C_CON[10] MST bit and/or I2C_SA[9-0] SA register and/or I2C_CNT[15-0] DCOUNT register and/or I2C_CON[0] STT bit and/or I2C_CON[1] STP bit.

Figure 12-22. I2C Controller Receiver Mode, Interrupt Method, in F/S and HS Modes

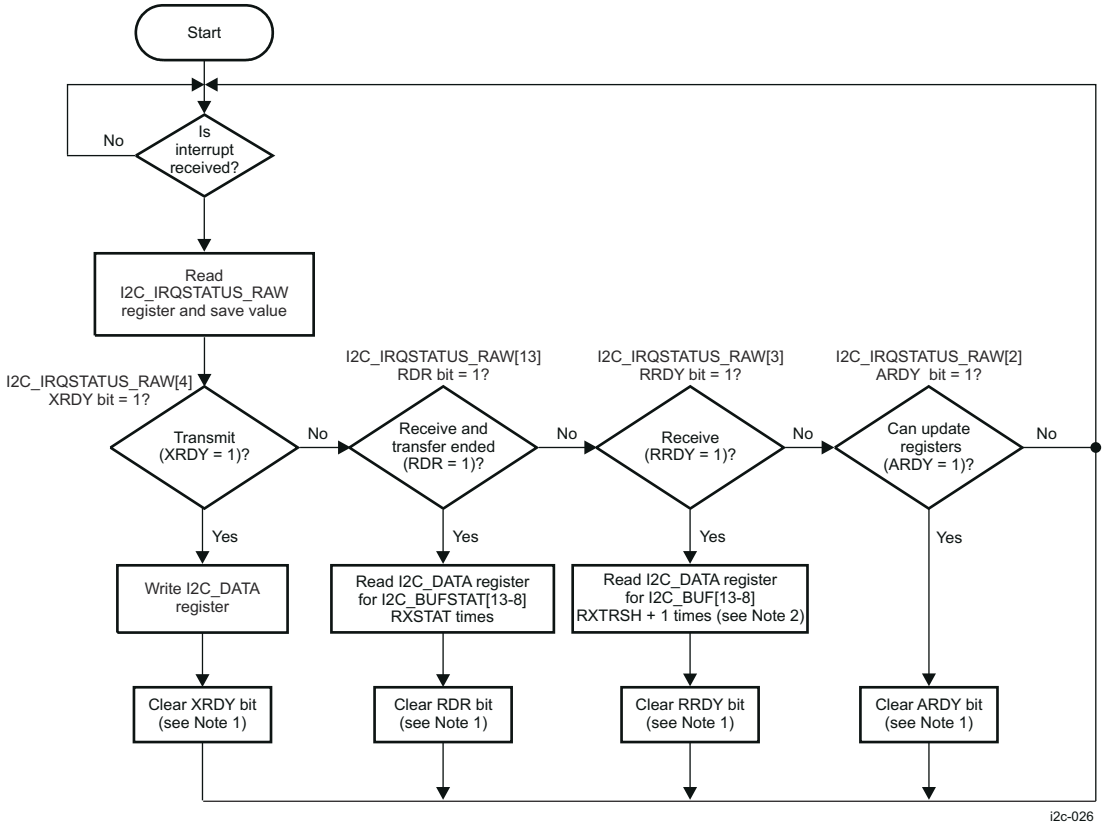
Copyright © 2026 Texas Instruments Incorporated



A. The XRDY, RDR, RRDY, and ARDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

B. In target transmitter mode, the amount of data requested by the external controller I^2C device is unknown; thus, the I2C_BUF[5-0] XTRSH bit field must be configured to 0x0 (TX threshold = 1).

Figure 12-23. I2C Target Transmitter/Receiver Mode, Polling



A. The XRDY, RDR, RRDY, and ARDY bits are cleared by writing 1 to each corresponding bit in the I2C_IRQSTATUS register.

Copyright © 2026 Texas Instruments Incorporated

B. In target transmitter mode, the amount of data requested by the external controller I^2C device is unknown; thus, the I2C_BUF[5-0] XTRSH bit field must be configured to 0x0 (TX threshold = 1).

Figure 12-24. I2C Target Transmitter/Receiver Mode, Interrupt

Copyright © 2026 Texas Instruments Incorporated

12.1.3 Multichannel Serial Peripheral Interface (MCSPI)

This section describes the Multichannel Serial Peripheral Interface (MCSPI) modules for the device.

12.1.3.1 MCSPI Overview

The MCSPI module is a multichannel transmit/receive, controller/peripheral synchronous serial bus.

Figure 12-25 shows the MCSPI overview.

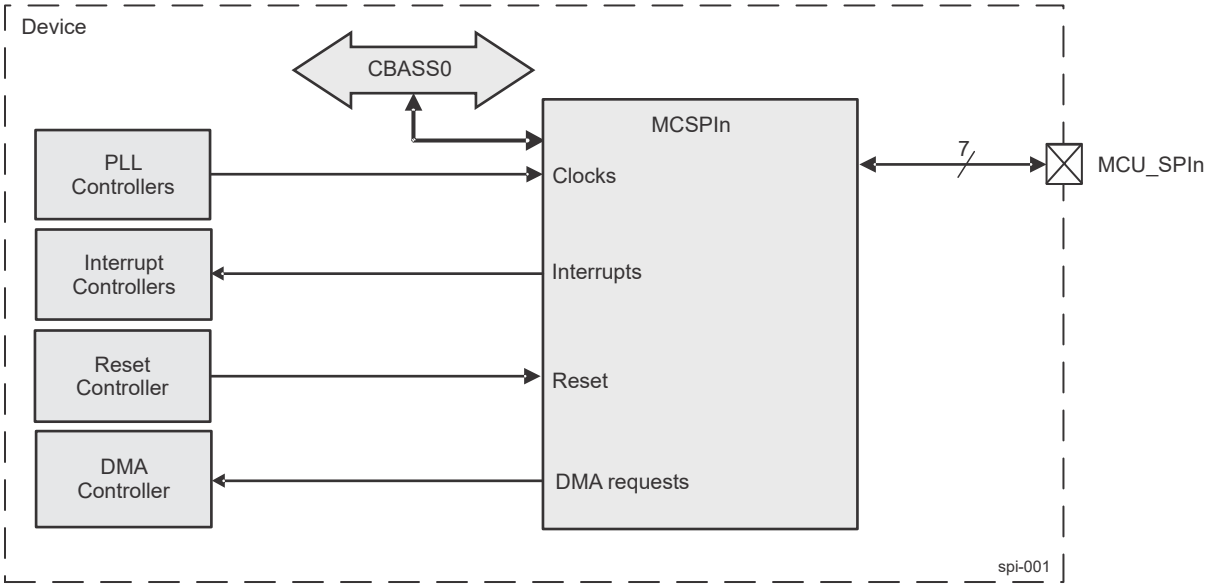


Figure 12-25. MCSPI Overview

n represents a valid instance of MCSPI in a domain.

12.1.3.1.1 MCSPI Features

The MCSPI modules include the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of MCSPI word lengths, ranging from 4 to 32 bits
- Up to four controller channels, or single channel in peripheral mode
- Controller multichannel mode:
  - Full duplex/half duplex
  - Transmit-only/receive-only/transmit-and-receive modes
  - Flexible input/output (I/O) port controls per channel
  - Programmable clock granularity
  - MCSPI configuration per channel. This means, clock definition, polarity enabling and word width
- Single interrupt line for multiple interrupt source events
- Enable the addition of a programmable start-bit for MCSPI transfer per channel (start-bit mode)
- Supports start-bit write command
- Supports start-bit pause and break sequence
- Programmable shift operations (1-32 bits)
- Programmable timing control between chip select and external clock generation
- Built-in FIFO available for a single channel.

12.1.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Copyright © 2026 Texas Instruments Incorporated

Note

Some features may not be available. See *Module Integration* for more information.

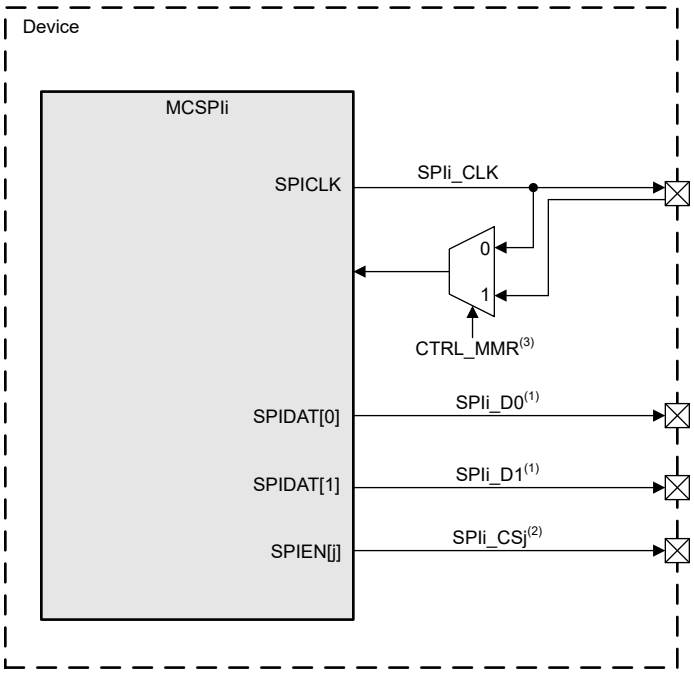
Copyright © 2026 Texas Instruments Incorporated

12.1.3.2 MCSPI Environment

This section describes the MCSPI external connections (environment).

12.1.3.2.1 Basic MCSPI Pins for Controller Mode

Figure 12-26 shows all of the MCSPI interface signals in controller mode.



A. Direction depends on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits
B. j = 0 to 3
C. The source of the loopback clock is defined by MCU_CTRL_MMR_CFG0_MCU_SPI0_CLKSEL[16] MSTR_LB_CLKSEL bit (and the respective register for MCU_MCSPI1) and MAIN_CTRL_MMR_CFG0_SPI0_CLKSEL[16] MSTR_LB_CLKSEL bit (and the respective for MCSPIi) in *Control Module (CTRL_MMR)*.

Note

i represents a MCSPI instance. See the device datasheet for available domains and MCSPI instances.

Figure 12-26. MCSPI Interface Signals in Controller Mode

Table 12-15 describes the MCSPI I/O signals in controller mode.

Table 12-15. MCSPI I/O Signals (Controller Mode)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
		MCU_MCSPIi(5)		
SPICLK	MCU_SPIi(5)_CLK	O	MCSPI Serial clock output for controller mode.	HiZ
SPIDAT[0]	MCU_SPIi(5)_D0	O(3)	MCSPI Data I/O for controller mode.	HiZ
SPIDAT[1]	MCU_SPIi(5)_D1	I(4)	MCSPI Data I/O for controller mode.	HiZ
SPIEN_[n]	MCU_SPIi(5)_CSi	O	MCSPI Chip-select i output for controller mode	HiZ
		MCSPIi(5)		
SPICLK	SPIi(5)_CLK	O	MCSPI Serial clock output for controller mode.	HiZ
SPIDAT[0]	SPIi(5)_D0	O(3)	MCSPI Data I/O for controller mode.	HiZ
SPIDAT[1]	SPIi(5)_D1	I(4)	MCSPI Data I/O for controller mode.	HiZ

Copyright © 2026 Texas Instruments Incorporated

Table 12-15. MCSPI I/O Signals (Controller Mode) (continued)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
SPIEN_[n]	SPIi(5)_CSi	O	MCSPI Chip-select i output for controller mode	HiZ

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) Example configuration only. Can be configured either as input or as output depending on MCSPI_CH(i)CONF[18] IS and MCSPI_CH(i)CONF[16] DPE0.
(4) Example configuration only. Can be configured either as input or as output depending on MCSPI_CH(i)CONF[18] IS and MCSPI_CH(i)CONF[17] DPE1.
(5) i represents a MCSPI instance. See the device datasheet for available domains and MCSPI instances.

Note

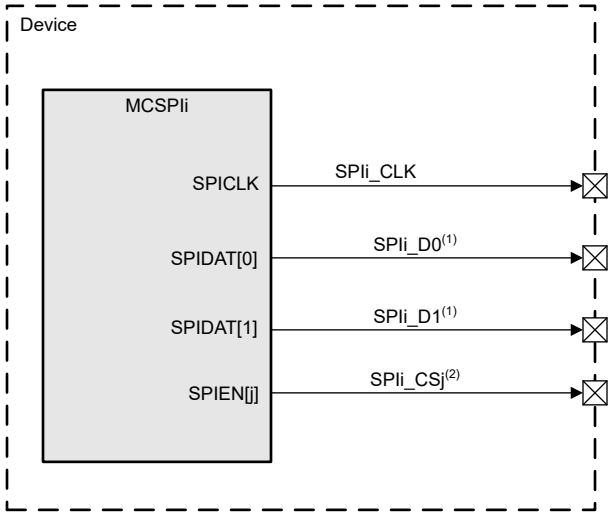
For SPIi(5)_CLK and MCU_SPIi(5)_CLK signals to work properly, the RXACTIVE bit of the appropriate MCU_PADCFG_CTRL_MMR_CFG0_PADCONFIGx/ MAIN_CTRL_MMR_CFG0_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.3.2.2 Basic MCSPI Pins for Peripheral Mode

Figure 12-27 shows all of the MCSPI interface signals in peripheral mode.



A. Direction depends on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits
B. j = 0 to 3

Note

i represents a MCSPI instance. See the device datasheet for available domains and MCSPI instances.

Figure 12-27. MCSPI Interface Signals in Peripheral Mode

Table 12-16 describes the MCSPI I/O signals in peripheral mode.

Copyright © 2026 Texas Instruments Incorporated

Table 12-16. MCSPI I/O Signals (Peripheral Mode)

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[1]
MCU_MCSPI[1-0]				
SPICLK	MCU_SPIi[5]_CLK	I	MCSPI serial clock input for peripheral mode.	HiZ
SPIDAT[0]	MCU_SPIi[5]_D0	I[2]	MCSPI Data I/O for peripheral mode.	HiZ
SPIDAT[1]	MCU_SPIi[5]_D1	O[3]	MCSPI Data I/O for peripheral mode.	HiZ
SPIEN_[n]	MCU_SPIi[5]_CSi	I[4]	MCSPI chip-select i input for peripheral mode.	HiZ
MCSPI[4-0]				
SPICLK	SPIi[5]_CLK	I	MCSPI serial clock input for peripheral mode.	HiZ
SPIDAT[0]	SPIi[5]_D0	I[2]	MCSPI Data I/O for peripheral mode.	HiZ
SPIDAT[1]	SPIi[5]_D1	O[3]	MCSPI Data I/O for peripheral mode.	HiZ
SPIEN_[n]	SPIi[5]_CSi	I[4]	MCSPI chip-select i input for peripheral mode.	HiZ

(1) HiZ = High Impedance
(2) Example configuration only. Can be configured either as input or as output depending on MCSPI_CH(i)CONF[18] IS and MCSPI_CH(i)CONF[16] DPE0.
(3) Example configuration only. Can be configured either as input or as output depending on MCSPI_CH(i)CONF[18] IS and MCSPI_CH(i)CONF[17] DPE1.
(4) The chip-select input in peripheral mode can be selected through the MCSPI_CH(i)CONF[22-21] SPIENSLV bit field.
(5) i represents a MCSPI instance. See the device datasheet for available domains and MCSPI instances.

Note

For SPI[4-0]_CLK and MCU_SPI[1-0]_CLK signals to work properly, the RXACTIVE bit of the appropriate CTRLMMR_MCU_PADCONFIGx/ CTRLMMR_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.2.3 MCSPI Protocol and Data Format

The synchronous MCSPI protocol allows a controller device to initiate serial data transfers to a peripheral device. A peripheral select line (SPIEN_[n]) allows selection of an individual peripheral MCSPI device. Peripheral devices that are not selected do not interfere with MCSPI bus activities.

MCSPI offers the flexibility to modify the following parameters to adapt to the device features:

- Word length

  MCSPI supports any MCSPI word ranging from 4 bits to 32 bits long (the MCSPI_CH(i)CONF[11-7] WL bit field).

  MCSPI word length can be changed between transmissions to allow the controller device to communicate with peripheral devices that have different requirements.
- MCSPI enable (SPIEN_[n], for channel i)

  The polarity of the MCSPI enable signals is programmable (the MCSPI_CH(i)CONF[6] EPOL bit). SPIEN_[n] signals can be active high or low.

  Assertion of the SPIEN_[n] signals is programmable and can be done manually or automatically. The manual assertion mode is available in single controller mode only. SPIEN_[n] can be kept active between words with the MCSPI_CH(i)CONF[20] FORCE bit.

  Two consecutive words for two different peripheral devices can go along with active SPIEN_[n] signals with different polarity.
- Programmable start-bit

  In start-bit mode a start-bit is added before the MCSPI word length to indicate how the next MCSPI word must be handled. The start-bit is enabled by setting the MCSPI_CH(i)CONF[23] SBE bit to 1. The MCSPI_CH(i)CONF[24] SBPOL bit defines the polarity of the start-bit.
- Programmable MCSPI clock
  - Bit rate

    In controller mode, the baud rate of the MCSPI serial clock is programmable using the 50-MHz reference clock (from the device clock management module). Table 12-17 lists the SPICLK bit rates obtained for data transfer when programming the clock divider (the MCSPI_CH(i)CONF[5-2] CLKD bit field).

Table 12-17. MCSPI Controller Clock Rates

Divider	Clock Rate
1	50 MHz(1)
2	25 MHz(1)
4	12.5 MHz
8	6.25 MHz
16	3.125 MHz
32	1.5625 MHz
64	781.25 kHz
128	390.625 kHz
256	~195 kHz
512	~97.7 kHz
1024	~48.8 kHz
2048	~24.4 kHz

Copyright © 2026 Texas Instruments Incorporated

Table 12-17. MCSPI Controller Clock Rates (continued)

Divider	Clock Rate
4096	~12.2 kHz

(1) These frequencies are not necessarily supported by all MCSPI modules. For more information, see the *Timing Requirements and Switching Characteristics* chapter in the device-specific Datasheet.

– Polarity and phase

The polarity (the MCSPI_CH(i)CONF[1] POL bit) and the phase (the MCSPI_CH(i)CONF[0] PHA bit) of the MCSPI serial clock (SPICLK) are configurable to offer four combinations. Software selects the right combination, depending on the device. See Table 12-18 and Figure 12-28.

Data transfer is not required to update polarity, but it is configuration specific.

- In multi-controller mode POL/PHA is only updated to a channel's settings when a transaction is necessary on that channel.
- In single-controller mode POL/PHA is only updated on CS assertion (assuming channel already enabled), which is now SW-controlled using the FORCE bitfield.

Table 12-18. Phase and Polarity Combinations

Polarity (POL)	Phase (PHA)	MCSPI Mode	Description
0	0	Mode 0	SPICLK is inactive low and sampling occurs at the rising edge.
0	1	Mode 1	SPICLK is inactive low and sampling occurs at the falling edge.
1	0	Mode 2	SPICLK is inactive high and sampling occurs at the falling edge.
1	1	Mode 3	SPICLK is inactive high and sampling occurs at the rising edge.

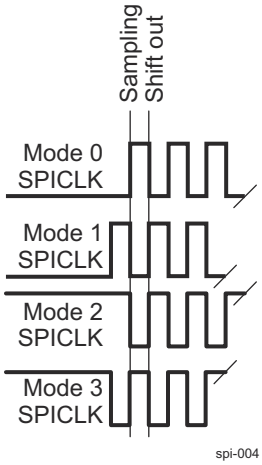


Figure 12-28. Phase and Polarity Combinations

12.1.3.2.3.1 Transfer Format

In controller and peripheral modes, the MCSPI drives the data lines when SPIEN_[n] is asserted.

Each word is transmitted starting with the most-significant bit (MSB).

This section explains the two cases of data transmission determined by the clock phase (PHA) and the type of data transmission using a start-bit (SBE) called the start-bit mode:

- Transmission in mode 0 and mode 2 (PHA = 0)

Copyright © 2026 Texas Instruments Incorporated

When PHA = 0, the first bit of the MCSPI word to transmit (on the controller or the peripheral data output pin) is valid one-half cycle of SPICLK after the assertion of SPIEN_[n].

Therefore, the first edge of the SPICLK line is used by the controller to sample the first data bit sent by the peripheral. On the same edge, the first data bit sent by the controller is sampled by the peripheral.

On the next SPICLK edge, the received data bit is shifted into the receive shift register and a new data bit is transmitted on the serial data line.

This process continues for a number of pulses on the SPICLK line defined by the MCSPI word length programmed in the controller device, with data being latched on odd-numbered edges and shifted on even-numbered edges, see Figure 12-29.

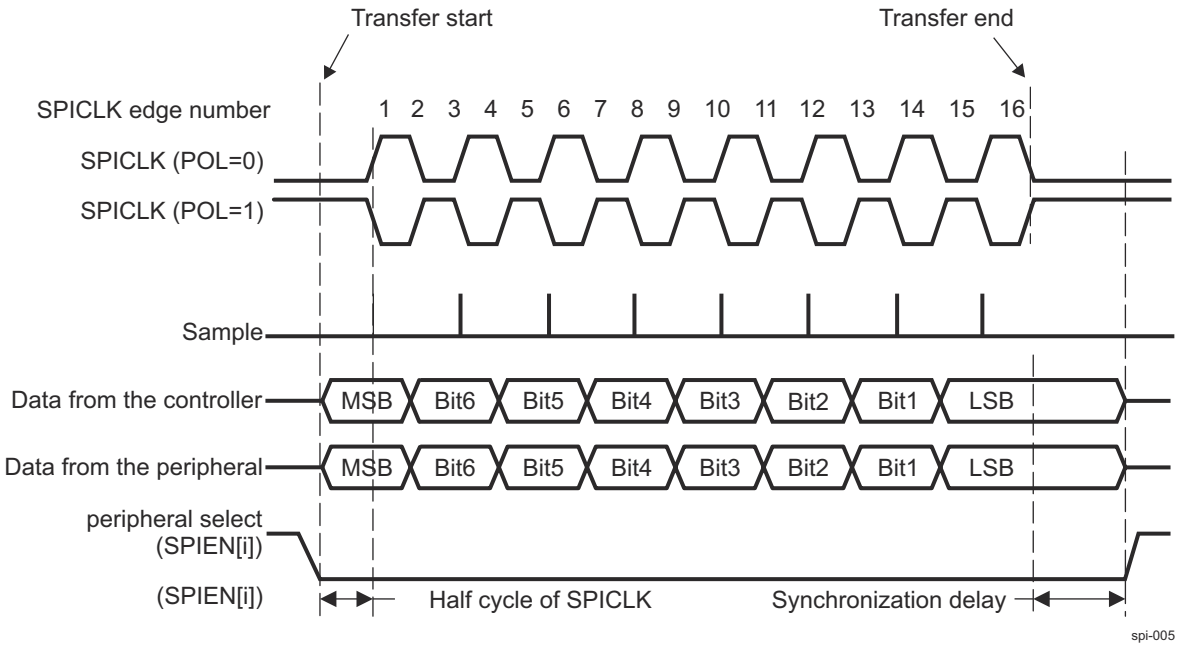


Figure 12-29. Full-Duplex Transfer Format With PHA = 0

- Transmission in mode 1 and mode 3 (PHA = 1)

  When PHA = 1, the first bit of the MCSPI word to transmit (on the controller or the peripheral data output pin) is valid on the following SPICLK edge (one-half cycle later). This is the sampling edge for the controller and peripheral. A synchronization delay is added between the activation of SPIEN_[n] and the first SPICLK edge.

  The received data bit is shifted into the shift register on the third SPICLK edge.

  This process continues for a number of pulses on the SPICLK line defined by the MCSPI word length programmed in the controller device, with data being latched on even-numbered edges and shifted on odd-numbered edges.

  Note

  The minimum synchronization delay is one cycle of SPICLK, if the frequency of SPICLK equals the frequency of MCSPI_FCLK (MCSPI functional clock) in controller mode. The minimum synchronization delay is one-half cycle of SPICLK, if the frequency of SPICLK is lower than the frequency of MCSPI_FCLK in the controller and peripheral modes.

- Transmission with a start-bit (SBE = 1)

  When the MCSPI_CH(i)CONF[23] SBE bit is set to 1, a start-bit is added before the MSB to indicate whether the next MCSPI word must be handled as a command or as data.

  Figure 12-30 shows an example of a data transfer with an extra start-bit.

Copyright © 2026 Texas Instruments Incorporated

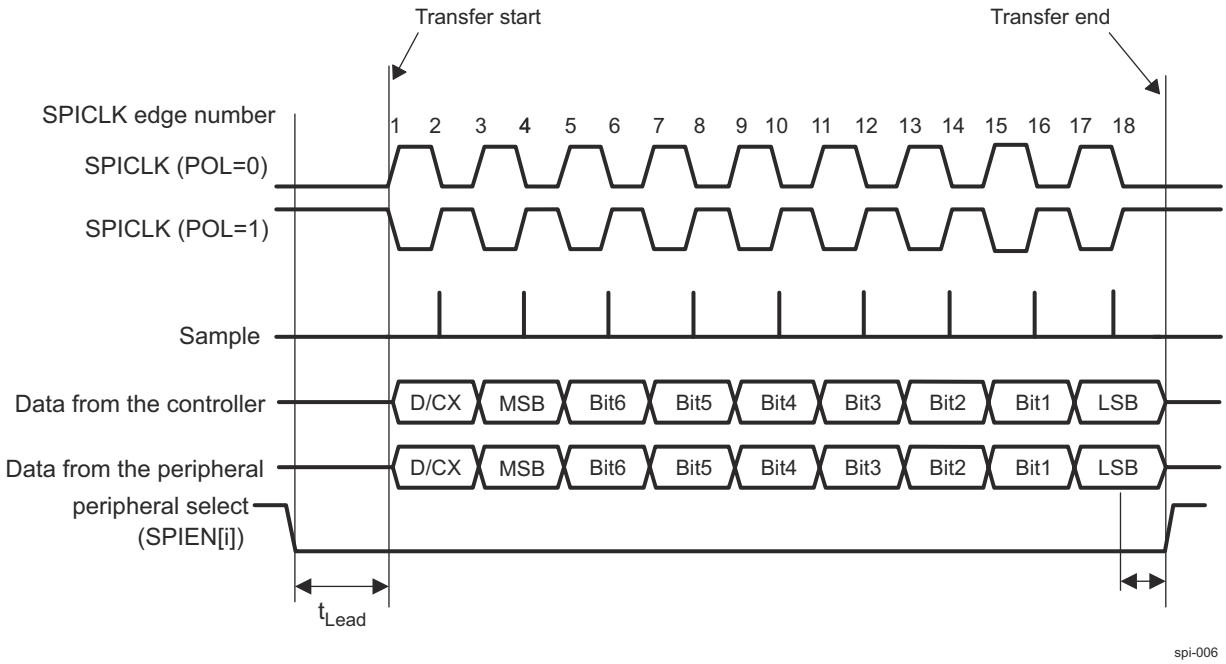
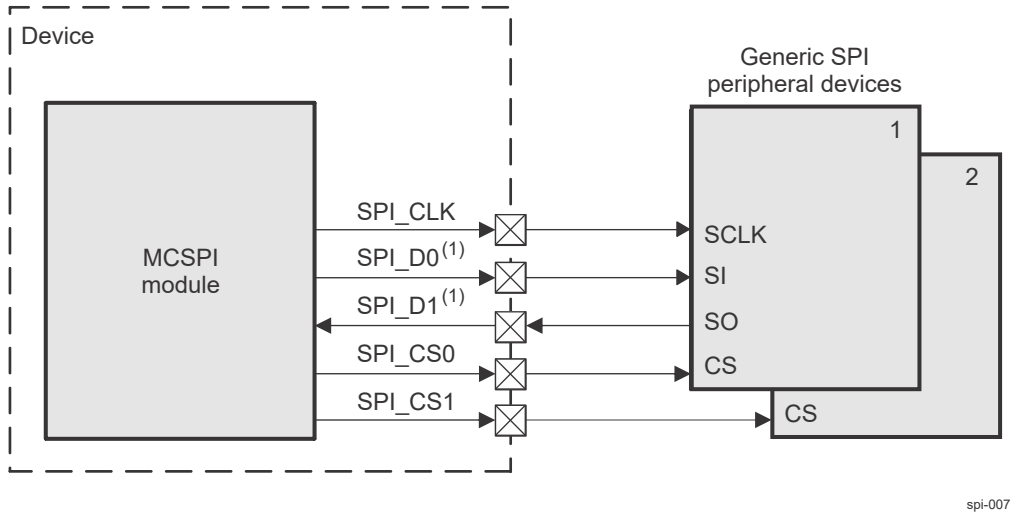


Figure 12-30. Extended MCSPI Transfer With a Start-Bit (SBE = 1)

12.1.3.2.4 MCSPI in Controller Mode

Figure 12-31 shows a case in controller mode (full-duplex) where the MCSPI module is connected with two peripheral devices.

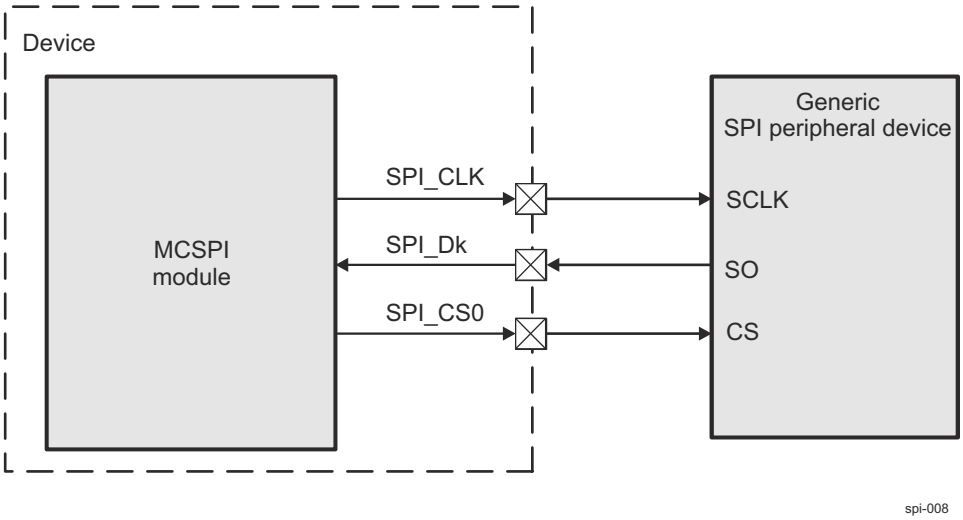


A. Direction depends on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-31. MCSPI Controller Mode (Full Duplex)

Figure 12-32 shows the controller single mode, which can also be configured in receive-only mode.

Copyright © 2026 Texas Instruments Incorporated



k = 0 or 1 depending on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

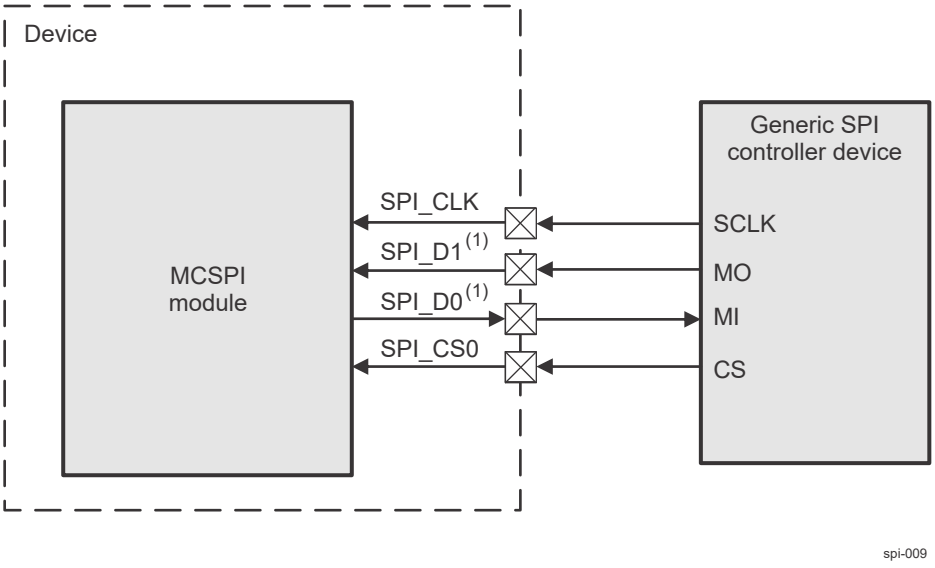
Figure 12-32. MCSPI Controller Single Mode (Receive Only)

12.1.3.2.5 MCSPI in Peripheral Mode

Figure 12-33 shows a case in peripheral mode (full-duplex).

Note

Only channel 0 can be configured as peripheral, but the chip-enable signal can be connected to any SPIEN_[n] pin and then rerouted internally to channel 0 (the MCSPI_CH0CONF[22-21] SPIENSLV bit field). For more information, see Section 12.1.3.4.4, *MCSPI Peripheral Mode*.

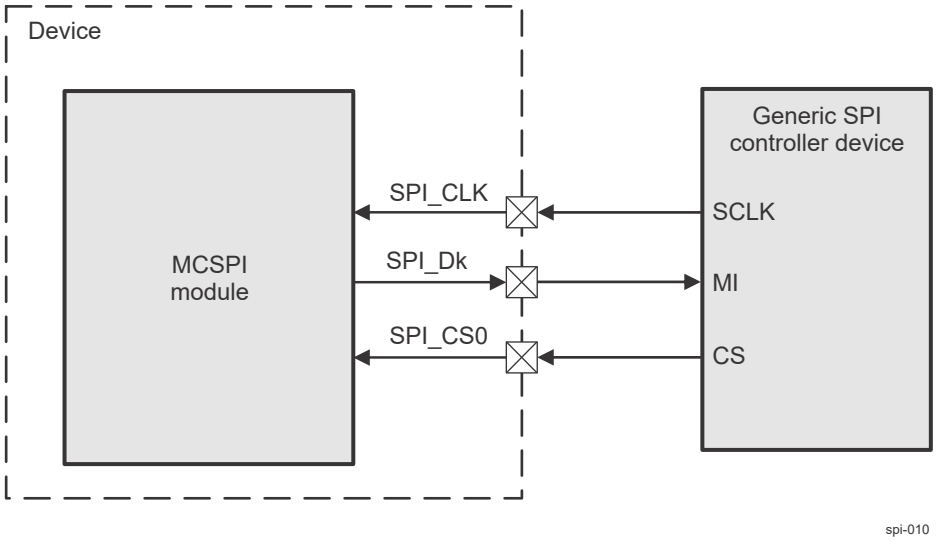


A. Direction depends on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-33. MCSPI Peripheral Mode (Full Duplex)

Figure 12-34 shows the peripheral single mode, which can also be configured in transmit-only mode.

Copyright © 2026 Texas Instruments Incorporated



k = 0 or 1 depending on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-34. MCSPI Peripheral Single Mode (Transmit Only)

12.1.3.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.4 MCSPI Functional Description

12.1.3.4.1 MCSPI Block Diagram

Figure 12-35 shows the MCSPI module.

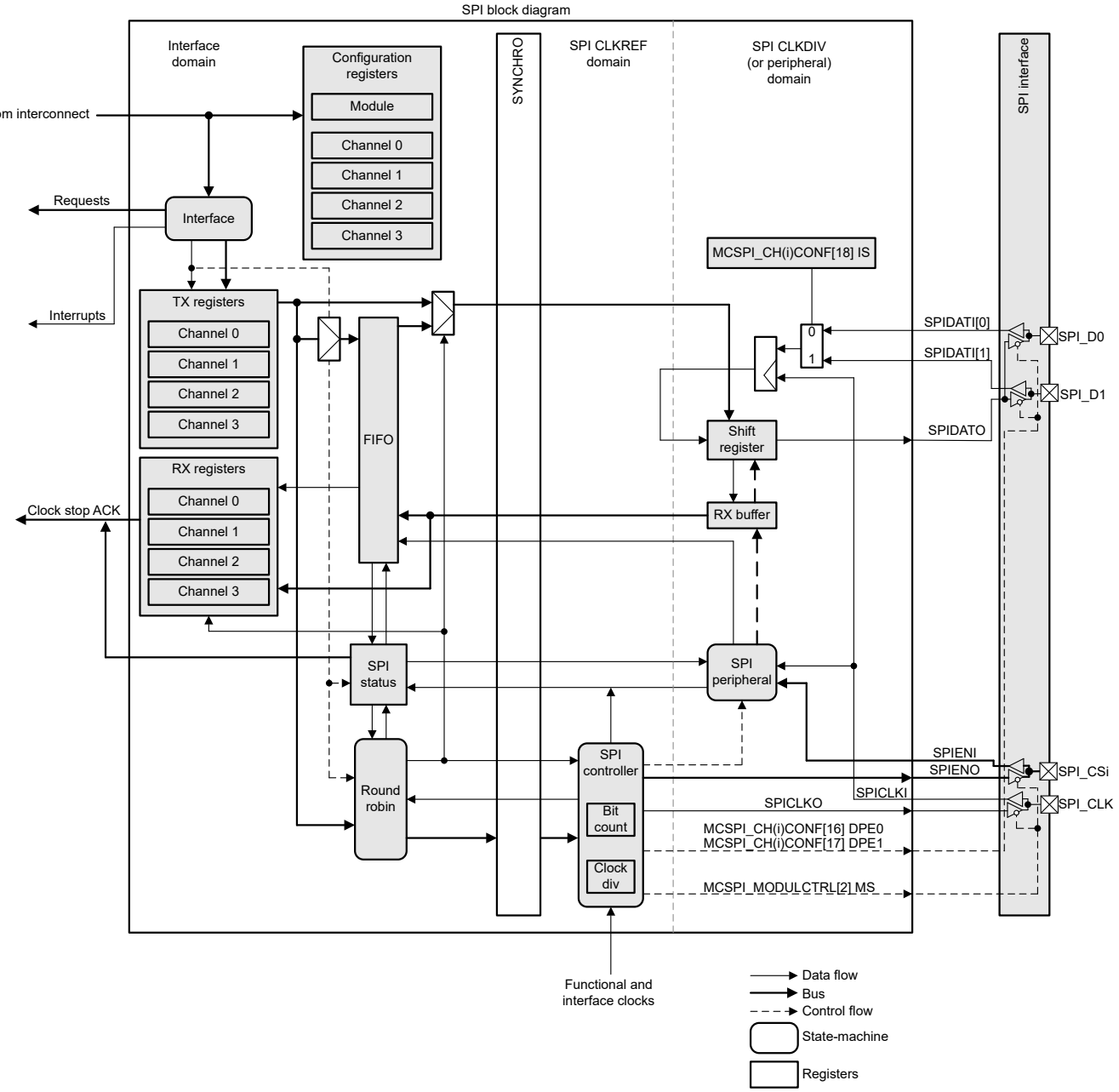


Figure 12-35. MCSPI Block Diagram

12.1.3.4.2 MCSPI Reset

The MCSPI module can be reset either by hardware or by software reset. All configuration registers and all state machines are reset by the hardware reset signal (MCSPI_RST). MCSPI can be reset by software through the MCSPI_SYSCONFIG[1] SOFTRESET bit. This bit has the same impact on the module as the hardware reset signal. The only exception is that the MCSPI_SYSCONFIG register is not affected by that software reset.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.4.3 MCSPI Controller Mode

12.1.3.4.3.1 Controller Mode Features

The MCSPI controller mode supports multichannel communication with up to four independent MCSPI communication channel contexts. The MCSPI initiates a data transfer on the data lines (SPIDAT[0] and SPIDAT[1]) and generates clock (SPICLK) and control (SPIEN_[n]) signals.

Connected to multiple external devices, the MCSPI exchanges data with one MCSPI device at a time through two main modes (available in peripheral mode):

- Two-data-pins interface mode (transmit-and-receive mode for full-duplex transmission)
- Single-data-pin interface mode (recommended for half-duplex transmission)

Note

There is a fixed chip select line allocation in multichannel controller mode. Channel i is mapped to SPIEN_[n].

Two DMA request events (read and write) allow synchronized accesses of the DMA controller with the activity of MCSPI.

Three interrupt events can be used for data transmission and reception in controller mode (for more information about interrupts, see Section 12.1.3.4.7.1, *Interrupt Events in Controller Mode*).

12.1.3.4.3.2 Controller Transmit-and-Receive Mode (Full Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on SPIDAT[0]) and received (shifted in serially on SPIDAT[1]) simultaneously on separate data lines.

The controller transmit-and-receive mode is programmable per channel (the MCSPI_CH(i)CONF[13-12] TRM bit field).

Channel access to the shift registers for transmission/reception is based on the MCSPI_TXi transmitter register state, the MCSPI_RXi receiver register state, and round-robin arbitration.

Channels that meet the following rules are included in the round-robin list of active channels scheduled for transmission and/or reception. The arbiter skips channels that do not meet the rules and searches in the rotation for the next enabled channel.

- Rule 1: Only enabled channels (the MCSPI_CH(i)CTRL[0] EN bit) can be scheduled for transmission and/or reception.
- Rule 2: If its MCSPI_TXi transmitter register is not empty (the MCSPI_CH(i)STAT[1] TXS bit), an enabled channel can be scheduled when the shift register is assigned. If the MCSPI_TXi register is empty when the shift register is assigned, the TXx_UNDERFLOW event is activated, and the next enabled channel with new data to transmit is scheduled (see also transmit-only mode).
- Rule 3: An enabled channel can be scheduled if its receive register is not full (the MCSPI_CH(i)STAT[0] RXS bit) when the shift register is assigned (see also receive-only mode). Therefore, the MCSPI_RXi register cannot be overwritten. The MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set to this mode.

When MCSPI word transfer completes (the MCSPI_CH(i)STAT[2] EOT bit is set), the updated MCSPI_TXi register of the next scheduled channel is loaded into the shift register. The serialization (transmit-and-receive) starts depending on the channel communication configuration. When serialization completes, the received data transfers to the channel receive register.

The serial clock (SPICLK) synchronizes shifting and sampling of the information on the two serial data lines (SPIDAT[0] and SPIDAT[1]). Each time a bit transfers out from the controller, 1 bit transfers in from the peripheral.

Figure 12-36 shows an example of a full-duplex system with a controller device on the left and a peripheral device on the right. After eight cycles of the serial clock SPICLK, WordA transfers from the controller to the peripheral. At the same time, WordB transfers from the peripheral to the controller.

Copyright © 2026 Texas Instruments Incorporated

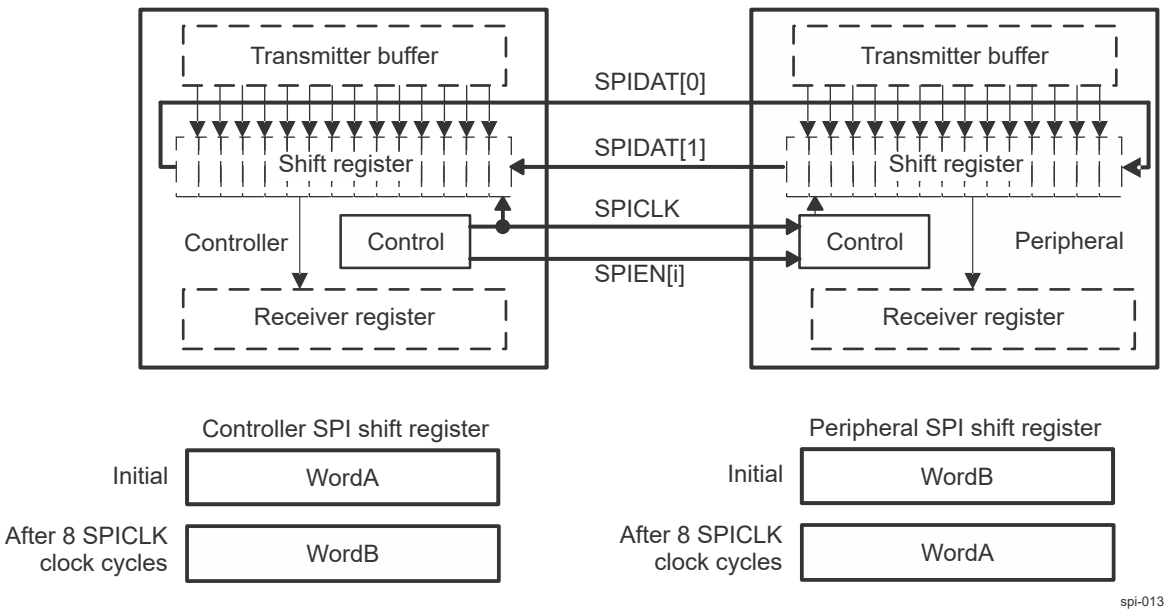


Figure 12-36. MCSPI Full-Duplex Transmission (Example)

12.1.3.4.3.3 Controller Transmit-Only Mode (Half Duplex)

The controller transmit-only mode prevents the processor from reading the MCSPI_RXi register (minimizing data movement) when only transmission is meaningful.

The controller transmit-only mode is programmable per channel (the MCSPI_CH(i)CONF[13-12] TRM bit field). Transmission starts only after data is loaded into the MCSPI_TXi register.

Rule 1 and Rule 2, defined in Section 12.1.3.4.3.2, apply in this mode.

Rule 3, defined in Section 12.1.3.4.3.2, does not apply.

In controller transmit-only mode, the MCSPI_RXi register state FULL does not prevent transmission and the MCSPI_RXi register is always overwritten with the new MCSPI word. This event is not significant when only transmission is meaningful. Thus, the RX0_OVERFLOW bit in the MCSPI_IRQSTATUS register is never set in this mode.

The hardware automatically disables the RX_FULL interrupt and the DMA read requests.

The transfer status is given by the MCSPI_CH(i)STAT[2] EOT bit.

12.1.3.4.3.4 Controller Receive-Only Mode (Half Duplex)

The controller receive mode prevents the processor from refilling the MCSPI_TXi register (minimizing data movement) when only reception is meaningful.

The controller receive mode is programmable per channel (the MCSPI_CH(i)CONF[13-12] TRM bit field).

The controller receive-only mode enables channel scheduling only on the empty state of the MCSPI_RXi register.

Rule 1 and Rule 3, defined in Section 12.1.3.4.3.2, apply in this mode.

Rule 2, defined in Section 12.1.3.4.3.2, does not apply.

In the controller receive-only mode, software must write dummy data to the MCSPI_TXi register. Only one dummy write is enough to receive any number of words from the peripheral. Software must ensure that the MCSPI_TXi register is always full (the TXi_EMPTY bits of MCSPI_IRQSTATUS) when receiving. The content of the MCSPI_TXi register is always loaded into the shift register when the shift register is assigned.

Copyright © 2026 Texas Instruments Incorporated

After writing the dummy data to the MCSPI_TXi register, the TXi_EMPTY and TXi_UNDERFLOW bits in the MCSPI_IRQSTATUS register are never set in receive-only mode.

The MCSPI_CH(i)STAT[2] EOT bit gives the status of serialization. The RXi_FULL bits of the MCSPI_IRQSTATUS register are set when received data is loaded from the shift register to the corresponding MCSPI_RXi register. The MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set in this mode.

12.1.3.4.3.5 Single-Channel Controller Mode

When the MCSPI is configured as a controller device with a single enabled channel (MCSPI_MODULCTRL[2] MS = 0 and MCSPI_MODULCTRL[0] SINGLE = 1), the assertion of the SPIEN_[n] signal is optional depending on device connected to the controller. In 3-pin mode (MCSPI_MODULCTRL[1] PIN34 = 1) the controller starts transmitting data when a write to the MCSPI_TXi register or the FIFO is performed. In 4-pin mode (MCSPI_MODULCTRL[1] PIN34 = 0) the assertion and de-assertion of SPIEN_[n] is controlled by software using the MCSPI_CH(i)CONF[20] FORCE bit.

12.1.3.4.3.5.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for the MCSPI word transfer to complete (wait until the MCSPI_CH(i)STAT[2] EOT bit is set to 1) before disabling the current channel and enabling a different channel.
- Disable the current channel, and then enable the other channel.

12.1.3.4.3.5.2 Force SPIEN_[n] Mode

Continuous transfers are allowed manually by keeping the SPIEN_[n] signal active for successive MCSPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the SPIEN_[n] line. This mode is supported by all channels and any controller sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the SPIEN_[n] active mode is supported when:

- A single channel is used (with the MCSPI_MODULCTRL[0] SINGLE bit set to 1).
- Transfer parameters are loaded in the configuration register of the appropriate channel (MCSPI_CH(i)CONF).

  The state of the SPIEN_[n] signal is programmable:
  - Writing 1 to the MCSPI_CH(i)CONF[20] FORCE bit drives the SPIEN_[n] line high when the MCSPI_CH(i)CONF[6] EPOL bit is set to 0. SPIEN_[n] is driven low when the MCSPI_CH(i)CONF[6] EPOL bit is set to 1.
  - Writing 0 to the MCSPI_CH(i)CONF[20] FORCE bit drives the SPIEN_[n] line low when the MCSPI_CH(i)CONF[6] EPOL bit is set to 0. SPIEN_[n] is driven high when the MCSPI_CH(i)CONF[6] EPOL bit is set to 1.
- A single channel is enabled (the MCSPI_CH(i)CTRL[0] EN bit is set to 1). The first enabled channel activates the SPIEN_[n] line.

When the channel is enabled, the SPIEN_[n] signal activates with the programmed polarity. As in the multichannel controller mode, the transfer start depends on the status of the MCSPI_TXi register (the MCSPI_CH(i)STAT[1] TXS bit), the status of the MCSPI_RXi register (the MCSPI_CH(i)STAT[1] RXS bit), and the defined mode (the MCSPI_CH(i)CONF[13-12] TRM bit field) of the channel enabled.

The MCSPI_CH(i)STAT[2] EOT bit gives the transfer status of each MCSPI word. The RXx_FULL bit in the MCSPI_IRQSTATUS register is set when received data is loaded from the shift register to the MCSPI_RXi register.

A change in the configuration parameters is propagated directly on the MCSPI interface. If the SPIEN_[n] signal is activated, ensure that the configuration is changed only between MCSPI words to avoid corrupting the current transfer.

Copyright © 2026 Texas Instruments Incorporated

Note

To avoid data corruption, SPIEN_[n] polarity and SPICLK phase and SPICLK polarity must not be modified when the SPIEN_[n] signal is activated.

A delay between MCSPI words that requires the connected MCSPI peripheral device to switch from one configuration to another (for instance, from transmit-only to receive-only) must be handled by software.

At the end of the last MCSPI word, the channel must be deactivated (the MCSPI_CH(i)CTRL[0] EN bit set to 0) and SPIEN_[n] can be forced to its INACTIVE state using the MCSPI_CH(i)CONF[20] FORCE bit.

Figure 12-37 and Figure 12-38 show successive transfers with SPIEN_[n] maintained active low with a different configuration for each MCSPI word in single-data-pin and dual-data-pin interface modes, respectively.

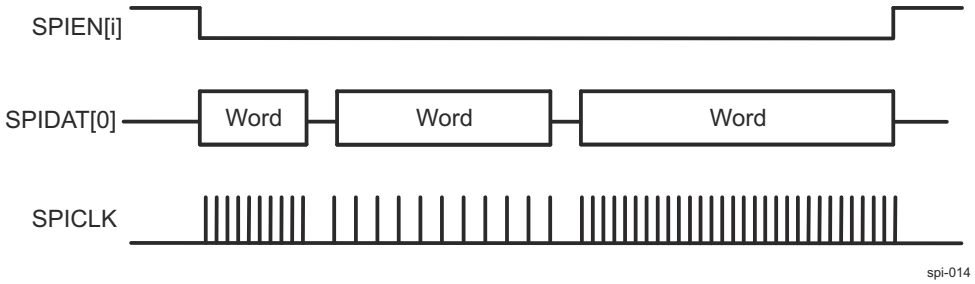


Figure 12-37. Continuous Transfers With SPIEN_[n] Maintained Active (Single-Data-Pin Interface Mode)

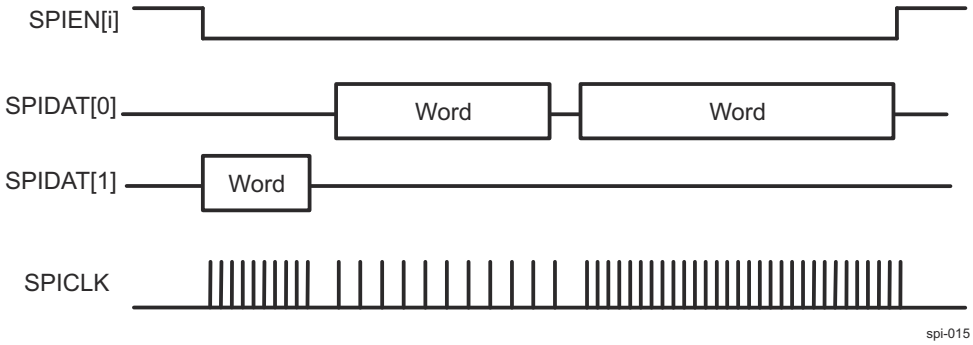


Figure 12-38. Continuous Transfers With SPIEN_[n] Maintained Active (Dual-Data-Pin Interface Mode)

Note

The SPIEN_[n] signal can be maintained active via software using the MCSPI_CH(i)CONF[20] FORCE bit only when the MCSPI_MODULCTRL[0] SINGLE bit is set to 0x1.

12.1.3.4.3.5.3 Turbo Mode

Turbo mode improves the throughput of the MCSPI interface when a single channel is enabled by allowing transfers until the shift register and the MCSPI_RXi register are full. Turbo mode is time saving when a transfer exceeds two words. This mode is programmable per channel (through the MCSPI_CH(i)CONF[9] TURBO bit).

When several channels are enabled, the TURBO bit has no effect and the channel access to the shift registers remains as previously described.

In turbo mode, Rule 1 and Rule 2 apply, but Rule 3 does not (see Section 12.1.3.4.3.2, *Controller Transmit-and-Receive Mode (Full Duplex)*). An enabled channel can be scheduled if its receive register is full (the MCSPI_CH(i)STAT[0] RXS bit) when the shift-register is assigned until the shift register is full.

Copyright © 2026 Texas Instruments Incorporated

The MCSPI_RXi register cannot be overwritten in turbo mode. Consequently, the MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set in this mode.

12.1.3.4.3.6 Start-Bit Mode

In start-bit mode, an extended bit is added before the MCSPI word to indicate whether the next MCSPI word must be handled as a command or as data. This feature is available only in controller mode. Start-bit mode cannot be used at the same time as turbo mode and/or force SPIEN_[n] mode. In this case, only one channel can be used; round-robin arbitration is not possible.

This mode is programmable per channel by setting the MCSPI_CH(i)CONF[23] SBE bit to 1. The polarity of the extended bit is programmable per channel. When the MCSPI_CH(i)CONF[24] SBPOL bit is set to 0, the MCSPI word must be handled as a command. When the MCSPI_CH(i)CONF[24] SBPOL bit is set to 1, the MCSPI word must be handled as data. Moreover, start-bit polarity can be changed dynamically during start-bit transfer without disabling the channel for reconfiguration; in this case, users must configure the MCSPI_CH(i)CONF[24] SBPOL bit before writing the MCSPI word to be transmitted to the TX register.

12.1.3.4.3.7 Chip-Select Timing Control

The chip-select (CS) timing control is available only in controller mode with automatic CS generation (the MCSPI_MODULCTRL[0] SINGLE bit set to 0) to add a programmable delay between CS assertion and first clock edge, or CS removal and last clock edge. This option is available only in 4-pin mode when MCSPI_MODULCTRL[1] PIN34 set to 0.

This mode is programmable per channel through the MCSPI_CH(i)CONF[26-25] TCS0 bit field.

Figure 12-39 shows the CS SPIEN timing controls.

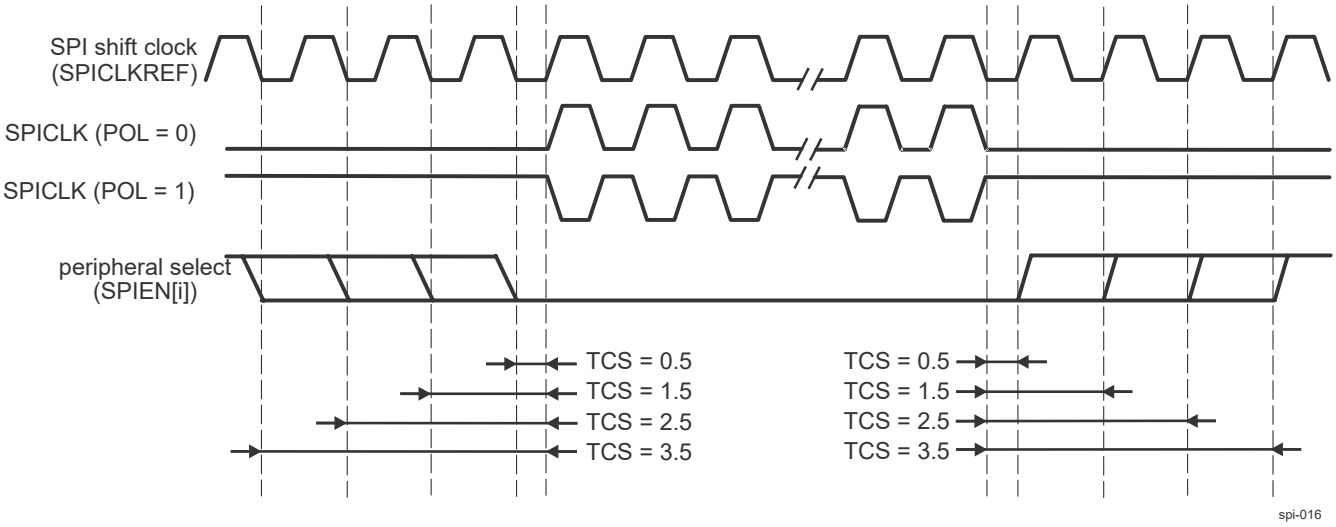


Figure 12-39. CS SPIEN Timing Controls

Note

Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between CS assertion and the first clock edge with PHA = 1 or between CS removal and the last clock edge with PHA = 0.

12.1.3.4.3.8 Programmable MCSPI Clock (SPICLK)

In controller mode, the baud rate of the MCSPI serial clock is programmable.

Copyright © 2026 Texas Instruments Incorporated

An internal reference clock, SPICLKREF, is used as input of a programmable divider (the MCSPI_CH(i)CONF[5-2] CLKD bit field) to generate the bit rate of the serial output clock SPICLK. Table 12-19 summarizes the supported divisor values.

Table 12-19. MCSPI Controller Clock Rates

Divider	Clock Rate
1	50 MHz(1)
2	25 MHz(1)
4	12.5 MHz
8	6.25 MHz
16	3.125 MHz
32	1.5625 MHz
64	781.25 kHz
128	~390 kHz
256	~195 kHz
512	~97.7 kHz
1024	~48.8 kHz
2048	~24.4 kHz
4096	~12.2 kHz
8192 and higher: Division not supported	–

(1) These frequencies are not necessarily supported by all MCSPI modules. For more information, see the *Timing Requirements and Switching Characteristics* chapter in the device-specific Datasheet.

12.1.3.4.3.8.1 Clock Ratio Granularity

By default, the clock division ratio is defined by the MCSPI_CH(i)CONF[5-2] CLKD bit field with power-of-2 granularity leading to a clock division in the range 1 to 4096; in this case, the duty cycle is always 50 percent. With the MCSPI_CH(i)CONF[29] CLKG bit, clock division granularity can be changed to one clock cycle; in that case the MCSPI_CH(i)CTRL[15-8] EXTCLK bit field is concatenated with the MCSPI_CH(i)CONF[5-2] CLKD bit field to give a 12-bit-wide division ratio in the range 1 to 4096.

When granularity is one clock cycle (the CLKG bit set to 1), for the odd value of the clock ratio, the clock high level lasts one clock cycle more than the low level, depending on the MCSPI_CH(i)CONF[1] POL and MCSPI_CH(i)CONF[0] PHA bits (see Table 12-20).

Table 12-20. CLKSPIO High/Low Time Computation

Clock Ratio F_{RATIO}	CLKSPIO High Time	CLKSPIO Low Time
1	T_{HIGH_REF}	T_{LOW_REF}
Even >= 2	T_ref * (F_{RATIO}/2)	T_ref * (F_{RATIO}/2)
Odd >= (POL = PHA)	T_ref * (F_{RATIO}– 1)/2	T_ref * (F_{RATIO} + 1)/2
Odd >= (POL =! PHA)	T_ref * (F_{RATIO}+ 1)/2	T_ref * (F_{RATIO} – 1)/2

Note

F_{RATIO} = SPICLK frequency (F_{OUT}) division ratio
T_{HIGH} = SPICLK high time period
T_{LOW} = SPICLK low time period
T_ref = MCSPI_FCLK period
T_{HIGH_REF} = MCSPI_FCLK high time period
T_{LOW_REF} = MCSPI_FCLK low time period

If the CLKG bit is set to 1; F_{RATIO} = EXTCLK concatenated with CLKD + 1.

Copyright © 2026 Texas Instruments Incorporated

For odd ratio values, the duty cycle is calculated as follows:

Duty_cycle = $(1 – 1/F_{RATIO})/2$

Table 12-21 shows examples of clock granularity with a clock source frequency of 50 MHz.

Table 12-21. Clock Granularity Examples

EXTCLK	CLKD	CLKG	F_{RATIO}	PHA	POL	T_{HIGH} (ns)	T_{LOW} (ns)	T_{PERIOD} (ns)	Duty Cycle	F_{OUT} (MHz)
X	0	0	1	X	X	10.0	10.0	20.0	50–50	50
X	1	0	2	X	X	20.0	20.0	40.0	50–50	25
X	2	0	4	X	X	40.0	40.0	80.0	50–50	12.5
X	3	0	8	X	X	80.0	80.0	160.0	50–50	6.2
0	0	1	1	X	X	10.0	10.0	20.0	50–50	50
0	1	1	2	X	X	20.0	20.0	40.0	50–50	25
0	2	1	3	1	0	40.0	20.0	60.0	66–33	16.6
0	2	1	3	1	1	20.0	40.0	60.0	33–66	16.6
0	3	1	4	X	X	40.0	40.0	80.0	50–50	12.5
5	0	1	81	1	0	820.0	800.0	1620.0	50.6–49.4	0.617
5	7	1	88	X	X	880.0	880.0	1760.0	50–50	0.568

12.1.3.4.4 MCSPI Peripheral Mode

To select the MCSPI peripheral mode, set the MCSPI_MODULCTRL[2] MS bit.

A MCSPI peripheral device can be connected to up to four external MCSPI controller devices but handles transactions with one MCSPI controller device at a time.

In peripheral mode, the MCSPI initiates data transfer on the data lines (SPIDAT[0] and SPIDAT[1]) when it is selected by an active control signal (SPIEN_[n]) and receives an MCSPI clock (SPICLK) from the external MCSPI controller device. Only channel 0 can be configured as a peripheral but through the MCSPI_CH0CONF_0[22-21] SPIENSLV bit field any of the SPIEN_[n] signals can be used to select the MCSPI module. In peripheral mode and when the MCSPI_MODULCTRL[1] PIN34 is set to 0x0 (default behaviour), the MCSPI uses the edge of SPIEN_[n] to detect word length. For this reason, SPIEN_[n] must become inactive between each word.

When the MCSPI_MODULCTRL[1] PIN34 is set to 0x0, the MCSPI does not support SPIEN_[n] active between MCSPI words. In this case, the MCSPI uses the edge to detect word length.

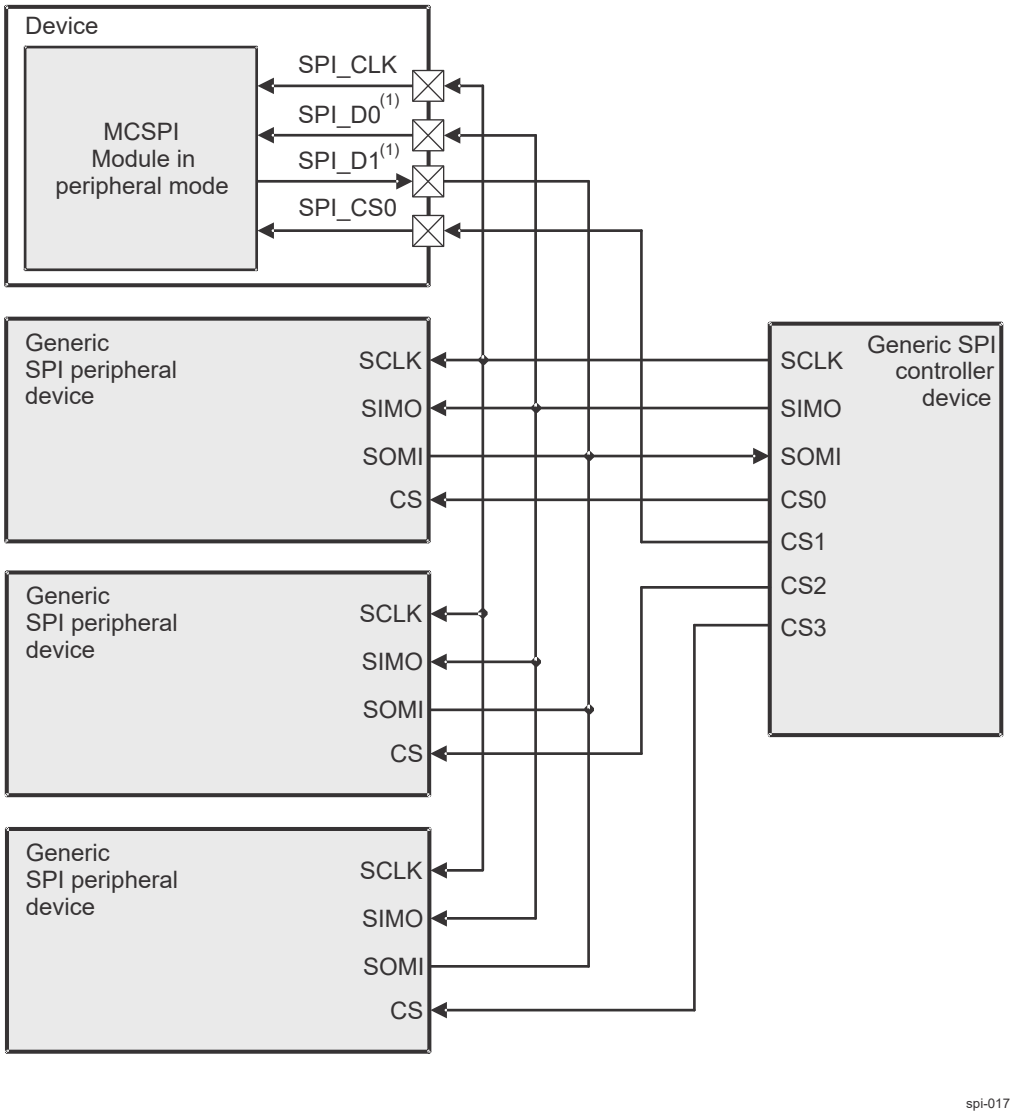
When the MCSPI_MODULCTRL[1] PIN34 is set to 0x1, a multiword transfer can be performed without needing the external MCSPI controller to deactivate SPIEN_[n] between each word as in this case the MCSPI module works in 3-pin peripheral mode and SPIEN_[n] is not needed.

12.1.3.4.4.1 Dedicated Resources

Only channel 0 can be enabled in peripheral mode.

Figure 12-40 shows an example of four peripherals wired on a single controller device.

Copyright © 2026 Texas Instruments Incorporated



A. Direction depends on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-40. Example of MCSPI Peripheral With One Controller and Multiple Peripheral Devices on Channel 0

Channel 0 in peripheral mode has the following resources:

- Its own channel enable, programmable with the MCSPI_CH0CTRL[0] EN bit. This channel must be enabled before transmission and reception.
- For this mode, the peripheral-select signal can be detected on any of the SPIEN_[n] ports. This is programmable with the MCSPI_CH0CONF[22-21] SPIENSLV bit field.
- Its own transmitter register, MCSPI_TX0, on top of the common transmit shift register. If the MCSPI_TX0 register is empty, the MCSPI_CH0STAT[1] TXS bit is set. If MCSPI is selected by an external controller (the active signal on the SPIEN_[n] port assigned to channel 0), the MCSPI_TX0 register content of channel 0 is always loaded into the shift register, whether its content is updated or not. The MCSPI_TX0 register must be loaded before MCSPI is selected by a controller.
- Its own receiver register, MCSPI_RX0, on top of the common receive shift register. If the MCSPI_RX0 register is full, the MCSPI_CH0STAT[0] RXS bit is set.

Copyright © 2026 Texas Instruments Incorporated

Note

The MCSPI_TXi and MCSPI_RXi registers are not used. Reading from or writing to a channel register other than channel 0 has no effect.

- Its own communication configuration with the following parameters through the MCSPI_CH0CONF:
  - Transmit and receive modes, programmable with the TRM field
  - Interface mode (two data pins or single data pin) and data pins assignment, both programmable with the IS and DPE bits. (The MCSPI modules are in peripheral mode after reset and must be properly configured for the modules to act in controller mode.)
  - MCSPI word length, programmable with the WL bit
  - SPIEN_[n] polarity, programmable with the EPOL bit
  - SPICLK polarity, programmable with the POL bit
  - SPICLK phase, programmable with the PHA bit

  The SPICLK frequency of a transfer is controlled by the external MCSPI controller connected to the MCSPI peripheral device. The MCSPI_CH0CONF[5-2] CLKD bit field is not used in peripheral mode.

Note

The configuration of the channel can be loaded in the MCSPI_CH0CONF only when the channel is disabled.

- Two DMA request events, read and write, synchronize read/write accesses of the DMA controller with the activity of MCSPI. DMA requests are asserted using the MCSPI_CH0CONF[15] DMAR bit for reading and the MCSPI_CH0CONF[14] DMAW bit for writing.
- Four interrupt events (see Section 12.1.3.4.7.2, *Interrupt Events in Peripheral Mode*).

12.1.3.4.4.2 Peripheral Transmit-and-Receive Mode

The peripheral receive mode is programmable (set the MCSPI_CH0CONF[13-12] TRM bit field to 0x0).

In peripheral transmit-and-receive mode, the MCSPI_TX0 register must be loaded before MCSPI is selected by an external MCSPI controller device.

After a channel is enabled, transmission and reception proceed with interrupt and DMA request events.

The MCSPI_TX0 register content is always loaded in the shift register whether it is updated or not. The event TX0_UNDERFLOW is activated accordingly and does not prevent transmission.

When the MCSPI word transfer completes (the MCSPI_CH0STAT[2] EOT bit is set to 1), the received data is transferred to the channel receive register.

To use MCSPI as a peripheral transmit-only device, the RX0_FULL and RX0_OVERFLOW interrupts and DMA read requests must be disabled due to the state of the MCSPI_RX0 register (see Section 12.1.3.4.7.2, *Interrupt Events in Peripheral Mode*).

12.1.3.4.4.3 Peripheral Transmit-Only Mode

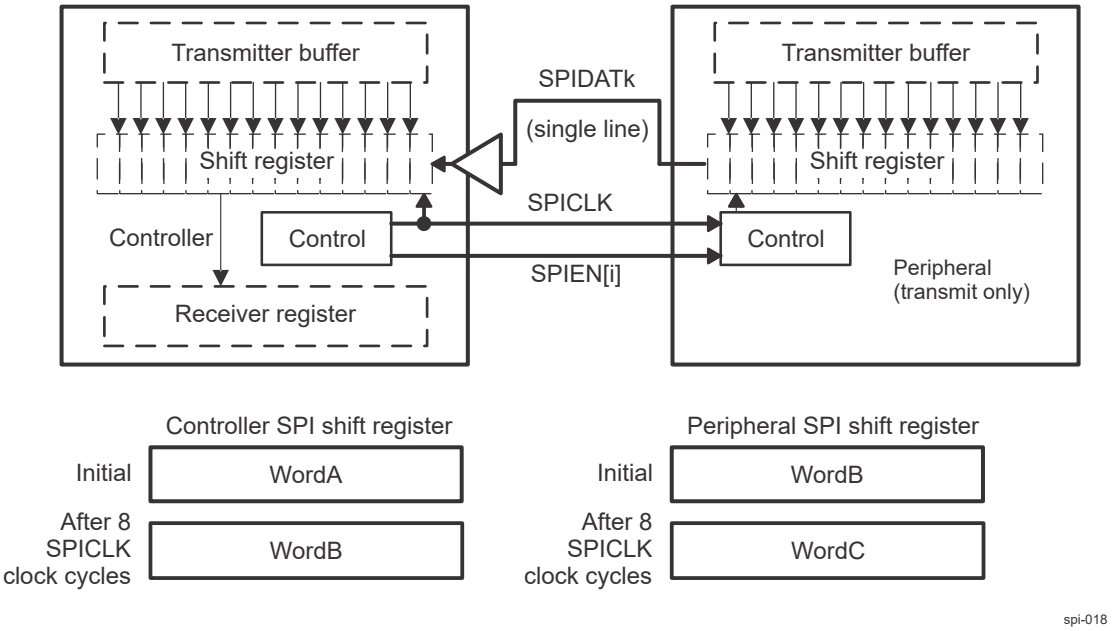
The peripheral transmit-only mode is programmable (set the MCSPI_CH0CONF[13-12] TRM bit field to 0x2) and avoids the requirement for the processor to read the MCSPI_RX0 register (minimizing data movement) only when transmission is meaningful.

To use the MCSPI as a peripheral transmit-only device, the RX0_FULL and RX0_OVERFLOW interrupts and DMA read requests must be disabled due to the state of the MCSPI_RX0 register.

When the MCSPI word transfer completes, the MCSPI_CH0STAT[2] EOT bit is set.

Figure 12-41 shows a half-duplex system with a controller device on the left and a transmit-only peripheral device on the right. Each time a bit transfers out from the peripheral, 1 bit transfers in the controller. After eight cycles of the serial clock SPICLK, WordB transfers from the peripheral to the controller.

Copyright © 2026 Texas Instruments Incorporated



k = 0 or 1 depending on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-41. MCSPI Half-Duplex Transmission (Transmit-Only Peripheral)

12.1.3.4.4.4 Peripheral Receive-Only Mode

The peripheral receive mode is programmable (set the MCSPI_CH0CONF[13-12] TRM bit field to 0x1).

In receive-only mode, the MCSPI_TX0 register must be loaded before the MCSPI is selected by an external MCSPI controller device. The MCSPI_TX0 register content is always loaded into the shift register whether it is updated or not. The TX0_UNDERFLOW event is activated accordingly and does not prevent transmission.

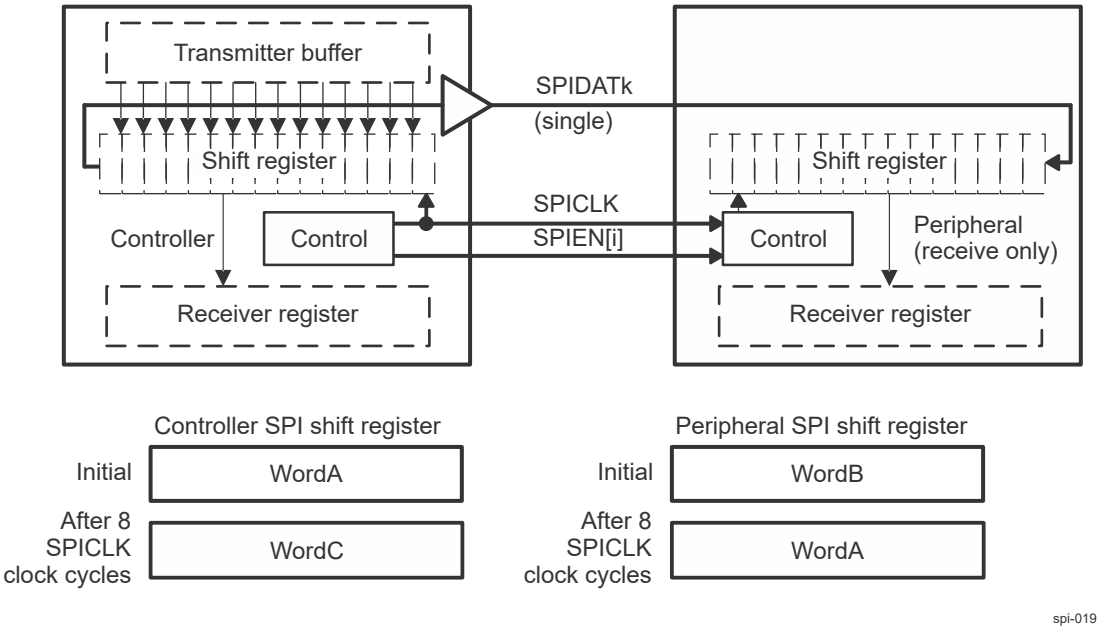
When the MCSPI word transfer completes (the MCSPI_CH0STAT[2] EOT bit is set to 1), the received data is transferred to the channel receive register.

To use the MCSPI as a peripheral receive-only device, the TX0_EMPTY and TX0_UNDERFLOW interrupts and the DMA write requests must be disabled due to the state of the MCSPI_TX0 register.

For a full-duplex transmission, the serial clock (SPICLK) synchronizes shifting and sampling of the information on the single serial data line. For full duplex, two data lines are required. If SPICLK synchronizes on a single serial data line, the data line should be half-duplex.

Figure 12-42 shows a half-duplex system with a controller device on the left and a receive-only peripheral device on the right. Each time a bit transfers out from the controller, 1 bit transfers in from the peripheral. After eight cycles of the serial clock SPICLK, WordA transfers from the controller to the peripheral.

Copyright © 2026 Texas Instruments Incorporated



k = 0 or 1 depending on MCSPI_CH(i)CONF[16] DPE0, MCSPI_CH(i)CONF[17] DPE1 and MCSPI_CH(i)CONF[18] IS bits

Figure 12-42. MCSPI Half-Duplex Transmission (Receive-Only Peripheral)

12.1.3.4.5 MCSPI 3-Pin or 4-Pin Mode

Depending on targeted application the MCSPI interface can be configured to use 3 or 4 pins through the MCSPI_MODULCTRL[1] PIN34 bit. If this bit is set to 0, MCSPI is in 4-pin mode using the SPICLK, SPIDAT[0], SPIDAT[1] and SPIEN_[n] signals. If PIN34 is set to 1 the controller is in 3-pin mode and SPIEN_[n] is not used. In this mode all options related to chip select management are useless (EPOL, FORCE and TCS0 bits of MCSPI_CH(i)CONF). 3-pin and 4-pin operation applies to both controller and peripheral modes.

12.1.3.4.6 MCSPI FIFO Buffer Management

The MCSPI controller has a built-in 64-byte buffer to unload the DMA or interrupt handler and improve data throughput.

This buffer can be used by only one channel at a time and is selected by setting the MCSPI_CH(i)CONF[28] FFER or MCSPI_CH(i)CONF[27] FFEW bit to 1. If several channels are selected and several FIFO enable bit fields are set to 1, the controller forces the buffer not to be used; the driver must set only one FIFO enable bit field.

The buffer can be used in the following modes:

- Controller or peripheral mode
- Transmit-only, receive-only, or transmit-and-receive mode
- Single channel or turbo mode, or normal round-robin mode. In round-robin mode the buffer is used by only one channel.

Every word length (MCSPI_CH(i)CONF[11-7] WL) is supported.

In transmit-and-receive mode, the buffer can be used in transmit (see Figure 12-43) or receive (see Figure 12-44) directions, or in both directions. If only one direction is chosen in transmit-and-receive mode, the full buffer is used for this direction. In both directions, the buffer is split into two halves, one for each direction (see Figure 12-45).

Copyright © 2026 Texas Instruments Incorporated

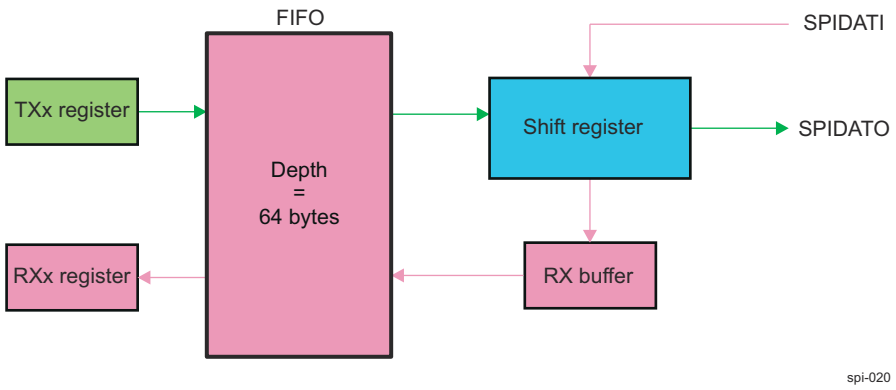


Figure 12-43. Buffer Used in Transmit Direction Only

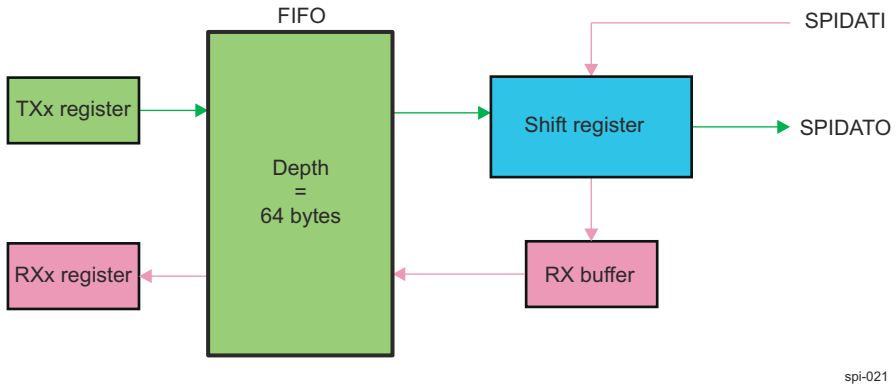


Figure 12-44. Buffer Used in Receive Direction Only

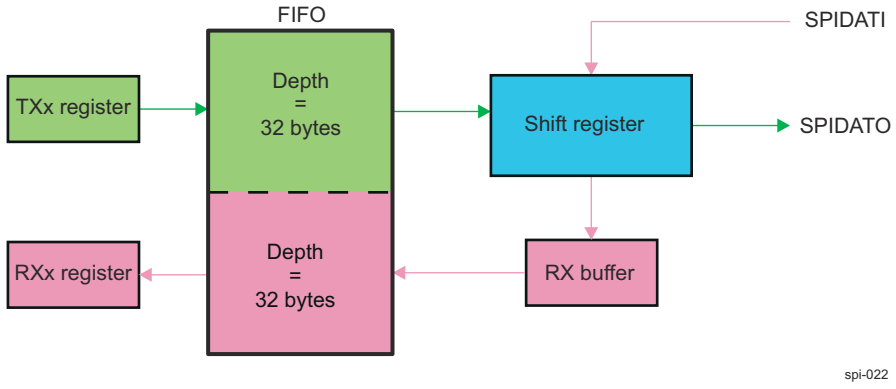


Figure 12-45. Buffer Used for Transmit and Receive Directions

Two levels (MCSPI_XFERLEVEL[5-0] AEL and MCSPI_XFERLEVEL[13-8] AFL) rule the buffer management. The granularity of these levels is 1 byte; it is not aligned with the MCSPI word length. The driver must set these values as a multiple of the MCSPI word length defined in WL. Table 12-22 lists the number of bytes written in the FIFO, depending on the word length.

Table 12-22. FIFO Writes, Word Length Relationship

	MCSPI Word Length (WL)		
	3 ≤ WL≤ 7	**8 ≤ WL ≤ 15**	**16 ≤ WL ≤ 31**
Number of bytes written in the FIFO	1 byte	2 bytes	4 bytes

Copyright © 2026 Texas Instruments Incorporated

The FIFO buffer pointers are reset when the corresponding channel is enabled or the FIFO configuration changes.

12.1.3.4.6.1 Buffer Almost Full

The MCSPI_XFERLEVEL[15-8] AFL bit field is needed when the buffer is used to receive an MCSPI word from a peripheral (the MCSPI_CH(i)CONF[28] FFER bit must be set to 1). It defines the almost-full buffer status. See Figure 12-46.

When the FIFO pointer reaches this level, an interrupt or a DMA request is sent to the processor to enable the system to read AFL + 1 bytes from the receive register.

Note

AFL + 1 must correspond to a multiple value of the MCSPI_CH(i)CONF[11-7] WL bit field.

When DMA is used, the request is de-asserted after the first receive register read.

No new request is asserted again as long as the system has not performed the correct number of read accesses.

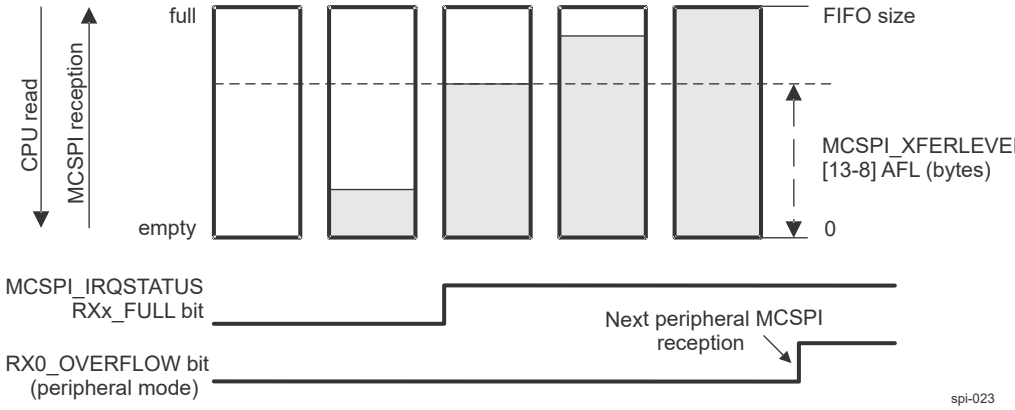


Figure 12-46. Buffer Almost Full Level (AFL)

Note

The MCSPI_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the request is asserted on the same conditions as the MCSPI_IRQSTATUS RXi_FULL flag.

12.1.3.4.6.2 Buffer Almost Empty

The MCSPI_XFERLEVEL[7-0] AEL bit field is needed when the buffer is used to transmit an MCSPI word to a peripheral (the MCSPI_CH(i)CONF[27] FFEW bit must be set to 1). It defines the almost-empty buffer status. See Figure 12-47.

When the FIFO pointer does not reach this level, an interrupt or a DMA request is sent to the processor to enable the system to write AEL + 1 bytes to the transmit register.

Note

AEL + 1 must correspond to a multiple value of the MCSPI_CH(i)CONF[11-7] WL bit field.

When DMA is used, the request is de-asserted after the first transmit register write.

No new request is asserted again as long as the system has not performed the correct number of write accesses.

Copyright © 2026 Texas Instruments Incorporated

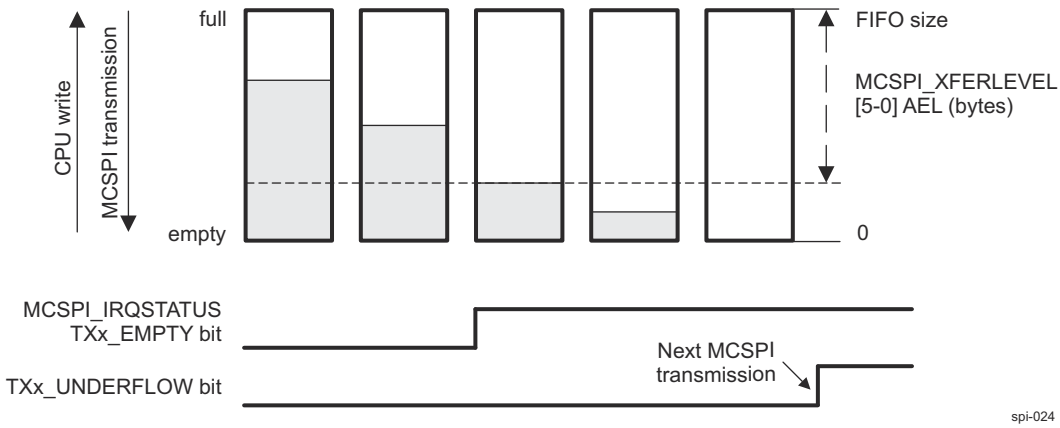


Figure 12-47. Buffer Almost Empty Level (AEL)

Note

The MCSPI_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the request is asserted on the same conditions as the MCSPI_IRQSTATUS TXi_EMPTY flag.

12.1.3.4.6.3 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user must previously configure in the MCSPI_XFERLEVEL register the AEL and AFL levels and especially the MCSPI_XFERLEVEL[31-16] WCNT bit field to define the number of MCSPI words to be transferred using the FIFO before enabling the channel.

This counter lets the controller stop the transfer correctly after a defined number of MCSPI word transfers. If WNCT is set to 0x0000, the counter is not used and the user must stop the transfer manually by disabling the channel; in this case, the user does not know how many MCSPI transfers have been done. For received words, software must poll the MCSPI_CH(i)STAT[5] RXFFE bit and read the MCSPI_RXi receive register to empty the FIFO buffer.

When the end-of-word count interrupt is generated (the MCSPI_IRQSTATUS[17] EOW bit is set), the user can disable the channel and poll the MCSPI_CH(i)STAT[5] RXFFE bit to know the last MCSPI words in the FIFO buffer and read them.

No new request is asserted as long as the system has not performed the correct number of write accesses.

12.1.3.4.6.4 Multiple MCSPI Word Access

The processor has the ability to perform multiple MCSPI word access to the receive or transmit registers within a single 32-bit interface access by setting the MCSPI_MODULCTRL[7] MOA to 1 under specific conditions:

- The channel selected has the FIFO enable.
- Only FIFO sense enabled support the kind of access.
- MCSPI_MODULCTRL[7] MOA is set to 1.
- Only 32-bit interface access and data width can be performed to receive or transmit registers, for other kind of access the processor must de-assert MCSPI_MODULCTRL[7] MOA bit.
- The level MCSPI_XFERLEVEL[7-0] AEL and MCSPI_XFERLEVEL[15-8] AFL must be 32-bit aligned, it means that AEL[0] = AEL[1] = 1 or AFL[0] = AFL[1] = 1.
- If MCSPI_XFERLEVEL[31-16] WCNT is used it must be configured according to MCSPI word length.
- The word length of MCSPI words allows to perform multiple MCSPI access, that means that MCSPI_CH(i)CONF[11-7] WL is <16.

The number of MCSPI word access depends on MCSPI word length:

Copyright © 2026 Texas Instruments Incorporated

- 3 ≤ WL ≤ 7, MCSPI word length smaller or equal to byte length, 4 MCSPI words accessed per 32-bit interface read/write. If word count is used (MCSPI_XFERLEVEL[31-16] WCNT), set the bit field to WCNT[0] = WCNT[1] = 0.
- 8 ≤ WL ≤ 15, MCSPI word length greater than byte or equal to 16-bit length, 2 MCSPI words accessed per 32-bit interface read/write. If word count is used (MCSPI_XFERLEVEL[31-16] WCNT]), set the bit field to WCNT[0] = 0.
- 16 ≤ WL Multiple MCSPI word access is not applicable.

12.1.3.4.6.5 First MCSPI Word Delay

Figure 12-48 shows the MCSPI controller ability to delay the first MCSPI word transfer to give time for system to complete some parallel processes or fill the FIFO in order to improve transfer bandwidth. This delay is applied only on first MCSPI word after MCSPI channel enabled and first write in transmit register. It is based on output clock frequency.

This option is meaningful in controller mode and single channel mode asserted through MCSPI_MODULCTRL[0] SINGLE.

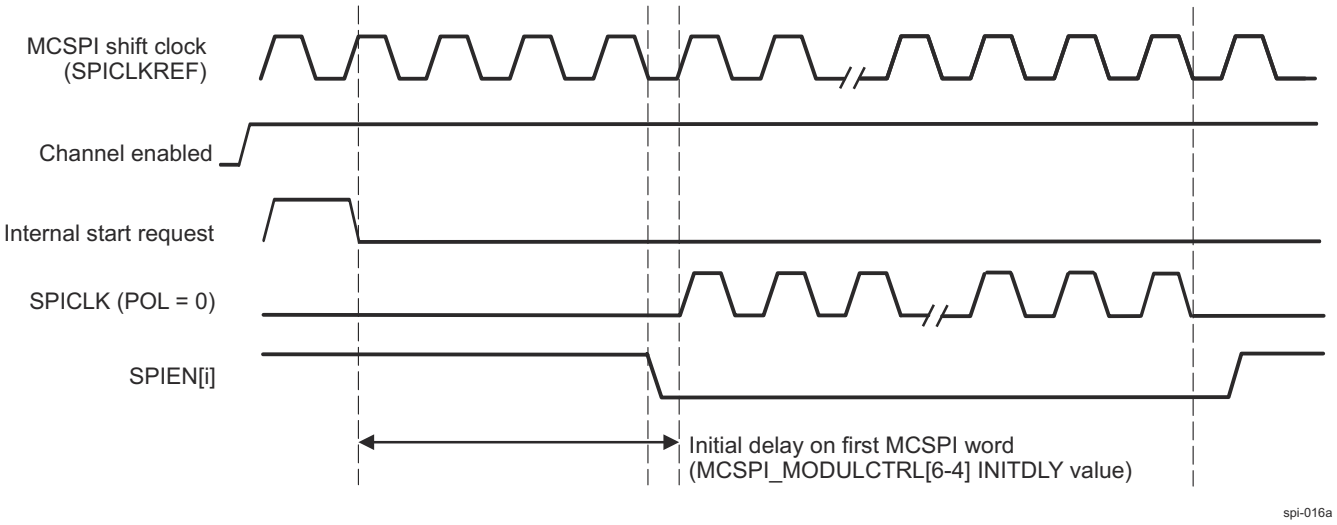


Figure 12-48. Controller Single Channel Initial Delay

Few delay values are available: No delay, 4/8/16/32 MCSPI cycles.

Its accuracy is half cycle in clock bypass mode and depends on clock polarity and phase.

12.1.3.4.7 MCSPI Interrupts

Each channel can issue interrupt events.

Each interrupt event has status bits in the MCSPI_IRQSTATUS register (RXi_FULL, TXi_UNDERFLOW, TXi_EMPTY, etc.) (where x = 0, 3) that indicate whether service is required. Each status bit has an interrupt enable bit (a mask) in the MCSPI_IRQENABLE register (RXi_FULL_ENABLE, TXi_UNDERFLOW_ENABLE, TXi_EMPTY_ENABLE, etc.).

When an interrupt occurs and a mask is later applied on it, the interrupt line is not asserted again, even if the interrupt source is not serviced.

The MCSPI supports interrupt-driven and polling operations.

12.1.3.4.7.1 Interrupt Events in Controller Mode

In controller mode, the interrupt events related to the state of the MCSPI_TXi register are TXi_EMPTY and TXi_UNDERFLOW. The interrupt event related to the state of the MCSPI_RXi register is RXi_FULL.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.4.7.1.1 TXx_EMPTY

The TXx_EMPTY event is activated when a channel is enabled and its MCSPI_TXi register is empty (transient event). Enabling a channel automatically triggers this event, except in controller receive-only mode (see Section 12.1.3.4.3.4, *Controller Receive-Only Mode*). When the FIFO buffer is enabled (the MCSPI_CH(i)CONF[27] FFEW bit is set to 1), the MCSPI_IRQSTATUS TXi_EMPTY bit is set as soon as there is enough space in the buffer to write a number of bytes defined by the MCSPI_XFERLEVEL[5-0] AEL bit field.

The MCSPI_TXi register must be loaded with data to remove the source of the interrupt; the MCSPI_IRQSTATUS TXi_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXi_EMPTY event is asserted as long as the processor has not performed the number of writes into the MCSPI_TXi register defined by the MCSPI_XFERLEVEL[5-0] AEL bit field. The processor must perform the correct number of writes.

12.1.3.4.7.1.2 TXx_UNDERFLOW

The event TXx_UNDERFLOW is activated when the channel is enabled and if the MCSPI_TXi register or the FIFO is empty (not updated with new data) when an external controller device starts a data transfer with the MCSPI (transmit and receive).

The TXi_UNDERFLOW is a harmless warning in controller mode.

To avoid having a TXi_UNDERFLOW event at the beginning of a transmission, the TXi_UNDERFLOW event is not activated when no data has been loaded into the MCSPI_TXi register, because the channel is enabled. To avoid having a TXx_UNDERFLOW event, the MCSPI_TXi register must seldom be loaded.

The MCSPI_IRQSTATUS TXi_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

12.1.3.4.7.1.3 RXx_ FULL

The RXx_FULL event is activated when a channel is enabled and the MCSPI_RXi register becomes filled (transient event). When the FIFO buffer is enabled (the MCSPI_CH(i)CONF[28] FFER bit is set to 1), RXi_ FULL is asserted as soon as the number of bytes held in the FIFO to be read reaches the MCSPI_XFERLEVEL[13-8] AFL threshold.

The MCSPI_RXi register must be read to remove the source of the interrupt; the MCSPI_IRQSTATUS RXi_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXi_FULL event is asserted as long as the processor has not performed AFL + 1 reads into MCSPI_RXi. The processor must perform the correct number of reads.

12.1.3.4.7.1.4 End Of Word Count

The MCSPI_IRQSTATUS[17] EOW event (end of word count) is activated when the channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller performs the number of transfers defined in the MCSPI_XFERLEVEL[31-16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The end of word count interrupt also indicates that the MCSPI transfer is halted on the channel using the FIFO buffer as soon as MCSPI_XFERLEVEL[31-16] WCNT is not reloaded and the channel is not re-enabled.

The MCSPI_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

12.1.3.4.7.2 Interrupt Events in Peripheral Mode

In peripheral mode, the interrupt events related to the state of the MCSPI_TXi register are TX0_EMPTY and TX0_UNDERFLOW. The interrupt events related to the state of the MCSPI_RXi are RX0_FULL

Copyright © 2026 Texas Instruments Incorporated

and RX0_OVERFLOW (channels 1, 2, and 3 do not have a receiver overflow status bit). See the MCSPI_IRQSTATUS register.

12.1.3.4.7.2.1 TXx_EMPTY

The TXi_EMPTY event is activated when a channel is enabled and its MCSPI_TXi register is empty. Enabling the channel automatically raises this event. If the FIFO buffer is enabled (the MCSPI_CH(i)CONF[27] FFEW bit is set to 1), the TXx_EMPTY event is asserted as soon as there is enough space in buffer to write a number of bytes defined by the MCSPI_XFERLEVEL[5-0] AEL bit field.

The MCSPI_TXi register must be loaded with data to remove the source of the interrupt; the MCSPI_IRQSTATUS TXi_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXi_EMPTY event is asserted as long as the processor has not performed the number of writes into the MCSPI_TXi register defined by MCSPI_XFERLEVEL[5-0] AEL bit field. The processor must perform the correct number of writes.

12.1.3.4.7.2.2 TXx_UNDERFLOW

The TXx_UNDERFLOW event is activated when a channel is enabled and if the MCSPI_TXi register is empty (not updated with new data) when an external controller device starts a data transfer with the MCSPI (transmit and receive).

When FIFO is enabled, the data emitted while the underflow event is raised is not the last data written in the FIFO but the next data in the FIFO (an old transmitted value or a dummy data in the FIFO has been reset).

TXx_UNDERFLOW indicates an error (data loss) in peripheral mode.

To avoid having a TXi_UNDERFLOW event at the beginning of a transmission, the TXi_UNDERFLOW event is not activated when no data has been loaded into the MCSPI_TXi register because the channel is enabled.

The MCSPI_IRQSTATUS TXx_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

12.1.3.4.7.2.3 RXx_FULL

The RXx_FULL event is activated when a channel is enabled and the MCSPI_RXi register is being filled (transient event). When the FIFO buffer is enabled (the MCSPI_CH(i)CONF[28] FFER bit is set to 1), RXi_FULL is asserted as soon as the number of bytes held in the buffer to read defined by the MCSPI_XFERLEVEL[13-8] AFL bit field.

The MCSPI_RXi register must be read to remove the source of the interrupt; the MCSPI_IRQSTATUS RXi_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXi_FULL event is asserted as long as the processor has not performed AFL + 1 reads into MCSPI_RXi. The processor must perform the correct number of reads.

12.1.3.4.7.2.4 RX0_OVERFLOW

The RX0_OVERFLOW event is activated in peripheral mode in transmit-and-receive mode or receive-only mode when a channel is enabled and the MCSPI_RXi register or FIFO is full when a h MCSPI word is received. The MCSPI_RXi register is always overwritten with the new MCSPI word. If the FIFO is enabled, data within the FIFO are overwritten; it must be considered as corrupted. The RX0_OVERFLOW event should not appear in peripheral mode using the FIFO.

The RX0_OVERFLOW event indicates an error (data loss) in peripheral mode.

The MCSPI_IRQSTATUS[3] RX0_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

Copyright © 2026 Texas Instruments Incorporated

12.1.3.4.7.2.5 End Of Word Count

The MCSPI_IRQSTATUS[17] EOW event (end of word count) is activated when the channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller performs the number of transfers defined in the MCSPI_XFERLEVEL[31-16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The end of word count interrupt also indicates that the MCSPI transfer is halted on the channel using the FIFO buffer as soon as WCNT is not reloaded and the channel is not re-enabled.

The MCSPI_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

12.1.3.4.7.3 Interrupt-Driven Operation

An interrupt enable bit in the MCSPI_IRQENABLE register can be set to enable each event to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the processor must:

1. Read the MCSPI_IRQSTATUS register to identify which event occurred.
2. Read the MCSPI_RXi register that corresponds to the event to remove the source of an RXi_FULL event or write into the MCSPI_TXi register that corresponds to the event to remove the source of a TXi_EMPTY event. No action is required to remove the source of the TXi_ UNDERFLOW and RX0_OVERFLOW events.
3. Set the corresponding bit of the MCSPI_IRQSTATUS register to 1 to clear an interrupt status and then release the interrupt line.

The interrupt status bit must always be reset after channel enabling and before events are enabled as interrupt sources.

12.1.3.4.7.4 Polling

When the interrupt capability of an event is disabled in the MCSPI_IRQENABLE register, the interrupt line is not asserted, but the status bits in the MCSPI_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.

Once the expected event occurs:

- RXx_FULL: To remove the source of the event, the processor must read the corresponding MCSPI_RXi register.
- TXx_EMPTY: To remove the source of the event, the processor must write into the corresponding MCSPI_TXi register.
- TXx_UNDERFLOW and RX0_OVERFLOW: No action is required to remove the source of the event.

To clear an interrupt, set the corresponding status bit of the MCSPI_IRQSTATUS register to 1. This does not affect the interrupt line state.

12.1.3.4.8 MCSPI DMA Requests

Each MCSPI channel, if enabled, can issue DMA requests. There are two DMA request lines per MCSPI channel (one for read and one for write).

The DMA read request line is asserted when the MCSPI channel is enabled and new data is available in the receive register of the MCSPI channel. A DMA read request can be individually masked with the MCSPI_CH(i)CONF[15] DMAR bit. The DMA read request line is de-asserted when reading of the MCSPI_RXi register of the MCSPI channel completes.

The DMA write request line is asserted when the MCSPI channel is enabled and the MCSPI_TXi register of the MCSPI channel is empty. A DMA write request can be individually masked with the MCSPI_CH(i)CONF[14] DMAW bit. The DMA write request line is de-asserted when loading of the MCSPI_TXi register of the channel completes.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.4.9 MCSPI Power Saving Management

Power consumption can be optimized by switching off internal clocks (interface and functional clock) when there is no activity.

12.1.3.4.9.1 Normal Mode

In normal mode, internal MCSPI module clocks are automatically switched off (autogated) when there is no activity in peripheral or controller mode.

Autogating of the module interface clock and functional clock occurs when the following conditions are met:

- The MCSPI_SYSCONFIG[0] AUTOIDLE bit is set.
- In controller mode, there is no data to transmit or receive in all channels.
- In peripheral mode, the MCSPI is not selected by the external controller and there are no register accesses.

Autogating of the module interface clock and functional clock stops when the following conditions are met:

- In controller mode, an internal access occurs.
- In peripheral mode, an internal access occurs or the MCSPI is selected by the external controller.

12.1.3.4.9.2 Idle Mode

Note

Some of the MCSPI features described in this section may not be supported on this family of devices. For more information, see *MCSPI Not Supported Features*.

At the power management level, when all conditions to shut off the MCSPI_FCLK or MCSPI_ICLK clocks are met, the corresponding LPSC asserts a clock stop request to the MCSPI. Although this procedure is completely hardware-oriented and out of software control, the method in which the MCSPI module acknowledges the clock stop request can be configured through the MCSPI_SYSCONFIG[4-3] SIDLEMODE bit field.

The settings of the SIDLEMODE bit field and the related acknowledgement modes are:

- Force-idle mode (the MCSPI_SYSCONFIG[4-3] SIDLEMODE bit field is set to 0x0): The MCSPI module acknowledges unconditionally the clock stop request, regardless of its internal operations. This mode must be used carefully in this case because it does not prevent the loss of data when the clock is switched off.
- No-idle mode (the SIDLEMODE bit field is set to 0x1): The MCSPI never acknowledges the clock stop request and is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient to save power because it does not allow shut off of MCSPI_FCLK and MCSPI_ICLK.
- Smart-idle mode (the SIDLEMODE bit field is set to 0x2): The MCSPI acknowledges the clock stop request, depending on its internal activity. MCSPI acknowledges the shut off of MCSPI_FCLK and MCSPI_ICLK only when all pending transactions, IRQs, or DMA requests are treated. This is the best approach for efficient system power management.

When configured in smart-idle mode, the MCSPI also offers an additional feature to control gating of MCSPI_FCLK or MCSPI_ICLK. The MCSPI_SYSCONFIG[9-8] CLOCKACTIVITY bit field determines which clock shuts down (MCSPI_FCLK, MCSPI_ICLK, neither clock, or both clocks).

The setting of the CLOCKACTIVITY bit field is used internally to the MCSPI to determine on which part of the module the conditions to acknowledge the clock stop request are tested. For example, if MCSPI_FCLK is not shut off on clock stop request, the MCSPI considers only MCSPI_ICLK and the associated pending activities before acknowledging the request.

Some MCSPI features are associated with MCSPI_ICLK and others with MCSPI_FCLK. Using the CLOCKACTIVITY bit field with the smart-idle mode ensures that the features associated with the clock that remains active are always enabled, even if MCSPI acknowledges the clock stop request.

12.1.3.4.9.2.1 Force-Idle Mode

Force-idle mode is enabled and exited as follows:

Copyright © 2026 Texas Instruments Incorporated

- Force-idle mode is enabled when the MCSPI_SYSCONFIG[4-3] SIDLEMODE bit field is set to 0x0.
  - In force-idle mode, the MCSPI responds unconditionally to the clock stop request by de-asserting unconditionally the interrupt and DMA request lines, if asserted.
  - The transition from normal mode to idle mode does not affect the interrupt event bits of the MCSPI_IRQSTATUS register.
  - In force-idle mode, because the module must be disabled, the interrupt and DMA request lines are likely de-asserted. The interface clock and MCSPI clock provided to the MCSPI can be switched off.
  - A clock stop request during an MCSPI data transfer can lead to an unexpected and unpredictable result. Software must avoid such a request.

Copyright © 2026 Texas Instruments Incorporated

12.1.3.5 MCSPI Programming Guide

This section describes the low-level hardware programming sequences for the configuration and use of the MCSPI module.

12.1.3.5.1 MCSPI Global Initialization

12.1.3.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the MCSPI module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the MCSPI. For further information, see *MCSPI Integration* and *MCSPI Environment*.

Table 12-23 lists the information on the global initialization of the surrounding modules.

Table 12-23. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts and if this applicable, see *Interrupts*.
MCU_M4FSS0_CORE0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MCU_M4FSS0_CORE0 interrupts and if this is applicable, see *Interrupts*.
R5FSS0/1_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0/1_CORE0/1 interrupts and if this is applicable, see *Interrupts*.
PRUSS	Device INTCs must be configured to enable the interrupt request generation. For information about enabling PRUSS interrupts and if this is applicable, see *Interrupts*.
PDMA0 and PDMA1	Device INTCs must be configured to enable the interrupt request generation.
MCU_PLLCTRL0 and PLLCTRL0	PLL controller's configuration must be done to enable the module clocks. For more information, see *Clocking*.

12.1.3.5.1.2 MCSPI Global Initialization

12.1.3.5.1.2.1 Main Sequence – MCSPI Global Initialization

The procedure in Table 12-24 can be used to initialize MCSPI when performing software reset.

Table 12-24. MCSPI Global Initialization

Step	Register/Bit Field/Programming Model	Value
Perform a software reset.	MCSPI_SYSCONFIG[1] SOFTRESET	1
Wait until reset is finished?	MCSPI_SYSSTATUS[0] RESETDONE	=1
Configure static settings (such as SPI controller or peripheral) as required.	MCSPI_MODULCTRL[8-0]	0x-
Write MCSPI_SYSCONFIG	MCSPI_SYSCONFIG	0x-

12.1.3.5.2 MCSPI Operational Mode Configuration

12.1.3.5.2.1 MCSPI Operational Modes

The selection of the working mode is done with the MCSPI_CH(i)CONF register.

Table 12-25. MCSPI Receive Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set receive mode for the channel.	MCSPI_CH(i)CONF[13-12] TRM	0x1
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPI_CH(i)CONF	0x-
Reset the status bits.	MCSPI_IRQSTATUS	0x0

Table 12-26. MCSPI Transmit Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set transmit mode for the channel.	MCSPI_CH(i)CONF[13-12] TRM	0x2

Copyright © 2026 Texas Instruments Incorporated

Table 12-26. MCSPI Transmit Mode Initialization (continued)

Step	Register/Bit Field/Programming Model	Value
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPI_CH(i)CONF	0x-
Reset the status bits.	MCSPI_IRQSTATUS	0x0

Table 12-27. MCSPI Transmit-and-Receive Mode Initialization

Step	Register/Bit Field/Programming Model	Value
Set transmit and receive mode for the channel.	MCSPI_CH(i)CONF[13-12] TRM	0x0
Configure SPI clock polarity/phase, clock divider, word length, and others for the channel.	MCSPI_CH(i)CONF	0x-
Reset the status bits.	MCSPI_IRQSTATUS	0x0

12.1.3.5.2.1.1 Common Transfer Sequence

MCSPI module allows the transfer of one or several words, according to different modes:

- CONTROLLER Normal, CONTROLLER Turbo, PERIPHERAL
- TRANSMIT–RECEIVE, TRANSMIT-ONLY, RECEIVE-ONLY
- Write and Read requests: Interrupts, DMA
- SPIEN_[n] lines assertion/deassertion: automatic, manual

For all these sequences, the host process contains the main process and the interrupt routines.

The interrupt routines are called on the interrupt signals or by an internal call if the module is used in polling mode.

Table 12-28 represents the main sequence which is common to all transfers.

In multi-channel controller mode, the sequences of different channels can be run simultaneously.

Table 12-28. Common Transfer Sequence (Main Process)

Step	Register/Bit Field/Programming Model	Value
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
Write MCSPI_IRQENABLE to enable interrupts	MCSPI_IRQENABLE	0x-
Write MCSPI_CH(i)CONF to configure the channel	MCSPI_CH(i)CONF	0x-
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait for the first write request (TX empty or DMA write)		
Write the transmitter register with data	MCSPI_TXi	0x-
Wait for the host event for end of transfer		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.2.1.2 End of Transfer Sequences

The end of transfer depends on the transfer mode. Table 12-29 summarizes the type of end of transfer per transfer mode and gives a reference to the appropriate section for details.

Table 12-29. End of Transfer Sequences

		TRANSMIT-AND-RECEIVE		TRANSMIT-ONLY		RECEIVE-ONLY	
		INTERRUPT	DMA	INTERRUPT	DMA	INTERRUPT	DMA
CONTROLLER Normal	End of transfer sequence	See Section 12.1.3.5.2.1.3		See Section 12.1.3.5.2.1.4.1	See Section 12.1.3.5.2.1.4.2	See Section 12.1.3.5.2.1.5.1	See Section 12.1.3.5.2.1.5.2
	Minimum number of word	1	1	1	1	1	2
	DMA transfer size		N		N		N-1

Copyright © 2026 Texas Instruments Incorporated

Table 12-29. End of Transfer Sequences (continued)

		TRANSMIT-AND-RECEIVE		TRANSMIT-ONLY		RECEIVE-ONLY	
		INTERRUPT	DMA	INTERRUPT	DMA	INTERRUPT	DMA
CONTROLLER Turbo	End of transfer sequence	See Section 12.1.3.5.2.1.3		See Section 12.1.3.5.2.1.4.1	See Section 12.1.3.5.2.1.4.2	See Section 12.1.3.5.2.1.6.1	See Section 12.1.3.5.2.1.6.2
	Minimum number of word	1	1	1	1	2	3
	DMA transfer size		N		N		N-2
PERIPHERAL	End of transfer sequence	See Section 12.1.3.5.2.1.3		See Section 12.1.3.5.2.1.4.1	See Section 12.1.3.5.2.1.4.2	See Section 12.1.3.5.2.1.7	
	Minimum number of word	1	1	1	1	1	1
	DMA transfer size		N		N	N	N

The transfer to execute has a size of N words.

The different sequences can be merged in one process to manage transfers of several types. The end of transfer sequences are described from the start of the channel.

In these sequences, some soft variables are used:

- write_count = 0
- read_count = 0
- channel_enable = FALSE
- last_transfer = FALSE
- last_request = FALSE

They are initialized before starting the channel.

12.1.3.5.2.1.3 Transmit-and-Receive (Controller and Peripheral)

If the requests are configured in DMA, write_count and read_count are assigned with 'N' when the DMA handlers have completed their 'N' CBASS0 accesses.

Table 12-30. Transmit-and-Receive (Controller and Peripheral) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait for write_count = N **AND** read_count = N		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

Table 12-31. Transmit-and-Receive (Controller and Peripheral) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: TXi_EMPTY		
Write the transmitter register with data	MCSPI_TXi	0x-
Increment write_count +1		
IF: RXi_FULL		
Read the receiver register	MCSPI_RXi	
Increment read_count +1		
ENDIF		

Copyright © 2026 Texas Instruments Incorporated

12.1.3.5.2.1.4 Transmit-Only (Controller and Peripheral)

12.1.3.5.2.1.4.1 Based on Interrupt Requests

Table 12-32. Transmit-Only With Interrupts (Controller and Peripheral) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until last transfer = TRUE		
Wait for end of transfer	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

Table 12-33. Transmit-Only With Interrupts (Controller and Peripheral) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: TXx_EMPTY **AND** write_count < N		
Write the transmitter register with data	MCSPI_TXi	0x-
Increment write_count +1		
ELSEIF: write_count ≥ N		
last_transfer = TRUE		
ENDIF		

12.1.3.5.2.1.4.2 Based on DMA Write Requests

When the DMA handler has completed its ‘N’ CBASS0 accesses, write_count is assigned with ‘N’.

Table 12-34. Transmit-Only With DMA (Controller and Peripheral) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until write_count = N		
Disable DMA write request	MCSPI_CH(i)CONF[14] DMAW	0
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

Table 12-35. Transmit-Only With DMA (Controller and Peripheral) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: TXi_EMPTY **AND** write_count = N		
last_transfer = TRUE		
ENDIF		

12.1.3.5.2.1.5 Controller Normal Receive-Only

12.1.3.5.2.1.5.1 Based on Interrupt Requests

Table 12-36. Receive-Only With Interrupt (Controller Normal) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until last_request = TRUE		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register	MCSPI_RXi	0x-

Copyright © 2026 Texas Instruments Incorporated

Table 12-36. Receive-Only With Interrupt (Controller Normal) (Main Process) (continued)

Step	Register/Bit Field/Programming Model	Value
Increment read_count +1		

Table 12-37. Receive-Only With Interrupt (Controller Normal) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: RXi_FULL **AND** read_count = N - 1		
last_request = TRUE		
ELSEIF: read_count ≠ N - 1		
Read the receiver register	MCSPI_RXi	0x-
Increment read_count +1		
ENDIF		

12.1.3.5.2.1.5.2 Based on DMA Read Requests

When the DMA handler has completed its ‘N-1’ CBASS0 accesses, read_count is assigned with ‘N-1’.

Table 12-38. Receive-Only With DMA (Controller Normal) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until read_count = N - 1		
Disable DMA read request	MCSPI_CH(i)CONF[15] DMAR	0
Wait until last_transfer = TRUE		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register	MCSPI_RXi	0x-
Increment read_count +1		

Table 12-39. Receive-Only With DMA (Controller Normal) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: RXi_FULL **AND** read_count = N		
last_transfer = TRUE		
ENDIF		

12.1.3.5.2.1.6 Controller Turbo Receive-Only

12.1.3.5.2.1.6.1 Based on Interrupt Requests

Table 12-40. Receive-Only With Interrupt (Controller Turbo) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until channel_enable = TRUE		
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0
Wait until channel_enable = FALSE		

Copyright © 2026 Texas Instruments Incorporated

Table 12-41. Receive-Only With Interrupt (Controller Turbo) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: RXi_FULL		
IF: read_count = N - 2		
last_transfer = TRUE		
Wait until channel_enable = FALSE		
ENDIF		
IF: read_count < N		
Read the receiver register	MCSPI_RXi	0x-
Increment read_count +1		
ENDIF		
ENDIF		

12.1.3.5.2.1.6.2 Based on DMA Read Requests

When the DMA handler has completed its ‘N-2’ CBASS0 accesses read_count is assigned with ‘N-2‘.

Table 12-42. Receive-Only With DMA (Controller Turbo) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until channel_enable = TRUE		
Wait until read_count = TRUE		
Disable DMA read request	MCSPI_CH(i)CONF[15] DMAR	0
Wait until last_transfer = TRUE		
Wait for end of transfer	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0
Wait until channel_enable = FALSE		

Table 12-43. Receive-Only With DMA (Controller Turbo) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: RXi_FULL		
IF: read_count = N - 2		
last_transfer = TRUE		
Wait until channel_enable = FALSE		
ENDIF		
IF: read_count < N		
Read the receiver register	MCSPI_RXi	0x-
Increment read_count +1		
ENDIF		
ENDIF		

12.1.3.5.2.1.7 Peripheral Receive-Only

If the requests are configured in DMA, read_count is assigned with ‘N’ when the DMA handler has completed its ‘N‘ CBASS0 accesses.

Copyright © 2026 Texas Instruments Incorporated

Table 12-44. Receive-Only (Peripheral) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
Wait until read_count = N		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

Table 12-45. Receive-Only (Peripheral) (Interrupt Routine)

Step	Register/Bit Field/Programming Model	Value
Read MCSPI_IRQSTATUS	MCSPI_IRQSTATUS	0x-
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS[channel i bits]	0b1111
IF: RXi_FULL		
Read the receiver register	MCSPI_RXi	0x-
Increment read_count +1		
ENDIF		

12.1.3.5.2.1.8 Transfer Procedures With FIFO

These flows describe the transfer with FIFO.

The MCSPI module allows the transfer of one or several words, according to different modes:

- CONTROLLER Normal, CONTROLLER Turbo, PERIPHERAL
- TRANSMIT–RECEIVE, TRANSMIT-ONLY, RECEIVE-ONLY
- Write and Read requests: IRQ, DMA

For all these flows, the host process contains the main process and the interrupt routine. This routine is called on the IRQ signals or by an internal call if the module is used in polling mode.

For more information, see Section 12.1.3.4.6, *MCSPI FIFO Buffer Management*.

Table 12-46. FIFO Mode Common Sequence (Controller) (Main Process)

Step	Register/Bit Field/Programming Model	Value
Write MCSPI_IRQSTATUS to reset channel status bits	MCSPI_IRQSTATUS	1
Write MCSPI_IRQENABLE to enable interrupts	MCSPI_IRQENABLE	1
Write MCSPI_CH(i)CONF to configure the channel	MCSPI_CH(i)CONF	0x-
Write MCSPI_XFERLEVEL	MCSPI_XFERLEVEL	0x-
Start the channel	MCSPI_CH(i)CTRL[0] EN	1
IF: Receive only		
Wait for the write request (TX empty or DMA write)		
Write for the transmitter register with data	MCSPI_TXi	0x-
ENDIF		
Wait for the host event for end of transfer		
Stop the channel	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.2.1.8.1 Common Transfer Sequence in FIFO Mode

This flow describes the host sequence for a transfer of any type defined in Section 12.1.3.5.2.1.8, *Transfer Procedures With FIFO*.

In multi-channel, only one channel can use the FIFO.

Before enabling the FIFO for a channel (MCSPI_CH(i)CONF[28] FFER and MCSPI_CH(i)CONF[27] FFEW bits), the host must check that the FIFO is not enabled for another channel, even if these channels are not used.

In transmit-and-receive mode, the FIFO can be enabled for write or read request only, without FIFO for the other request.

Copyright © 2026 Texas Instruments Incorporated

In Peripheral mode, the channel 0 only can be activated. The correct SPIEN line is chosen in MCSPI_CH0CONF[22-21] SPIENSLV bits.

The MCSPI module can start the transfer only when the first write request has been released by writing the MCSPI_TXi register, even in receive-only mode (only one write request occurs in this case).

12.1.3.5.2.1.8.2 End of Transfer Sequences in FIFO Mode

Table 12-47 summarizes the type of end of transfer per transfer mode and gives a reference to the appropriate section for details.

Table 12-47. End of Transfer Sequences in FIFO Mode

Word count	TRANSMIT AND RECEIVE	TRANSMIT-ONLY	RECEIVE-ONLY
Yes	See Figure 12-49	See Figure 12-51	See Figure 12-52
No	See Figure 12-50	See Figure 12-51	See Figure 12-53

The end of transfer sequences are described from the start of the channel.

In these sequences, some soft variables are used:

- write_count = N
- read_count = N
- last_request = FALSE

They are initialized before starting the channel.

12.1.3.5.2.1.8.3 Transmit-and-Receive With Word Count

Figure 12-49 shows the flow of a transfer in transmit-and-receive mode, with word count.

Copyright © 2026 Texas Instruments Incorporated

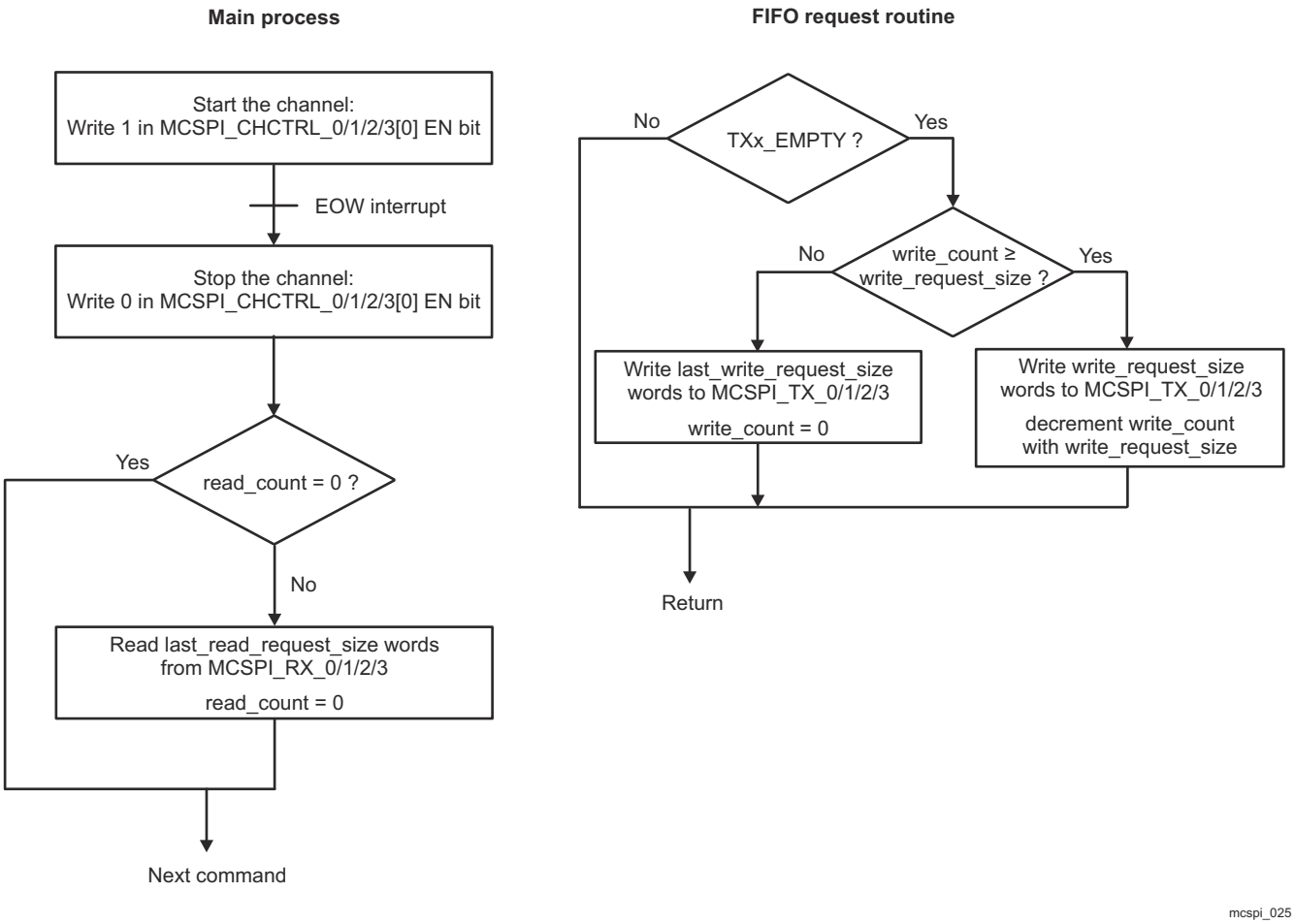


Figure 12-49. FIFO Mode Transmit-and-Receive With Word Count (Controller)

12.1.3.5.2.1.8.4 Transmit-and-Receive Without Word Count

Figure 12-50 shows the flow of a transfer in transmit-and-receive mode, without word count.

Copyright © 2026 Texas Instruments Incorporated

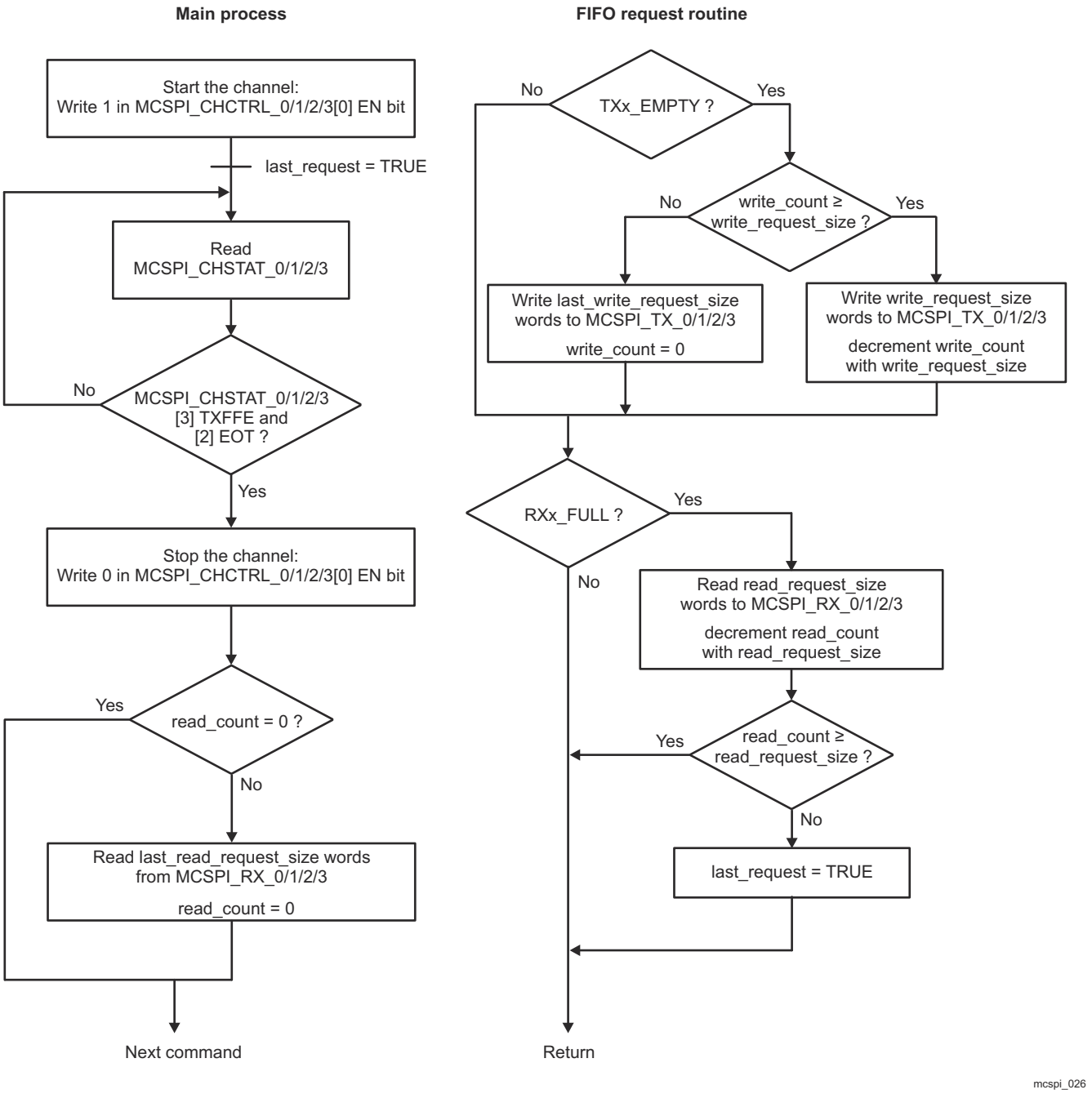


Figure 12-50. FIFO Mode Transmit-and-Receive Without Word Count (Controller)

12.1.3.5.2.1.8.5 Transmit-Only

Figure 12-51 shows the flow of a transfer in transmit-only mode, with or without word count. The difference between word count enabled or not is just on the condition after starting the channel:

- word count enable: wait for EOW interrupt
- word count disable: wait for write_count = 0

Copyright © 2026 Texas Instruments Incorporated

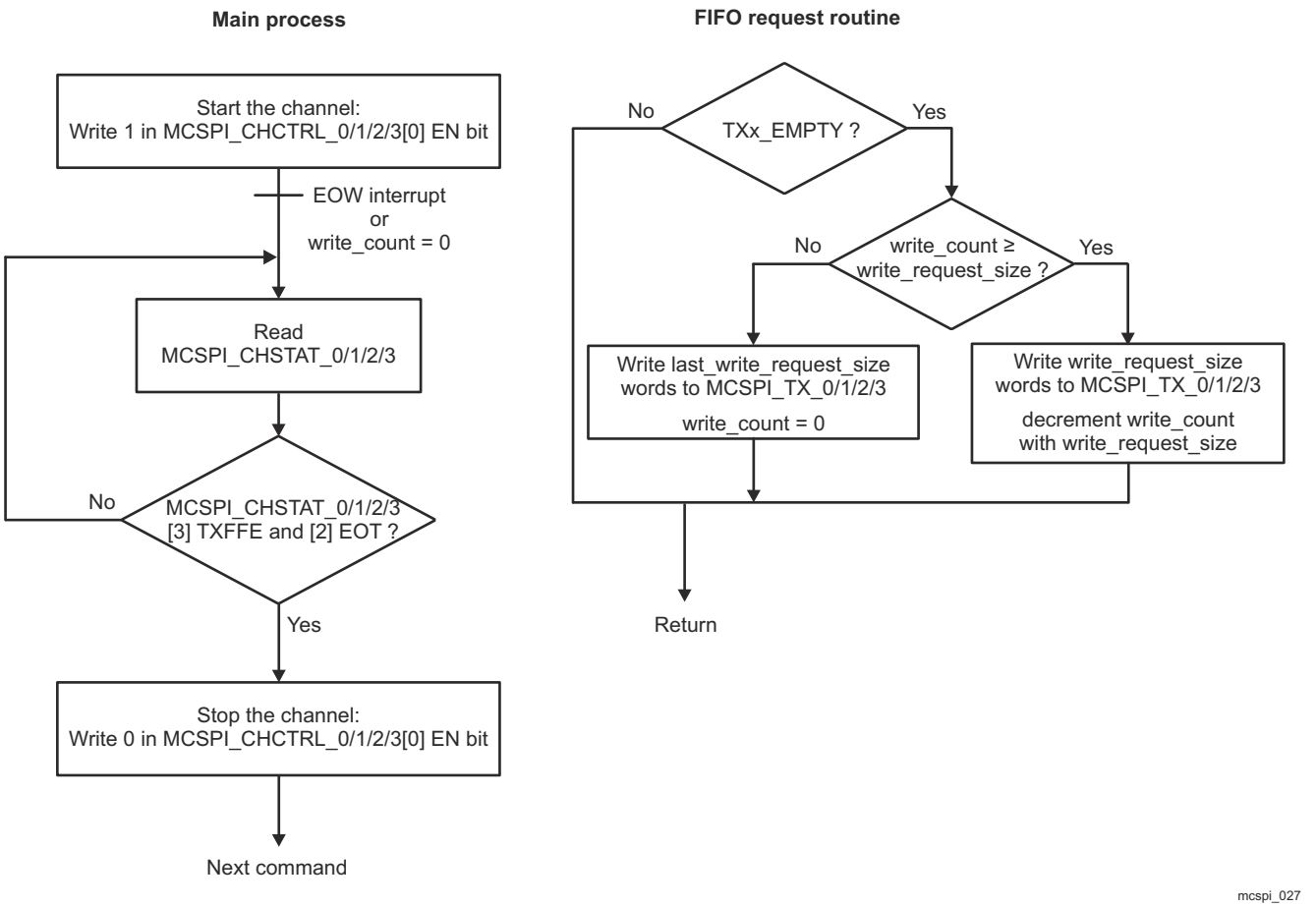


Figure 12-51. FIFO Mode Transmit-Only (Controller)

12.1.3.5.2.1.8.6 Receive-Only With Word Count

Figure 12-52 shows the flow of a transfer in receive-only mode, with word count.

Copyright © 2026 Texas Instruments Incorporated

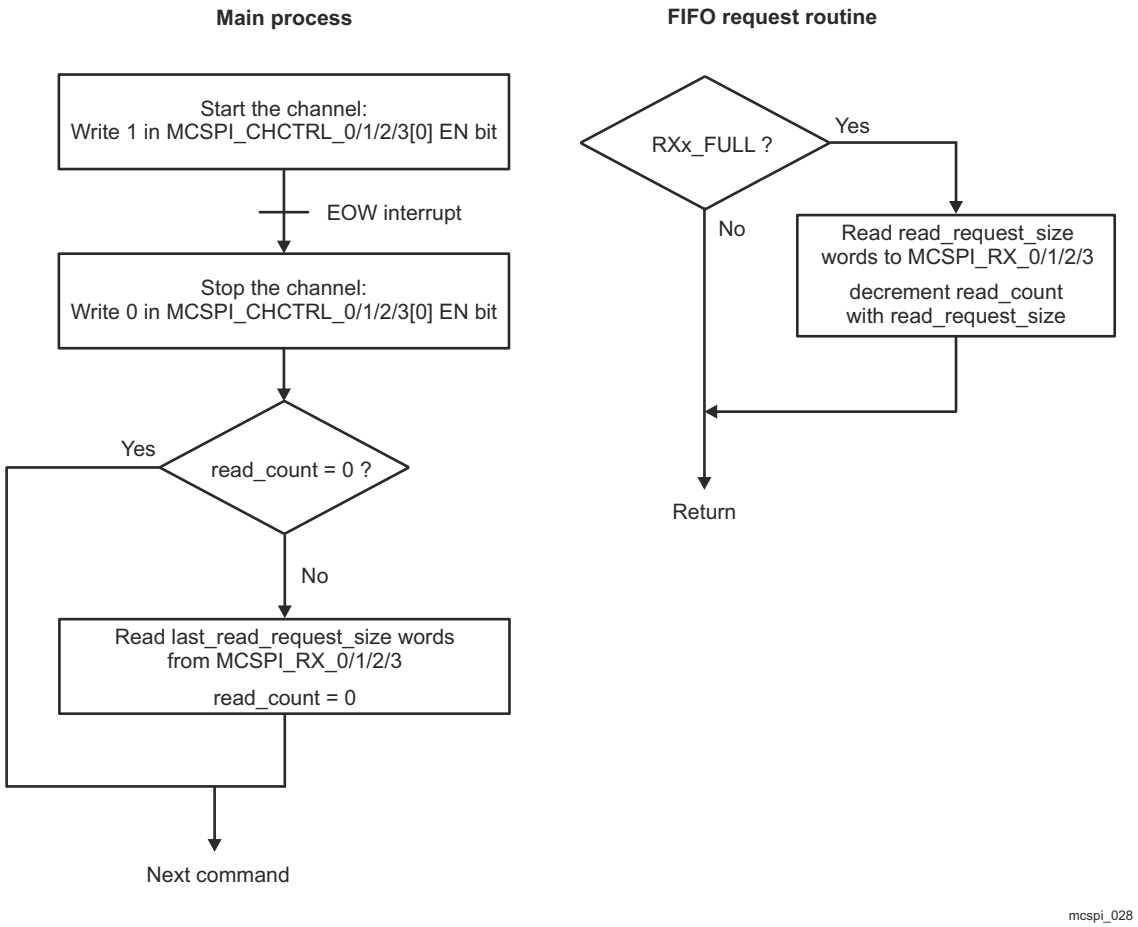


Figure 12-52. FIFO Mode Receive-Only With Word Count (Controller)

12.1.3.5.2.1.8.7 Receive-Only Without Word Count

Figure 12-53 shows the flow of a transfer in receive-only mode, with word count.

Copyright © 2026 Texas Instruments Incorporated

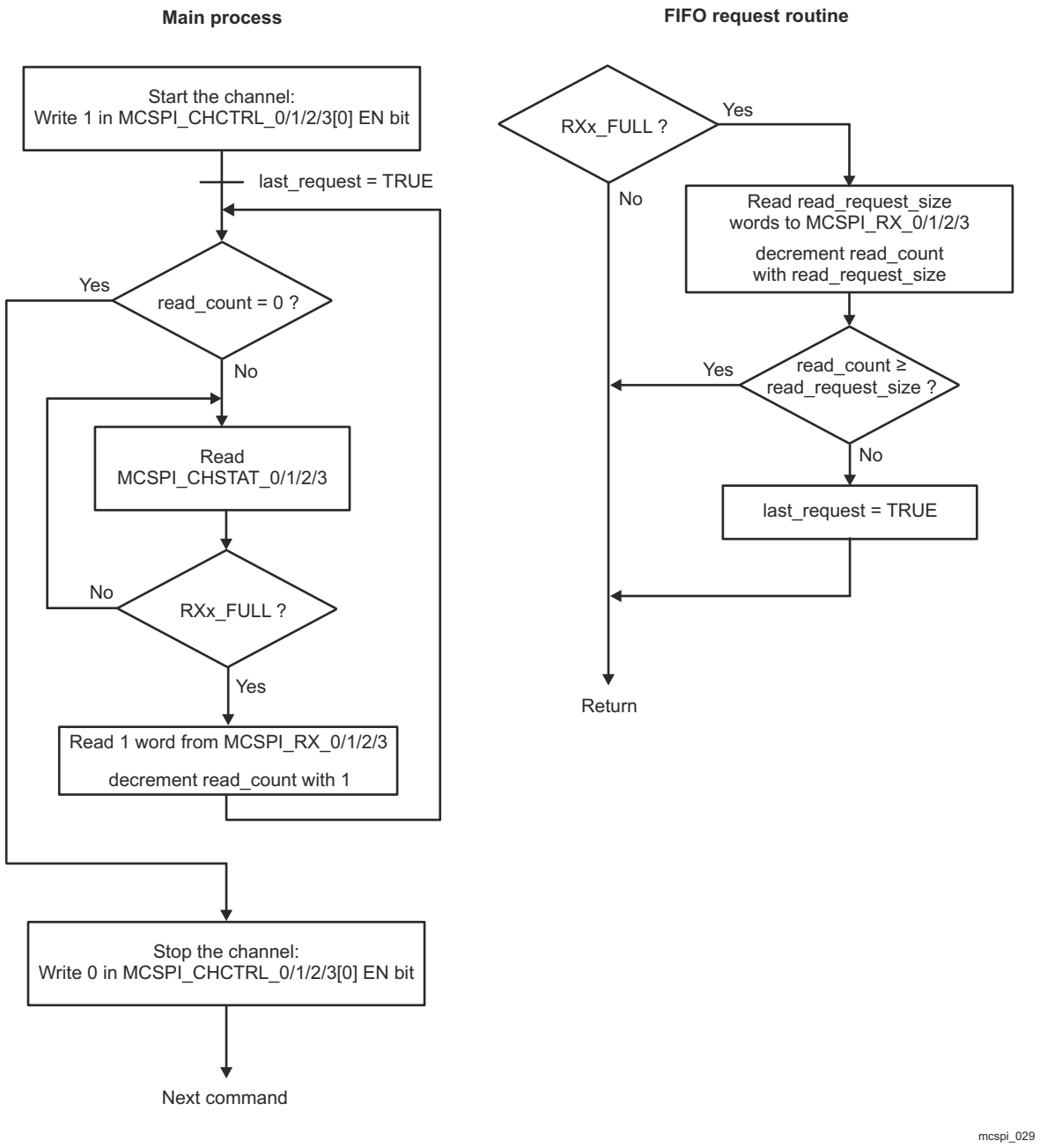


Figure 12-53. FIFO Mode Receive-Only Without Word Count (Controller)

12.1.3.5.2.1.9 Common Transfer Procedures Without FIFO – Polling Method

12.1.3.5.2.1.9.1 Receive-Only Procedure – Polling Method

Table 12-48 lists the receive-only procedure using the polling method.

Table 12-48. Receive-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-25.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Wait for end-of-transfer.	MCSPI_CH(i)STAT[2] EOT	=1
Read the receiver register.	MCSPI_RXi	0x-

Copyright © 2026 Texas Instruments Incorporated

Table 12-48. Receive-Only Procedure – Polling Method (continued)

Step	Register/Bit Field/Programming Model	Value
Stop the channel if no more data is expected.	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.2.1.9.2 Receive-Only Procedure – Interrupt Method

Table 12-49 lists the receive-only procedure using the interrupt method.

Table 12-49. Receive-Only Procedure – Interrupt Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-25.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Enable the interrupt for the receiver register.	MCSPI_IRQENABLE[2] RX_FULL_ENABLE	1
Wait for interrupt.		
Read the status register.	MCSPI_IRQSTATUS[2] RX_FULL	1
Disable the interrupt if no more data is expected.	MCSPI_IRQENABLE[2] RX_FULL_ENABLE	0
Stop the channel if no more data is expected.	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register.	MCSPI_RXi	0x-

12.1.3.5.2.1.9.3 Transmit-Only Procedure – Polling Method

Table 12-50 lists the transmit-only procedure using the polling method.

Table 12-50. Transmit-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-26.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Write the transmitter register with data.	MCSPI_TXi	0x-
Wait until end of transfer?	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel.	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.2.1.9.4 Transmit-and-Receive Procedure – Polling Method

Table 12-51 lists the transmit-and-receive procedure using the polling method.

Table 12-51. Transmit-and-Receive Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-27.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Write the transmitter register with data.	MCSPI_TXi	0x-
Wait until transmit/receive word?	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel.	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register.	MCSPI_RXi	0x-

12.1.3.5.3 Common Transfer Procedures Without FIFO – Polling Method

12.1.3.5.3.1 Receive-Only Procedure – Polling Method

Table 12-52 lists the receive-only procedure using the polling method.

Table 12-52. Receive-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-25.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Wait for end-of-transfer.	MCSPI_CH(i)STAT[2] EOT	=1

Copyright © 2026 Texas Instruments Incorporated

Table 12-52. Receive-Only Procedure – Polling Method (continued)

Step	Register/Bit Field/Programming Model	Value
Read the receiver register.	MCSPI_RXi	0x-
Stop the channel if no more data is expected.	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.3.2 Receive-Only Procedure – Interrupt Method

Table 12-53 lists the receive-only procedure using the interrupt method.

Table 12-53. Receive-Only Procedure – Interrupt Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-25.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Enable the interrupt for the receiver register.	MCSPI_IRQENABLE[2] RX_FULL_ENABLE	1
Wait for interrupt.		
Read the status register.	MCSPI_IRQSTATUS[2] RX_FULL	1
Disable the interrupt if no more data is expected.	MCSPI_IRQENABLE[2] RX_FULL_ENABLE	0
Stop the channel if no more data is expected.	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register.	MCSPI_RXi	0x-

12.1.3.5.3.3 Transmit-Only Procedure – Polling Method

Table 12-54 lists the transmit-only procedure using the polling method.

Table 12-54. Transmit-Only Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-26.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Write the transmitter register with data.	MCSPI_TXi	0x-
Wait until end of transfer?	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel.	MCSPI_CH(i)CTRL[0] EN	0

12.1.3.5.3.4 Transmit-and-Receive Procedure – Polling Method

Table 12-55 lists the transmit-and-receive procedure using the polling method.

Table 12-55. Transmit-and-Receive Procedure – Polling Method

Step	Register/Bit Field/Programming Model	Value
Configure the channel according to the mode.	See Table 12-27.	
Start the channel.	MCSPI_CH(i)CTRL[0] EN	1
Write the transmitter register with data.	MCSPI_TXi	0x-
Wait until transmit/receive word?	MCSPI_CH(i)STAT[2] EOT	=1
Stop the channel.	MCSPI_CH(i)CTRL[0] EN	0
Read the receiver register.	MCSPI_RXi	0x-

Copyright © 2026 Texas Instruments Incorporated

12.1.4 Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the function, operation, and configuration of the Universal Asynchronous Receiver/Transmitter (UART)/RS-485/Infrared Data Association (IrDA)/Consumer Infrared (CIR) module in the device.

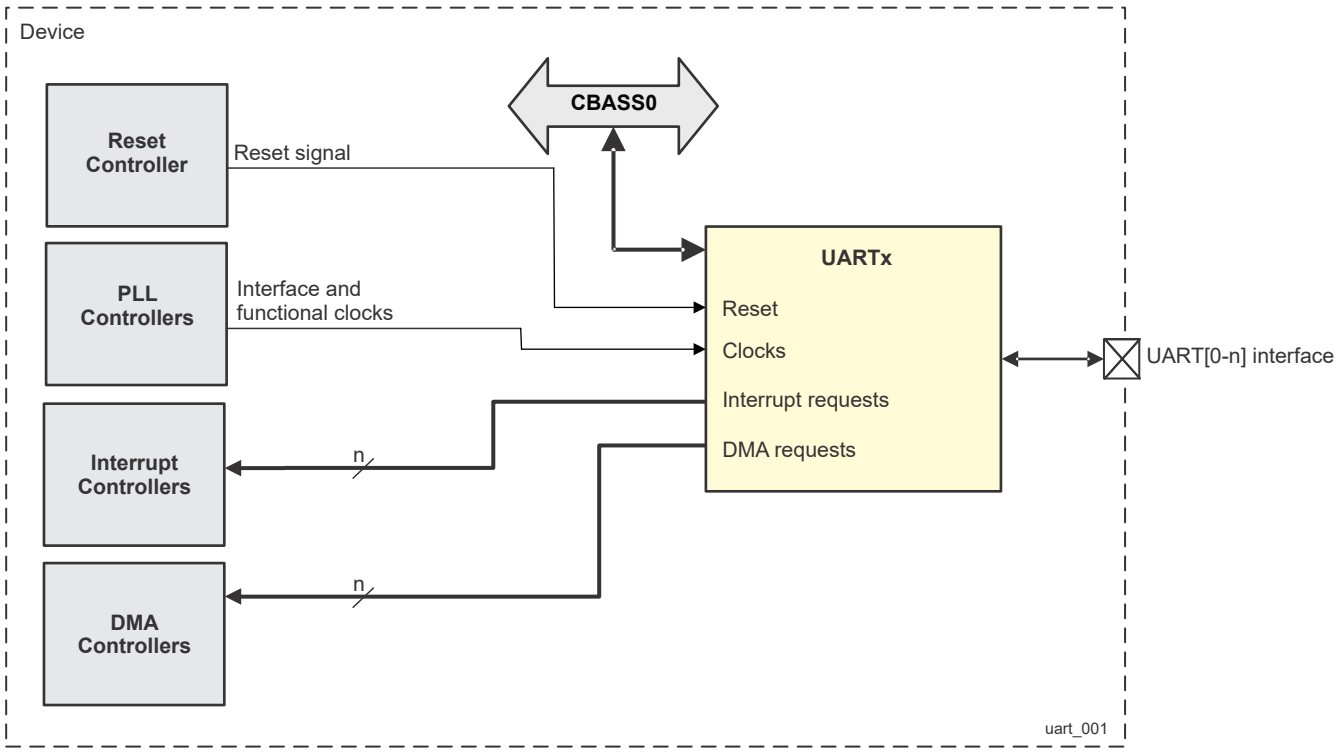
Note

UART and USART acronyms are used interchangeably in this section.

12.1.4.1 UART Overview

The UART is a peripheral that utilizes the DMA for data transfer or interrupt polling via host CPU. All UART modules support IrDA and CIR modes when 48 MHz function clock is used. Each UART can be used for configuration and data exchange with a number of external peripheral devices or interprocessor communication between devices.

Figure 12-54 shows the UART modules overview.



A. x represents a valid instance of UART in a domain. See the device datasheet for specifics.
B. n represents the maximum number of available UART signals -1.

Figure 12-54. UART Modules Overview

12.1.4.1.1 UART Features

The UART includes the following features:

- 16C750-compatible
- RS-485 external transceiver auto flow control support
- 64-byte FIFO buffer for receiver and 64-byte FIFO buffer for transmitter
- Programmable interrupt trigger levels for FIFOs
- Programmable sleep mode
- The 48 MHz functional clock is default option and allows baud rates up to 3.6 Mbps
- Auto-baud between 1200 bits/s and 115.2 Kbits/s (only when 48 MHz function clock is used)

Copyright © 2026 Texas Instruments Incorporated

- Optional multi-drop transmission
- Configurable time-guard feature
- Configurable data format:
  - Parity bit: Even, odd, none
  - Stop-bit: 1, 1.5, 2 bit(s)
- Flow control: Hardware (RTS/CTS) or software (XON/XOFF)
- False start bit detection
- Line break generation and detection
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities
- Modem control functions (CTS, RTS)
- UART0 module in MAIN domain has extended modem control signals (DCD, RI, DTR, DSR)

12.1.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.1.4.1.3 IrDA Features

The IrDA includes the following features:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR), and fast infrared (FIR) communications:
  - Slow infrared (SIR 115.2 KBAUD), medium infrared (MIR 0.576 MBAUD) and fast infrared (FIR 4.0 MBAUD) operations (very fast infrared (VFIR) is not supported)
  - Frame formatting: addition of variable beginning-of-frame (xBOF) characters and end-of-frame (EOF) characters
  - Uplink/downlink cyclic redundancy check (CRC) generation/detection
  - Asynchronous transparency (automatic insertion of break character)
  - Eight-entry status FIFO (with selectable trigger levels) to monitor frame length and frame errors
  - Framing error, CRC error, illegal symbol (FIR), and abort pattern (SIR, MIR) detection
- IrDA mode when 48 MHz function clock is used

12.1.4.1.4 CIR Features

The CIR mode uses a variable pulse-width modulation (PWM) technique (based on multiples of a programmable *t* period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on a user-definable frame structure and packet content.

The CIR includes the following features to provide CIR support for remote-control applications:

- Transmit and receive mode
- Free data format (supports any remote-control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3, or 1/4 carrier duty cycle
- CIR mode when 48 MHz function clock is used

Copyright © 2026 Texas Instruments Incorporated

12.1.4.2 UART Environment

The , modules are hereinafter referred to as UART module.

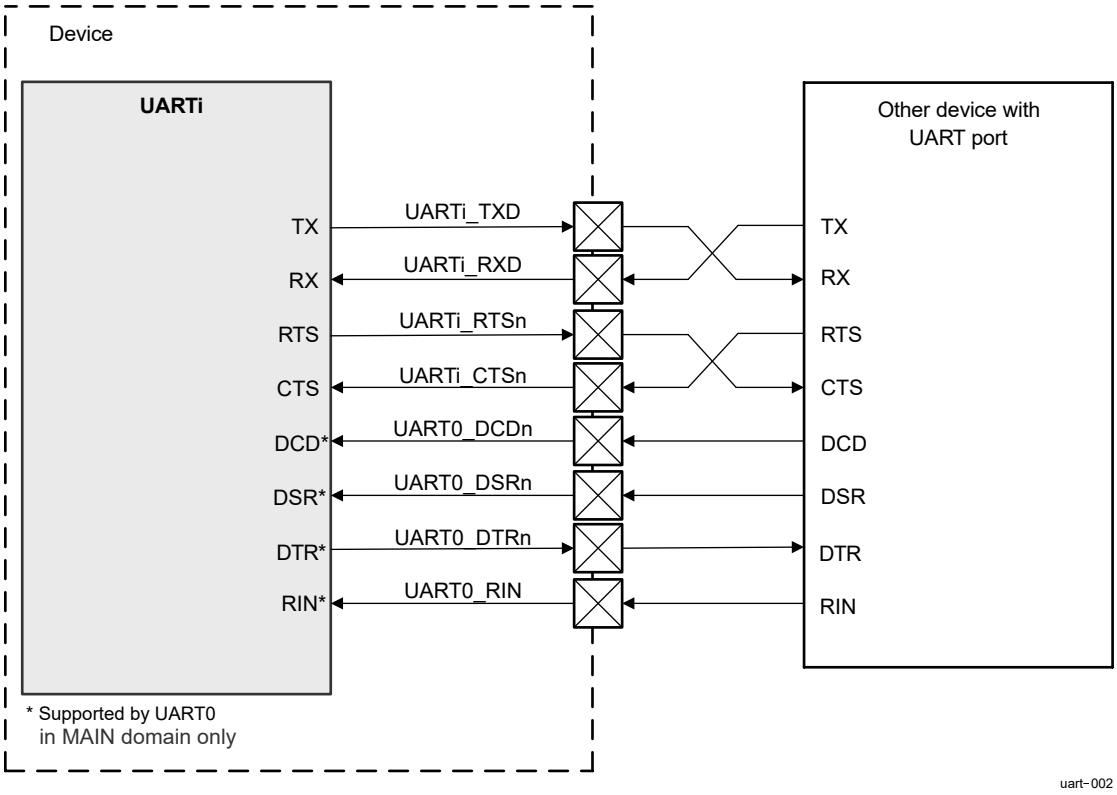
This section describes the UART/RS-485/IrDA/CIR external connections (environment).

- The UART interface is described in Section 12.1.4.2.1, *UART Functional Interfaces*.
- The RS-485 interface is described in Section 12.1.4.2.2, *RS-485 Functional Interfaces*.
- The IrDA interface is described in Section 12.1.4.2.3, *IrDA Functional Interfaces*.
- The CIR interface is described in Section 12.1.4.2.4, *CIR Functional Interfaces*.

12.1.4.2.1 UART Functional Interfaces

12.1.4.2.1.1 System Using UART Communication With Hardware Handshake

Each UART instance can be easily connected to the UART port of an external IC (see Figure 12-55).



A. i represents a valid instance of UART in a domain. See the device datasheet for valid domains and instances.

Figure 12-55. UART Mode Interface Signals

Copyright © 2026 Texas Instruments Incorporated

12.1.4.2.1.2 UART Interface Description

Table 12-56 lists the UART interface input/output (I/O) signals.

Table 12-56. UART I/O Signals

Module Pin Name	Device Level Signal Name	I/O[1]	Description	Module Pin Reset Value[2]
WKUP_UARTi[10]				
RX	WKUP_UARTi[10]_RXD	I	Serial data input	HiZ
TX	WKUP_UARTi[10]_TXD	O	Serial data output[3]	1
CTS	WKUP_UARTi[10]_CTSn	I	Clear to send[4]	HiZ
RTS	WKUP_UARTi[10]_RTSn	O	Request to send[5]	1
MCU_UARTi[10]				
RX	MCU_UARTi[10]_RXD	I	Serial data input	HiZ
TX	MCU_UARTi[10]_TXD	O	Serial data output[3]	1
CTS	MCU_UARTi[10]_CTSn	I	Clear to send[4]	HiZ
RTS	MCU_UARTi[10]_RTSn	O	Request to send[5]	1
UARTi[10] Modem Signals				
RX	UARTi[10]_RX	I	Serial data input	HiZ
TX	UARTi[10]_TX	O	Serial data output[3]	1
CTS	UARTi[10]_CTSn	I	Clear to send[4]	HiZ
RTS	UARTi[10]_RTSn	O	Request to send[5]	1
DCD	UARTi[10]_DCDn	I	Data Carrier Detect[6]	HiZ
DSR	UARTi[10]_DSRn	I	Data Set Ready[7]	HiZ
DTR	UARTi[10]_DTRn	O	Data Terminal Ready[8]	1
RIN	UARTi[10]_RIN	I	Ring Indicator[9]	HiZ

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) Because this pin is active high in IrDA mode and the output is muxed, this pin is set to low on reset (when the UART_MDR1[2-0] bit field is set to 0x7) and takes the defined inactive level of that signal corresponding to when and how the UART_MDR1 register is programmed; that is, the output is 1 (inactive for UART modem modes) and 0 (inactive for IrDA modes).
(4) Active-low modem status signal. Reading the UART_MSR[4] NCTS_STS bit checks the condition of CTS. Reading the UART_MSR[0] CTS_STS bit checks a change of state of CTS since the last read of the modem status register. The auto-CTS mode uses CTS to control the transmitter.
(5) When active (low), the module is ready to receive data. Setting the UART_MCR[1] RTS bit activates RTS signal, which becomes inactive as the result of a module reset, loopback mode, or clearing the UART_MCR[1] RTS bit. In auto-RTS mode, RTS signal becomes inactive as a result of the receiver threshold logic.
(6) Active-low modem status signal. The condition of DCD can be checked by reading the UART_MSR[7] NCD_STS bit. Any change in its state can be detected by reading the UART_MSR[3] DCD_STS bit.
(7) Active-low modem status signal. Reading the UART_MSR[5] NDSR_STS bit checks the condition of DSR. Reading the UART_MSR[1] DSR_STS bit checks a change of state of DSR since the last read of the UART_MSR register.
(8) When active (low), this signal informs the modem that the module is ready to communicate. It is activated by setting the UART_MCR[0] DTR bit.
(9) Active-low modem status signal. The condition of RIN can be checked by reading the UART_MSR[6] NRI_STS bit. Any change in its state can be detected by reading the UART_MSR[2] RI_STS bit.
(10) i represents a UART instance. See the device datasheet for available domains and UART instances.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.4.2.1.3 UART Protocol and Data Format

The UART device operates in three modes:

- UART 16× (<= 230.4 kbps)

Copyright © 2026 Texas Instruments Incorporated

- UART 16× with autobauding (>= 1200 bps and <= 115.2 kbps)
- UART 13× (>= 460.8 kbps)

CAUTION

To be used as a UART, the operating mode must be programmed appropriately in the UART_MDR1[2-0] MODE_SELECT bit field to select UART, IrDA, or CIR mode, and the UART_MDR3[4] DIR_EN bit field to select RS-485 mode.

The UART uses a wired interface for serial communication with a remote device.

The UART is functionally compatible with the TL16C750 UART and earlier designs such as the TL16C550.

Figure 12-56 shows the UART frame data format.

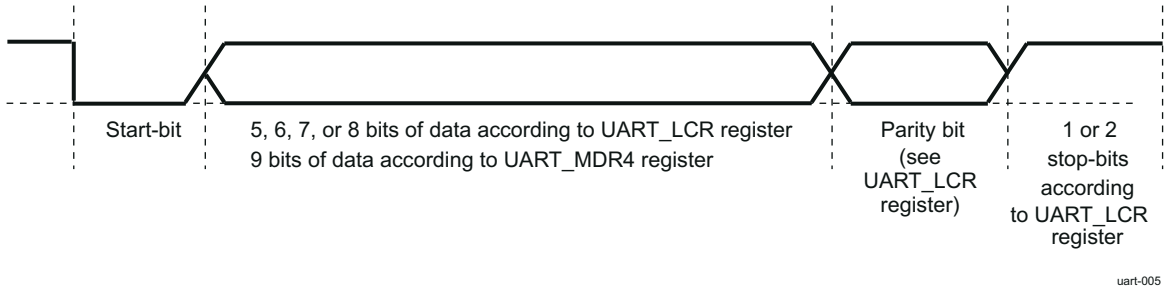
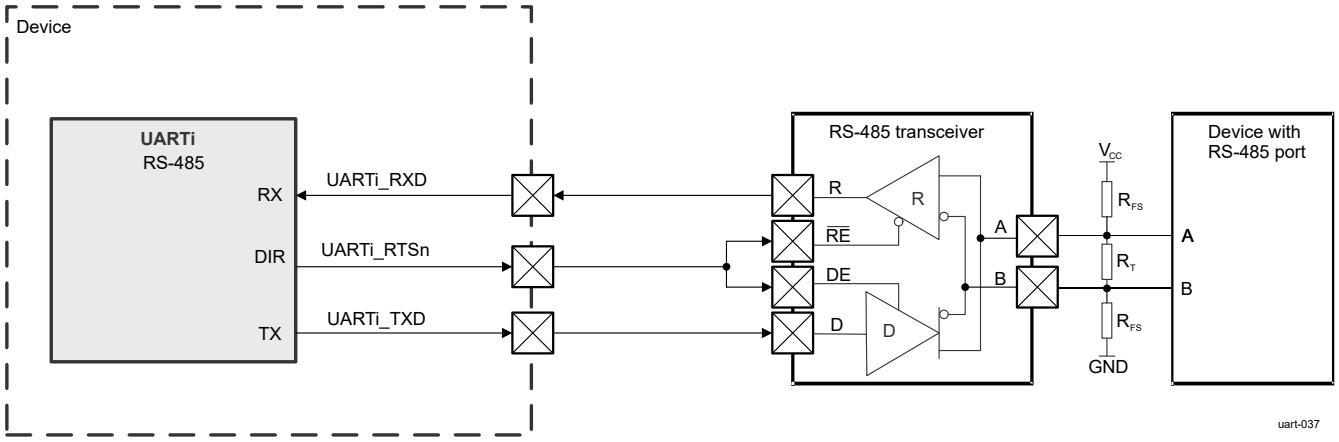


Figure 12-56. UART Frame Data Format

12.1.4.2.2 RS-485 Functional Interfaces

12.1.4.2.2.1 System Using RS-485 Communication

The RS-485 network physical layer consists of two-wire differential bus, usually twisted pair. External RS-485 transceiver IC is needed to access a RS-485 bus by the RS-485 mode. Figure 12-57 shows an example connection of an UART instance in RS-485 mode.



A. i represents a valid instance of UART in a domain. See the device data sheet for valid domains and instances.

Figure 12-57. RS-485 Mode Interface Signals

12.1.4.2.2.2 RS-485 Interface Description

Table 12-57 lists the RS-485 interface input/output (I/O) signals.

Table 12-57. UART I/O Signals (RS-485 Mode)

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
			WKUP_UARTi[3]	

Copyright © 2026 Texas Instruments Incorporated

Table 12-57. UART I/O Signals (RS-485 Mode) (continued)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
RX	WKUP_UARTi(3)_RXD	I	Serial data input	HiZ
TX	WKUP_UARTi(3)_TXD	O	Serial data output	1
DIR	WKUP_UARTi(3)_RTSn	O	RS-485 Direction	1
			MCU_UARTi(3)	
RX	MCU_UARTi(3)_RXD	I	Serial data input	HiZ
TX	MCU_UARTi(3)_TXD	O	Serial data output	1
DIR	MCU_UARTi(3)_RTSn	O	RS-485 Direction	1
			UARTi(3)	
RX	UARTi(3)_RXD	I	Serial data input	HiZ
TX	UARTi(3)_TXD	O	Serial data output	1
DIR	UARTi(3)_RTSn	O	RS-485 Direction	1

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) i represents a valid UART instance. See the device datasheet for available domains and UART instances.

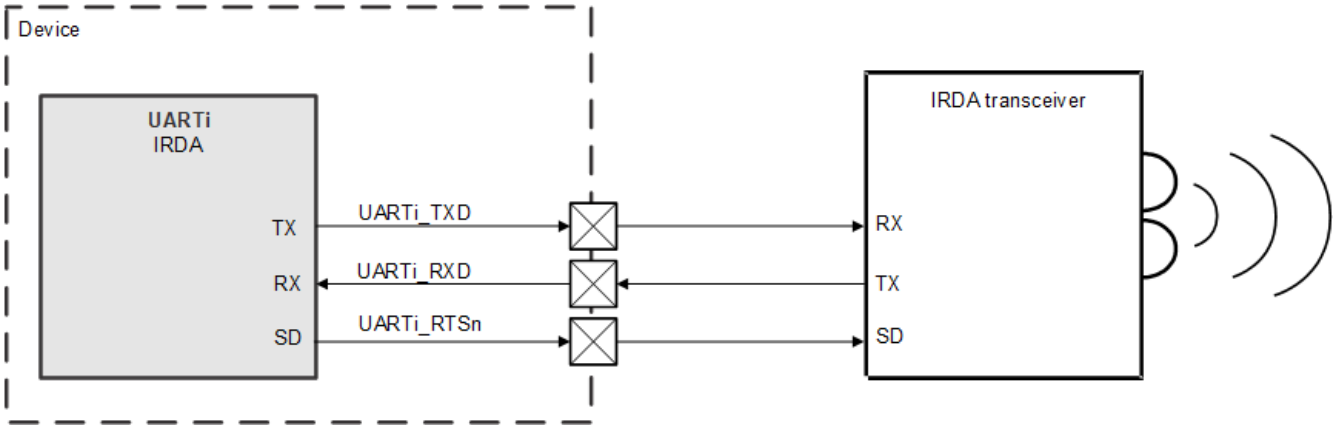
Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.4.2.3 IrDA Functional Interfaces

12.1.4.2.3.1 System Using IrDA Communication Protocol

Figure 12-58 shows an example connection of an UART instance to an external infrared transceiver in the IrDA modes (FIR, SIR, and MIR).



A. i represents a valid instance of UART in a domain. See the device data sheet for valid domains and instances.

Figure 12-58. IrDA Mode Interface Signals

Copyright © 2026 Texas Instruments Incorporated

12.1.4.2.3.2 IrDA Interface Description

Table 12-58 lists the IrDA interface I/O signals.

Table 12-58. UART I/O Signals (IrDA Mode)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
RX		I	Serial data input	HiZ
TX		O	Serial data output in IrDA modes (SIR, MIR, and FIR).(3)	0
SD		O	SD mode is used to configure the transceivers.(4)	1
RX		I	Serial data input	HiZ
TX		O	Serial data output in IrDA modes (SIR, MIR, and FIR).(3)	0
SD		O	SD mode is used to configure the transceivers.(4)	1

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) In other modes, this pin is set to the reset value (inactive state).
(4) The SD pinout (see UART_ACREG[6] SD_MOD bit).

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.4.2.3.3 IrDA Protocol and Data Format

12.1.4.2.3.3.1 SIR Mode

In SIR mode, data is transferred between the Host CPU and peripheral devices at speeds of up to 115.2 baud. A SIR transmit frame begins with start flags (a single 0xC0, a multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data and a CRC-16, and ends with a stop flag (0xC1).

The bit format for a single word uses 1 start-bit, 8 data bits, and 1 stop-bit, and is unaffected by the use and settings of the UART_LCR register.

The UART_BLR[6] XBOF_TYPE bit selects whether the 0xC0 or 0xFF start patterns are used when multiple start flags are required.

The SIR transmit state-machine attaches start flags, CRC-16, and stop flags, and checks the outgoing data to establish whether data transparency is required.

The SIR transparency is carried out if the outgoing data between the start and stop flags contains 0xC0, 0xC1, or 0x7D. If one of these start flags is about to be transmitted, the SIR state-machine sends an escape character (0x7D), inverts the fifth bit of the real data to be sent, and then sends this data immediately after the 0x7D character.

The SIR receive state-machine recovers the receive clock, removes the start flags and any transparency from the incoming data, and determines the frame boundary with reception of the stop flag. The SIR state-machine also checks for errors such as a frame abort (0x7D character followed immediately by a 0xC1 stop flag without transparency), a CRC error, or a frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART_LSR_IRDA) to find possible errors of the received frame.

Note

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See the description of the UART_ACREG[5] DIS_IR_RX bit. This applies to all three modes: SIR, MIR, and FIR.

Copyright © 2026 Texas Instruments Incorporated

Infrared output in SIR mode can be 1.6-µs or 3/16 encoding, selected by the UART_ACREG[7] PULSE_TYPE bit. In 1.6-µs encoding, the infrared pulse width is 1.6 µs; and in 3/16th encoding, the infrared pulse width is 3/16th of a bit duration (1/baud rate).

For back-to-back frames, the transmitting device must send at least two start flags at the start of each frame.

Note

Reception supports variable-length stop-bits.

12.1.4.2.3.3.1.1 Frame Format

Figure 12-59 shows the IrDA SIR frame format.

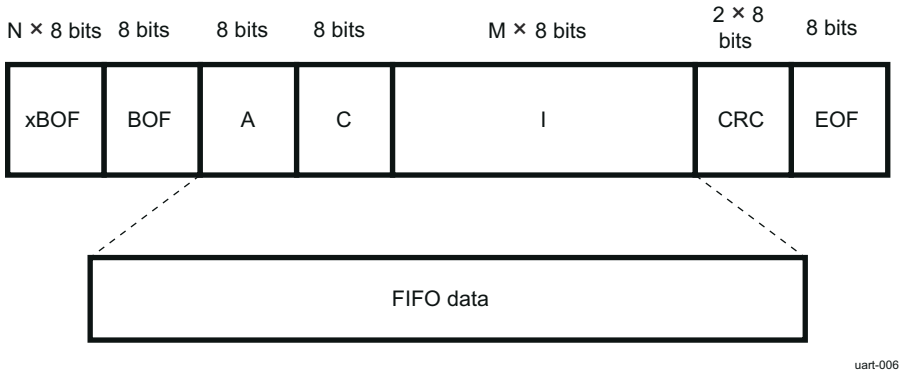


Figure 12-59. IrDA SIR Frame Format

The CRC is applied on the address (A), control (C), and information (I) bytes.

Note

The two words of CRC are written to the FIFO in reception.

12.1.4.2.3.3.1.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), the controller performs certain tasks:

- In transmission:
  - Inserts a control escape (CE) byte preceding the byte
  - Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20)

  The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).
- In reception:

  For the A, C, I, and CRC fields:
  - Compares the byte with the CE byte; if they are not equal, sends the byte to the CRC detector and stores it in the RX FIFO.
  - If the byte is equal to the CE byte, discards the CE byte
  - Complements bit 5 of the byte following the CE
  - Sends the complemented byte to the CRC detector and stores it in the RX FIFO

12.1.4.2.3.3.1.3 Abort Sequence

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

Copyright © 2026 Texas Instruments Incorporated

When a 0x7D character that is followed immediately by a 0xC1 character is received without transparency, the receiver treats the frame as an aborted frame.

12.1.4.2.3.3.1.4 Pulse Shaping

The SIR mode supports the 3/16 and the 1.6-µs pulse duration methods. The UART_ACREG[7] PULSE_TYPE bit selects the pulse-width method in transmit mode.

12.1.4.2.3.3.1.5 Encoder

Serial data from the transmit state-machine are encoded to transmit data to the optoelectronics. While the TX FIFO output is high, the TX line is always low, and the counter used to form a pulse on TX is cleared continuously.

After the TX FIFO output resets to 0, TX rises on the falling edge of the seventh 16XCLK. On the falling edge of the tenth 16XCLK pulse, TX falls, creating a 3-clock-wide pulse. While the TX FIFO output stays low, a pulse is transmitted during the seventh clock to the tenth clock of each 16-clock bit cycle.

Figure 12-60 shows the IrDA SIR encoding mechanism.

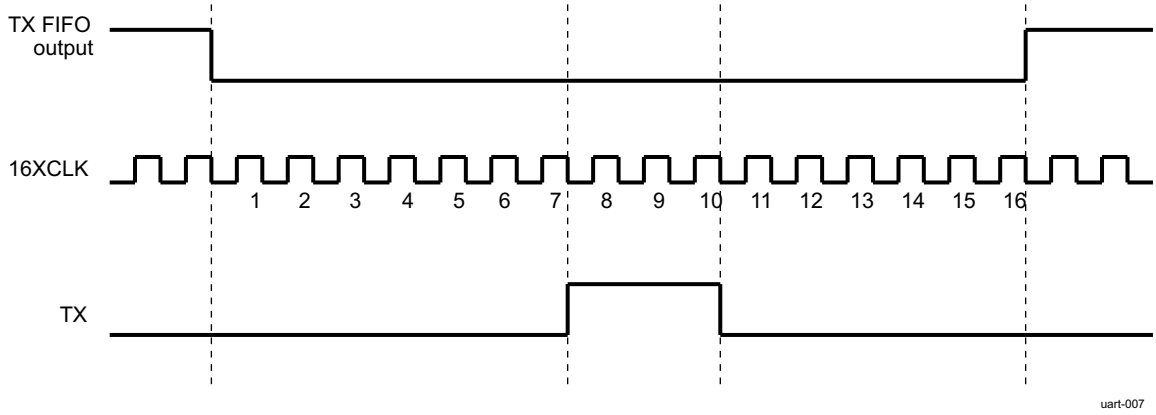


Figure 12-60. IrDA SIR Encoding Mechanism

12.1.4.2.3.3.1.6 Decoder

After reset, the RX FIFO input is high and the 4-bit counter is cleared. When a rising edge is detected on RX, the RX FIFO input falls on the next rising edge of 16XCLK with sufficient setup time. The RX FIFO input stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RX, the RX FIFO input remains high.

Figure 12-61 shows the IrDA SIR decoding mechanism.

Copyright © 2026 Texas Instruments Incorporated

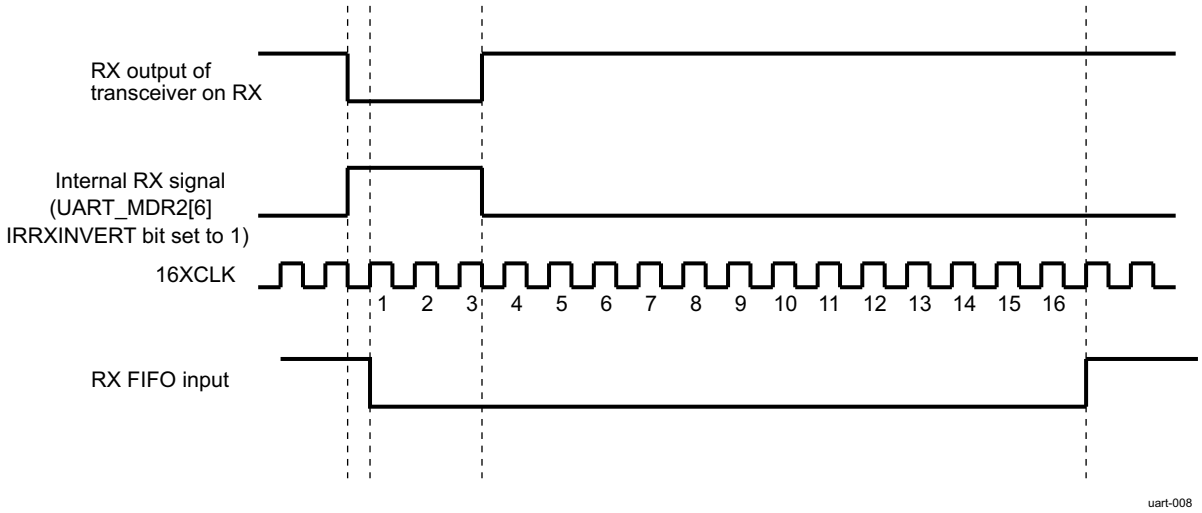


Figure 12-61. IrDA SIR Decoding Mechanism

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. The operation of the RX input can be disabled using the UART_ACREG[5] DIS_IR_RX bit. The UART_MDR2[6] IRRXINVERT bit can invert the signal from the transceiver (RX) pin to the IR RX logic in the UART. This inversion is performed by default.

12.1.4.2.3.3.1.7 IR Address Checking

In all IR modes, when address checking is enabled by setting the UART_EFR[1-0] bit field (see Table 12-59), only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multipoint infrared environment. To program two frame addresses that the UARTi receives in IrDA mode, use the UART_XON1_ADDR1[7-0] and UART_XON2_ADDR2[7-0] bit fields.

Table 12-59. UART_EFR[1-0] IR Address Checking Options

UART_EFR[1]	UART_EFR[0]	IR Address Checking
0	0	All address-checking operations disabled
0	1	Only address 1 checking enabled
1	0	Only address 2 checking enabled
1	1	All address-checking operations enabled

12.1.4.2.3.3.2 SIR Free-Format Mode

To allow complete software flexibility when transmitting and receiving infrared data packets, the SIR free-format (FF) mode is a subfunction of the existing SIR mode. In FF mode, all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values.

The FF mode corresponds to a UART mode with a pulse modulation of 3/16 of baud rate pulse width.

For example, a normal SIR packet has BOF control and CRC error-checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs.

Figure 12-62 shows SIR FF mode.

Copyright © 2026 Texas Instruments Incorporated

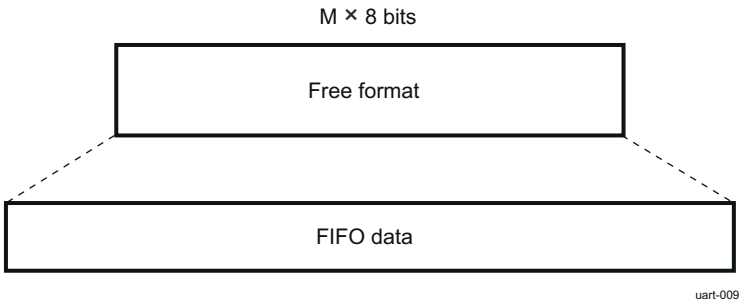


Figure 12-62. SIR FF Mode

In SIR FF mode, the Host CPU software must construct (that is, encode and decode) the entire FIFO data packet.

12.1.4.2.3.3.3 MIR Mode

In MIR mode, data is transferred between the Host CPU and the peripheral devices at 0.576 Mbps or 1.152 Mbps. A MIR transmit frame starts with at least two start flags, followed by a frame data and a CRC-16, and ends with a stop flag (see Figure 12-63).

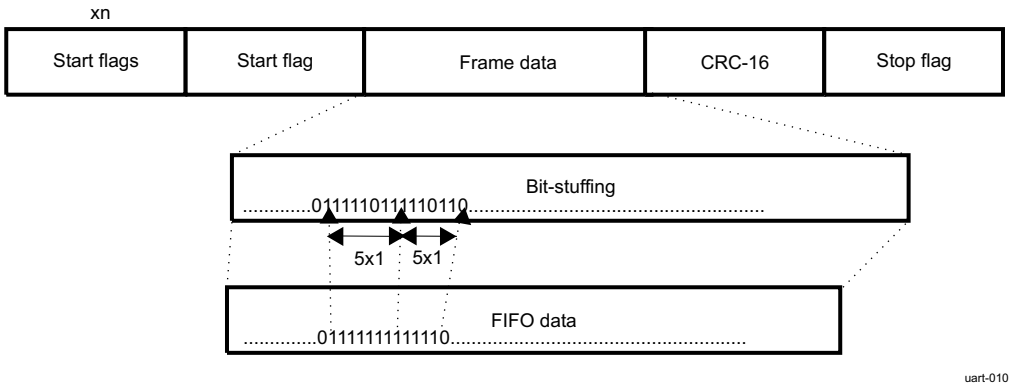


Figure 12-63. MIR Transmit Frame Format

On transmit, the MIR state-machine attaches start flags, a CRC-16, and stop flags, as in SIR mode. All fields are transmitted least-significant bit (LSB) of each byte first.

In MIR mode:

- The state-machine looks for consecutive 1s in the frame data and automatically inserts 0 after five consecutive 1s (this is called bit-stuffing).
- 0x7E is used for start and stop flags (unambiguously, not data, because of bit-stuffing).
- An abort sequence requires a minimum of seven consecutive 1s (unambiguously, not data, because of bit-stuffing).
- Back-to-back frames are allowed with three or more stop flags between them. If two consecutive frames are not back to back, the gap between the last stop flag of the first frame and the start flag of the second frame must be separated by at least seven bit durations.

On receive, the MIR receive state-machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors such as frame abort, CRC error, and frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART_LSR_IRDA) to detect errors of the received frame.

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.2.3.3.3.1 MIR Encoder/Decoder

To meet the MIR baud rate tolerance of 0.1 percent with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag, and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected.

The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4, but it is within the tolerances defined by IrDA specifications.

Figure 12-64 shows the MIR baud rate adjustment mechanism.

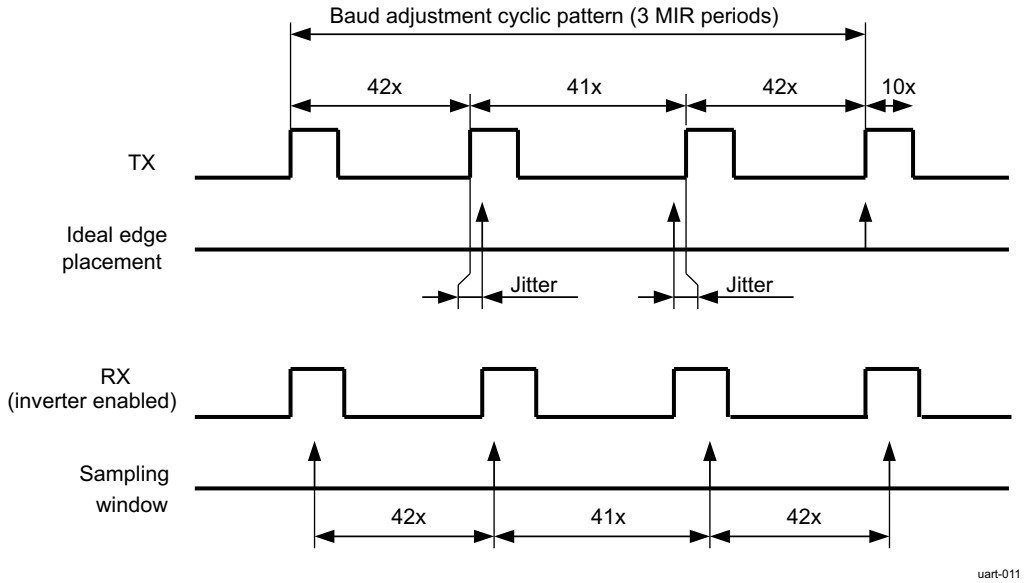


Figure 12-64. MIR Baud Rate Adjustment Mechanism

12.1.4.2.3.3.3.2 SIP Generation

In the MIR and FIR operation modes, the transmitter must send a serial infrared interaction pulse (SIP) at least once every 500 ms. The SIP informs slow devices (operating in SIR mode) that the medium is occupied.

Figure 12-65 shows the SIP.

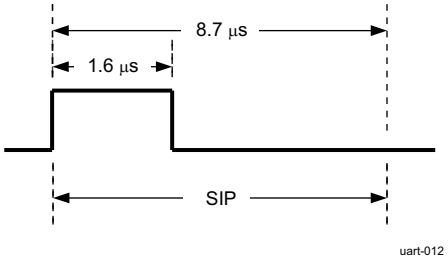


Figure 12-65. SIP

12.1.4.2.3.3.4 FIR Mode

In FIR mode, data is transferred between the Host CPU and the peripheral devices at 4 Mbps. A FIR transmit frame starts with a preamble that is followed by a start flag, frame data, CRC-32, and ends with a stop flag.

Figure 12-66 shows the FIR transmit frame format.

Figure 12-66. FIR Transmit Frame Format

Preamble (16x)	Start flag	Frame data	CRC-32	Stop flag

Copyright © 2026 Texas Instruments Incorporated

On transmit, the FIR transmit state-machine attaches the preamble, start flag, CRC-32, and stop flag. An abort sequence requires at least two transmissions of 0000. Back-to-back frames are allowed, but each frame must be complete.

The state-machine also encodes the transmit data into 4-PPM format (see Table 12-60) and generates the SIP (see Section 12.1.4.2.3.3.3.2, *SIP Generation*).

Table 12-60. 4-PPM Format

Data Bit Pair (Bin)	4-PPM Data Symbol (Bin)
00	1000
01	0100
10	0010
11	0001

The four symbols described in Table 12-60 are the legal, encoded data symbols. All other combinations are illegal for encoding data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag because they are unambiguously not data (see Table 12-61).

Table 12-61. FIR Preamble, Start Flag, and Stop Flag

Frame Part	Transmitted Frame (Bin)
Preamble	1000 0000 1010 1000 (16 repeated transmissions)
Start flag	0000 1100 0000 1100 0110 0000 0110 0000
Stop flag	0000 1100 0000 1100 0000 0110 0000 0110

All fields are transmitted LSBs of each byte first (see Table 12-62).

Table 12-62. FIR Data Byte Transmission Order Example

Data Byte (Hex)	Data Byte Pair (Bin)	4-PPM Data Symbol (Bin)	Transmission Order
0x0B	00	1000	4
	00	1000	3
	10	0010	2
	11	0001	1

On receive, the FIR receive state-machine recovers the receive clock, removes the preamble and the start flag, decodes the 4-PPM incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the Host CPU reads the line status register (UART_LSR_IRDA) to detect errors of the received frame.

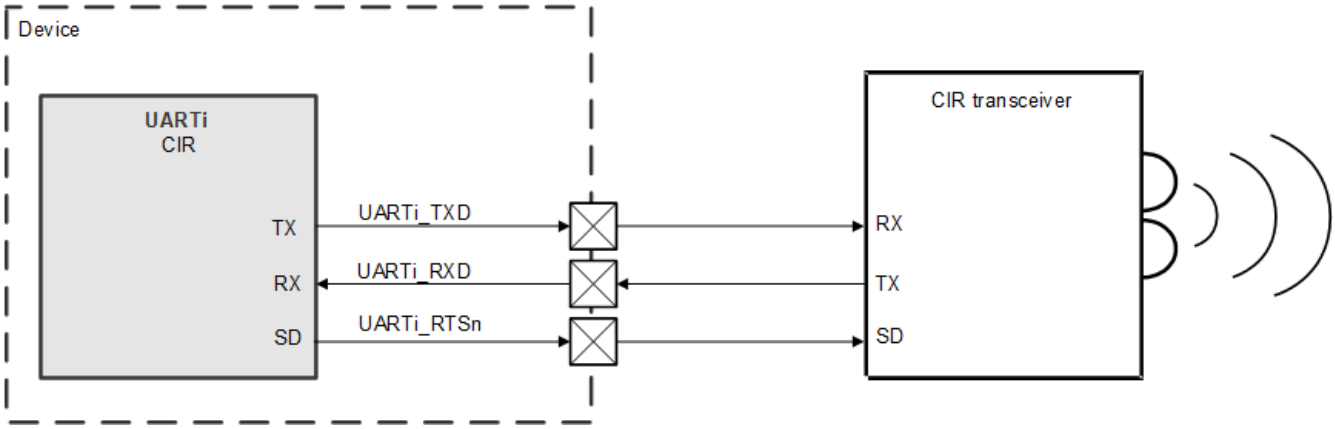
The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

12.1.4.2.4 CIR Functional Interfaces

12.1.4.2.4.1 System Using CIR Communication Protocol With Remote Control

All UART modules can be connected to an external infrared transceiver in CIR mode. Figure 12-67 shows an example connection of a UART instance in CIR mode.

Copyright © 2026 Texas Instruments Incorporated



A. i represents a valid instance of UART in a domain. See the device data sheet for valid domains.

Figure 12-67. CIR Mode Interface Signals

12.1.4.2.4.2 CIR Interface Description

Table 12-63 lists the CIR interface I/O signals.

Table 12-63. UART I/O Signals (CIR Mode)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
RX		I	Serial data input	HiZ
TX		O	Serial data output in CIR mode. (3)	0
SD		O	SD mode is used to configure the transceivers.(4)	1
UART[0-9]				
RX		I	Serial data input	HiZ
TX		O	Serial data output in CIR mode. (3)	0
SD		O	SD mode is used to configure the transceivers.(4)	1

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) In other modes, this pin is set to the reset value (inactive state).
(4) The SD pinout is an inverted value of the UART_ACREG[6] SD_MOD bit.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.1.4.2.4.3 CIR Protocol and Data Format

In CIR mode, the infrared operation functions as a programmable (universal) remote control.

The CIR mode uses a variable PWM technique (based on multiples of a programmable *t* period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on user-defined frame structure and packet content.

12.1.4.2.4.3.1 Carrier Modulation

Each modulated pulse that constitutes a digit is a train of on/off pulses (see Figure 12-68).

Copyright © 2026 Texas Instruments Incorporated

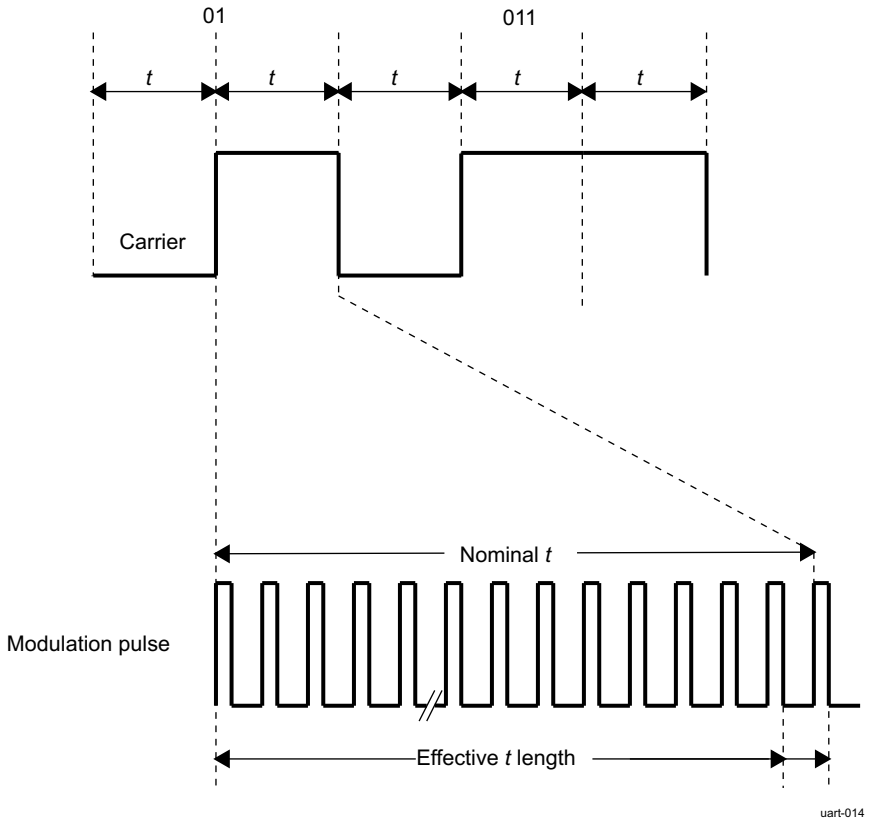


Figure 12-68. CIR Pulse Modulation

12.1.4.2.4.3.2 Pulse Duty Cycle

The programmer can choose one of four duty cycles for modulation pulses by setting the appropriate value in the UART_MDR2[5-4] CIR_PULSE_MODE bit field (1/4, 1/3, 5/12, or 1/2).

Figure 12-69 shows the CIR modulation duty cycles.

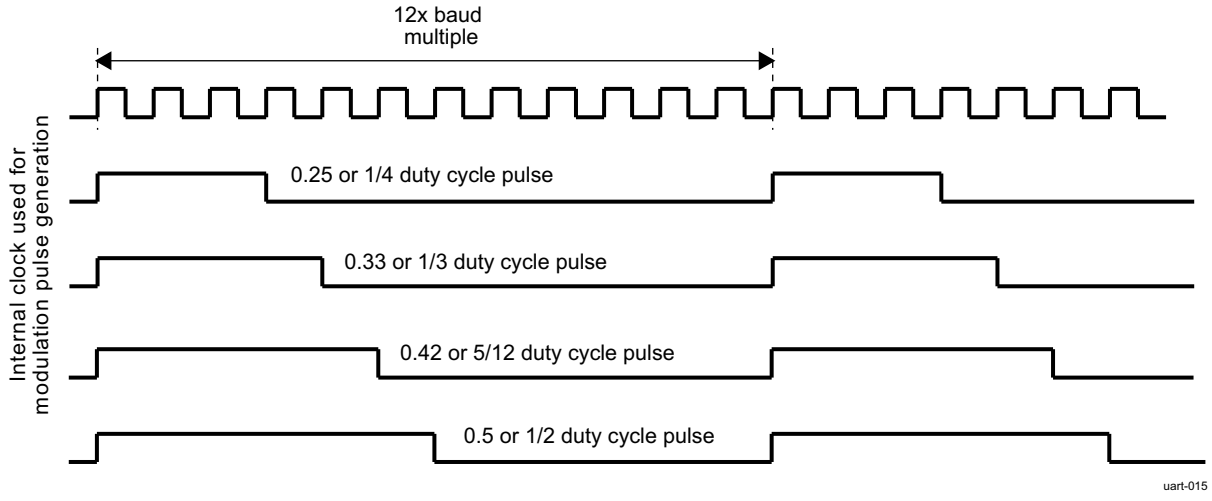


Figure 12-69. CIR Modulation Duty Cycle

The transmission logic ensures that all pulses are transmitted completely (no cutoff during transmission). While transmitting continuous bytes back-to-back, no delay is inserted between 2 transmitted bytes. Thus, software must handle the delay between consecutively transmitted bytes if the receiving end requires it.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.2.4.3.3 Consumer IR Encoding/Decoding

There are two methods of encoding for remote-control applications:

- Pulse duration encoding (time-extended bit forms): A variable pulse distance, or duration, in which the difference between logic 1 and logic 0 is the length of the pulse width
- Biphase encoding: The encoding of logic 0 and logic 1 is in the change of signal level from 1 to 0 or 0 to 1, respectively.

Japanese manufacturers favor pulse duration encoding; European manufacturers favor biphase encoding.

CIR mode uses a completely flexible free-format encoding in which 1 is transmitted from the TX FIFO as a modulated pulse with duration *t*.

Similarly, 0 is transmitted as a blank duration T. The Host CPU constructs and deciphers the protocol of the data. For example, the RC-5 protocol using Manchester encoding can be emulated as using a 01 pair for 1 and a 10 pair for 0 (see Figure 12-70).

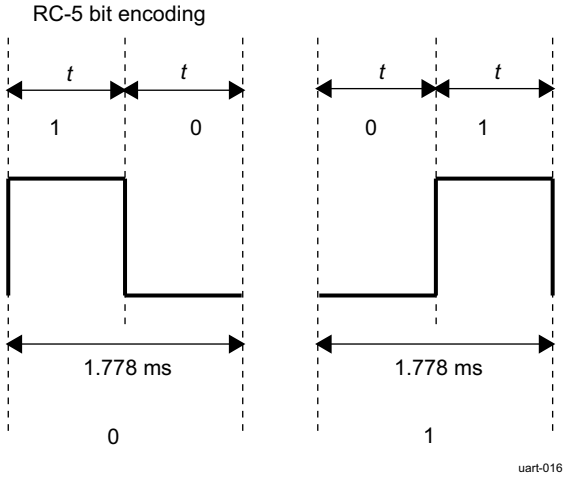


Figure 12-70. UART RC-5 Bit Encoding

Because CIR mode logic does not impose a fixed format for infrared packets of data, the Host CPU software can define the format using simple data structures that are then modulated into an industry standard, such as RC-5 or SIRC. To send a sequence of 0101 in RC-5, the Host CPU software must write an 8-bit binary character of 10011001 to the data FIFO of the UART.

For SIRC, the modulation length (multiples of *t*) is used to distinguish between 1 and 0. The subsequent SIRC digits show the difference in encoding between this and, for example, RC-5. The pulse width is extended for one digit.

Figure 12-71 shows SIRC bit encoding.

Copyright © 2026 Texas Instruments Incorporated

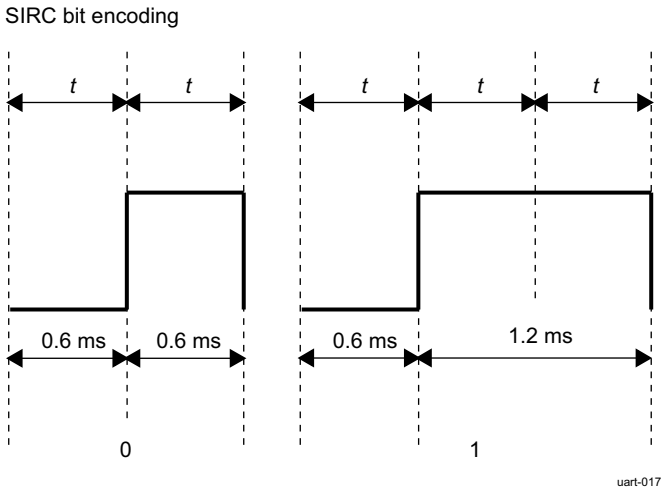


Figure 12-71. UART SIRC Bit Encoding

To construct comprehensive packets constituting remote-control commands, the Host CPU software must combine a number of 8-bit data characters in a sequence that follows one of the universally accepted formats.

Figure 12-72 shows a standard RC-5 frame as detected by UART in CIR mode (the SIRC format follows this). Each field in RC-5 can be considered as two *t* pulses (digital bits) from the TX FIFO.

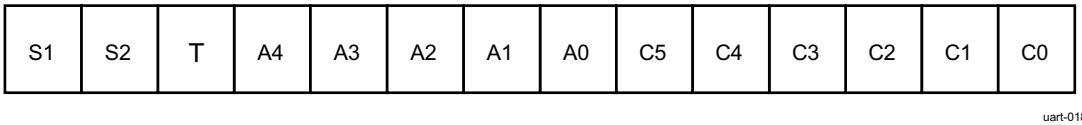


Figure 12-72. UART RC-5 Standard Packet Format

Where:

S1, S2:	Start-bits (always 1)
T:	Toggle bit
A4..A0:	Address (or system) bits
C5..C0:	Command bits

The toggle bit T changes when a new command is transmitted to detect when the same key is pressed twice (effectively receiving the same data from the host consecutively). A brief delay in the transmission of the same command is detected by the use of the toggle bit because a code is sent while the Host CPU transmits characters to the UART for transmission. The address bits define the machine or device for which the infrared transmission is intended, and the command defines the operation.

To accommodate an extended RC-5 format, the S2 bit is replaced by an additional command bit (C6) that lets the command range increase to 7 bits. This format is known as the extended RC-5 format.

The SIRC encoding uses the duration of modulation for mark and space; therefore, the duration of data bits in the standard frame length varies.

Figure 12-73 shows the packet format and bit encoding. As Figure 12-74 shows, 1 start-bit of 2.4 ms and control codes are followed by data that constitute the entire frame.

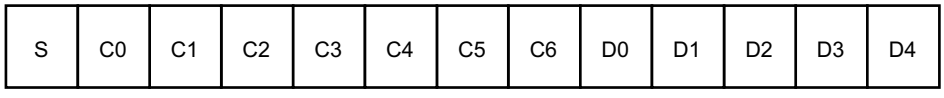


Figure 12-73. UART SIRC Packet Format

Copyright © 2026 Texas Instruments Incorporated

Note

The encoding must take a standard duration, but the contents of the data can vary. This implies that the control software for sending and receiving data packets must exercise a scheme of interpacket delay, where successive packets can be sent only after a real-time delay expires.

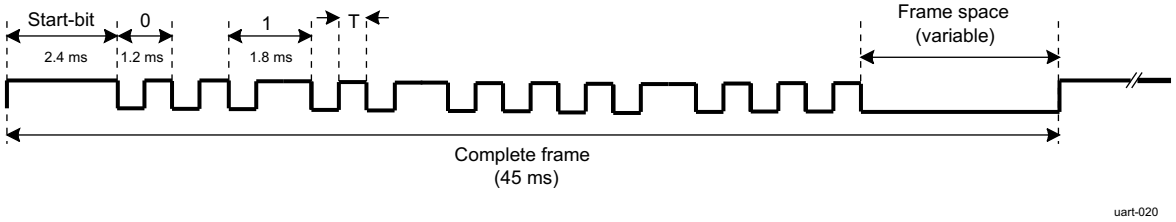


Figure 12-74. UART SIRC Bit Transmission Example

Note

This document does not describe all encoding methods and techniques; the previous information discusses the considerations required to employ different encoding methods for different industry-standard protocols. See industry-standard documentation for specific methods of encoding and protocol use.

12.1.4.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4 UART Functional Description

12.1.4.4.1 UART Block Diagram

The UART module can be divided into three main blocks:

- FIFO management
- Mode selection
- Protocol formatting

FIFO management is common to all functions and enables the transmission and reception of data from the host processor point of view.

There are two modes:

- Function mode: Routes the data to the chosen function (UART, RS-485, IrDA, or CIR) and enables the mechanism corresponding to the chosen function.
- Register mode: Enables conditional access to registers.

For more information about mode configuration, see Section 12.1.4.4.7, *Mode Selection*.

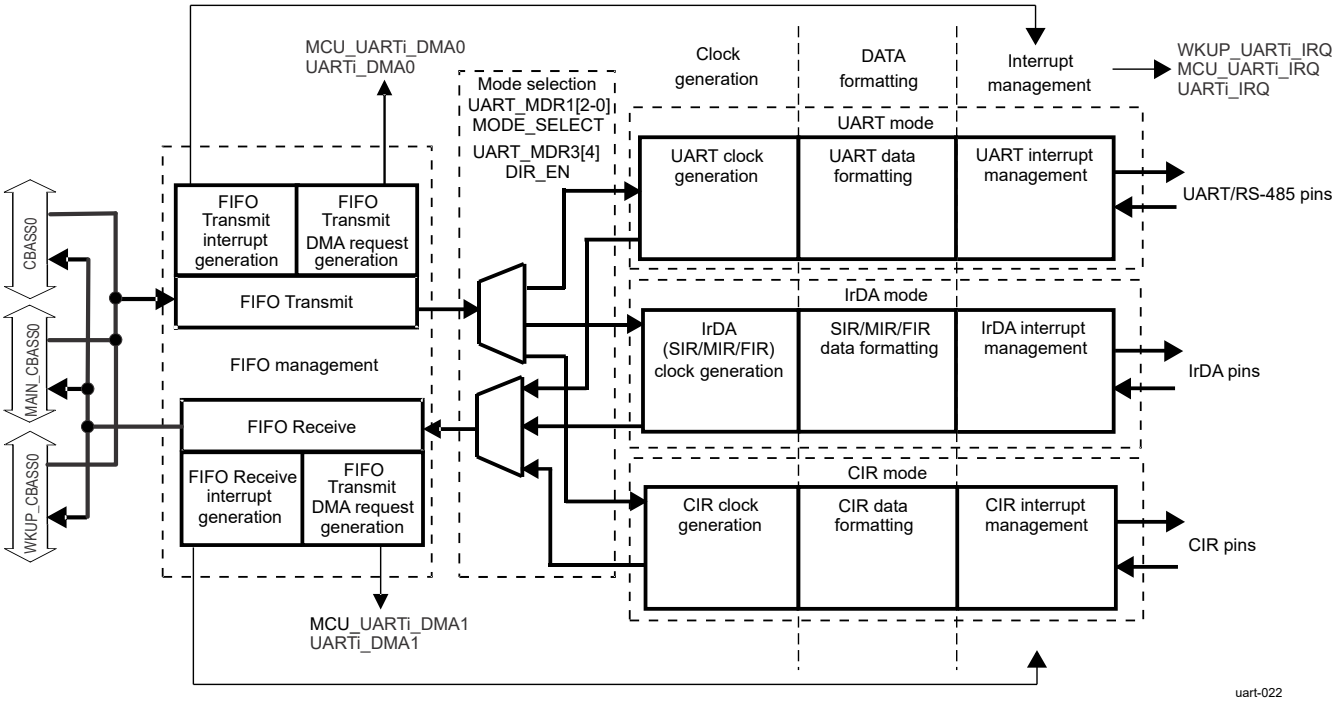
Protocol formatting has three subcategories:

- Clock generation: The 48-MHz input clock generates all necessary clocks.
- Data formatting: Each function uses its own state-machine that is responsible for the transition between FIFO data and frame data associated with it.
- Interrupt management: Different interrupt types are generated depending on the chosen function. In each mode, when an interrupt is generated, the UART_IIR_UART register indicates the interrupt type.
  - UART mode interrupts: Seven interrupts prioritized in six different levels
  - IrDA mode interrupts: Eight interrupts. The interrupt line is activated when any interrupt is generated (there is no priority).
  - CIR mode interrupts: A subset of existing IrDA mode interrupts is used.

In parallel with these functional blocks, a power-saving strategy exists for each function.

Figure 12-75 is the UART block diagram.

Copyright © 2026 Texas Instruments Incorporated



A. i represents a valid instance of UART in a domain. See the device datasheet for valid domains and instances.

Figure 12-75. UART Functional Block Diagram

12.1.4.4.2 UART Clock Configuration

Each UART uses a 48-MHz functional clock for its logic and to generate external interface signals. Each UART uses an interface clock for register accesses.

12.1.4.4.3 UART Software Reset

The UART_SYSC[1] SOFTRESET bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to hardware reset.

12.1.4.4.3.1 Independent TX/RX

The receiver and transmitter are enabled by default after reset. Software can choose to disable, re-enable or to reset either the RX or the TX side independently of the other through the UART_ECR register.

12.1.4.4.4 UART Power Management

12.1.4.4.4.1 UART Mode Power Management

12.1.4.4.4.1.1 Module Power Saving

In UART modes, sleep mode is enabled by setting the UART_IER_UART[4] SLEEP_MODE bit to 1 (when the UART_EFR[4] ENHANCED_EN bit is set to 1).

Sleep mode is entered when all of the following conditions exist:

- The serial data input line, RX, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- The only pending interrupts are THR interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set to modem mode. Therefore, even if the UART has no key role functionally, it must be initialized in a functional mode to take advantage of sleep mode.

Copyright © 2026 Texas Instruments Incorporated

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked by these clocks, this greatly reduces power consumption. The module wakes up when a change is detected on the RX line, when data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

An interrupt can be generated on a wake-up event by setting the UART_SCR[4] RX_CTS_WU_EN bit to 1. To understand how to manage the interrupt, see Section 12.1.4.4.5.1.2, *Wake-Up Interrupt*.

Note

There must be no writing to the divisor latches, UART_DLL and UART_DLH, to set the baud clock (BCLK) while in sleep mode. It is advisable to disable sleep mode using the UART_IER_UART[4] SLEEP_MODE bit before writing to the UART_DLL or UART_DLH register.

12.1.4.4.4.1.2 System Power Saving

Sleep and auto-idle modes are embedded power-saving features. Power-reduction techniques can be applied at the system level by shutting down certain internal clock and power domains of the device.

The UART supports an idle req/idle ack handshaking protocol used at the system level to shut down the UART clocks in a clean and controlled manner and to switch the UART from interrupt-generation mode to wake-up generation mode for unmasked events (see the UART_SYSC[2] ENAWAKEUP bit and the UART_WER register).

For more information, see *Power* in the *Device Configuration*.

12.1.4.4.4.2 IrDA Mode Power Management

12.1.4.4.4.2.1 Module Power Saving

In IrDA modes, sleep mode is enabled by setting the UART_MDR1[3] IR_SLEEP bit to 1.

Sleep mode is entered when all of the following conditions exist:

- The serial data input line, RXD, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when a change is detected on the RXD line or when data is written to the TX FIFO.

12.1.4.4.4.2.2 System Power Saving

System power saving for the IrDA mode has the same function as for the UART mode (see Section 12.1.4.4.4.1.2, *System Power Saving*).

12.1.4.4.4.3 CIR Mode Power Management

12.1.4.4.4.3.1 Module Power Saving

Module power saving for the CIR mode has the same function as for the IrDA mode (see Section 12.1.4.4.4.2.1, *Module Power Saving*).

12.1.4.4.4.3.2 System Power Saving

System power saving for the CIR mode has the same function as for the UART mode (see Section 12.1.4.4.4.1.2, *System Power Saving*).

12.1.4.4.4.4 Local Power Management

Table 12-64 describes power-management features available for the UART.

Copyright © 2026 Texas Instruments Incorporated

Note

For information about source clock gating and the sleep/wake-up transitions description, see *Power* in the *Device Configuration*.

Table 12-64. UART Local Power-Management Features

Feature	Registers	Description
Clock autogating	UART_SYSC[0] AUTOIDLE	This bit allows local power optimization in the module by gating the clock on interface activity or gating the UARTi_FCLK clock on internal activity.
Peripheral idle modes	UART_SYSC[4-3] IDLEMODE	Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available.
Clock activity	N/A	Feature not available
Controller standby modes	N/A	Feature not available
Global wake-up enable	UART_SYSC[2] ENAWAKEUP	This bit enables the wake-up feature at module level.
Wake-Up sources enable	N/A	Feature not available

12.1.4.4.5 UART Interrupt Requests

12.1.4.4.5.1 UART Mode Interrupt Management

12.1.4.4.5.1.1 UART Interrupts

The UART mode includes seven possible interrupts prioritized to six levels.

When an interrupt is generated, the interrupt identification register (UART_IIR_UART) sets the UART_IIR_UART[0] IT_PENDING bit to 0 to indicate that an interrupt is pending, and indicates the type of interrupt through the UART_IIR_UART[5-1] bit field. Table 12-65 summarizes the interrupt control functions.

Table 12-65. UART Mode Interrupts

IIR[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
000001	N/A	No Interrupt	N/A	N/A
000110	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read the UART_RHR register. OE: Read the UART_LSR_UART register.
001100	2	RX time-out	Stale data in RX FIFO	Read the UART_RHR register.
000100	2	RHR interrupt	DRDY (data ready) (FIFO disabled) RX FIFO above trigger level (FIFO enabled)	Read the UART_RHR register until the interrupt condition disappears
000010	3	THR interrupt	TFE (THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR until the interrupt condition disappears
000000	4	Modem status	See the UART_MSR register.	Read the MSR register
010000	5	XOFF interrupt/ special character interrupt	Receive XOFF characters/special character	Receive XON character(s), if XOFF interrupt/read of the UART_IIR_UART register, if special character interrupt
100000	6	CTS, RTS	RTS pin or CTS pin change state from active (low) to inactive (high)	Read the UART_IIR_UART register

For the receiver-line status interrupt, the UART_LSR_UART[7] RX_FIFO_STS bit generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the UART_IIR_UART register.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.5.1.2 Wake-Up Interrupt

Wake-up interrupt is a special interrupt that works differently from other interrupts. This interrupt is enabled when the UART_SCR[4] RX_CTS_WU_EN bit is set to 1. The UART_IIR_UART register is not modified when this occurs; the UART_SSR[1] RX_CTS_WU_STS bit must be checked to detect a wake-up event.

When a wake-up interrupt occurs, it can be cleared only by resetting the UART_SCR[4] RX_CTS_WU_EN bit. This bit must be reenabled (set to 1) after the current wake-up interrupt event is processed to detect the next incoming wake-up event.

12.1.4.4.5.2 IrDA Mode Interrupt Management

12.1.4.4.5.2.1 IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UART_IER_IRDA). The interrupt status of the device can be checked by reading the interrupt identification register (UART_IIR_IRDA).

The UART, IrDA, and CIR modes have different interrupts in the UART module and, therefore, different UART_IER_IRDA and UART_IIR_IRDA mappings, depending on the selected mode.

The IrDA modes have eight possible interrupts (see Table 12-66). The interrupt line is activated when any interrupt is generated (there is no priority).

Table 12-66. IrDA Mode Interrupts

IIR_IRDA Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disabled) RX FIFO above trigger level (FIFO enabled)	Read the UART_RHR register until the interrupt condition disappears.
1	THR interrupt	TFE (UART_THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR until the interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO is available to be read at the UART_RHR port.	Read the UART_RHR register.
3	RX overrun	Write to the UART_RHR register when the RX FIFO is full.	Read UART_RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX status	UART_THR empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error OR Transmission of the last bit of the IrDA frame completed successfully.	Read the UART_RESUME register OR Read the UART_IIR_IRDA register.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into the STATUS FIFO.	Read the STATUS FIFO (read until empty - maximum of eight reads required).
7	Received EOF	Received end-of-frame	Read the UART_IIR_IRDA register.

12.1.4.4.5.2.2 Wake-Up Interrupts

The wake-up interrupt for IrDA mode has the same function as that for UART mode (see Section 12.1.4.4.5.1.2, *Wake-Up Interrupt*).

12.1.4.4.5.3 CIR Mode Interrupt Management

12.1.4.4.5.3.1 CIR Interrupts

The CIR function generates interrupts that can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UART_IER_CIR). The interrupt status of the device can be checked by reading the interrupt identification register (UART_IIR_CIR).

The UART, IrDA, and CIR modes have different interrupts in the UART module and, therefore, different UART_IER_CIR and UART_IIR_CIR mappings, depending on the selected mode.

Copyright © 2026 Texas Instruments Incorporated

Table 12-67 lists the interrupt modes to be maintained. In CIR mode, the sole purpose of the UART_IIR_CIR[5] TX_STATUS_IT bit is to indicate that the last bit of infrared data was passed to the TX pin.

Table 12-67. CIR Mode Interrupts

IIR_CIR Bit Number	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
1	THR interrupt	TFE (THR empty) (FIFO disabled) TX FIFO below trigger level (FIFO enabled)	Write to the UART_THR register until the interrupt condition disappears
2	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
3	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
4	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
5	TX status	Transmission of the last bit of the frame is complete successfully	Read the UART_IIR_CIR register
6	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
7	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode

12.1.4.4.5.3.2 Wake-Up Interrupts

The wake-up interrupt for CIR mode has the same function as that for UART mode (see Section 12.1.4.4.5.1.2, *Wake-Up Interrupt*).

12.1.4.4.6 UART FIFO Management

The FIFO is accessed by reading and writing the UART_RHR and UART_THR registers. Parameters are controlled using the FIFO control register (UART_FCR) and supplementary control register (UART_SCR). Reading the UART_SSR[0] TX_FIFO_FULL bit at 1 means the FIFO is full.

The UART_TLR register controls the FIFO trigger level, which enables DMA and interrupt generation. After reset, transmit (TX) and receive (RX) FIFOs are disabled; thus, the trigger level is the default value of 1 byte. Figure 12-76 shows the FIFO management registers.

Note

Data in the UART_RHR register is not overwritten when an overflow occurs.

Note

The UART_SFLSR, UART_SFREGL, and UART_SFREGH status registers are used in IrDA mode only. For information about their use, see Section 12.1.4.4.8.3.3, *IrDA Data Formatting*.

Note

Bits UART_FCR[2] TX_FIFO_CLEAR and UART_FCR[1] RX_FIFO_CLEAR are automatically cleared by hardware after 4 × + 5 × UARTi_FCLK clock cycles. This delay is needed to finish the resetting of the corresponding FIFO and DMA control registers.

Copyright © 2026 Texas Instruments Incorporated

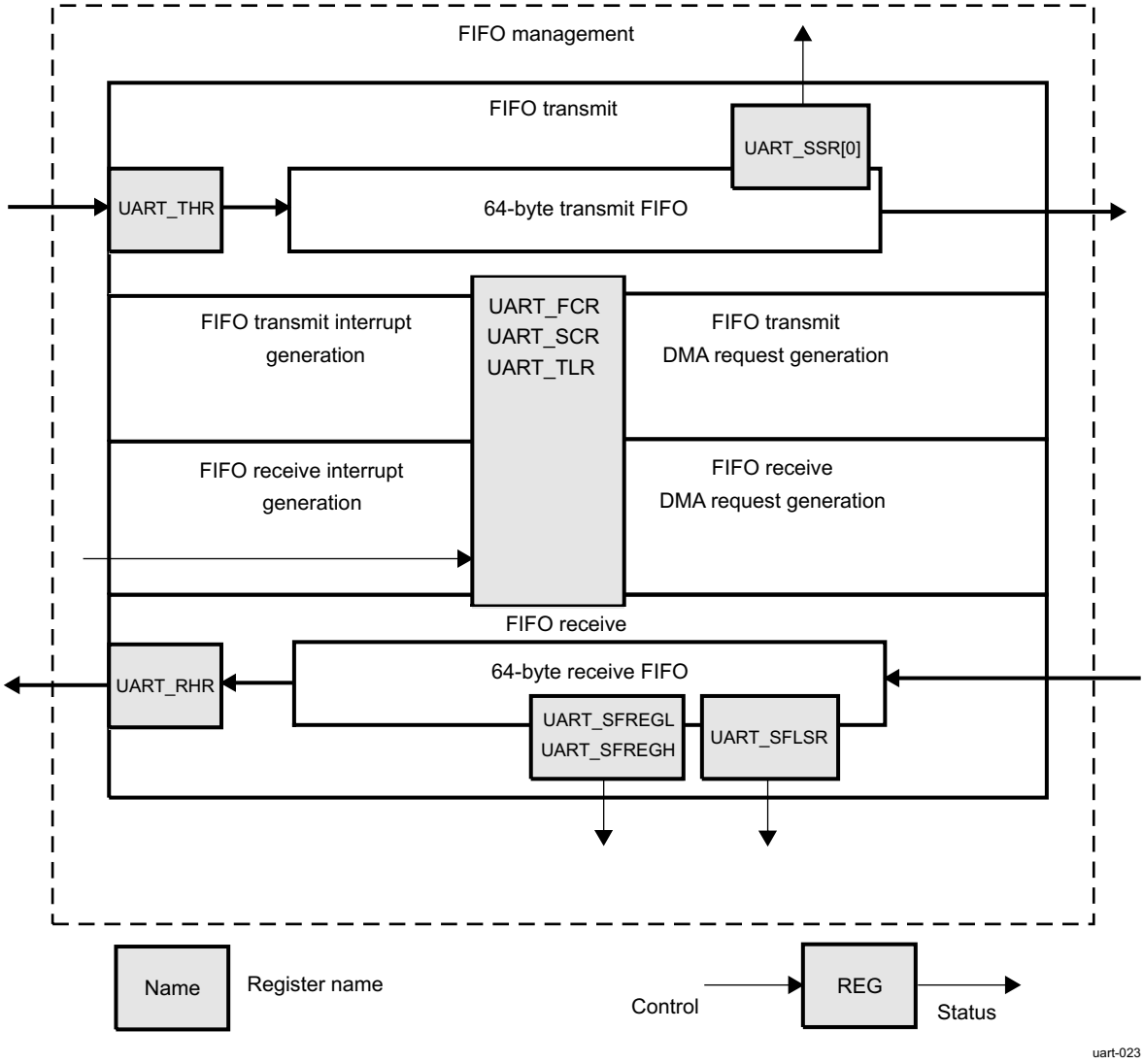


Figure 12-76. UART FIFO Management Registers

12.1.4.4.6.1 FIFO Trigger

12.1.4.4.6.1.1 Transmit FIFO Trigger

Table 12-68 lists the TX FIFO trigger level settings.

Table 12-68. UART TX FIFO Trigger Level Setting Summary

SCR[6]	TLR[3:0]	TX FIFO Trigger Level
0	= 0x0	Defined by the UART_FCR[5-4] TX_FIFO_TRIG bit field (8,16, 32, or 56 spaces)
0	!= 0x0	Defined by the UART_TLR[3-0] TX_FIFO_TRIG_DMA bit field (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TX_FIFO_TRIG_DMA and TX_FIFO_TRIG (from 1 to 63 spaces with a granularity of 1 space) **Note:** The combination of TX_FIFO_TRIG_DMA = 0x0 and TX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of 1 space required). All zeros result in unpredictable behavior.

12.1.4.4.6.1.2 Receive FIFO Trigger

Table 12-69 lists the RX FIFO trigger-level settings.

Copyright © 2026 Texas Instruments Incorporated

Table 12-69. UART RX FIFO Trigger-Level Setting Summary

SCR[7]	TLR[7:4]	RX FIFO Trigger Level
0	= 0x0	Defined by the UART_FCR[7-6] RX_FIFO_TRIG bit field (8,16, 56, or 60 characters)
0	!= 0x0	Defined by the UART_TLR[7-4] RX_FIFO_TRIG_DMA bit field (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of RX_FIFO_TRIG_DMA and RX_FIFO_TRIG (from 1 to 63 characters with a granularity of 1 character) **Note:** The combination of RX_FIFO_TRIG_DMA = 0x0 and RX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of 1 character required). All zeros result in unpredictable behavior.

The receive threshold is programmed using the UART_TCR[7-4] RX_FIFO_TRIG_START and UART_TCR[3-0] RX_FIFO_TRIG_HALT bit fields:

- Trigger levels from 0 to 60 bytes are available with a granularity of 4 (trigger level = 4 × [4-bit register value]).
- To ensure correct device operation, ensure that RX_FIFO_TRIG_HALT > RX_FIFO_TRIG when auto-RTS is enabled.

  Delay = [4 + 16 × (1 + CHAR_LENGTH + Parity + Stop – 0.5)] × Baud_rate + 4 × FCLK

  Note

  The RTS signal is deasserted after the UART module receives the data over RX_FIFO_TRIG_HALT. Delay means how long the UART module takes to deassert the RTS signal after reaching RX_FIFO_TRIG_HALT.

- In FIFO interrupt mode with flow control, ensure that the trigger level to HALT transmission is greater than or equal to the RX FIFO trigger level (the UART_TCR[7-4] RX_FIFO_TRIG_START bit field or the UART_FCR[7-6] RX_FIFO_TRIG bit field); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent when a byte is received.

12.1.4.4.6.2 FIFO Interrupt Mode

In FIFO interrupt mode (the FIFO control register UART_FCR[0] FIFO_EN bit is set to 1 and relevant interrupts are enabled by the UART_IER_UART register), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the RX/TX FIFO threshold (the UART_TLR[7-4] RX_FIFO_TRIG_DMA and UART_TLR[3-0] TX_FIFO_TRIG_DMA bit fields or the UART_FCR[7-6] RX_FIFO_TRIG and UART_FCR[5-4] TX_FIFO_TRIG bit fields, respectively) is reached.

The interrupt signals instruct the Host CPU to transfer data to the destination (from the UART in receive mode and/or from any source to the UART FIFO in transmit mode).

When UART flow control is enabled with interrupt capabilities, the UART flow control FIFO threshold (the UART_TCR[3-0] RX_FIFO_TRIG_HALT bit field) must be greater than or equal to the RX FIFO threshold.

Figure 12-77 shows the generation of the RX FIFO interrupt request.

Copyright © 2026 Texas Instruments Incorporated

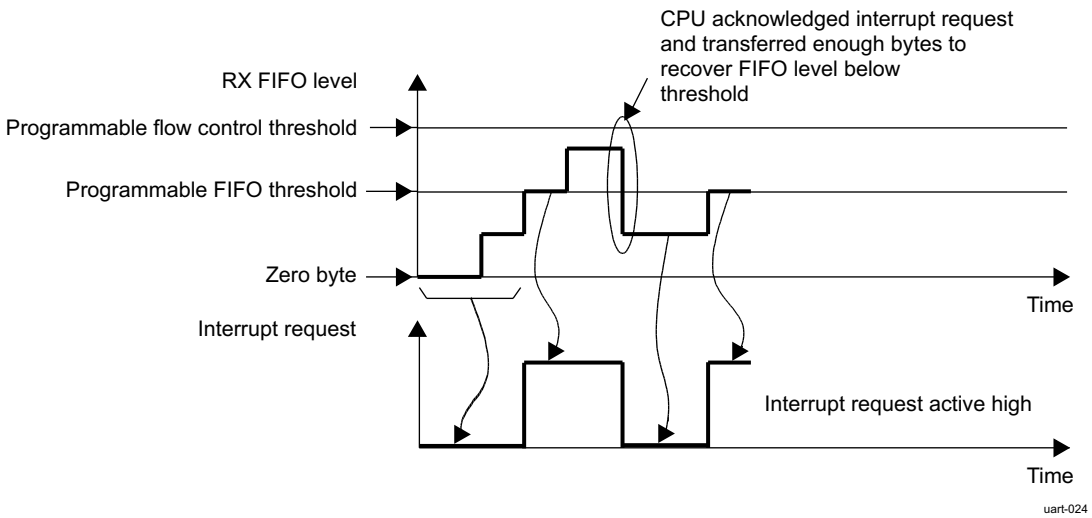


Figure 12-77. UART RX FIFO Interrupt Request Generation

In receive mode, no interrupt is generated until the RX FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the Host CPU has handled enough bytes to put the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Figure 12-78 shows the generation of the TX FIFO interrupt request.

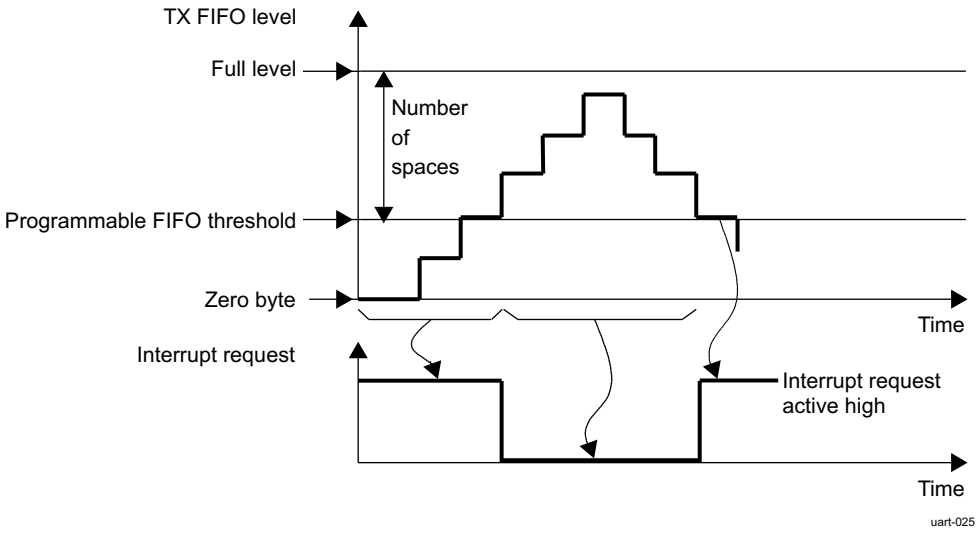


Figure 12-78. UART TX FIFO Interrupt Request Generation

In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is deasserted when the TX FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements is transmitted to go below the TX FIFO threshold.

12.1.4.4.6.3 FIFO Polled Mode Operation

In FIFO polled mode (the UART_FCR[0] FIFO_EN bit is set to 0 and the relevant interrupts are disabled by the UART_IER_UART register), the status of the receiver and transmitter can be checked by polling the line status register (UART_LSR_UART).

This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically determined by sending interrupts to the Host CPU.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.6.4 FIFO DMA Mode Operation

Although the DMA operation includes four modes (DMA modes 0 through 3), the information in *UART Hardware Requests*, assumes that mode 1 is used. (Mode 2 and mode 3 are legacy modes that use only one DMA request for each module.)

In mode 2, the remaining DMA request is used for RX. In mode 3, the remaining DMA request is used for TX.

DMA requests in mode 2 and mode 3 use the signals (where i = 0 to).

signals are not used by the module in mode 2 and mode 3:

The DMA mode and signals usage can be selected as follows:

- When the UART_SCR[0] DMA_MODE_CTL bit is set to 0, setting the UART_FCR[3] DMA_MODE bit to 0 enables DMA mode 0. Setting the UART_FCR[3] DMA_MODE bit to 1 enables DMA mode 1.
- When the UART_SCR[0] DMA_MODE_CTL bit is set to 1, the UART_SCR[2-1] DMA_MODE_2 bit field determines DMA mode 0 to mode 3 based on the supplementary control register (UART_SCR) description.

For example:

- If no DMA operation is desired, set the UART_SCR[0] DMA_MODE_CTL bit to 1 and the UART_SCR[2-1] DMA_MODE_2 bit field to 0x0. (The DMA_MODE bit is discarded.)
- If DMA mode 1 is desired, set the UART_SCR[0] DMA_MODE_CTL bit to 0 and the UART_FCR[3] DMA_MODE bit to 1, or set the UART_SCR[0] DMA_MODE_CTL bit to 1 and the UART_SCR[2-1] DMA_MODE_2 bit field to 01. (The UART_FCR[3] DMA_MODE bit is discarded.)

If the FIFOs are disabled (the UART_FCR[0] FIFO_EN bit is set to 0), the DMA occurs in single-character transfers.

When DMA mode 0 is programmed, the signals associated with DMA operation are not active.

Depending on UART_MDR3[2] SET_DMA_TX_THRESHOLD, the threshold can be programmed different ways:

- SET_TX_DMA_THRESHOLD = 1:

  The threshold value will be the value of the UART_TX_DMA_THRESHOLD register. If SET_TX_DMA_THRESHOLD + TX trigger spaces 64, then the default method of threshold is used: threshold value = TX FIFO size.
- SET_TX_DMA_THRESHOLD = 0:

  The threshold value = TX FIFO size TX trigger space. The TX DMA line is asserted if the TX FIFO level is lower then the threshold. It remains asserted until TX trigger spaces number of bytes are written into the FIFO. The DMA line is then deasserted and the FIFO level is compared with the threshold value.

12.1.4.4.6.4.1 DMA sequence to disable TX DMA

In order to disable TX DMA if it is not needed anymore (e.g. all transfers are done and UART idle mode is desired), the following sequence must be use

1. DMA mode 1 is set (both TX/RX DMA) by registers UART_SCR[0] DMA_MODE_CTL = 0 and UART_FCR[3] DMA_MODE = 1:
   a. Set the UART_SCR[2-1] DMA_MODE_2 bit fields to 01 (DMA mode 1)
   b. Set the UART_SCR[0] DMA_MODE_CTL bit to1 (this setting of UART_SCR[0] DMA_MODE-CTL will ignores UART_FCR[3] DMA_MODE_CTL bit)

      Note

      It is strongly suggested to do steps ‘a’ and ‘b’ in two separate write in order to avoid malfunction of the device.

   c. Set the UART_FCR[3] DMA_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART_SCR[0] DMA_MODE_CTL bit. Be sure that all data was read out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON

Copyright © 2026 Texas Instruments Incorporated

protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART_ACREG[5] DIS_IR_RX bit

Note

There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

d. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
e. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 10 (DMA mode 2, RX only).
f. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
g. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.

2. DMA mode 1 is set (both TX/RX DMA) by registers UART_FCR[3] DMA_MODE = 0 and UART_SCR[0] DMA_MODE_CTL = 1, UART_SCR[2-1] DMA_MODE_2 = 01. It is almost the same as above, but steps ‘a’, and ‘b’ can be skipped:
   a. Set the UART_FCR[3] DMA_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART_SCR[0] DMA_MODE_CTL bit. Be sure that all data was read out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART_ACREG[5] DIS_IR_RX bit

   Note

   There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

   b. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
   c. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 10 (DMA mode 2, RX only).
   d. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
   e. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.

3. DMA mode 3 is set (TX DMA only) by registers UART_FCR[3] DMA_MODE = 0 and UART_SCR[0] DMA_MODE_CTL = 1, UART_SCR[2-1] DMA_MODE_2 = 11. It is the same as above:
   a. Set the UART_FCR[3] DMA_MODE bit to 0. It is not necessary but suggested to avoid restore of DMA mode 1 during accidental reset of UART_SCR[0] DMA_MODE_CTL bit. Be sure that all data was read out from RX FIFO and if it possible disable the RX side. In UART mode the RTS/CTS or XOFF/XON protocol can be used. In IrDA modes RX can be forcibly disabled by setting UART_ACREG[5] DIS_IR_RX bit

   Note

   There can be RX DATA loss during the next steps if all DATA was not read out or there was an ongoing reception!

   b. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO).
   c. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 10 (DMA mode 2, RX only).
   d. Set the UART_FCR[2-1] DMA_MODE bit field to 11 (clear TX and RX FIFO and the DMA request again).
   e. Set the UART_SCR[2-1] DMA_MODE_2 bit field to 00 (no DMA) or keep 10 if RX DMA is needed.

12.1.4.4.6.4.2 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 12-79 through Figure 12-82 show the supported DMA operations.

Copyright © 2026 Texas Instruments Incorporated

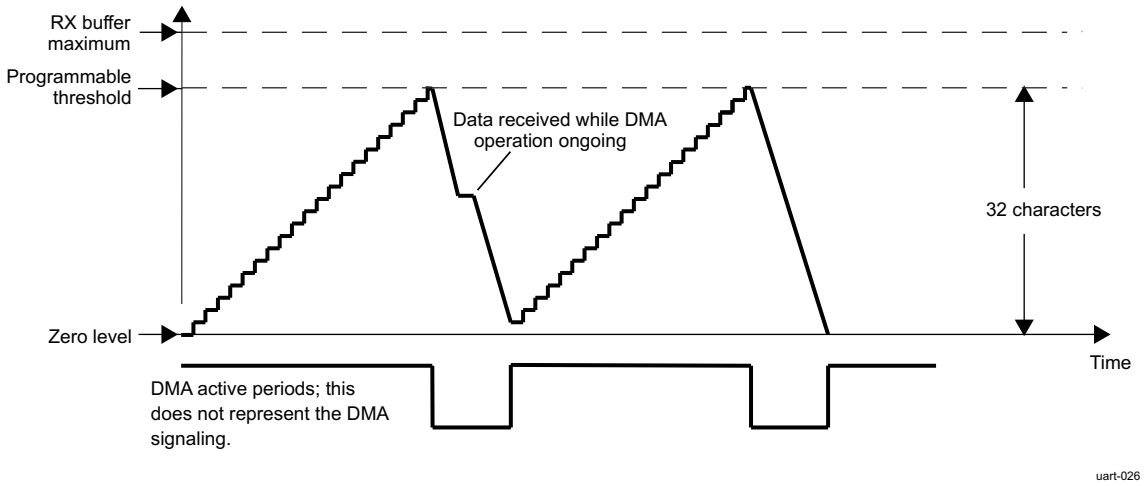


Figure 12-79. UART Receive FIFO DMA Request Generation (32 Characters)

In receive mode, a DMA request is generated when the RX FIFO reaches its threshold level defined in the trigger level register (UART_TLR). This request is deasserted when the number of bytes defined by the threshold level is read by the device DMA controllers.

In transmit mode, a DMA request is automatically asserted when the TX FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the UART_TLR register is written by the device DMA controllers. If an insufficient number of characters is written, the DMA request stays active.

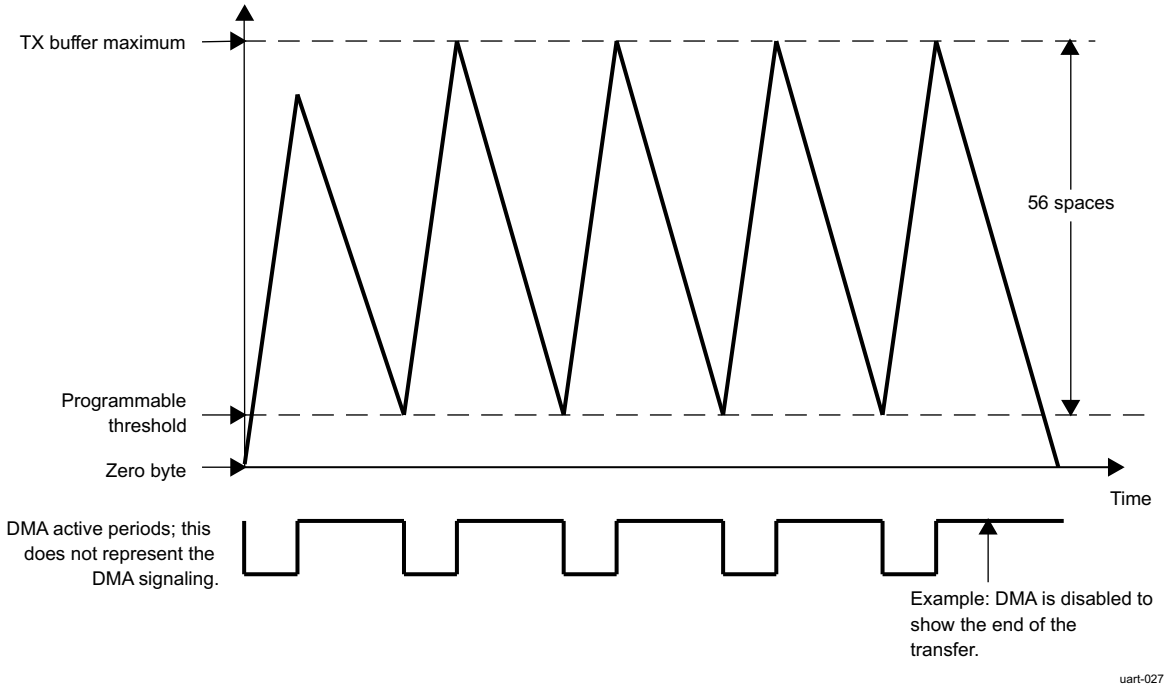


Figure 12-80. UART Transmit FIFO DMA Request Generation (56 Spaces)

The DMA request is again asserted if the FIFO can receive the number of bytes defined by the UART_TLR register.

The threshold can be programmed in a number of ways. Figure 12-80 shows a DMA transfer operating with a space setting of 56 that can arise from using the auto settings in the UART_FCR[5-4] TX_FIFO_TRIG bit field or the UART_TLR[3-0] TX_FIFO_TRIG_DMA bit field concatenated with the TX_FIFO_TRIG bit field.

Copyright © 2026 Texas Instruments Incorporated

The setting of 56 spaces in the UART module must correlate with the settings of the device DMA controllers, so that the buffer does not overflow (program the DMA request size of the LH controller to equal the number of spaces in the UART module).

Figure 12-81 shows an example with eight spaces to show the buffer level crossing the space threshold. The LH DMA controller settings must correspond to those of the UART module.

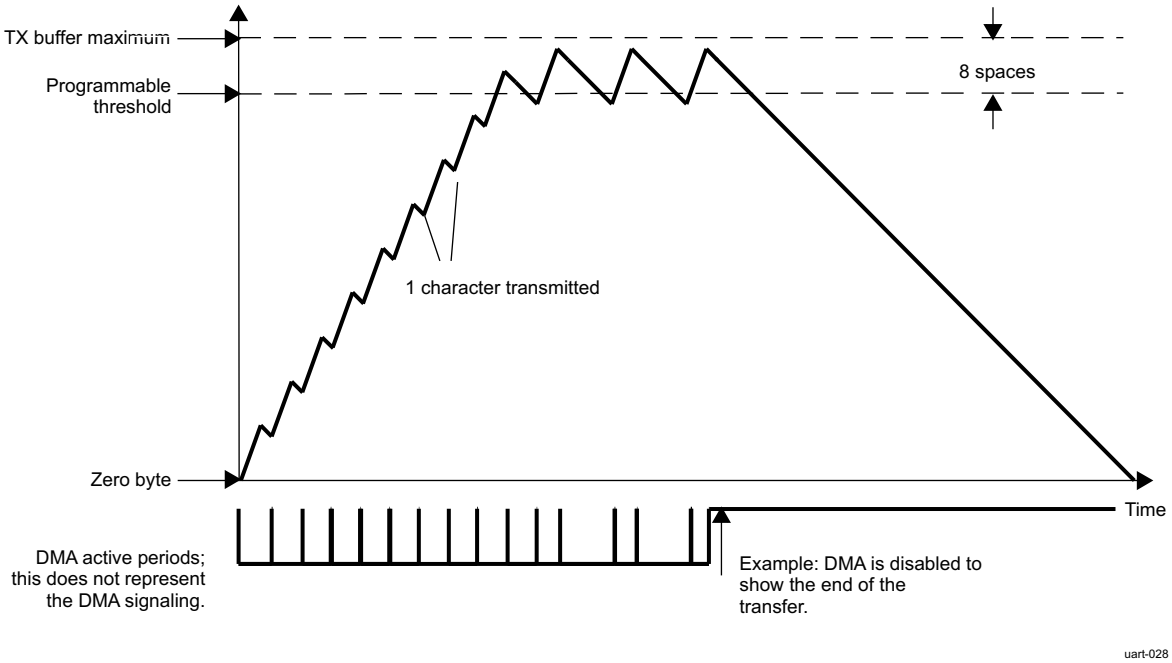


Figure 12-81. UART Transmit FIFO DMA Request Generation (8 Spaces)

The next example shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer (see Figure 12-82). The buffer is filled faster than the baud rate at which data is transmitted to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer.

On two occasions, the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the UART_DLL and UART_DLH registers.

The DMA settings must correspond to the system LH DMA controller settings to ensure correct operation of this logic.

Copyright © 2026 Texas Instruments Incorporated

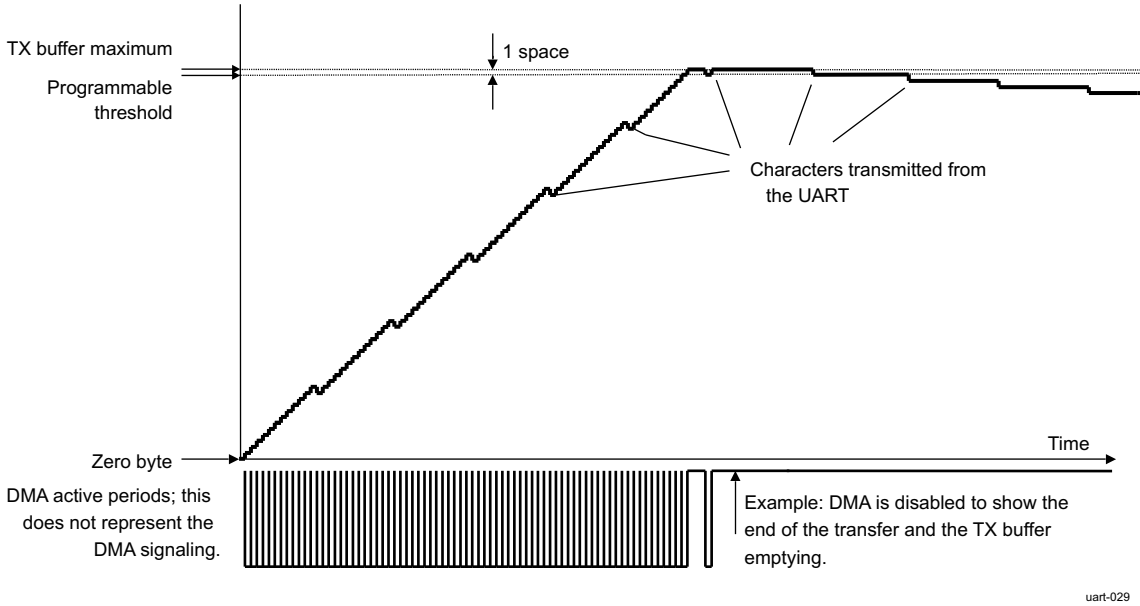


Figure 12-82. UART Transmit FIFO DMA Request Generation (1 Space)

The final example illustrates the setting of eight spaces but setting the TX DMA threshold directly by setting UART_MDR3[1] NONDEFAULT_FREQ bit and UART_TX_DMA_THRESHOLD register (see Figure 12-83). In the example, the UART_TX_DMA_THRESHOLD[5-0] TX_DMA_THRESHOLD = 3 and the trigger level is 8. The buffer is filled at a faster rate than the BAUD rate transmits data to the TX pin. The buffer is filled with 8 bytes and the DMA operations stop transferring data to the transmit buffer. When the buffer is emptied to the threshold level by transmission, the DMA operation activates again to fill the buffer with 8 bytes.

Eventually, the buffer will be emptied at the rate specified by the BAUD Rate settings of the UART_DLL and UART_DLH registers.

If the selected threshold level + trigger level exceeds max buffer size, then the original TX DMA threshold method is used to prevent TX overrun, regardless of the UART_MDR3[1] NONDEFAULT_FREQ value.

The DMA settings should correspond to the system Local Host DMA controller settings in order to ensure the correct operation of this logic.

Copyright © 2026 Texas Instruments Incorporated

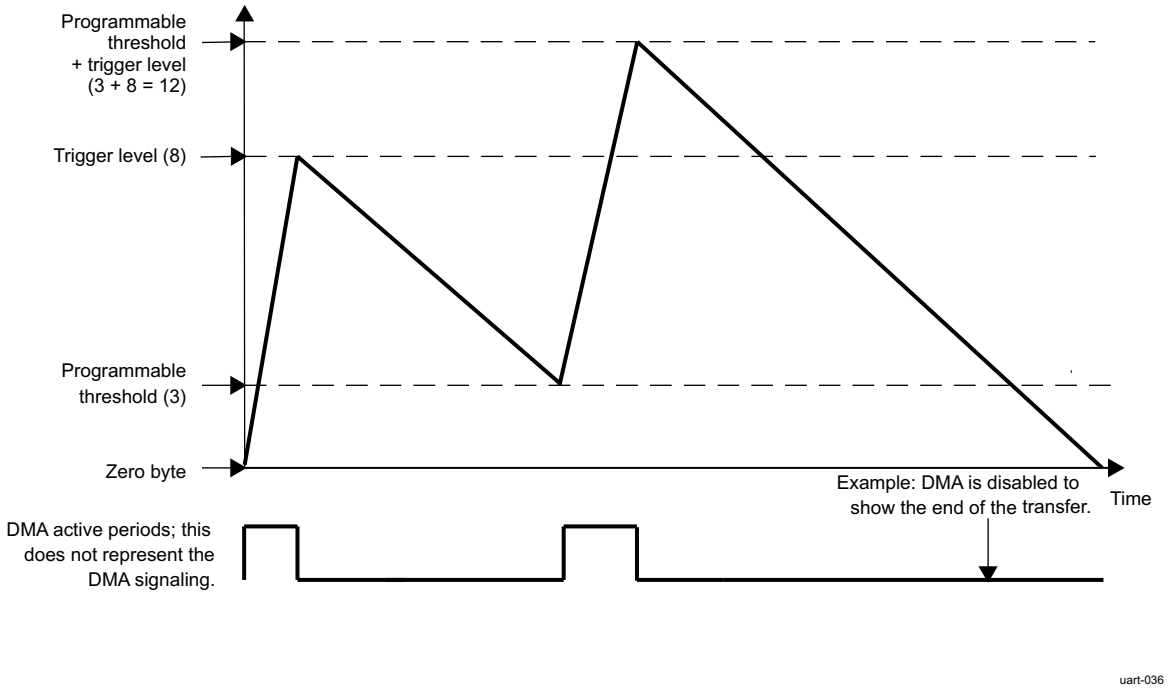


Figure 12-83. UART Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming. (Threshold = 3; Spaces = 8)

12.1.4.4.6.4.3 DMA Transmission

Figure 12-84 shows DMA transmission.

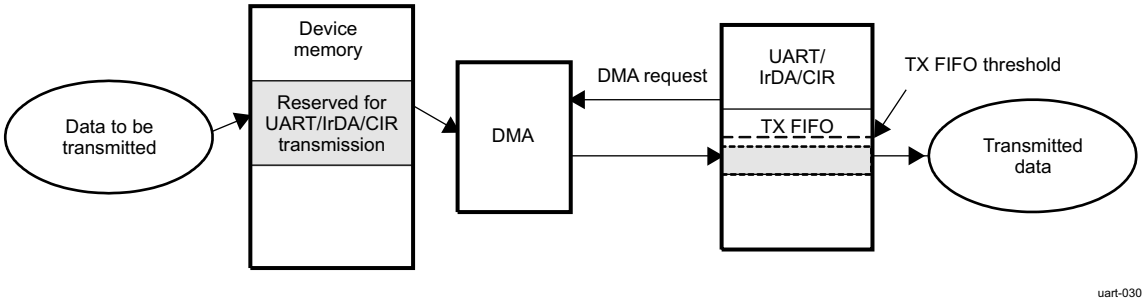


Figure 12-84. DMA Transmission

1. Data to be transmitted are put in the device memory reserved for UART transmission by the DMA:
   a. Until the TX FIFO trigger level is not reached, a DMA request is generated
   b. An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
2. Data in the TX FIFO are automatically transmitted.
3. The end of the transmission is signaled by the UART_THR empty (TX FIFO empty).

Note

In IrDA mode, the transmission does not end immediately after the TX FIFO empties, at which point the last data byte, the CRC field, and the stop flag still must be transmitted; thus, the end of transmission occurs a few milliseconds after the UART_THR register empties.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.6.4.4 DMA Reception

Figure 12-85 shows DMA reception.

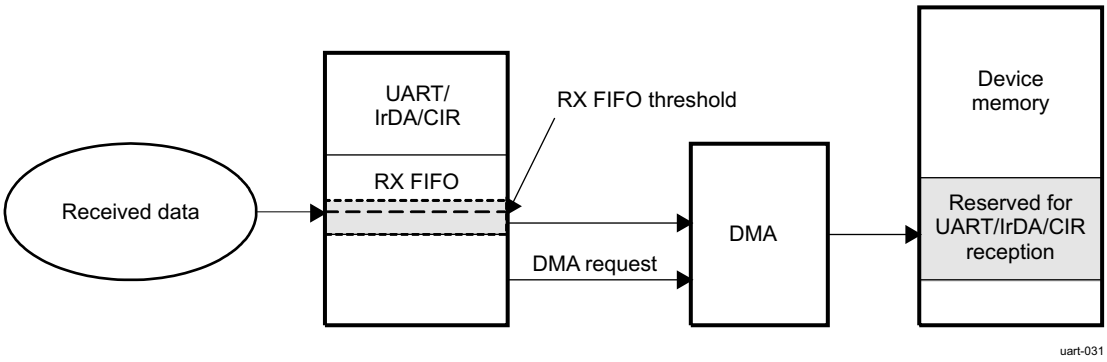


Figure 12-85. DMA Reception

1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA:
   a. At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
   b. An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

12.1.4.4.7 UART Mode Selection

12.1.4.4.7.1 Register Access Modes

12.1.4.4.7.1.1 Operational Mode and Configuration Modes

Register access depends on the register access mode, although register access modes are not correlated to functional mode selection. Three different modes are available:

- Operational mode
- Configuration mode A
- Configuration mode B

Operational mode is the selected mode when the function is active; serial data transfer can be performed in this mode.

Configuration mode A and configuration mode B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in the operational mode. The modes are used when the module is inactive (no serial data transfer processed) and only for initialization or reconfiguration of the module.

The value of the UART_LCR register determines the register access mode (see Table 12-70).

Table 12-70. UART Register Access Mode Programming (Using UART_LCR)

Mode	Condition
Configuration mode A	UART_LCR[7] = 0x1 and UART_LCR[7-0] != 0xBF
Configuration mode B	UART_LCR[7] = 0x1 and UART_LCR[7-0] = 0xBF
Operational mode	UART_LCR[7] = 0x0

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.7.1.2 Register Access Submode

In each access register mode (operational mode or configuration mode A/B), some register accesses are conditional on the programming of a submode (MSR_SPR, TCR_TLR, and XOFF). These registers are identified in Table 12-93, *UART Load FIFO Triggers Defined by the Concatenated Value*.

Table 12-71 through Table 12-73 summarize the register access submodes.

Table 12-71. UART Subconfiguration Mode A Summary

Mode	Condition
MSR_SPR	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

Table 12-72. UART Subconfiguration Mode B Summary

Mode	Condition
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1
XOFF	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)

Table 12-73. UART Suboperational Mode Summary

Mode	Condition
MSR_SPR	UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

12.1.4.4.7.1.3 Registers Available for the Register Access Modes

Table 12-74 lists the names of the register bits in each access register mode. Gray shading indicates that the register does not depend on the register access mode (available in all modes).

Table 12-74. UART Register Access Mode Overview

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART	UART_IER_UART
0x008	UART_IIR_UART	UART_FCR	UART_EFR	UART_EFR	UART_IIR_UART	UART_FCR
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_ADDR1	UART_XON1_ADDR1	UART_MCR	UART_MCR
0x014	UART_LSR_UART	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR_UART	–
0x018	UART_MSR / UART_TCR	UART_TCR	UART_TCR / UART_XOFF1	UART_TCR / UART_XOFF1	UART_MSR / UART_TCR	UART_TCR
0x01C	UART_SPR / UART_TLR	UART_SPR / UART_TLR	UART_TLR / UART_XOFF2	UART_TLR / UART_XOFF2	UART_SPR / UART_TLR	UART_SPR / UART_TLR
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH
0x038	UART_UASR	–	UART_UASR	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR

Copyright © 2026 Texas Instruments Incorporated

Table 12-74. UART Register Access Mode Overview (continued)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

12.1.4.4.7.2 UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection

To select a mode, set the UART_MDR1[2:0] MODE_SELECT bit field (see Table 12-75).

Table 12-75. UART Mode Selection

Value	Mode
0x0:	UART 16× mode
0x1:	SIR mode
0x2:	UART 16× auto-baud
0x3:	UART 13× mode
0x4:	MIR mode
0x5:	FIR mode
0x6:	CIR mode

MODE_SELECT is effective when the module is in operational mode (see Section 12.1.4.4.7.1, *Register Access Modes*).

To select a RS-485 mode, set the UART_MDR3[4] DIR_EN bit field to 0x1.

12.1.4.4.7.2.1 Registers Available for the UART Function

Only the registers listed in Table 12-76 are used for the UART function.

Table 12-76. UART Mode Register Overview

Address Offset	Registers[1] [2]					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (UART)	UART_IER_UART (UART)
0x008	UART_IIR_UART	UART_FCR	UART_EFR [4]	UART_EFR [4]	UART_IIR_UART (UART)	UART_FCR (UART)

Copyright © 2026 Texas Instruments Incorporated

Table 12-76. UART Mode Register Overview (continued)

Address Offset	Registers(1) (2)					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_ADDR1	UART_XON1_ADDR1	UART_MCR	UART_MCR
0x014	UART_LSR_UART (UART)	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR_UART (UART)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_XOFF1/ UART_TCR	UART_XOFF1/ UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR/ UART_XOFF2	UART_TLR/ UART_XOFF2	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]	UART_MDR1 [2-0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	–	–	–	–	–	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	UART_UASR	–	UART_UASR	–	–	–
0x03C	–	–	–	–	–	–
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	–	–
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions (described separately in *UART Registers*).
(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

12.1.4.4.7.2.2 Registers Available for the IrDA Function

Only the registers listed in Table 12-77 are used for the IrDA function.

Table 12-77. IrDA Mode Register Overview

Address Offset	Registers(1) (2)					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR

Copyright © 2026 Texas Instruments Incorporated

Table 12-77. IrDA Mode Register Overview (continued)

Address Offset	Registers(1) (2)					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (IrDA)	UART_IER_UART (IrDA)
0x008	UART_IIR_UART	UART_FCR	UART_EFR [4]	UART_EFR [4]	UART_IIR_UART (IrDA)	UART_FCR (IrDA)
0x00C	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]
0x010	–	–	UART_XON1_ADDR1	UART_XON1_ADDR1	–	–
0x014	UART_LSR_UART (IrDA)	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR_UART (IrDA)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR	UART_TLR	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH
0x038	–	–	–	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

(1) REGISTER_NAME(IrDA) notation indicates that the register exists for other functions (UART or CIR), but fields have different meanings for other functions (described separately in *UART Registers*).
(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the IrDA function.

12.1.4.4.7.2.3 Registers Available for the CIR Function

Only the registers listed in Table 12-78 are used for the CIR function.

Copyright © 2026 Texas Instruments Incorporated

Table 12-78. CIR Mode Register Overview

Address Offset	Registers[1] [2]					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	–	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER_UART (CIR)	UART_IER_UART (CIR)
0x008	UART_IIR_UART	UART_FCR	UART_EFR	UART_EFR	UART_IIR_UART (CIR)	UART_FCR (CIR)
0x00C	UART_LCR	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]	UART_LCR [7]
0x010	–	–	–	–	–	–
0x014	UART_LSR_UART (CIR)	–	–	–	UART_LSR_UART (CIR)	–
0x018	UART_MSR/ UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/ UART_TCR	UART_TCR
0x01C	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR	UART_TLR	UART_TLR	UART_TLR/ UART_SPR	UART_TLR/ UART_SPR
0x020	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]	UART_MDR1 [3-0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	UART_RESUME	–	UART_RESUME	–	UART_RESUME	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	–	–	–	–	–	–
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]	UART_WER [6-4]
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

(1) REGISTER_NAME(CIR) notation indicates that the register exists for other functions (IrDA or UART), but fields have different meanings for other functions (described separately in *UART Registers*).
(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the CIR function.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.8 UART Protocol Formatting

12.1.4.4.8.1 UART Mode

12.1.4.4.8.1.1 UART Clock Generation: Baud Rate Generation

The UART function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 12-86 shows the baud rate generator and associated controls.

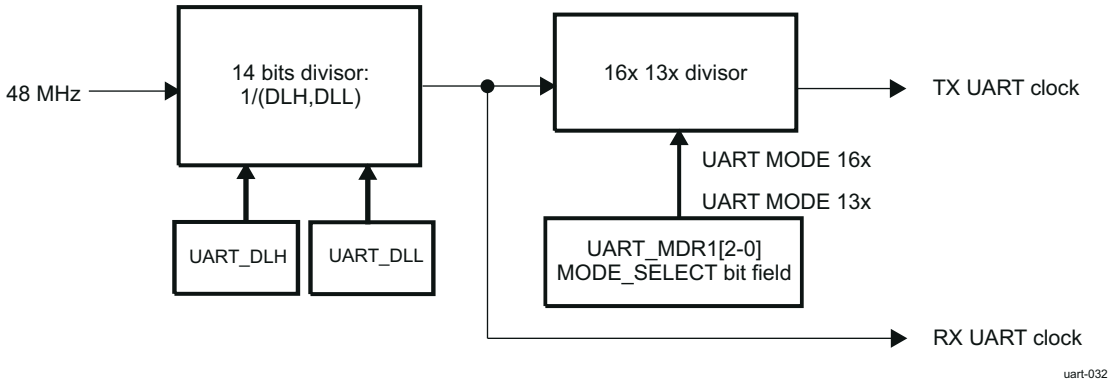


Figure 12-86. UART Baud Rate Generation

CAUTION

Before initializing or modifying clock parameter controls (UART_DLH, UART_DLL), UART_MDR1[2-0] MODE_SELECT = DISABLE must be set to 0x7. Failure to observe this rule can result in unpredictable module behavior.

12.1.4.4.8.1.2 Choosing the Appropriate Divisor Value

Two divisor values are:

- UART 16× mode: Divisor value = Operating frequency / (16× baud rate)
- UART 13× mode: Divisor value = Operating frequency / (13× baud rate)

Table 12-79 describes the UART baud rate settings.

Table 12-79. UART Baud Rate Settings (48MHz, 160MHz, 192MHz Clock)

Baud Rate	Baud Multiple	Source Clock(MHz)	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
0.3kbps	16x	48	10000	0x27, 0x10	0.3kbps	0
0.6kbps	16x	48	5000	0x13, 0x88	0.6kbps	0
1.2kbps	16x	48	2500	0x09, 0xC4	1.2kbps	0
2.4kbps	16x	48	1250	0x04, 0xE2	2.4kbps	0
4.8kbps	16x	48	625	0x02, 0x71	4.8kbps	0
9.6kbps	16x	48	312	0x01, 0x38	9.615385kbps	-0.16
9.6kbps	16x	192	1250	0x04, 0xE2	9.6kbps	0
14.4kbps	16x	48	208	0x00, 0xD0	14.42308kbps	-0.16
19.2kbps	16x	192	625	0x02, 0x71	19.2kbps	0
28.8kbps	16x	48	104	0x00, 0x68	28.84615kbps	-0.16
38.4kbps	16x	48	78	0x00, 0x4E	38.46154kbps	-0.16
57.6kbps	16x	48	52	0x00, 0x34	57.69231kbps	-0.16
115.2kbps	16x	48	26	0x00, 0x1A	115.3846kbps	-0.16
230.4kbps	16x	48	13	0x00, 0x0D	230.7692kbps	-0.16

Copyright © 2026 Texas Instruments Incorporated

Table 12-79. UART Baud Rate Settings (48MHz, 160MHz, 192MHz Clock) (continued)

Baud Rate	Baud Multiple	Source Clock(MHz)	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
460.8kbps	13x	48	8	0x00, 0x08	461.5385kbps	-0.16
921.6kbps	13x	48	4	0x00, 0x04	923.0769kbps	-0.16
1.843Mbps	13x	48	2	0x00, 0x02	1.846154Mbps	-0.17
2.5Mbps	16x	160	4	0x00, 0x04	2.5Mbps	0
3.0Mbps	16x	48	1	0x00, 0x01	3.0Mbps	0
3.6884Mbps	13x	48	1	0x00, 0x01	3.692308Mbps	-0.11
4Mbps	16x	192	3	0x00, 0x03	4Mbps	0
5Mbps	16x	160	2	0x00, 0x02	5Mbps	0
10Mbps	16x	160	1	0x00, 0x01	10Mbps	0
12Mbps	16x	192	1	0x00, 0x01	12Mbps	0

12.1.4.4.8.1.3 UART Data Formatting

The UART can use hardware flow control to manage transmission and reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals.

The UART is enhanced with the autobauding function. In control mode, autobauding lets the speed, the number of bits per character, and the parity selected be set automatically.

12.1.4.4.8.1.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the UART_LCR register.

Character length is specified using the UART_LCR[1-0] CHAR_LENGTH bit field.

The number of stop-bits is specified using the UART_LCR[2] NB_STOP bit.

The parity bit is programmed using the UART_LCR[5-3] PARITY_EN, UART_LCR[5-3] PARITY_TYPE_1, and UART_LCR[5-3] PARITY_TYPE_2 bit fields (see Table 12-80).

Table 12-80. UART Parity Bit Encoding

PARITY_EN	PARITY_TYPE_1	PARITY_TYPE_2	Parity
0	N/A	N/A	No parity
1	0	0	Odd parity
1	1	0	Even parity
1	0	1	Forced 1
1	1	1	Forced 0

12.1.4.4.8.1.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled and disabled independently by programming the UART_EFR[7] AUTO_CTS_EN and UART_EFR[6] AUTO_RTS_EN bit fields, respectively.

With auto-CTS, CTS signal must be active before the module can transmit data.

Auto-RTS activates the RTS output only when there is enough room in the RX FIFO to receive data. It deactivates the RTS output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the UART_TCR register determine the levels at which RTS is activated and deactivated.

If auto-CTS and auto-RTS are enabled, data transmission does not occur unless the RX FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If auto-CTS and auto-RTS are not enabled, overrun errors occur if the transmit data rate exceeds the RX FIFO latency.

- Auto-RTS:

Copyright © 2026 Texas Instruments Incorporated

Auto-RTS data flow control originates in the receiver block. The RX FIFO trigger levels used in auto-RTS are stored in the UART_TCR register. RTS is active if the RX FIFO level is below the HALT trigger level in the UART_TCR[3-0] RX_FIFO_TRIG_HALT bit field. When the RX FIFO HALT trigger level is reached, RTS is deasserted. The sending device (for example, another UART) can send an additional byte after the trigger level is reached because it may not recognize the deassertion of RTS until it begins sending the additional byte.

RTS is automatically reasserted when the RX FIFO reaches the RESUME trigger level programmed by the UART_TCR[7-4] RX_FIFO_TRIG_START bit field. This reassertion requests the sending device to resume transmission.

In this case, RTS is an active-low signal.

- Auto-CTS:

  The transmitter circuitry checks CTS before sending the next data byte. When CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the next byte, CTS must be deasserted before the middle of the last stop-bit currently sent.

  The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the CTS state changes do not have to trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can result.

  In this case, CTS is an active-low signal.

12.1.4.4.8.1.3.3 Software Flow Control

Software flow control is enabled through the enhanced feature register (UART_EFR) and the modem control register (UART_MCR). Different combinations of software flow control can be enabled by setting different combinations of the UART_EFR[3-0] bit field (see Table 12-81).

Two other enhanced features relate to software flow control:

- XON-any function (UART_MCR[5] XON_EN): Operation resumes after receiving any character after the XOFF character is recognized. If special character detect is enabled and special character is received after XOFF1, it does not resume transmission. The special character is stored in the RX FIFO.

  Note

  The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (UART_EFR[5] SPECIAL_CHAR_DETEC T): Incoming data is compared to XOFF2. When the special character is detected, the XOFF interrupt (UART_IIR_UART) is set, but it does not halt transmission. The XOFF interrupt is cleared by a read of UART_IIR_UART. The special character is transferred to the RX FIFO. Special character does not work with XON2, XOFF2, or sequential XOFFs.

Table 12-81. UART_EFR[3:0] Software Flow Control Options

Bit 3	Bit 2	Bit 1	Bit 0	TX, RX Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2(1)
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2(1)

(1) In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.
XON1 is defined in the UART_XON1_ADDR1[7-0] XON_WORD1 bit field. XON2 is defined in the UART_XON2_ADDR2[7-0] XON_WORD2 bit field.

Copyright © 2026 Texas Instruments Incorporated

XOFF1 is defined in the UART_XOFF1[7-0] XOFF_WORD1 bit field. XOFF2 is defined in the UART_XOFF2[7-0] XOFF_WORD2 bit field.

12.1.4.4.8.1.3.3.1 Receive (RX)

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases, XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission stops after transmission of the current character completes. Detection of XOFF also sets the UART_IIR_UART[4] bit (if enabled by UART_IER_UART[5]) and causes the interrupt line to go low.

To resume transmission, an XON1/2 character must be received (in certain cases, XON1 and XON2 must be received sequentially). When the correct XON characters are received, the UART_IIR_UART[4] bit is cleared and the XOFF interrupt disappears.

Note

When a parity, framing, or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when UART_EFR[1-0] = 0x2, if XON1 and XOFF1 characters are received, they are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (UART_EFR[1-0] = 0x3), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

12.1.4.4.8.1.3.3.2 Transmit (TX)

Two XOFF1 characters are transmitted when the RX FIFO passes the trigger level programmed by UART_TCR[3-0] RX_FIFO_TRIG_HALT. As soon as the RX FIFO reaches the trigger level programmed by UART_TCR[7-4] RX_FIFO_TRIG_START, two XON1 characters are sent, so the data transfer recovers.

Note

If software flow control is disabled after an XOFF character is sent, the module transmits XON characters automatically to enable normal transmission.

The transmission of XOFF(s)/XON(s) follows the same protocol as transmission of an ordinary byte from the TX FIFO. This means that even if the word length is 5, 6, or 7 characters, the 5, 6, or 7 LSBs of XOFF1/2 and XON1/2 are transmitted. The 5, 6, or 7 bits of a character are seldom transmitted, but this function is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

12.1.4.4.8.1.3.4 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an "at" (AT) command (ASCII code). These characteristics are used to receive data after an AT and to send data.

The following AT commands are valid:

AT	DATA	<CR>
at	DATA	<CR>
A/		
a/		

A line break during the acquisition of the sequence AT is not recognized, and an echo function is not implemented in hardware.

Copyright © 2026 Texas Instruments Incorporated

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always follows an AT, and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data, including the final <CR> (0x0D), are saved to the RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved in the RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The UART_UASR register reflects the correct settings for the baud rate detected. Interrupt activity can continue in this fashion when a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:
  - 115.2K baud
  - 57.6K baud
  - 38.4K baud
  - 28.8K baud
  - 19.2K baud
  - 14.4K baud
  - 9.6K baud
  - 4.8K baud
  - 2.4K baud
  - 1.2K baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

Note

The combination of 7-bit character plus space parity is not supported.

Autobauding mode is selected when the UART_MDR1[2-0] MODE_SELECT bit field is set to 0x2. In UART autobauding mode, UART_DLL, UART_DLH, and UART_LCR[5-0] bit field settings are not used; instead, the UART_UASR register is updated with the configuration detected by the autobauding logic.

UASR Autobauding Status Register Use

This register is used to set up transmission according to the characteristics of the previous reception instead of the UART_LCR, UART_DLL, and UART_DLH registers when the UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), the UART_MDR1[2-0] MODE_SELECT bit field must be set to reset state (0x7) and then to the UART in autobauding mode (0x2) or to the UART in standard mode (0x0).

Use limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115.2 bps (10 possibilities)

12.1.4.4.8.1.3.5 Error Detection

When the UART_LSR_UART register is read, the UART_LSR_UART[4:2] bit field reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (the next character to be read). Therefore, reading the UART_LSR_UART register and then reading the UART_RHR register identifies errors in a character.

Copyright © 2026 Texas Instruments Incorporated

Reading the UART_RHR register updates the BI, FE, and PE bits (see Table 12-65 for the UART mode interrupts).

The UART_LSR_UART[7] RX_FIFO_STS bit is set when there is an error in the RX FIFO and is cleared only when no errors remain in the RX FIFO.

Note

Reading the UART_LSR_UART register does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the UART_RHR register.

Reading the UART_LSR_UART register clears the OE bit if it is set (see Table 12-65 for the UART mode interrupts).

12.1.4.4.8.1.3.6 Overrun During Receive

Overrun during receive occurs if the RX state-machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the Host CPU with the UART_IIR_UART[5-1] IT_TYPE bit field set to 0x3 (receiver line status error) and discards the remaining portion of the frame.

Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the Host CPU must:

- Reset the RX FIFO.
- Read the UART_RESUME register, which clears the internal flag.

12.1.4.4.8.1.3.7 Time-Out and Break Conditions

12.1.4.4.8.1.3.7.1 Time-Out Counter

An RX idle condition is detected when the receiver line (RX) is high for a time that equals 4x the programmed word length + 12 bits or manually configured amount of baud clocks, if a value other zero is set in the timeout register. RX is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on RX.

There are two modes of operation:

- In default operation on the UART_EFR2[6] TIMEOUT_BEHAVE is set to 0. For the time-out interrupt, the counter counts only when there is data in the RX FIFO, and the count is reset when there is activity on RX or when the UART_RHR register is read.
- Optionally, for choose to enable the timeout counter even if no character has been received by setting UART_EFR2[6] TIMEOUT_BEHAVE bit. This will generate periodic interrupts if the RX line remains idle. In this mode the counter will auto-reset when a timeout has been reached. Reading the UART_IIR_UART will clear the interrupt, but not the counter.

12.1.4.4.8.1.3.7.2 Break Condition

When a break condition occurs, TX is pulled low. A break condition is activated by setting the UART_LCR[6] BREAK_EN bit. The break condition is not aligned on word stream (a break condition can occur in the middle of a character). The only way to send a break condition on a full character is:

1. Reset the TX FIFO (if enabled).
2. Wait for the transmit shift register to empty (the UART_LSR_UART[6] TX_SR_E bit is set to 1).
3. Take a guard time according to stop-bit definition.
4. Set the BREAK_EN bit to 1.

The break condition is asserted while the BREAK_EN bit is set to 1.

The time-out counter and break condition apply only to UART modem operation and not to IrDA/CIR mode operation.

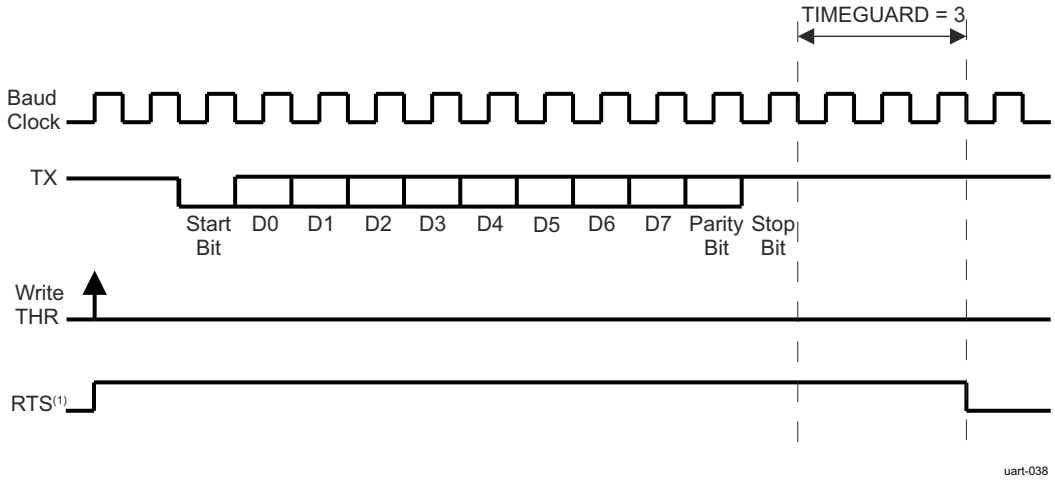
Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.8.2 RS-485 Mode

12.1.4.4.8.2.1 RS-485 External Transceiver Direction Control

The UART_MDR3[4] DIR_EN bit enables hardware control over an external transceiver to support RS-485. The direction signal comes across the DIR port. The direction polarity is controlled by the UART_MDR3[3] DIR_POL bit. The direction is determined by the hardware monitoring the TX FIFO and the TX shift register. When both are empty the transceiver is set to RX. There is a guard band delay counter of 3 bit clock cycles after the TX shift register is going empty to allow time for the stop bit to transition through the transceiver before a direction change to receive might be applied.

Figure 12-87 shows the direction control.



(1) Assumes DIR_POL = 1

Figure 12-87. RS-485 External Transceiver Direction Control

12.1.4.4.8.3 IrDA Mode

12.1.4.4.8.3.1 IrDA Clock Generation: Baud Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 12-88 shows the baud rate generator and associated controls.

Copyright © 2026 Texas Instruments Incorporated

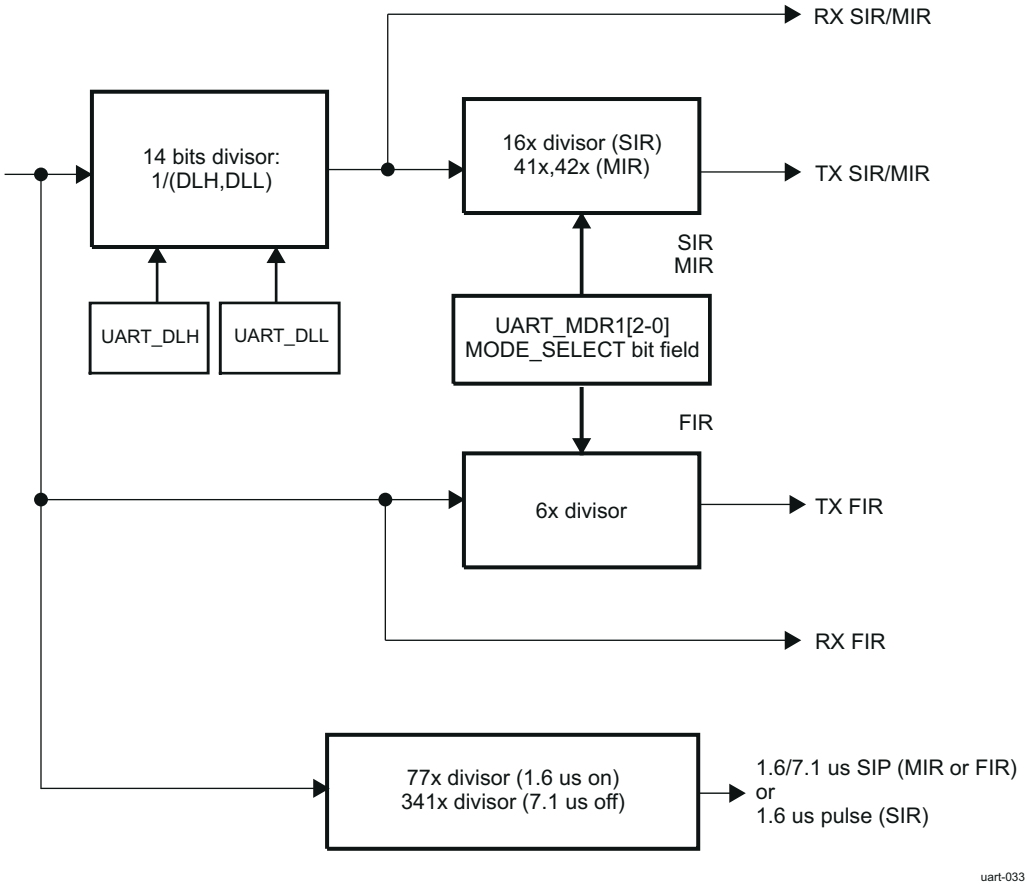


Figure 12-88. IrDA Baud Rate Generator

CAUTION

Before initializing or modifying clock parameter controls (UART_DLH, UART_DLL), MODE_SELECT=DISABLE (UART_MDR1[2-0] MODE_SELECT) must be set to 0x7). Failure to observe this rule can result in unpredictable module behavior.

12.1.4.4.8.3.2 Choosing the Appropriate Divisor Value

Three divisor values are:

- SIR mode: Divisor value = Operating frequency/(16× baud rate)
- MIR mode: Divisor value = Operating frequency/(41×/42× baud rate)
- FIR mode: Divisor value = None

Table 12-82 lists the IrDA baud rate settings.

Table 12-82. IrDA Baud Rate Settings

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)	Source Jitter (%)	Pulse Duration
2.4 kbps	SIR	16x	3/16	1250	2.4 kbps	0	0	78.1 µs
9.6 kbps	SIR	16x	3/16	312	9.6153 kbps	+0.16	0	19.5 µs
19.2 kbps	SIR	16x	3/16	156	19.231 kbps	+0.16	0	9.75 µs
38.4 kbps	SIR	16x	3/16	78	38.462 kbps	+0.16	0	4.87 µs
57.6 kbps	SIR	16x	3/16	52	57.692 kbps	+0.16	0	3.25 µs
115.2 kbps	SIR	16x	3/16	26	115.38 kbps	+0.16	0	1.62 µs

Copyright © 2026 Texas Instruments Incorporated

Table 12-82. IrDA Baud Rate Settings (continued)

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)	Source Jitter (%)	Pulse Duration
0.576 Mbps	MIR	41×/42×	1/4	2	0.5756 Mbps(1)	0	+1.63/-0.80	416 ns
1.152 Mbps	MIR	41×/42×	1/4	1	1.1511 Mbps(1)	0	+1.63/-0.80	208 ns
4 Mbps	FIR	6×	4 PPM	–	4 Mbps	0	0	125 ns

(1) Average value

Note

Baud rate error and source jitter table values do not include 48-MHz reference clock error and jitter.

12.1.4.4.8.3.3 IrDA Data Formatting

The methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

12.1.4.4.8.3.3.1 IR RX Polarity Control

The UART_MDR2[6] IRRXINVERT bit provides the flexibility to invert the RX pin in the UART to ensure that the protocol at the output of the transceiver has the same polarity at module level. By default, the RX pin is inverted because most transceivers invert the IR receive pin.

12.1.4.4.8.3.3.2 IrDA Reception Control

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Operation of the RX input can be disabled by the UART_ACREG[5] DIS_IR_RX bit.

12.1.4.4.8.3.3.3 IR Address Checking

In all IR modes, when address checking is enabled, only frames intended for the device are written to the RX FIFO. This restriction avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives, with the UART_XON1_ADDR1[7-0] XON_WORD1 and UART_XON2_ADDR2[7-0] XON_WORD2 bit fields.

Setting the UART_EFR[0] bit to 1 selects address1 checking. Setting the UART_EFR[1] bit to 1 selects address2 checking. Setting the UART_EFR[1-0] bit field to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for private and public addresses.

If address checking is disabled, all received frames write to the RX FIFO.

12.1.4.4.8.3.3.4 Frame Closing

A transmission frame can be terminated in two ways:

- Frame-length method: Set the UART_MDR1[7] FRAME_END_MODE bit to 0. The Host CPU writes the value of the frame length to the UART_TXFLH and UART_TXFLL registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the value of the frame length.
- Set-EOT bit method: Set the UART_MDR1[7] FRAME_END_MODE bit to 1. The Host CPU writes 1 to the UART_ACREG[0] EOT bit just before it writes the last byte to the TX FIFO. When the Host CPU writes the last byte to the TX FIFO, the device internally sets the tag bit for that character in the TX FIFO. As the TX state-machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and correctly terminate the frame.

12.1.4.4.8.3.3.5 Store and Controlled Transmission

In store and controlled transmission (SCT) mode, the Host CPU starts writing data to the TX FIFO. Then, after writing a part of a frame (for a bigger frame) or an entire frame (a small frame; that is, a supervisory frame), the Host CPU writes 1 to the UART_ACREG[2] SCTX_EN bit (deferred TX start) to start transmission.

Copyright © 2026 Texas Instruments Incorporated

SCT mode is enabled by setting the UART_MDR1[5] SCT bit to 1. This transmission method differs from normal mode, in which data transmission starts immediately after data is written to the TX FIFO. SCT mode is useful for sending short frames without TX underrun.

12.1.4.4.8.3.3.6 Error Detection

When the UART_LSR_UART register is read, the UART_LSR_UART[4-2] bit field reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read).

The error is triggered by an interrupt (for IrDA mode interrupts, see Table 12-66). The STATUS FIFO must be read until empty (a maximum of eight reads is required).

12.1.4.4.8.3.3.7 Underrun During Transmission

Underrun during transmission occurs when the TX FIFO is empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission.

Underrun also causes an internal flag to be set, which disables additional transmissions. Before the next frame can be transmitted, the Host CPU must:

- Reset the TX FIFO.
- Read the UART_RESUME register, which clears the internal flag.

This function can be disabled by the UART_ACREG[4] DIS_TX_UNDERRUN bit, compensated by the extension of the stop-bit in transmission if the TX FIFO is empty.

12.1.4.4.8.3.3.8 Overrun During Receive

Overrun during receive for the IrDA mode has the same function as that for the UART mode (see Section 12.1.4.4.8.1.3.6, *Overrun During Receive*).

12.1.4.4.8.3.3.9 Status FIFO

In IrDA modes, a status FIFO records the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written to the status FIFO.

Reading the UART_SFREGH[3-0] MSB and UART_SFREGL[3-0] (LSB) bit fields obtains the frame length. The frame error status is read in the UART_SFLSR register. Reading the UART_SFLSR register increments the status FIFO read pointer. Because the status FIFO is eight entries deep, it can hold the status of eight frames.

The Host CPU uses the frame-length information to locate the frame boundary in the received frame data. The Host CPU can screen bad frames using the error status information and can later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA mode because the Host CPU must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

12.1.4.4.8.3.3.10 Multi-drop Parity Mode with Address Match

Multi-drop mode is enabled in the UART_EFR2 register.

Address matching mode is only available with 8 bit character length setting. UART_LCR[1-0] CHAR_LENGTH bit fields should always be set to 0x11 (8 bits) prior to enabling the feature.

This mode allows the transmitter to send data on a line where multiple receivers are connected, when supported. In this mode, a set parity bit is used to mark an address, and a parity of 0 denotes data.

This setting affects how the parity is generated. Writing a 0x1 into the UART_ECR[0] A_MULTIDROP bit will set the parity bit for the next byte to be sent, which will then be considered an address, for sending a data frame, the UART_ECR[0] A_MULTIDROP bit has to be cleared.

On reception if the feature is enabled by setting the UART_EFR2[2] MULTIDROP bit to 0x1 incoming frames with parity set to 0x1 are treated as address frames and with parity set to 0x0 as data frames. The receiver will drop all data frames until a matching address frame was found.

Copyright © 2026 Texas Instruments Incorporated

The matching address is determined by the values set in UART_MAR, UART_MMR and UART_MBR registers and the value set in UART_EFR2[7] BROADCAST bit.

Table 12-83 summarizes the operation of address matching based on the mentioned values.

Table 12-83. Details of address matching

Received frame	Received parity	Frame type	UART_MAR	UART_MMR	UART_MBR	UART_EFR2[7] BROADCAST	Operation of receiver	Address matching
0xXX(2)	0	DATA	X(1)	X(1)	0xXX(2)	X(1)	Drops data until matching address found	N/A
0xXX(2)	1	ADDRESS	0xXX(2)	0x00	0xXX(2)	0	Matches any address	Yes
0xEF	1	ADDRESS	0xXX(2)	0xXX(2)	0xEF	1	Matches broadcast address	Yes
0x1A	1	ADDRESS	0x1A	0xFF	0xXX(2)	0	Single address match	Yes
0xF5	1	ADDRESS	0xF3	0xF9	0xXX(2)	0	Group address match	Yes

(1) X indicates a do not care bit value
(2) 0xXX indicates a do not care 8 bit hexadecimal value

The possible values for matching address can be calculated in the following way:

- Single and Group addresses can be formed by masking the UART_MAR registers value with the value set in the UART_MMR register, bits set to 0x0 in the UART_MMR register result in do not care values.
- Broadcast addresses can be set in the UART_MBR register if broadcast address is enabled in the UART_EFR2[7] BROADCAST bit, the module will match on received address frames containing the broadcast address.
- For more details, see example below:
  - UART_MAR: 0xF3, UART_MMR: 0xF9, UART_MBR: 0xFF
  - Single and Group addresses: 0xF1, 0xF3, 0xF5, 0xF7
  - Broadcast addresses: 0xFF

If an address match occurred the matching address value can be obtained from the UART_RHR register in the following way:

- If the FIFO is disabled or the threshold is set to 0x1, the matching address can be directly read from UART_RHR as the FIFO will not be overwritten.
- If the FIFO is enabled or the threshold is greater than 0x1, the matching address will be the latest frame in the FIFO with a parity error bit set.

For received data, the parity error bit in the UART_LSR_UART register is set when a bit with a parity of 0x1 is received indicating an address frame and the received address matches based on the values of UART_MAR, UART_MMR, UART_MBR and UART_EFR2[2] MULTIDROP bit.

In Multi-drop mode no parity is used, as the parity bit is used to differentiate address and data frames. The parity error bit is used for indicating an address match.

For enabling the interrupt generation for address matching UART_IER_UART[2] LINE_STS_IT bit has to be set to 0x1.

An interrupt for the matching address can be identified by reading the UART_IIR_UART[5-1] IT_TYPE bit fields, a value of 0x00011 indicates a receiver line status error. After the UART_LSR_UART[2] RX_PE bit has to be

Copyright © 2026 Texas Instruments Incorporated

read, a value of 0x1 indicates that an address match occurred. The reception of a frame is indicated with a value of 0x1 in the UART_LSR_UART[0] RX_FIFO_E bit as the matching value is written into the FIFO regardless of the frame type (data or address). UART_LSR_UART[7] RX_FIFO_STS bit will also be set to 0x1 as the parity error bit is used to indicate a matching address.

Note that the operation of the UART_LSR_UART[2] RX_PE bit depends on the value set in UART_EFR2[2] MULTIDROP bit. If UART_EFR2[2] MULTIDROP bit is set to 0x0, UART_LSR_UART[2] RX_PE bit is used to indicate a received parity error. If UART_EFR2[2] MULTIDROP bit is set to 0x1, the receiver is in Multi-drop Address Match mode, thus the value in UART_LSR_UART[2] RX_PE bit is used to indicate an address match.

The interrupt is cleared the same way in both operation modes: reading the UART_LSR_UART register updates the values.

12.1.4.4.8.3.3.11 Time-guard

The time-guard feature enables the UART interface to operate with slow remote devices.

When set, it will insert a number of idle states between transmitting two characters, the length of which can be set in the UART_TIMEGUARD register. The value in the register defines the number of baud clocks of idle period to insert.

This idle state essentially acts like a long stop bit.

12.1.4.4.8.3.4 SIR Mode Data Formatting

This section provides specific instructions for SIR mode programming.

12.1.4.4.8.3.4.1 Abort Sequence

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

A transmission frame can be aborted by setting the UART_ACREG[1] ABORT_EN bit to 1. When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

When a 0x7D character followed immediately by a 0xC1 character is received without transparency, the receiver treats a frame as an aborted frame.

CAUTION

When the TX FIFO is not empty and the UART_MDR1[5] SCT bit is set to 1, the UART IrDA starts a new transfer with data of a previous frame when the aborted frame is sent. Therefore, the TX FIFO must be reset before sending an aborted frame.

12.1.4.4.8.3.4.2 Pulse Shaping

SIR mode supports the 3/16 or the 1.6-µs pulse duration methods. The UART_ACREG[7] PULSE_TYPE bit selects the pulse width method in the transmit mode.

12.1.4.4.8.3.4.3 SIR Free Format Programming

The SIR FF mode is selected by setting the module in the UART mode (UART_MDR1[2-0] MODE_SELECT = 0x0) and the UART_MDR2[3] PULSE bit to 1 to allow pulse shaping.

Because the bit format stays the same, some UART mode configuration registers must be set at specific values:

- UART_LCR[1-0] CHAR_LENGTH bit field = 0x3 (8 data bits)
- UART_LCR[2] NB_STOP bit = 0x0 (1 stop-bit)
- UART_LCR[3] PARITY_EN bit = 0x0 (no parity)

The UART mode interrupts are used for the SIR FF mode, but many are not relevant (XOFF, RTS, CTS, modem status register, etc.).

Copyright © 2026 Texas Instruments Incorporated

12.1.4.4.8.3.5 MIR and FIR Mode Data Formatting

This section describes common instructions for FIR and MIR mode programming.

At the end of a frame reception, the CPU reads the line status register (UART_LSR_UART) to detect errors in the received frame.

When the UART_MDR1[6] SIP_MODE bit is set to 1, the TX state-machine always sends one SIP at the end of a transmission frame. However, when the SIP_MODE bit is set to 0, SIP transmission depends on the UART_ACREG[3] SEND_SIP bit.

The CPU can set the SEND_SIP bit at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not have to send the SIP at the end of each frame, thus reducing the overhead required.

12.1.4.4.8.4 CIR Mode

12.1.4.4.8.4.1 CIR Mode Clock Generation

Depending on the encoding method (variable pulse distance/biphase), the Host CPU must develop a data structure that combines 1 and 0 with a *t* period to encode the complete frame to transmit. This can then be transmitted to the infrared output with a modulation method, as shown in Figure 12-89.

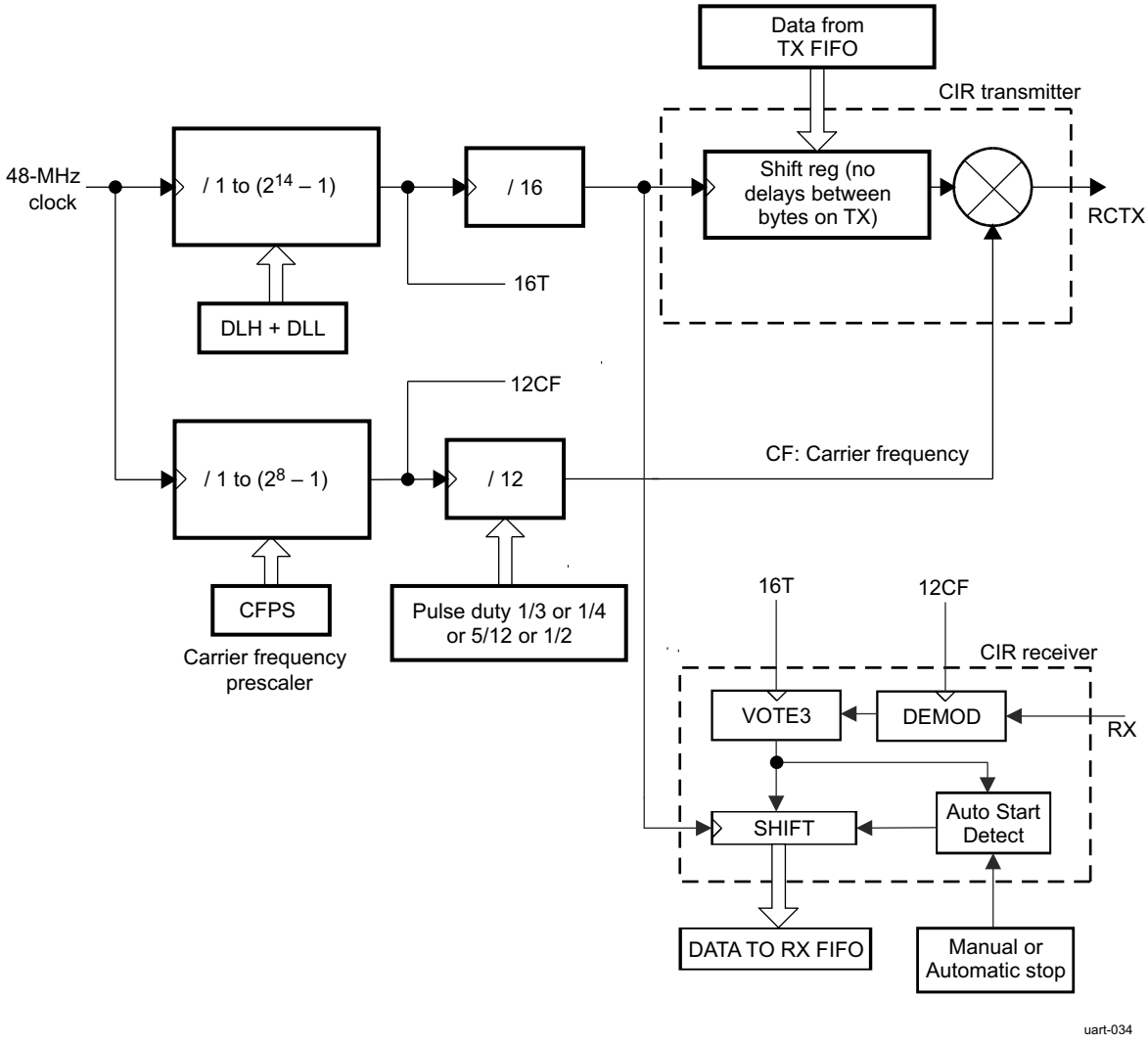


Figure 12-89. CIR Mode Block Components

Copyright © 2026 Texas Instruments Incorporated

Based on the requested modulation frequency, the UART_CFPS register must be set with the correct dividing value to provide an accurate pulse frequency:

Dividing value = (FCLK / 12) / MODfreq

Where:

FCLK = System clock frequency (48 MHz)

12 = Real value of baud multiple

MODfreq = Effective frequency of the modulation (MHz)

Example: For a targeted modulation frequency of 36 kHz, the value of CFPS must be set to 0x7 (decimal), which provides a modulation frequency of 36.04 kHz.

Note

The UART_CFPS register starts with a reset value of 105 (decimal), which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user-defined by the pulse duty register bits in the UART_MDR2 register. Table 12-84 shows the duty cycle.

Table 12-84. CIR Duty Cycle

UART_MDR2[5-4] CIR_PULSE_MODE	Duty Cycle (High-Level)
00	1/4
01	1/3
10	5/12
11	1/2

12.1.4.4.8.4.2 CIR Data Formatting

The methods described in this section apply to all CIR modes.

12.1.4.4.8.4.2.1 IR RX Polarity Control

The IR RX polarity control for CIR mode has the same function as that for IrDA mode (see Section 12.1.4.4.8.3.3.1, *IR RX Polarity Control*).

12.1.4.4.8.4.2.2 CIR Transmission

In transmission, the Host CPU software must exercise an element of real-time control to transmit data packets, each of which must be emitted at a constant delay from the start-bits of each individual packet. Thus, when sending a series of packets, the packet-to-packet delay must respect a specific delay. Two methods can be used to control this delay:

- Filling the TX FIFO with a number of zero bits that are transmitted with a *t* period
- Using an external system timer to control the delay between each start-of-frame or between the end of a frame and the start of the next one. This can be performed by:
  - Controlling the start of the frame using the UART_MDR1[5] SCT bit and the UART_ACREG[2] SCTX_EN bit, depending on the timer status
  - Using the UART_IIR_UART[5] TX_STATUS_IT interrupt bit to preload the next frame in the TX FIFO and to control the start of the timer (in case of control delay between the end of a frame and the start of the next frame)

12.1.4.4.8.4.2.3 CIR Reception

In reception, there are 2 ways to stop it

Copyright © 2026 Texas Instruments Incorporated

- The Host CPU can disable the reception by setting the UART_ACREG[5] DIS_IR_RX bit to 1. When it considers that the reception is finished because a large number of 0 has been received. To receive a new frame, the UART_ACREG[5] DIS_IR_RX bit must be set to 0.
- Using a specific mechanism, depending on the value set in the BOF length register (UART_EBLR), allows stopping automatically the reception. If the value set in the UART_EBLR register is different than 0, this features is enabled and count a number of bit received at 0. When the counter achieved the value defined in the UART_EBLR register, the reception is automatically stop and UART_IIR_CIR[2] RX_STOP_IT bit is set. When a 1 is detected on the RX pin, the reception is automatically enabled.

There is a limitation when receiving data in UART CIR mode. The IrDA transceivers on the market have a common characteristic that shrinks the hold time of the received modulation pulse. The UART filtering schema on receiving is based on the same encoding mechanism used in transmission.

For the following scenario:

- shift register period: 0.9us
- modulation frequency: 36kHz
- duty cycle: 1/4 of a modulation frequency period

So the data sent in those conditions would look like 7us pulses within 28us period. The UART expects to receive similar incoming data on receive, but available transceiver timing characteristics typically send 2us modulated pulses. Those will be filtered out and RX FIFO will not receive any data.

This does not affect UART CIR mode in transmission.

Note

The CIR RX demodulation can be bypassed by setting the UART_MDR3[0] DISABLE_CIR_RX_DEMOD bit.

Copyright © 2026 Texas Instruments Incorporated

12.1.4.5 UART Programming Guide

This section describes the procedure for operating the UART with FIFO and DMA or interrupts. This three-part procedure ensures the quick start of the UART. It does not cover every UART feature.

The first programming model covers software reset of the UART. The second programming model describes FIFO and DMA configuration. The last programming model describes protocol, baud rate, and interrupt configuration.

Note

Each programming model can be used independently of the other two; for instance, reconfiguring the FIFOs and DMA settings only.

Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see Section 12.1.4.4.7.1, *Register Access Modes*.

12.1.4.5.1 UART Global Initialization

12.1.4.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the UART module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the UART.

For more information, see , *UART Integration*.

12.1.4.5.1.2 UART Module Global Initialization

The procedure in Table 12-85 can be used to initialize UART when performing software reset.

Table 12-85. UART Global Initialization

Step	Register/Bit Field/Programming Model	Value
Perform a software reset.	UART_SYSC[1] SOFTRESET	1
Wait until reset is finished.	UART_SYSS[0] RESETDONE	=1

12.1.4.5.2 UART Mode selection

Table 12-86 describes how to set different register access mode.

Table 12-86. UART Configure Register Access Mode

Step	Register/Bit Field/Programming Model	Value
Set the register access mode A	UART_LCR[7] DIV_EN	1
	UART_LCR[7-0]	≠0xBF
Set the register access mode B	UART_LCR[7-0]	0xBF
Set the operational mode	UART_LCR[7] DIV_EN	0

12.1.4.5.3 UART Submode selection

This section describes how to set different register access submode.

Table 12-87. UART Configure Register Access Submode TCR_TLR

Step	Register/Bit Field/Programming Model	Value
Configure the submode TCR_TLR		
Configure mode B	see Table 12-86	
Enable writing to register bits UART_MCR[7-5]	UART_EFR[4] ENHANCED_EN	1
Configure mode A	see Table 12-86	0x1

Copyright © 2026 Texas Instruments Incorporated

Table 12-87. UART Configure Register Access Submode TCR_TLR (continued)

Step	Register/Bit Field/Programming Model	Value
Set the submode TCR_TLR	UART_MCR[6] TCR_TLR	1

Table 12-88. UART Configure Register Access Submode MSR_SPR

Step	Register/Bit Field/Programming Model	Value
First option: configure the submode MSR_SPR		
Configure mode B	see Table 12-86	
Set the submode MSR_SPR	UART_EFR[4] ENHANCED_EN	0
Second option: configure the submode MSR_SPR		
Configure mode B	see Table 12-86	
Enable writing to register bits UART_MCR[7-5]	UART_EFR[4] ENHANCED_EN	1
Set the submode MSR_SPR	UART_MCR[6] TCR_TLR	0

Table 12-89. UART Configure Register Access Submode XOFF

Step	Register/Bit Field/Programming Model	Value
Configure of the XOFF		
Configure B	see Table 12-86	
Set the submode XOFF	UART_EFR[4] ENHANCED_EN	0

12.1.4.5.4 UART Load FIFO trigger and DMA mode settings

12.1.4.5.4.1 DMA mode Settings

To enable and configure program the DMA mode, perform the following steps:

Table 12-90. DMA Mode Settings

Step	Register/Bit Field/Programming Model	Value
Set the option of DMA mode configuration	UART_SCR[0] DMA_MODE_CTL	-
IF Configure DMA mode 0 and 1	UART_SCR[0] DMA_MODE_CTL	=0
Select the DMA mode, for more information see Section 12.1.4.4.6.4	UART_FCR[3] DMA_MODE	-
IF Configure DMA mode from 0 to 3	UART_SCR[0] DMA_MODE_CTL	=1
Select the DMA mode, for more information see Section 12.1.4.4.6.4	UART_SCR[2-1] DMA_MODE_2	-

12.1.4.5.4.2 FIFO Trigger Settings

In this section is described configuration and settings of FIFO trigger level, which enable DMA and interrupt generation.

Table 12-91. Load FIFO Triggers Defined by the FCR

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see Table 12-87	0x-
Set the desire RX FIFO trigger level	UART_FCR[5-4] TX_FIFO_TRIG	0x-
Set the desire TX FIFO trigger level	UART_FCR[7-6] RX_FIFO_TRIG	0x-

Table 12-92. Load FIFO Triggers Defined by the TLR

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see Table 12-87	0x-
Set the desire RX FIFO trigger level	UART_TLR[7-4] RX_FIFO_TRIG_DMA	0x-
Set the desire TX FIFO trigger level	UART_TLR[3-0] TX_FIFO_TRIG_DMA	0x-

Copyright © 2026 Texas Instruments Incorporated

Table 12-93. Load FIFO Triggers Defined by the Concatenated Value

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see Table 12-87	0x-
Set the register bit	UART_SCR[7] RX_TRIG_GRANU1	1
Set the desire RX FIFO trigger level	UART_TLR[7-4] RX_FIFO_TRIG_DMA UART_FCR[7-6] RX_FIFO_TRIG	0x-
Set the register bit	UART_SCR[6] TX_TRIG_GRANU1	1
Set the desire TX FIFO trigger level	UART_TLR[3-0] TX_FIFO_TRIG_DMA UART_FCR[5-4] TX_FIFO_TRIG	0x-

12.1.4.5.5 UART Protocol, Baud rate and interrupt settings

12.1.4.5.5.1 Baud rate settings

Table 12-94. UART Baud Rate Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Switch to register configuration mode B	see Table 12-86	
Enable access to UART_IER_UART[7-4]	UART_EFR[4] ENHANCED_EN	1
Switch register operational mode	see Table 12-86	
Disable sleep mode	UART_IER_UART[4] SLEEP_MODE	0
Switch to register configuration mode A or B	see Table 12-86	
Set the appropriate divisor value	UART_DLL[7-0] CLOCK_LSB UART_DLH[5-0] CLOCK_MSB	0x-

12.1.4.5.5.2 Interrupt settings

Table 12-95. UART Interrupt Settings

Step	Register/Bit Field/Programming Model	Value
Switch to register configuration mode B	see Table 12-86	0x7
Enable access to UART_IER_UART[7-4]	UART_EFR[4] ENHANCED_EN	1
Switch register operational mode	see Table 12-86	
Set the desired interrupt configuration (0: Disable the interrupt; 1: Enable the interrupt)	UART_IER_UART[7] CTS_IT UART_IER_UART[6] RTS_IT UART_IER_UART[5] XOFF_IT UART_IER_UART[4] SLEEP_MODE UART_IER_UART[3] MODEM_STS_IT UART_IER_UART[2] LINE_STS_IT UART_IER_UART[1] THR_IT UART_IER_UART[0] RHR_IT	0x-

12.1.4.5.5.3 Protocol settings

Load the desired protocol formatting (parity, stop-bit, character length) and switch to register operational mode.

Table 12-96. UART Protocol Settings

Step	Register/Bit Field/Programming Model	Value
Load desired protocol formatting, see Section 12.1.4.4.8.1.3.1 , *Frame Formatting*	UART_LCR[5] PARITY_TYPE_2 UART_LCR[4] PARITY_TYPE_1 UART_LCR[3] PARITY_EN UART_LCR[2] NB_STOP UART_LCR[1-0] PARITY_LENGTH	0x-

Copyright © 2026 Texas Instruments Incorporated

Table 12-96. UART Protocol Settings (continued)

Step	Register/Bit Field/Programming Model	Value
Switch to register operational mode	UART_LCR[7] DIV_EN	0
	UART_LCR[6] BREAK_EN	

12.1.4.5.5.4 UART/RS-485/IrDA(SIR/MIR/FIR)/CIR

Table 12-97. UART Mode Selection

Step	Register/Bit Field/Programming Model	Value
Load the desired UART/IrDA (SIR, MIR, FIR)/CIR modes, see Section 12.1.4.4.7.2, *UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection*	UART_MDR1[2-0] MODE_SELECT	0x-
Load the desire RS-485 mode, see Section 12.1.4.4.7.2, *UART/RS-485/IrDA (SIR, MIR, FIR)/CIR Mode Selection*	UART_MDR3[4] DIR_EN	0x1

12.1.4.5.5.5 UART Multi-drop Parity Address Match Mode Configuration

Table 12-98. UART Multi-drop Parity Address Match Mode Configuration

Step	Register/Bit Field/Programming Model	Value
Disable receive mode	UART_ECR[3] RX_EN	0
Enable Multi-drop parity Address match mode	UART_EFR2[2] MULTIDROP	1
Set the matching device address	UART_MAR[7-0] ADDRESS	0x-
Set the address match masking	UART_MMR[7-0] MASK	0x-
Set the broadcast address match	UART_MBR[7-0] BROADCAST_ADDRESS	0x-
Enable broadcast address matching if needed	UART_EFR2[7] BROADCAST	1
Enable receive mode	UART_ECR[3] RX_EN	1

12.1.4.5.6 UART Hardware and Software Flow Control Configuration

This section describes the programming steps to enable and configure hardware and software flow control. Hardware and software flow control cannot be used at the same time.

12.1.4.5.6.1 Hardware Flow Control Configuration

Table 12-99. UART Hardware Flow Control Configuration

Step	Register/Bit Field/Programming Model	Value
Configure register submode TCR_TLR	see Table 12-87	0x7
Load the start and halt trigger value.	UART_TCR[7-4] AUTO_RTS_START	0x-
	UART_TCR[3-0] AUTO_RTS_HALT	
Enable or disable receive and transmit hardware flow control mode.	UART_EFR[7] AUTO_CTS_EN	0x-
	UART_EFR[6] AUTO_RTS_EN	

12.1.4.5.6.2 Software Flow Control Configuration

Table 12-100. UART Software Flow Control Configuration

Step	Register/Bit Field/Programming Model	Value
Set the register access submode XOFF	see Table 12-89	
Load the software control characters	UART_XON1_ADDR1[7-0] XON_WORD1	0x-
	UART_XON2_ADDR2[7-0] XON_WORD2	
	UART_XOFF1[7-0] XOFF_WORD1	
	UART_XOFF2[7-0] XOFF_WORD2	
Set the register access submode TCR_TLR	see Table 12-87	
Enable or disable XON any function (0: Disable; 1: Enable).	UART_MCR[5] XON_EN	--

Copyright © 2026 Texas Instruments Incorporated

Table 12-100. UART Software Flow Control Configuration (continued)

Step	Register/Bit Field/Programming Model	Value
Load start and halt trigger value for software flow control	UART_TCR[7-4] AUTO_RTS_START	0x-
	UART_TCR[3-0] AUTO_RTS_HALT	
Enable or disable special character function (0: Disable; 1: Enable)	UART_EFR[5] SPEC_CHAR	0x-
Set the software flow control mode	UART_EFR[3-0] SW_FLOW_CONTROL	0x-

12.1.4.5.7 IrDA Programming Model

12.1.4.5.7.1 SIR mode

12.1.4.5.7.1.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with parity forced to 1, baud rate = 115.2 kbps, FIFOs disabled, 2 stop-bits, and 8-bit word length:

Table 12-101. SIR Mode Receive Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate(115.2 Kbps)	UART_DLL[7-0] CLOCK_LSB	0x1A
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

12.1.4.5.7.1.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 6-byte frame with no parity, baud rate = 115.2 kbps, FIFOs disabled, 3/16 encoding, 2 stop-bits, and 7-bit word length:

Table 12-102. SIR Mode Transmit Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (115.2 Kbps)	UART_DLL[7-0] CLOCK_LSB	0x1A
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	0x1
Set transmit frame length to 6 bytes	UART_TXFLL[7-0] TXFLL	0x06
Set the seven starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
Set SIR pulse width to be 1.6 µs	UART_ACREG[7] PULSE_TYPE	1

12.1.4.5.7.2 MIR mode

12.1.4.5.7.2.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with no parity, baud rate = 1.152 Mpbs, and FIFOs disabled.

Copyright © 2026 Texas Instruments Incorporated

Table 12-103. MIR Mode Receive Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (1.152 bps)	UART_DLL[7-0] CLOCK_LSB	0x01
	UART_DLH[5-0] CLOCK_MSB	0x00
Set MIR mode	UART_MDR1[2-0] MODE_SELECT	0x4
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Force outputs DTR and RTS to active	UART_MCR[1-0]	0x3
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

12.1.4.5.7.2.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 60-byte frame with no parity, baud rate = 1.152 Mpbs, and FIFOs disabled.

Table 12-104. MIR Mode Transmit Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Load the baud rate (115.2 kbps)	UART_DLL[7-0] CLOCK_LSB	0x01
	UART_DLH[5-0] CLOCK_MSB	0x00
Set SIR mode	UART_MDR1[2-0] MODE_SELECT	0x4
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	0x1
Set transmit frame length to 60 bytes	UART_TXFLL[7-0] TXFLL	0x3C
Set the eight additional starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
SIP is sent at the end of transmission	UART_ACREG[3] SEND_SIP	1

12.1.4.5.7.3 FIR mode

12.1.4.5.7.3.1 Receive

The following programming model explains how to program the module to receive the IrDA frame with no parity, baud rate = 4 Mbps, FIFOs enabled, 8-bit word length.

Table 12-105. FIR Mode Receive Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Enable access to change UART_FCR[0]	UART_DLL[7-0] CLOCK_LSB	0x0
	UART_DLH[7-0] CLOCK_MSB	
FIFO clear and enable	UART_FCR[2-0]	0x7
Set the FIFO trigger level	see Section 12.1.4.5.4, *Load FIFO trigger and DMA mode settings*	
Set FIR mode	UART_MDR1[2-0] MODE_SELECT	0x5
Set frame length	UART_RXFLL[7-0] RXFLL	0xA
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00

Copyright © 2026 Texas Instruments Incorporated

Table 12-105. FIR Mode Receive Settings (continued)

Step	Register/Bit Field/Programming Model	Value
Enable the UART_RHR interrupt	UART_IER_IRDA[0] RHR_IT	1

12.1.4.5.7.3.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 4-byte frame with no parity, baud rate = 4 Mbps, FIFOs enabled, and 8-bit word length.

Table 12-106. FIR Mode Transmit Settings

Step	Register/Bit Field/Programming Model	Value
Disable UART mode	UART_MDR1[2-0] MODE_SELECT	0x7
Grant access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x80
Enable access to change UART_FCR[0]	UART_DLL[7-0] CLOCK_LSB	0x0
	UART_DLH[5-0] CLOCK_MSB	
FIFO clear and enable	UART_FCR[2-0]	0x7
Set the FIFO trigger level	see Section 12.1.4.5.4, *Load FIFO trigger and DMA mode settings*	
Set FIR mode	UART_MDR1[2-0] MODE_SELECT	0x1
Disable access to the UART_DLL and UART_DLH registers	UART_LCR[7-0]	0x00
Set FIR mode and enable auto-SIP mode	UART_MDR1[7-0]	0x45
Set frame length	UART_TXFLL[7-0] TXFLL	0x4
	UART_TXFLH[7-0] TXFLH	0x0
Force output DTR to active	UART_MCR[0] DTR	1
Enable the UART_THR interrupt	UART_IER_IRDA[1] THR_IT	1
Set the eight additional starts of frame transmission	UART_EBLR[7-0] EBLR	0x08
SIP is sent at the end of transmission	UART_ACREG[3] SEND_SIP	1

12.2 High-speed Serial Interfaces

Copyright © 2026 Texas Instruments Incorporated

12.2.1 Gigabit Ethernet Switch (CPSW3G)

This chapter describes the Gigabit Ethernet Switch (CPSW3G) subsystem in the device.

12.2.1.1 CPSW0 Overview

The 3-port Gigabit Ethernet Switch (CPSW0) subsystem provides Ethernet packet communication for the device and can be configured as an Ethernet switch.

Figure 12-90 shows the CPSW0 module overview.

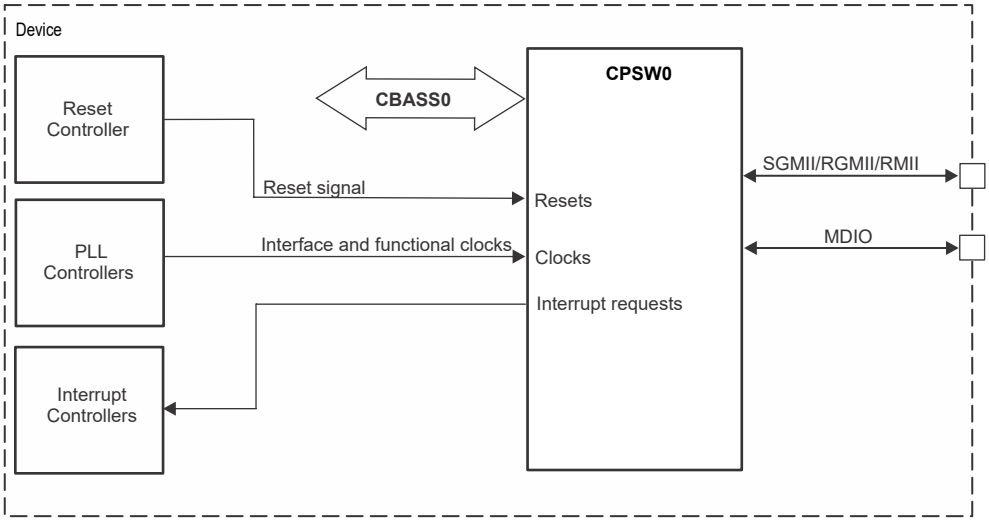


Figure 12-90. CPSW0 Overview

12.2.1.1.1 CPSW0 Features

The 3-port CPSW0 subsystem provides the following features:

- Two Ethernet ports (port 1 and 2) with selectable SGMII, RGMII and RMII interfaces and an internal Communications Port Programming Interface (CPPI) port (port 0)
- Synchronous 10/100/1000 Mbit operation
- Flexible logical FIFO-based packet buffer structure
- Cut through switch support
- Eight priority level Quality Of Service (QOS) support (802.1p)
- Support for Audio/Video Bridging (P802.1Qav/D6.0)
- Support for IEEE 1588 Clock Synchronization (2008 Annex D, Annex E and Annex F)
  - Timestamp module capable of time stamping external timesync events like Pulse-Per-Second and also generating Pulse-Per-Second outputs
  - CPTS module that supports time stamping for IEEE1588 with support for 4 hardware push events and generation of compare output pulses
- DSCP Priority Mapping (IPv4 and IPv6)
- Energy Efficient Ethernet (EEE) support (802.3az)
- Flow Control Support (802.3x)
- Wire rate switching (802.1d)
- Non-Blocking switch fabric
- Time Sensitive Network Support
  - IEEE P802.3br Interspersing Express Traffic
  - IEEE 802.1Qbv Enhancements for Scheduled Traffic
- Address Lookup Engine (ALE)
  - 512 ALE table entries
  - Configurable number of addresses plus VLANs

Copyright © 2026 Texas Instruments Incorporated

  - Wire rate lookup
  - Host controlled time-based aging and/or auto-aging
  - Spanning tree support
  - L2 address lock and L2 filtering support
  - MAC authentication (802.1x)
  - Receive-based or destination-based Multicast and Broadcast rate limits
  - MAC address blocking
  - Source port locking
  - OUI (Vendor ID) host accept/deny feature
  - Configurable number of classifier/policers (32)
  - VLAN support
    - 802.1Q compliant
      - Auto add port VLAN for untagged frames on ingress
      - Auto VLAN removal on egress and auto pad to minimum frame size
- EtherStats and 802.3Stats Remote Network Monitoring (RMON) statistics gathering (per port statistics)
- Ethernet Mac transmit to Ethernet Mac receive Loopback mode (digital loopback) supported
- CPSGMII Loopback Modes (transmit to receive)
- Maximum frame size of 2024 bytes
- Management Data Input/Output (MDIO) module for PHY Management with Clause 45 support
- Programmable interrupt control with selected interrupt pacing
- Host port CPPI Streaming Packet Interface (CPPI_GCLK)
- Digital loopback and FIFO loopback modes supported
- Emulation support
- Full duplex mode supported in 10/100/1000 Mbps. Half-duplex mode supported only in 10/100 Mbps modes only.
- RAM Error Detection and Correction (SECDED)

12.2.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.2.1.1.3 CPSW Terminology

Copyright © 2026 Texas Instruments Incorporated

Terminology:

AVB	Audio Video Bridging
AVBTP	Audio Video Bridging Trasport Protocol
BCCA	Best Controller Clock Algorithm
CFI	Canonical Format Indicator
CPPI	Communications Port Programming Interface
CPSW	Common Platform Switch
DLR	Device Level Ring
DSCP	Differentiated Services Code Point
EEE	Energy Efficient Ethernet
EMAC	Ethernet Media Access Control
EOP	End of Packet
EOQ	End of Queue
IPG	Inter-Packet Gap
LPI	Low Power Indicator
MDIO	Management Data Input/Output
MOF	Middle of Frame
OUI	Organizationally Unique Identifier
PFC	Priority based Flow Control
PTP	Precision Time Protocol
RMON	Remote Monitoring
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SCR	Switched Central Resource
SRP	Stream Reservation Protocol
TOS	Type of Service
VLAN	Virtual Local Area Network

Copyright © 2026 Texas Instruments Incorporated

12.2.1.2 CPSW0 Environment

This section describes the CPSW0 external connections (environment).

12.2.1.2.1 CPSW0 RMII Interface

Figure 12-91 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system. The individual CPSW0 and MDIO signals for the RMII interface are summarized in Table 12-107.

The CPSW0 module integrated in the device supports internal and external clock sources in RMII mode. Figure 12-91 shows the internal clock source for RMII_MHZ_50_CLK clock. It is 50 MHz clock source that is provided on the CLKOUT0 device pin. This clock has to be routed on the PCB to the RMII_REF_CLK device pin and the external PHY, RMII clock input.

For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

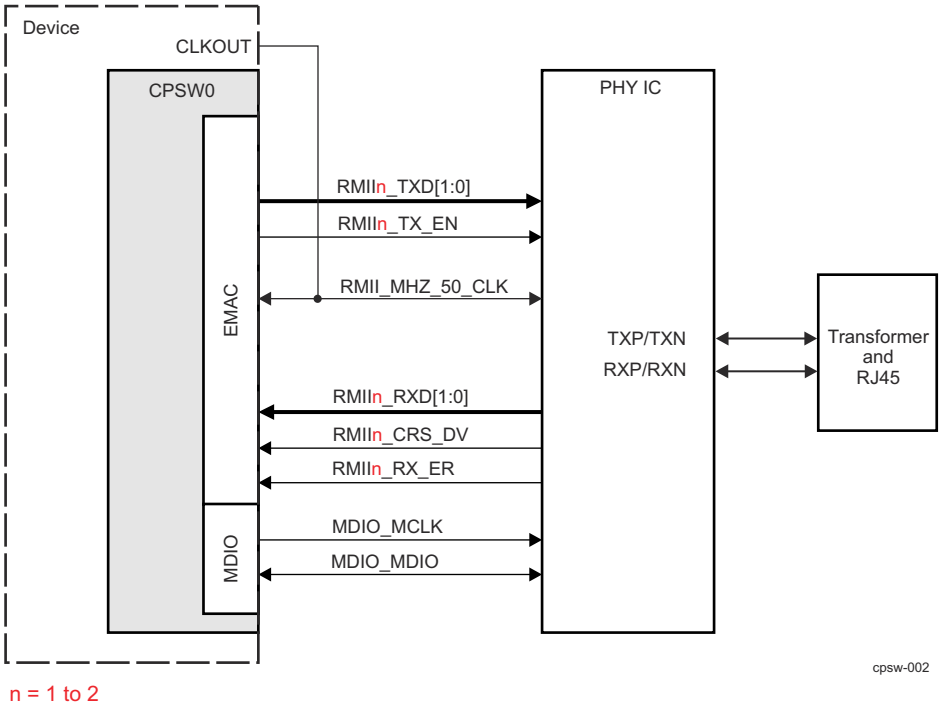


Figure 12-91. RMII Interface Typical Application (Internal Clock Source)

Figure 12-92 shows the external clock source for RMII_MHZ_50_CLK clock. In this case a 50 MHz clock is available on the PCB and it can be sourced from an oscillator or from the Ethernet PHY. This externally generated clock should be routed to both RMII_REF_CLK device pin and the external PHY, RMII clock input.

Copyright © 2026 Texas Instruments Incorporated

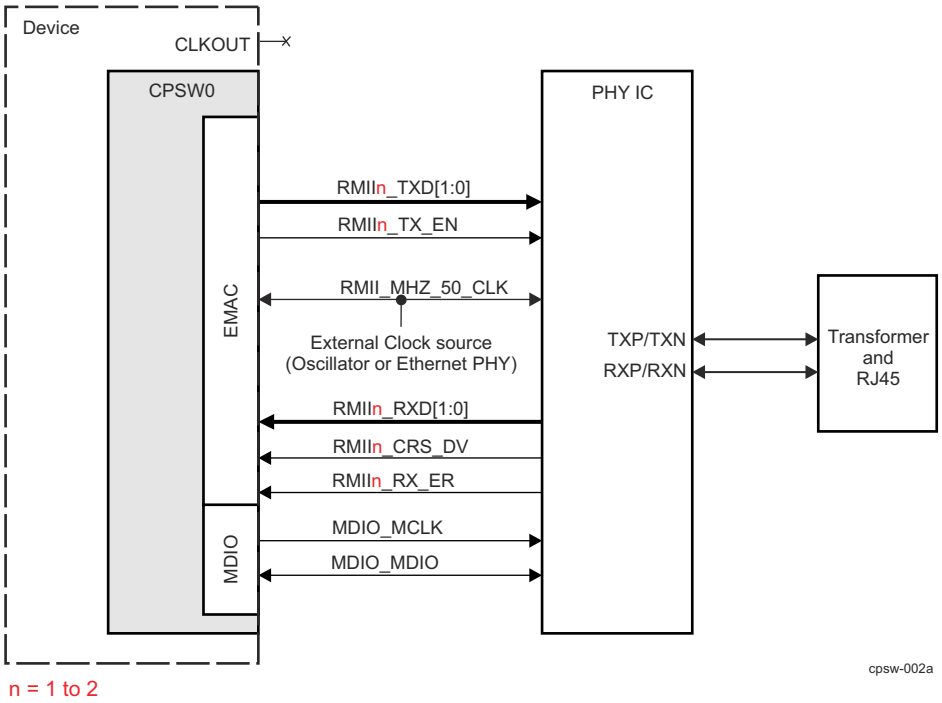


Figure 12-92. RMII Interface Typical Application (External Clock Source)

Table 12-107. RMII I/O Description

Signal[2]	Device Pin(s)	I/O[1]	Description
RMIIn_TXD[1:0]	RMIIn_TXD[1:0]	O	Transmit data. The transmit data pins are a collection of 2 bits of data. TXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMIIn_TX_EN is asserted.
RMIIn_TX_EN	RMIIn_TX_EN	O	RMII transmit enable. The transmit enable signal indicates that the RMIIn_TXD[1:0] pins are generating data for use by the PHY. RMIIn_TX_EN is synchronous to RMII_MHZ_50_CLK.
RMII_MHZ_50_CLK	RMII_REF_CLK	I	RMII 50MHz reference clock. The reference clock is used to synchronize all RMII signals. RMII_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMIIn_RXD[1:0]	RMIIn_RXD[1:0]	I	Receive data. The receive data pins are a collection of 2 bits of data. RXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMIIn_CRS_DV is asserted and RMIIn_RX_ER is de-asserted.
RMIIn_CRS_DV	RMIIn_CRS_DV	I	Carrier sense/receive data valid. Multiplexed signal between carrier sense and receive data valid.
RMIIn_RX_ER	RMIIn_RX_ER	I	Receive error. The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_MCLK	MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO0_MDIO data pin.
MDIO_MDIO	MDIO0_MDIO	I/O	MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output
(2) n 1 to 2

Copyright © 2026 Texas Instruments Incorporated

12.2.1.2.2 CPSW0 RGMII Interface

Figure 12-93 shows a device with integrated EMAC and MDIO interfaced via a RGMII connection in a typical system. The individual CPSW0 and MDIO signals for the RGMII interface are summarized in Table 12-108.

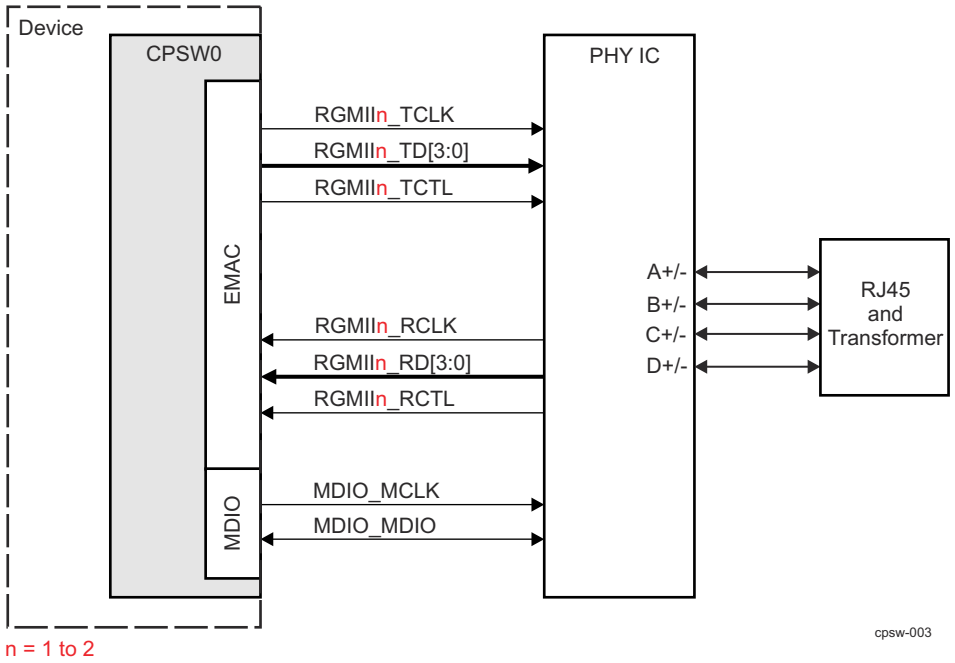


Figure 12-93. RGMII Interface Typical Application

Table 12-108. RGMII I/O Description

Signal[2]	Device Pin(s)	I/O[1]	Description
RGMIIn_TD[3:0]	RGMIIn_TD[3:0]	O	The transmit data pins are a collection of 4 bits of data. TD0 is the least-significant bit (LSB). The signals are valid only when RGMIIn_TCTL is asserted.
RGMIIn_TCTL	RGMIIn_TX_CTL	O	Transmit Control/enable. The transmit enable signal indicates that the TD pins are generating data for use by the PHY.
RGMIIn_TCLK	RGMIIn_TXC	O	The transmit reference clock. The clock is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, and 125 MHz at 1000 Mbps of operation.
RGMIIn_RD[3:0]	RGMIIn_RD[3:0]	I	The receive data pins are a collection of 4 bits of data. RD0 is the least-significant bit (LSB). The signals are valid only when RGMIIn_RX_CTL is asserted
RGMIIn_RCTL	RGMIIn_RX_CTL	I	The receive data valid/control signal indicates that the RD pins are nibble data for use by the EMAC.
RGMIIn_RCLK	RGMIIn_RXC	I	The receive clock is a continuous clock that provides the timing reference for receive operations. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation, 25 MHz at 100 Mbps operation, 125 MHz at 1000 Mbps of operation.
MDIO_MCLK	MDIO0_MDC	O	Management data clock (MDIO_MCLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO0_MDIO pin.
MDIO_MDIO	MDIO0_MDIO	I/O	The MDIO0_MDIO pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO0_MDIO pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

(1) I = Input; O = Output
(2) n 1 to 2

Copyright © 2026 Texas Instruments Incorporated

Note

The Control Module registers assign the specific function to the device pads. For more information on Control Module settings, see *Pad Configuration Registers* in *Control Module (CTRL_MMR)* and the device-specific Datasheet.

12.2.1.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4 CPSW0 Functional Description

The three-port switch Ethernet subsystem module (CPSW) is compliant to the IEEE Std 802.3 Specification. CPSW top level functional block diagram is shown in *CPSW Top Level Block Diagram*.

12.2.1.4.1 Functional Block Diagram

The three-port Ethernet subsystem consists of:

- CPSW_3G
- One RGMIIn (where n = 1 to 2) interface module
- One RMIIn (where n = 1 to 2) interface module
- One Host Port 0 CPPI Packet Streaming Interface
- CPSW subsystem control registers (REG)
- One MDIO interface module
- One Interrupt Controller module

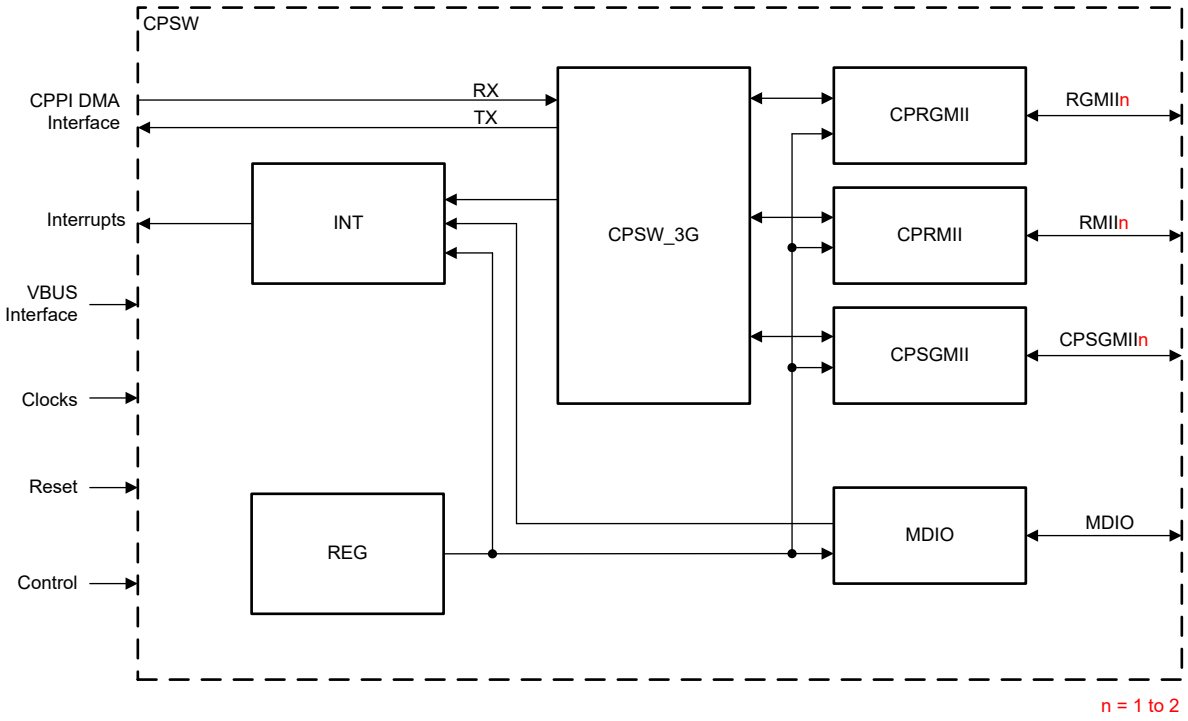


Figure 12-94. CPSW Top Level Block Diagram

12.2.1.4.2 CPSW Ports

The Ethernet Subsystem has three ports. Port 0 is the Host port (internal to the Subsystem). Port 1 and 2 are the external ports connected to SGMII, RGMII and RMII interfaces as per the interface selected.

Naming conventions followed in this chapter:

- Port0 is referred to the CPPI Host Port
- Port1 and 2 are referred to the interfaces SGMII/RGMII/RMII

12.2.1.4.2.1 Interface Mode Selection

The three-port switch (CPSW) Ethernet Subsystem has one 10/100/1000 Ethernet port with selectable SGMII, RMII and RGMII interfaces.

The interface modes for all 2 Ethernet ports are selected by configuring the Ethernet interface mode selection bitfield (PORT_MODE_SEL) in the ENET1_CTRL and ENET2_CTRL registers.

See the device-specific Datasheet for configuring the pin mux mode as per the interface selected.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.3 Clocking

12.2.1.4.3.1 Subsystem Clocking

CPSW clocking summary is shown in *CPSW Integration*.

12.2.1.4.3.2 Interface Clocking

Data is transmitted and received with respect to the reference clocks of the interface pins.

12.2.1.4.3.2.1 RGMII Interface Clocking

RGMII_RXC, RGMII_TXC frequencies are:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps
- 125 MHz at 1000 Mbps

12.2.1.4.3.2.2 RMII Interface Clocking

RMII interface clock RMII_50MHZ_CLK frequency is:

- 50 MHz at 10 Mbps
- 50 MHz at 100 Mbps

For more details on RMII clocking, please see *CPSW0 Integration*

CLKOUT_CTRL bits are used to enable and select the clock source for CLKOUT device pin.

12.2.1.4.3.2.3 MDIO Clocking

The MDIO clock is based on a divide-down of the interface (CPPI_ICLK) clock. The application software or driver must control the divide-down value.

See the CPSW3_MDIO_CONTROL_REG register for configuring the Clock Divider ([15-0]CLKDIV) value.

12.2.1.4.4 Software IDLE

The submodule software idle register bits enable CPSW operation to be completely or partially suspended by software control. There are two CPSW submodules that contain software idle register bits. Each of the two submodules may be individually commanded to enter the idle state. The idle state is entered at packet boundaries, and no further packet operations will occur on an idled submodule until the idle command is removed. The CPSW module enters the idle state when all two submodules are commanded to enter and have entered the idle state. Idle status is determined by reading or polling the two submodule idle bits. The CPSW_3G is in the idle state when all two submodules are in the idle state. The CPSW3_CPSW_NU_SOFT_IDLE_REG[0] SOFT_IDLE bit may be set if desired after the submodules are in the idle state. The SOFT_IDLE bit causes packets to not be transferred from one FIFO to another FIFO internal to the switch.

12.2.1.4.5 Interrupt Functionality

CPSW Ethernet Subsystem has six interrupt outputs:

- Cn_FH_PEND_INTR - FHost (from host to Ethernet) level interrupt
- Cn_TH_PEND_INTR - THost (from Ethernet to host) level interrupt
- Cn_TH_THRESH_PEND_INTR - THost (from Ethernet to host) non-paced level interrupt
- Cn_MISC_PEND_INTR - Miscellaneous non-paced pending interrupt
- ECC_SEC_PEND_INTR: ECC SEC pending interrupt - from CPSW ECC module. This interrupt is also included in the C0_MISC_PEND_INTR if enabled or can be used separately if desired.
- ECC_DED_PEND_INTR: ECC DED pending interrupt - from CPSW ECC module. This interrupt is also included in the C0_MISC_PEND_INTR if enabled or can be used separately.

Note

n = 0 to $num_cores-1

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.5.1 EVNT_PEND Interrupt

See Section 12.2.1.4.7, *Common Platform Time Sync (CPTS)* for more details on this interrupt.

12.2.1.4.5.2 Statistics Interrupt (STAT_PEND0)

The statistics level interrupt (STAT_PEND0) will be asserted, if enabled when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is cleared by writing to decrement any statistics values greater than 8000 0000h (such that their new values are less than 8000 0000h). The statistics are mapped into internal memory space and are 32-bits wide. The raw and masked statistics interrupt status may be read by reading the CPSW3_CPSW_NU_CPTS_INTSTAT_RAW_REG and CPSW3_CPSW_NU_CPTS_INTSTAT_MASKED_REG registers, respectively.

The statistics interrupt is enabled by setting to 1h the TS_PEND_EN bit in the CPSW3_CPSW_NU_CPTS_INT_ENABLE_REG register.

12.2.1.4.5.3 ECC DED Level Interrupt (ECC_DED_INT)

Level interrupt (ECC_DED_INT) indicating an ECC double error has been detected.

12.2.1.4.5.4 ECC SEC Level Interrupt (ECC_SEC_INT)

Level interrupt (ECC_SEC_INT) indicating an ECC single error has been detected and corrected.

12.2.1.4.5.5 MDIO Interrupts

Normal Mode: (CPSW3_MDIO_POLL_REG[30] STATECHANGEMODE = 0h)

MDIO_LINKINT event is set if there is a change in the link state of the PHY corresponding to the address in the PHYADR_MON field of the CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k register and the corresponding LINKINT_ENABLE bit is set. The MDIO_LINKINT event is also captured in the MDIO CPSW3_MDIO_LINK_INT_MASKED_REG register. When the GO bit in the CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register transitions from 1 to 0, indicating the completion of a user access, and the corresponding USERINTMASKSET bit in the CPSW3_MDIO_USER_INT_MASK_SET_REG register is set, the MDIO_USERINT signal is asserted. The MDIO_USERINT event is also captured in the CPSW3_MDIO_USER_INT_MASKED_REG register.

State Change Mode: (CPSW3_MDIO_POLL_REG[30] STATECHANGEMODE = 1h)

In State Change Mode, the MDIO will assert MDIO_LINKINT[0] when any bit in the MDIO CPSW3_MDIO_ALIVE_REG or CPSW3_MDIO_LINK_REG registers changes due to MDIO operations. The MDIO_LINKINT event is also captured in the CPSW3_MDIO_LINK_INT_MASKED_REG register. MDIO_LINKINT[1] output is unused in State Change Mode. The CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k registers are unused in state change mode.

Manual Mode: (CPSW3_MDIO_POLL_REG[31] MANUALMODE = 1h)

Manual Mode allows software to directly control the MDIO serial clock output (CPSW3_MDIO_MANUAL_IF_REG[2] MDIO_MDCLK_O) and the MDIO serial data output enable (CPSW3_MDIO_MANUAL_IF_REG[1] MDIO_OE). Manual Interface Mode is enabled when the MANUALMODE bit is set in the CPSW3_MDIO_POLL_REG register. Manual Mode is intended to be used by software for slow speed general purpose IO operations and not for MDIO PHY operations.

12.2.1.4.6 CPSW_3G

The CPSW_3G RMII/ RGMII interface is compliant to the IEEE Std 802.3 Specification.

The CPSW_3G contains two Ethernet port interfaces (Ethernet port 1 and 2), one CPPI packet streaming interface host port (port 0), Common Platform Time Sync (CPTS), ALE Engine and Statistics (STATS). A top-level block diagram of the CPSW_3G is shown in Figure 12-95.

Copyright © 2026 Texas Instruments Incorporated

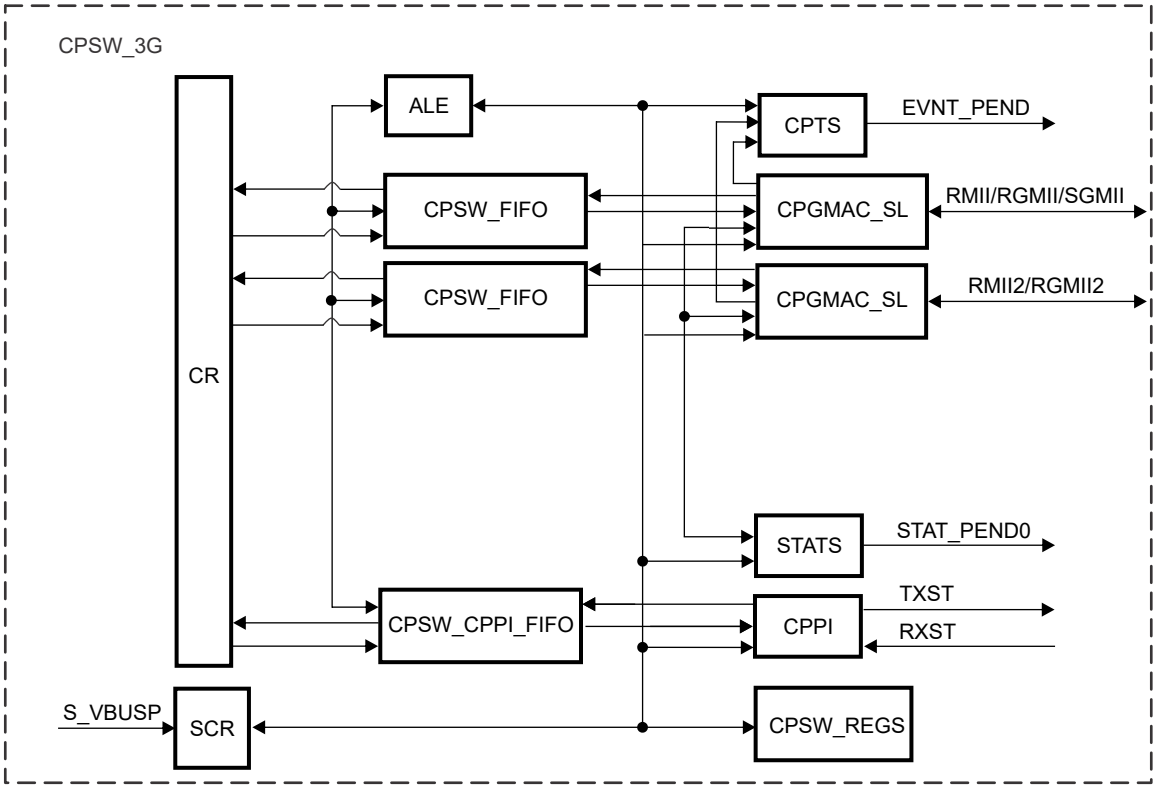


Figure 12-95. CPSW_3G Block Diagram

12.2.1.4.6.1 Address Lookup Engine (ALE)

The Address Lookup Engine (ALE) is a sub-block of the CPSW Switch and it processes all received packets and determines to which port(s) the packet should be forwarded. The ALE uses the incoming packet received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded. The ALE outputs the port mask to the switch fabric that indicates the port(s) the packet should be forwarded to. The ALE is enabled when the ENABLE_ALE bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register is set. All packets are dropped when the ENABLE bit is cleared to 0.

12.2.1.4.6.1.1 Error Handling

In normal operation, the Ethernet port modules are configured to issue an abort, instead of an end of packet, at the end of a packet that contains an error (runt, frag, oversize, jabber, crc, alignment, code etc.) or at the end of a MAC control packet. However, when the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG configuration bit(s) RX_CEF_EN, RX_CSF_EN, or RX_CMF_EN are set, error frames, short frames or MAC control frames have a normal end of packet instead of an abort at the end of the packet. When the ALE receives a packet that contains errors (due to a set header error bit), or a MAC control frame and does not receive an abort, the packet will be forwarded only to the host port (port 0). Packets with errors that are forwarded to the host have no VLAN untagging or drop due to rate limiting. No ALE learning occurs on packets with errors or mac control frames. Learning is based on source address and lookup is based on destination address. Directed packets from the host are not learned, updated, or touched.

12.2.1.4.6.1.2 Bypass Operations

The ALE may be configured to operate in bypass mode by setting the ENABLE_BYPASS bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register. When in bypass mode, all Ethernet port received packets

Copyright © 2026 Texas Instruments Incorporated

are forwarded only to the host port (port 0). In bypass mode, the ALE processes host port transmit packets the same as in normal mode. In general, packets would be directed by the host in bypass mode.

12.2.1.4.6.1.3 OUI Deny or Accept

The ALE may be configured to operate in OUI deny mode by setting the ENABLE_OUI_DENY bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register. When in OUI deny mode, a packet with a non-matching OUI source address will be dropped unless the destination address matches a multicast table entry with the SUPER bit set. When ENABLE_OUI_DENY bit is cleared, any packet source address matching an OUI address table entry will be dropped to the host unless the destination address matches with a supervisory address table entry. Broadcast packets will be dropped unless the broadcast address is entered into the table with the SUPER bit set. Unicast packets will be dropped unless the unicast address is in the table with BLOCK and SECURE both set (supervisory unicast packet).

12.2.1.4.6.1.4 Statistics Counting

ALE sends many statistics along with the frame routing so the CPSW can count them on a per port basis. There are many reasons to drop frames the below drop events are individually counted in CPSW per port statistics counters.

Table 12-109. Learned Address Control Bits

Abbreviation	Description
vddc	Dual VLAN or Double VLAN drop counter.
frdc	IPv4 fragmented drop counter.
nldc	IPv4 Protocol or IPv6 Next Header drops due to next header limit drop counter.
etdc	Ether Type field is <= 1500 and the frame size is too small drop counter.
smdc	The Source Address has bit 40 set drop counter.
badc	The Source or Destination Address blocked drop counter.
svdc	The Source Address is locked to a different port drop counter.
des	The Source and Destination are equal and the Source Address is not in the table drop counter.
adc	The Source Address is not in the table in authentication mode drop counter.
vicdc	The VLAN ingress check failed drop counter.
rldc	Rate Limit Drop Counter.
over	Lookup Overflow, the rate of the ALE clock is not sufficient to handle the incoming frame rate (Should never happen).
drop	Packet is dropped and not forwarded.

12.2.1.4.6.1.5 Automotive Security Features

The ALE has many automotive security features that most enterprise switches do not require.

- VLANs can be configured to not allow fragmented IPv4 frames. That is a VLAN can be configured to not allow fragmented IPv4 traffic.
- VLANs can be configured to only allow up to four different IPv4 Protocols or IPv6. Next Header values, for example a VLAN can be configured to only allow TCP traffic in both IPv4 and IPv6 packets.
- Drop invalid Source Addresses, that is drop Source Addresses with bit 40 set (Multicast/Broadcast indicator on Destination Addresses)

Copyright © 2026 Texas Instruments Incorporated

- IEEE802.3 Length Check, drop frames that the IEEE802.3 Length is not contained within the frame. (Ether Types 0-1500)
- Any Source Address can be secured to a port dropping any attempts from other ports to masquerade as a service.
- Any source or destination address can be blocked.
- Per Port or Per VLAN ingress checking, dropping traffic from non-member ports.
- Classification, Policing on L2 and L3 information.

12.2.1.4.6.1.6 CPSW Switching Solutions

The host port can operate in many different modes as well depending on the functionality of the host. It is important to understand the modes and configure them properly.

The ALE Table is designed to maximize the modes without compromise of the system functionality as well.

12.2.1.4.6.1.6.1 Basics of 3-port Switch Type

The 3-port switch has a host port, and that port can operate in two fundamental modes. Bridge mode allows the host to extent the switched domain to another network like Wi-Fi or another multi-port switch. In this case the host must be able to see unknown unicast addresses so they can be broadcast to the other network. In Port mode the host need not see any unknown unicast traffic. The CPSW3_CPSW_NU_ALE_ALE_CONTROL[8] EN_HOST_UNI_FLOOD bit determines the host mode for unknown unicast traffic. This bit should only be set if the user is bridging two or more networks together.

The 3-port switch can operate like a two-port switch using the ALE table just for the host info, the only adder is now other VLANs need to be supported for transit traffic between the external ports. This can easily be done using the default VLAN rules so no ALE table entries are used.

The 3-port switch can also operate in a fully authenticated environment where all network nodes are registered via the 802.1x based protocols. In this case the ALE table is filled with network node addresses that have been authenticated

12.2.1.4.6.1.7 VLAN Routing and OAM Operations

12.2.1.4.6.1.7.1 InterVLAN Routing

The CPSW module supports wire rate InterVLAN routing for a small number of routes, that is the host will setup an ALE classifier with and associated egress operation that will cause the CPSW to perform particular egress operations. The ALE can optionally check time to live validity as well.

The ALE uses the classifier along with an egress opcode, destination port mask and TTL check field to tell the CPSW to manipulate the packet on the egress. The CPSW will use the opcode along with a per port operation table to process the packet. By setting up the CPSW egress operation table you can replace the DA, SA and VLAN along with optionally updating the time to live IP header field. This allows the CPSW to perform the routing function for a small set of routes without getting the local host/CPU involved.

The Egress opcode will only be use for a classifier match and the packet would normally be sent only to the host. That is the host would have routed the packet but the CPSW has been configured to do the work instead. In the event that the time-to-live check feature is enabled and the time-to-live is either 0 or 1, the packet will not get the egress opcode and instead be sent to the host as if the route is not setup. This allows the host to deal with invalid TTL fields.

12.2.1.4.6.1.7.2 OAM Operations

The ALE supports OAM loopback on ports so that a remote link can be tested. That is a port placed in OAM loopback will echo packets received on a port back to the port with an egress op of 0xFF which will swap the SA and DA on egress in the CPSW.

Any supervisory packet will not be affected, so the spanning tree and other bridging functions are not affected.

Packets will only be echoed if the port is in OAM loopback mode, the received packet in not a supervisor packet. The port is in a forwarding state and the packet received DA!=SA and there are no errors in the packet.

Copyright © 2026 Texas Instruments Incorporated

When a port is in OAM loopback the port will not egress traffic from other ports, no address for loop backed traffic will be learned if enabled. Any packet received on the OAM loopback port with an error will be process as if the port is not in OAM. That is if the host has enabled copy errored frames the errored frames will be sent to the host instead.

12.2.1.4.6.1.8 Supervisory packets

Multicast supervisory packets are designated by the SUPER bit in the table entry. Unicast supervisory packets are indicated when BLOCK and SECURE are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing. The purpose of supervisory packets is to allow packets that would be otherwise blocked to be forwaded for special purposes.

12.2.1.4.6.1.9 Address Table Entry

The ALE table contains multiple table entry types. Each table entry represents a free entry, an address, a VLAN, an address/VLAN pair, or an OUI address. Software should ensure that there are not double address entries in the table. The double entry used would be indeterminate. Reserved table bits must be written with zeroes.

Source Address learning occurs for packets with a unicast, multicast or broadcast destination address and a unicast or multicast (including broadcast) source address. Multicast source addresses have the group bit (bit 40) cleared before ALE processing begins, changing the multicast source address to a unicast source address. A multicast address of all ones is the broadcast address which may be added to the table. A learned unicast source address is added to the table with the following control bits:

Table 12-110. Learned Address Control Bits

Bit(s)	Value
Ageable	1
Touch	1
BLOCK	0
SECURE	0

If a received packet has a source address that is equal to the destination address then the following occurs:

- The address is learned if the address is not found in the table.
- The address is updated if the address is found.
- The packet is dropped.

Table Entry Type

00 - Free Entry

01 - Address Entry : unicast or multicast determined by destination **address bit 40.**

10 - VLAN entry

11 - VLAN Address Entry : unicast or multicast determined by **address bit 40.**

12.2.1.4.6.1.9.1 Multicast Address Table Entry

Table 12-111. Multicast Address Table Entry Bit Values

70:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	PORT_MASK	SUPER	Reserved	MCAST_FWD_STATE	ENTRY_TYPE (01)	Reserved	MULTICAST_ADDRESS

Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

Copyright © 2026 Texas Instruments Incorporated

1: Supervisory packet

Port Mask(1:0) (PORT_MASK)

This 2-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

Multicast Forward State (MCAST_FWD_STATE)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit PORT_MASK has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00 - Forwarding

01 - Blocking/Forwarding/Learning

10 - Forwarding/Learning

11 - Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

Table Entry Type (ENTRY_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Packet Address (MULTICAST_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

12.2.1.4.6.1.9.2 OUI Unicast Address Table Entry

Table 12-112. OUI Unicast Address Table Entry Bit Values

70:64	63:62	61:60	59:48	47:24	23:0
Reserved	UNICAST_TYPE (10)	ENTRY_TYPE (01)	Reserved	UNICAST_OUI	Reserved

Unicast Type (UNICAST_TYPE)

This field indicates the type of unicast address the table entry contains.

00 - Unicast address that is not ageable.

01 - Ageable unicast address that has not been touched.

10 - OUI address - lower 24-bits are don't cares (not ageable).

11 - Ageable unicast address that has been touched.

Table Entry Type (ENTRY_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Packet Address (UNICAST_OUI)

For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.1.9.3 Unicast Address Table Entry (Bit 40 == 0)

Table 12-113. Unicast Address Table Entry Bit Values

70:69	68	67:66	65	64	63	62	61:60	59:48	47:0
RESERVED	TRUNK	PORT_NUMBER	BLOCK	SECURE	TOUCH	AGEABLE	ENTRY_TYPE (3h)	RESERVED	UNICAST_ADDRESS

Trunk Indicator (TRUNK)

0h = The port bits in the entry are the port number

1h = The port bits in the entry are the trunk numbe

Port Number (PORT_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0h = Address is not blocked.

1h = Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/ Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry PORT_NUMBER.

0h = Received port number is a don't care.

1h = Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

Touch Indicator (TOUCH)

Only valid when AGEABLE it a 1h.

0h = Ageable unicast address has not been touched

1h = Ageable unicast address that has been touched

Ageable (AGEABLE)

This bit indicates that the address is ageable.

0h = Unicast address that is not ageable

1h = Unicast address that is ageable

Table Entry Type (ENTRY_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Copyright © 2026 Texas Instruments Incorporated

Packet Address (UNICAST_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

12.2.1.4.6.1.9.4 Multicast Address Table Entry (Bit 40==1)

Table 12-114. Multicast Address Table Entry Bit Values

70:69	68:66	65	64	63:62	61:60	59:48	47:0
RESERVED	PORT_MASK	SUPER	IGNMBITS	FWDSTLVL	ENTRY_TYPE (1h)	RESERVED	MULTICAST_ADDRESS

Port Mask(2:0) (PORT_MASK)

This 3-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

Forward State Level (FWDSTLVL)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s).

A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

0h = Forwarding

1h = Blocking/Forwarding/Learning

2h = Forwarding/Learning

3h = Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

Table Entry Type (ENTRY_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Packet Address (MULTICAST_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

12.2.1.4.6.1.9.5 VLAN/Unicast Address Table Entry (Bit 40 == 0)

Table 12-115. Unicast Address Table Entry Bit Values

70:69	68	67:66	65	64	63	62	61:60	59:48	47:0
RESERVED	TRUNK	PORT_NUMBER	BLOCK	SECURE	TOUCH	AGEABLE	ENTRY_TYPE (3h)	VLAN_ID	UNICAST_ADDRESS

Copyright © 2026 Texas Instruments Incorporated

Trunk Indicator (TRUNK)

0h = The port bits in the entry are the port number

1h = The port bits in the entry are the trunk number

Port Number (PORT_NUMBER)

This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

Block (BLOCK)

The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0h = Address is not blocked.

1h = Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/ Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

Secure (SECURE)

This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry PORT_NUMBER.

0h = Received port number is a don't care.

1h = Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

Touch Indicator (TOUCH)

Only valid when AGEABLE is a 1h.

0h = Ageable unicast address has not been touched

1h = Ageable unicast address that has been touched

Ageable (AGEABLE)

This bit indicates that the address is ageable.

0h = Unicast address that is not ageable

1h = Unicast address that is ageable

Table Entry Type (ENTRY_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

3h: VLAN address entry. Unicast or multicast determined by address bit 40.

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

Packet Address (UNICAST_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.1.9.6 VLAN/Multicast Address Table Entry (Bit 40==1)

Table 12-116. VLAN/Multicast Address Table Entry Bit Values

70:69	68:66	65	64	63:62	61:60	59:48	47:0
RESERVED	PORT_MASK	SUPER	IGNMBITS	FWDSTLVL	ENTRY_TYPE (11)	VLAN_ID	MULTICAST_ADDRESS

Port Mask(2:0) (PORT_MASK)

This 3-bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

Forward State Level (FWDSTLVL)

Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s).

A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

0h = Forwarding

1h = Blocking/Forwarding/Learning

2h = Forwarding/Learning

3h = Forwarding

The forward state test returns a true value if both the RX and TX ports are in the required state.

Table Entry Type (ENTRY_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

11: VLAN address entry. Unicast or multicast determined by address bit 40.

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

Packet Address (MULTICAST_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

12.2.1.4.6.1.9.7 Inner VLAN Table Entry

Table 12-117. Inner VLAN Table Entry

70:69	68:66	65	64:62	61:60	59:48	47	46:39	38:36
RESERVED	NO_LEARN_MASK	VLAN_FORCE_INGRESS_CHECK	0h	ENTRY_TYPE (1h)	VLAN_ID	NOFRAG	RESERVED	REG_MCAST_FLOOD_INDEX

Copyright © 2026 Texas Instruments Incorporated

35:27	26:24	23	22:15	14:12	11:3	2:0
RESERVED	FORCE_UNTAGGED_EGRESS	LMTNXTHDR	RESERVED	UREGMSK	RESERVED	VLAN_MEMBER_LIST

No Learn Mask (NO_LEARN_MASK)

When a bit is set in this mask, a packet with an unknown source address received on the associated port will not be learned (i.e. When a VLAN packet is received and the source address is not in the table, the source address will not be added to the table).

VLAN Force Ingress Check (VLAN_FORCE_INGRESS_CHECK)

If the receive port is not a member of this VLAN then the packet is dropped. This is similar to the ly_REG_Py_VID_INGRESS_CHECK bit in the CPSW3_CPSW_NU_ALE_I0_ALE_PORTCTL0 register except this check is for this VLAN only (not all VLANs).

Table Entry Type (ENTRY_TYPE)

2h: VLAN entry

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

(NOFRAG)

VLAN No IPv4 Fragmented frames Control - Causes IPv4 fragmented IP frames to be dropped.

Registered Multicast Flood Index (REG_MCAST_FLOOD_INDEX)

This field indicates which port(s) are the registered multicast flood mask.

Force Untagged Packet Egress (FORCE_UNTAGGED_EGRESS)

This field causes the packet VLAN tag to be removed on egress for the specified port(s) (except on port 0).

VLAN Limit Next Header Control (LMTNXTHDR)

This bit causes frames to be dropped if the Protocol/Nxt Header does not match the CPSW3_CPSW_NU_ALE_ALE_NXT_HDR register values.

VLAN Unregister Multicast Mask (UREGMSK)

This field indicates which port(s) are the unregistered multicast flood mask.

VLAN Member List (VLAN_MEMBER_LIST)

This field indicates which port(s) are members of the associated VLAN. One bit per port.

12.2.1.4.6.1.9.8 Outer VLAN Table Entry

Table 12-118. Outer VLAN Table Entry

70:69	68:66	65	64:62	61:60	59:48	47	46:39	38:36
RESERVED	NO_LEARN_MASK	VLAN_FORCE_INGRESS_CHECK	2h	ENTRY_TYPE (2h)	VLAN_ID	NOFRAG	RESERVED	REG_MCAST_FLOOD_INDEX

35:27	26:24	23	22:15	14:12	11:3	2:0
RESERVED	FORCE_UNTAGGED_EGRESS	LMTNXTHDR	RESERVED	UREGMSK	RESERVED	VLAN_MEMBER_LIST

No Learn Mask (NO_LEARN_MASK)

When a bit is set in this mask, a packet with an unknown source address received on the associated port will not be learned (i.e. When a VLAN packet is received and the source address is not in the table, the source address will not be added to the table).

Copyright © 2026 Texas Instruments Incorporated

VLAN Force Ingress Check (VLAN_FORCE_INGRESS_CHECK)

If the receive port is not a member of this VLAN then the packet is dropped. This is similar to the Iy_REG_Py_VID_INGRESS_CHECK bit in the CPSW3_CPSW_NU_ALE_I0_ALE_PORTCTL0 register except this check is for this VLAN only (not all VLANs).

Table Entry Type (ENTRY_TYPE)

2h: VLAN entry

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

(NOFRAG)

VLAN No IPv4 Fragmented frames Control - Causes IPv4 fragmented IP frames to be dropped.

Registered Multicast Flood Index (REG_MCAST_FLOOD_INDEX)

Index into CPSW_ALE_MSK_MUX0 to CPSW_Ix_ALE_MSK_MUXx register array that is used to create the registered multicast flood mask.

Force Untagged Packet Egress (FORCE_UNTAGGED_EGRESS)

This field causes the packet VLAN tag to be removed on egress for the specified port(s) (except on port 0).

VLAN Limit Next Header Control (LMTNXTHDR)

This bit causes frames to be dropped if the Protocol/Nxt Header does not match the CPSW3_CPSW_NU_ALE_ALE_NXT_HDR register values.

VLAN Unregister Multicast Mask (UREGMSK)

This field indicates which port(s) are the unregistered multicast flood mask.

VLAN Member List (VLAN_MEMBER_LIST)

This field indicates which port(s) are members of the associated VLAN. One bit per port.

12.2.1.4.6.1.9.9 EtherType Table Entry

Table 12-119. EtherType Table Entry

70:65	64:62	61:60	59:16	15:0
RESERVED	4h	ENTRY_TYPE (2h)	RESERVED	ETHERTYPE

Table Entry Type (ENTRY_TYPE)

2h: VLAN entry

Ether Type (ETHERTYPE)

16-bits Ether Type field.

12.2.1.4.6.1.9.10 IPv4 Table Entry

Table 12-120. IPv4 Table Entry

70	69:65	64:62	61:60	59:32	31:0
RESERVED	IGNMBITS	6h	ENTRY_TYPE (2h)	RESERVED	IPV4ADR

Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

Table Entry Type (ENTRY_TYPE)

Copyright © 2026 Texas Instruments Incorporated

2h: VLAN entry

IPv4 Address (IPV4ADR)

32-bit IPv4 Address. Any ignored bits must be zero value in the table entry.

12.2.1.4.6.1.9.11 IPv6 Table Entry High

Table 12-121. IPv6 Table Entry High

70:64	63	62	61:60	59:0
IGNMBITS	RESERVED	(1h)	ENTRY_TYPE (2h)	IPV6ADR[127:68]

Ignore Multicast Bits (IGNMBITS)

Indication that the Multicast Address has ignored bits.

Table Entry Type (ENTRY_TYPE)

2h: VLAN entry

IPv6 Address - upper 64 bits (IPV6ADR[127:68])

This address is plit into three fields in the IPv6 High and IPv6 Low table entry. Any ignored bits must be zero in the table entry(s).

12.2.1.4.6.1.9.12 IPv6 Table Entry Low

Table 12-122. IPv6 Table Entry Low

70:63	62	61:60	59:0
IPV6ADR[67:60]	1h	ENTRY_TYPE (2h)	IPV6ADR[59:0]

Table Entry Type (ENTRY_TYPE)

2h: VLAN entry

IPv6 Address - upper 8 bits (IPV6ADR[67:60])

IPv6 Address - upper 60 bits (IPV6ADR[59:0])

This address is plit into three fields in the IPv6 High and IPv6 Low table entry. Any ignored bits must be zero in the table entry(s).

Note: IPV6 table address entries operate differently than all other table entry types. IPV6 table entries have a high entry concatenated with a low entry. The high entry must have an entry_pointer value of the low entry_pointer plus 0x40. Bit six of the entry_pointer must be set for the high entry and must be zero for the low entry.

12.2.1.4.6.1.10 Multicast Address

Multicast addresses are addresses with bit 40 set. Only destination addresses can be Multicast addresses. The group bit (bit 40) of the source address is reserved in the IEEE tandard.

A multicast address of all ones is the broadcast address which can be added to the lookup table if forwading of broadcast packets need be modified.

12.2.1.4.6.1.10.1 Multicast Ranges

Added IgnMbits to indicate at least one bit of the multicast address is ignored. Up to 10 bits of the multicast address can be ignored to provide the ability to create multiple multicast address ranges.

if ((IgnMbits)&(MultiCastAddress[0] ==0x000)) { MultiCastAddress[0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[1:0]==0x001)) { MultiCastAddress[1:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[2:0]==0x003)) { MultiCastAddress[2:0] is ignored in compare}

Copyright © 2026 Texas Instruments Incorporated

if ((IgnMbits)&(MultiCastAddress[3:0]==0x007)) { MultiCastAddress[3:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[4:0]==0x00F)) { MultiCastAddress[4:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[5:0]==0x01F)) { MultiCastAddress[5:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[6:0]==0x03F)) { MultiCastAddress[6:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[7:0]==0x07F)) { MultiCastAddress[7:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[8:0]==0x0FF)) { MultiCastAddress[8:0] is ignored in compare}

if ((IgnMbits)&(MultiCastAddress[9:0]==0x1FF)) { MultiCastAddress[9:0] is ignored in compare}

Below is 'C' code to modify ALE MultiCastAddress and IgnMbits when iNumOfBitsToIgnore is greater than zero. Where fGenMask(iOffset,iBitsToMask) creates a Mask for the value provided. For example fGenMast(0,5) will return 0x1F.

```
if(iNumOfBitsToIgnore)
{
int iIgnClrMsk,iIgnSetMsk;
iIgnClrMsk=fGenMask(0, iNumOfBitsToIgnore);
iIgnSetMsk=fGenMask(0, iNumOfBitsToIgnore -1);
MultiCastAddress &= ~iIgnClrMsk;
MultiCastAddress |= iIgnSetMsk;
IgnMbits = 1;
}
else
{
IgnMbits = 0;
}
```

Multicast Addresses or Ranges can overlap, in the event of an overlap; the higher ALE index will be used.

12.2.1.4.6.1.11 Aging and Auto Aging

The ALE supports software control or automatic aging of agable addresses.

Any time an agable address is seen as a source address entering from a port the source address entry will be marked as touched.

If the aging timer expires or the software sets the [29] AGE_OUT_NOW bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL, the aging process will be started.

The aging process will read each ALE entry and for all entries that are an address with or without VLAN that is also agable, the touch check process will be done.

The touch check process will test the TOUCH bit and if clear, the entry will be marked as free, else the TOUCH bit will be cleared.

What this means is that if the aging interval was programmed as one second, any unused entry could stay in the ALE table for 1.000001 to 1.999999 seconds

12.2.1.4.6.1.12 ALE Policing and Classification

The ALE has a number of configurable classifier engines (policers) that can be used for classification. Classification is a subset of the policing function and uses a policer without the color marking or rate

Copyright © 2026 Texas Instruments Incorporated

limiting functions. A policer is a hardware engine that is used for policing. The POLCNTDIV8 field in the CPSW3_CPSW_NU_ALE_ALE_STATUS register indicates the number of policers available to be used for classification. Each policer can be enabled to match on one or more of any of the below packet fields for classification. All but Port and Priority are index references to the ALE table entries.

- Port Number
- Priority extracted from VLAN, mapped from DSCP if enabled, or Default Port Priority
- Organization Network Unique identifier - ONU
- Destination Address - DA
- Source Address - SA
- Outer VLANID -S-VLANID
- Inner VLANID -C-VLANID
- Ether Type
- IP Source Address - IPSA with full CIDR masking for IPv4 and IPv6
- IP Destination Address - IPSA with full CIDR masking for IPv4 and IPv6
- Support Host Thread/Flow ID mapping based on any packet classification above

12.2.1.4.6.1.12.1 ALE Policing

The policing function on each policer engine is implemented as dual-counter three-color marking engine as described in the IETF RFC2698. The first counter is the Committed Information Rate (CIR) counter and the second counter is the Peak Information Rate (PIR) counter. The policing function can use either or both counters. Based on the counter values the packet color is determined. The color is used to determine whether the packet is dropped or forwarded. The ALE has a local feature that can drop packets regardless of queue state.

The policing rates are determined by the below equations:

CIR policing rate in Mbit/s = ((ALE frequency in Mhz) * CPSW3_CPSW_NU_ALE_POLICECFG7[31-0] CIR_IDLE_INC_VAL) / 32768

PIR policing rate in Mbit/s = ((ALE frequency in Mhz) * CPSW3_CPSW_NU_ALE_POLICECFG6[31-0] PIR_IDLE_INC_VAL) / 32768

Each policer has 10 different match operations (see Section 12.2.1.4.6.1.12). Since multiple policing entries can be hit on a single packet this provides the ability to create precise traffic stream control.

Packets are colored at ALE lookup time. Packets can be colored RED, YELLOW, or GREEN. If multiple policers are configured for a packet stream then the packet color is merged from all matching (hit) policers. If any policer is RED then the packet is marked RED. Else if any policer is YELLOW then the packet is marked YELLOW. Otherwise the packet is marked GREEN.

The Policing engine supports several modes such that packets that don't hit a policing/classifier match can be treated as RED, YELLOW, GREEN or policer 0 color. Using policer 0 allows for a system to regulate unregulated traffic.

12.2.1.4.6.1.12.2 Classifier to Host Thread Mapping

The ALE module allows Host Thread mapping based on any packet classification. That is the ALE can generate a thread ID used by the host based on ALE classifier matches.

When enabled the highest classifier match can map to a particular thread ID value.

The ALE also supports an optional default Thread ID value in the event that no classifier match.

Each Thread ID, including the default thread ID, has an enable functionality such that, if no classifier matches occur the default value is used, if the default is not enabled, the switch will use the {port,priority} value instead. If multiple classifier matches occur, the highest matching entry with a thread enable bit set will be used.

Three registers are used for ALE classification thread mapping configuration (CPSW3_CPSW_NU_ALE_THREADMAPDEF, CPSW3_CPSW_NU_ALE_THREADMAPCTL and

Copyright © 2026 Texas Instruments Incorporated

CPSW3_CPSW_NU_ALE_THREADMAPVAL). The three thread mapping registers are used independently and are separate from the other ALE policing registers. The CPSW3_CPSW_NU_ALE_THREADMAPCTL register allows the CPSW3_CPSW_NU_ALE_THREADMAPVAL register contents to be written to the selected classifier. There is a CPSW3_CPSW_NU_ALE_THREADMAPDEF register that is used for all classifiers. The thread mapping registers can be written or changed at any time but any packets that are already processed will not have their thread altered.

12.2.1.4.6.1.12.3 ALE Classification

When the policers are configured as classifiers, the color marking and policing functions of the policing/classifier engines are not used. One or multiple classifiers can be configured to match on a single packet. For example, a classifier can be enabled to match on priority while another classifier could match IP address.

12.2.1.4.6.1.12.3.1 Classifier to CPPI Transmit Flow ID Mapping

The ALE can generate a 6-bit transmit CPPI Flow ID based on classifier matches that can be used instead of the switch default transmit Flow ID mapping. The switch default flow ID is the remapped received packet priority (0 to 7). Thread and flow ID are used interchangeably for this since there is a single hardware thread (TXST_THREAD_MREADY) but there are 6-bits of FLOW_ID in the transmit CPPI INFO word 0. When enabled, the highest classifier match can map to a particular 6-bit flow ID value that is associated with the classifier. The ALE also supports an optional ALE default thread/flow ID value in the event that no classifiers match. Each thread/flow ID, including the ALE default thread/flow ID, has an enable such that the ALE default thread/Flow ID is used if enabled and if no matches occur (instead of the remapped received packet priority). If the ALE default is not enabled and no matches occur then the switch default value will be used. If multiple classifier matches occur, the highest match with a thread enable bit set will be used. The resultant flow ID has the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_FLOW_ID_OFFSET_REG register value added to it to determine the actual value in the INFO 0 Flow ID field.

Three registers are used for ALE classification thread/flow ID mapping configuration (CPSW3_CPSW_NU_ALE_THREADMAPDEF, CPSW3_CPSW_NU_ALE_THREADMAPCTL and CPSW3_CPSW_NU_ALE_THREADMAPVAL). The three thread mapping registers are used independently and are separate from the other ALE policing registers. The CPSW3_CPSW_NU_ALE_THREADMAPCTL register allows the CPSW3_CPSW_NU_ALE_THREADMAPVAL register contents to be written to the selected classifier. There is a single CPSW3_CPSW_NU_ALE_THREADMAPDEF that is used for all classifiers. The thread mapping registers can be written or changed at any time but any packets that are already processed will not have their thread altered.

12.2.1.4.6.1.13 Mirroring

The ALE supports three mirroring modes: destination port, source port and or table entry.

Destination port mirroring allows packets from any ingress port or trunk which ends up switching to a particular egress destination port or trunk to be mirrored to yet another egress destination port or trunk. For example any traffic from any port that is switched to port 'A' can be also mirrored to port 'B'. (MIRROR_DP=A, MIRROR_DEN=1h, MIRROR_TOP=B in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register).

Source port mirroring allows packets received on any enabled ingress source port or trunk to be switched to the mirror egress port as well as the actual egress destination ports. For example traffic received on ingress port 'A' can be switched to egress port 'B' as well as the intended egress destination port.(Iy_REG_Py_MIRROR_SP=1h in the CPSW3_CPSW_NU_ALE_I0_ALE_PORTCTL0 register, MIRROR_SEN=1h, MIRROR_TOP=B in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register).

Table entry mirroring allows for any MAC Address, MAC Address with VLAN, ONU Address or VLAN entry that matches on ingress to be switched to the egress destination as well as the actual egress destination. For example all traffic for VLAN ID of 35 can be mirrored to port 'B'. That is any traffic switched on VLAN ID of 35 will be mirrored. ({VLAN ID of 35 in ALE Table entry index=C}, MIRROR_MIDX=C, MIRROR_MEN=1h, MIRROR_TOP=B)

Copyright © 2026 Texas Instruments Incorporated

In the event that mirrored packets are mirrored to or from a port that is also the mirror port the packet will not be duplicated or marked as a mirror packet since the packet has already been on the port as ingress or egress. The packet sent to the mirror port may have modified VLAN info based on the port and VLAN lookup table entries. The mirror port need not be a member of the VLAN ID it is mirroring, the ALE will forward traffic to the mirror port after ingress and egress filters are applied.

The switch may decide to drop any mirror traffic based on switch buffer thresholds as to prevent required traffic from becoming congested.

Port mirroring is controlled by register fields in CPSW3_CPSW_NU_ALE_ALE_CONTROL, CPSW3_CPSW_NU_ALE_ALE_CTRL2 and the port control registers.

- MIRROR_DP - The destination port that will have its traffic mirrored (CPSW3_CPSW_NU_ALE_ALE_CONTROL register).
- MIRROR_TOP - The port to which mirrored traffic is sent (CPSW3_CPSW_NU_ALE_ALE_CONTROL register).
- MIRROR_MEN - The enable for mirroring traffic that matches a supported lookup table entry (CPSW3_CPSW_NU_ALE_ALE_CONTROL register).
- MIRROR_DEN - The Enable for destination port mirroring (CPSW3_CPSW_NU_ALE_ALE_CONTROL register).
- MIRROR_SEN - The Enable for source port mirroring (CPSW3_CPSW_NU_ALE_ALE_CONTROL register).
- MIRROR_MIDX - The index of a lookup table entry that will be mirrored CPSW3_CPSW_NU_ALE_ALE_CTRL2 register).
- Iy_REG_Py_MIRROR_SP - The enable for the Source port to be mirrored. Although multiple source ports can be mirrored concurrently, a mirror traffic bandwidth issue may occur on the mirror egress port (CPSW3_CPSW_NU_ALE_I0_ALE_PORTCTL0 register).

12.2.1.4.6.1.14 Trunking

The ALE supports port trunking of any port in any of four trunk groups. That is, four trunk groups can be supported with up to eight ports in each trunk group. There are no port adjacency rules for trunk groups. When ports are a member of a trunk group, addresses added and used in the lookup table will refer to the trunk group rather than port as indicated in the lookup table entries. If ports are removed from a trunk group, the ALE will redistribute the traffic based on the crc polynomial of enabled fields and the remaining ports within the trunk group. A trunk group may contain only one port. Packet priority, DA, SA, C-VLAN ID, IPv4SA, IPv4DA, IPv6SA, and/or IPv6DA can be used in the hash to generate destination port within the trunk group. If all hash enables are disabled, the packet can be directed to a particular port within the trunk group which allows for testing paths etc. A host directed frame is directed to the directed port regardless of trunk group settings.

Trunking is controlled through fields in the CPSW3_CPSW_NU_ALE_ALE_CTRL2 register and in each ALE CPSW3_CPSW_NU_ALE_I0_ALE_PORTCTL0 register:

- TRK_EN_DST - Enable destination address hashing for trunk port calculation.
- TRK_EN_SRC - Enable source address hashing for trunk port calculation.
- TRK_EN_PRI - Enable priority hashing for trunk port calculation.
- TRK_EN_IVLAN - Enable inner C-VLAN ID hashing for trunk port calculation.
- TRK_EN_SIP - Enable source IP address hashing for trunk port calculation.
- TRK_EN_DIP - Enable destination IP address hashing for trunk port calculation.
- TRK_BASE - Hashing formula starting value and test port offset.
- Iy_REG_Py_TRUNKEN - Enable this port as a trunk group
- Iy_REG_Py_TRUNKNUM - Trunk group number defines this port as a member of a particular trunk group.

12.2.1.4.6.1.15 DSCP

The ALE can map DSCP field to priority prior to classification matching. When enabled the DSCP is mapped via 64 priority entries such that any DSCP value can be mapped to any of the eight priorities. When a packet is received without a VLAN priority this remapped priority can be used instead of the default Port VLAN priority field. See CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_RX_DSCP_MAP_REG_y and

Copyright © 2026 Texas Instruments Incorporated

CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_DSCP_MAP_REG registers in the Register Manual section for DSCP mapping.

12.2.1.4.6.1.16 Packet Forwarding Processes

There are four processes that an incoming received packet may go through to determine packet forwarding. The processes are *Ingress Filtering*, *VLAN_Aware Lookup*, and *Egress.*

Packet processing begins in the Ingress Filtering process. Each port has an associated packet forwarding state that can be one of four values (Disabled, Blocked, Learning, or Forwarding). The default state for all ports is Disabled. The host sets the packet forwarding state for each port.

In the packet ingress process (receive packet process), there is a forward state test for unicast destination addresses and a forward state test for multicast addresses. The multicast forward state test indicates the port states required for the receiving port in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state for the packet to be forwarded for transmission. The MCAST_FWD_STATE indicates the required port state for the receiving port as indicated in the preceding table. The unicast forward state test indicates the port state required for the receiving port in order to forward the unicast packet. The transmit port must be in the Forwarding state in order to forward the packet. The BLOCK and SECURE bits determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state. The transmit port must be in the Forwarding state regardless. The forward state test used in the ingress process is determined by the destination address packet type (multicast/unicast).

In general, packets received with errors are dropped by the address lookup engine without learning, updating, or touching the address. The error condition and the abort are indicated by the Ethernet port to the ALE. Packets with errors may be passed to the host (not aborted) by a Ethernet port port, if the port has the RX_CMF_EN, RX_CEF_EN, or RX_CSF_EN bit(s) set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG registers. Error packets that are passed to the host by the Ethernet port are considered to be bypass packets by the ALE and are sent only to the host. Error packets do not learn, update, or touch addresses regardless of whether they are aborted or sent to the host. Packets with long or short errors received by the host are dropped. Packets with errors received by the host are forwarded as normal.

The following control bits are in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG registers:

- [22] RX_CEF_EN - enables frames that are fragments, long, jabber, CRC, code, and alignment errors to be forwarded
- [23] RX_CSF_EN - enables short frames to be forwarded
- [24] RX_CMF_EN - enables MAC control frames to be forwarded.

12.2.1.4.6.1.16.1 Ingress Filtering Process

Condition and action
If ((ALE BYPASS) and (host port is not the receive port)) then use host portmask and go to Egress process
if (directed packet) then use directed port number and go to Egress process
If (Rx Iy_REG_Py_PORTSTATE is Disabled) then discard the packet
if ((ALE BYPASS or error packet) and (host port is not the receive port)) then use host portmask and go to Egress process
if (((BLOCK) and (unicast source address found)) or ((BLOCK) and (unicast destination address found))) then discard the packet

Copyright © 2026 Texas Instruments Incorporated

if ((ENABLE_RATE_LIMIT) and (rate limit exceeded) and (not BCAST_MCAST_CTL) then if (((Multicast/Broadcast destination address found) and (not SUPER)) or (Multicast/Broadcast destination address not found)) then discard the packet
if ((not forward state test valid) and (destination address found)) then discard the packet to any port not meeting the requirements • Unicast destination addresses use the unicast forward state test and multicast destination addresses use the multicast forward state test.
if ((destination address not found) and ((not transmit port forwarding) or (not receive port forwarding))) then discard the packet to any ports not meeting the above requirements
if (source address found) and (secure) and (not block) and (receive port number != port_number)) then discard the packet
if ((not super) and (drop_untagged) and ((non-tagged packet) or ((priority tagged) and not(en_vid0_mode)) then discard the packet

If (VLAN_Unaware)
CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG = “000000”
CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST = “111111”
CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST = “111111”
UVLAN_MEMBER_LIST = “111111”
else if (VLAN not found)
CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG = CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG
CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST = CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST
CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST = CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST
CPSW3_CPSW_NU_ALE_ALE_UVLAN_MEMBER = CPSW3_CPSW_NU_ALE_ALE_UVLAN_MEMBER
else
CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG = found CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG
CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST = found CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST
CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST = found CPSW3_CPSW_NU_ALE_ALE_UVLAN_RMCAST
UVLAN_MEMBER_LIST = found UVLAN_MEMBER_LIST

if ((not SUPER) and (Iy_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member)) then discard the packet
if ((ENABLE_AUTH_MODE) and (source address not found) and not(destination address found and (SUPER))) then discard the packet
if (destination address equals source address) then discard the packet
if (VLAN_AWARE) goto VLAN_Aware_Lookup process else goto VLAN_Unaware_Lookup process

12.2.1.4.6.1.16.2 VLAN_Aware Lookup Process

Condition and action
if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER) then portmask is the logical "AND" of the PORT_NUMBER and UVLAN_MEMBER_LIST less the host port and goto Egress process

Copyright © 2026 Texas Instruments Incorporated

if ((unicast packet) and (destination address found with or without VLAN) and (not SUPER)) then portmask is the logical “AND” of the PORT_NUMBER and the UVLAN_MEMBER_LIST and goto Egress process
if ((unicast packet) and (destination address found with or without VLAN) and (SUPER)) then portmask is the PORT_NUMBER and goto Egress process
if (Unicast packet) # destination address not found then portmask is VLAN member LIST less host port and goto Egress process
if ((Multicast packet) and (destination address found with or without VLAN) and (not SUPER)) then portmask is the logical “AND” of CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST and found destination address/VLAN portmask (PORT_MASK) and UVLAN_MEMBER_LIST and goto Egress process
if ((Multicast packet) and (destination address found with or without VLAN) and (SUPER)) then portmask is the PORT_MASK and goto Egress process
if (Multicast packet) # destination address not found then portmask is the logical “AND” of CPSW3_CPSW_NU_ALE_ALE_UVLAN_URCAST and UVLAN_MEMBER_LIST then goto Egress process
if (Broadcast packet) then use found UVLAN_MEMBER_LIST and goto Egress process

Note

The UVLAN_MEMBER_LIST, UVLAN_UNREG_MCAST_FLOOD_MASK, UVLAN_REG_MCAST_FLOOD_MASK and UVLAN_FORCE_UNTAGGED_EGRESS are set in the Section 12.2.1.4.6.1.16.1 *Ingress Filtering Process*, based on VLAN_Unaware, Unknown_VLAN rules and VLAN table entries.

12.2.1.4.6.1.16.3 Egress Process

Condition and action
Clear Rx port from portmask (don’t send packet to Rx port).
Clear disabled ports from portmask.
if ((ENABLE_OUI_DENY) and (OUI source address not found) and (not ALE BYPASS) and (not error packet) and ((not mcast destination address) and (SUPER))) then Clear host port from portmask
if ((not ENABLE_OUI_DENY) and (OUI source address found) and (not ALE BYPASS) and (not error packet) and not ((mcast destination address) and (SUPER))) then Clear host port from portmask
if ((ENABLE_RATE_LIMIT) and (BCAST_MCAST_CTL)) then if (not SUPER) and (rate limit exceeded on any tx port) then clear rate limited tx port from portmask If address not found then SUPER cannot be set.
If portmask is zero then discard packet
Send packet to portmask ports.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.1.16.4 Learning/Updating/Touching Processes

The learning, updating, and touching processes are applied to each receive packet that is not aborted. The processes are concurrent with the packet forwarding process. In addition to the following, a packet must be received without error in order to learn/update/touch an address.

12.2.1.4.6.1.16.4.1 Learning Process

The learning process is applied to each receive packet that is not aborted. The learning process is a concurrent process with the packet forwarding process.

Condition and action
If (directed) then do not learn, update, or set touched else continue
If (not (Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (Iy_REG_Py_NO_LEARN)) then do not learn address
if ((Non-tagged packet) and (Iy_REG_Py_DROP_UN_TAGGED)) then do not learn address
if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = “000”)) then do not learn address
if ((Iy_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not learn address
if ((source address found) and (receive port_number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address else continue
if ((source address found) and (receive port number != PORT_NUMBER)) then update address else continue
if ((source address not found) and (VLAN_AWARE) and not (LEARN_NO_VLANID)) then learn address with VLAN
if ((source address not found) and ((not VLAN_AWARE) or (VLAN_AWARE and LEARN_NO_VLANID))) then learn address without VLAN

12.2.1.4.6.1.16.4.2 Updating Process

Condition and action
if (dlr_unicast) then do not update address
If (not(Learning or Forwarding) or (ENABLE_AUTH_MODE) or (packet error) or (Iy_REG_Py_NO_SA_UPDATE)) then do not update address
if ((Non-tagged packet) and (Iy_REG_Py_DROP_UN_TAGGED)) then do not update address
if ((VLAN_AWARE) and (VLAN not found) and (unknown UVLAN_MEMBER_LIST = “000”)) then do not update address
if ((Iy_REG_Py_VID_INGRESS_CHECK) and (Rx port is not VLAN member) and (VLAN found)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER) and (SECURE or BLOCK)) then do not update address
if ((source address found) and (receive port number != PORT_NUMBER)) then update address

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.1.16.4.3 Touching Process

if ((source address found) and (ageable) and (not touched)) then set touched

12.2.1.4.6.1.17 VLAN Aware Mode

The CPSW is in VLAN aware mode when the VLAN_AWARE bit is set in the CPSW3_CONTROL_REG register.

In VLAN aware mode, transmitted packet data is changed depending on the packet type (PKT_TYPE), packet priority (PKT_PRI), and VLAN information.

The VLAN_LTYPE_SEL value is selected by the S_CN_SWITCH bit in the CPSW3_CONTROL_REG register and is either the VLAN_LTYPE_INNER (8100h default) or VLAN_LTYPE_OUTER (88A8h default) value.

12.2.1.4.6.1.18 VLAN Unaware Mode

An egress port is operating in the VLAN unaware mode when the VLAN_AWARE bit in the CPSW3_CONTROL_REG register is cleared to 0h. In VLAN unaware mode, transmit (egress) packets are not modified on egress.

12.2.1.4.6.2 Packet Priority Handling

Packets are received on three ports, two Ethernet port and one CPPI host port. Received packets have a received packet priority (0 to 7, with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN_LTYPE_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP_IPV4_EN is set in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG or CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CONTROL_REG, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP_IPV6_EN is set in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG or CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CONTROL_REG, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority.

The received packet priority is mapped through the receive ports associated packet-priority-to-header-packet-priority-mapping register (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG) to obtain the header packet priority. The header packet priority is the hardware switch priority. The header packet priority is also used as the actual transmit packet priority if the VLAN information is to be sent on egress.

The header packet priority is mapped at each destination FIFO through the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TX_PRI_MAP_REG register (header priority to switch priority mapping register) to obtain the hardware switch priority (hardware queue 0 through 7).

12.2.1.4.6.2.1 Priority Mapping and Transmit VLAN Priority

There are three priorities that are used inside the Gigabit Ethernet Switch: **packet priority**, **header packet priority**, and **switch priority**. The **header packet priority** is used as the outgoing VLAN priority if the packet is egressing from the switch with a VLAN tag. The **switch priority** determines which of the 8 FIFO priority queues the packet uses during egress.

Figure 12-96 below, as well as the corresponding explanation that follows, explains each of the priorities, how they are determined, and how they are used. A number in parentheses in the figure indicates a process

Copyright © 2026 Texas Instruments Incorporated

(Ethernet port ingress, host port egress, etc.). Each bullet in the text following the diagram explains one of the 5 processes pointed out in the figure.

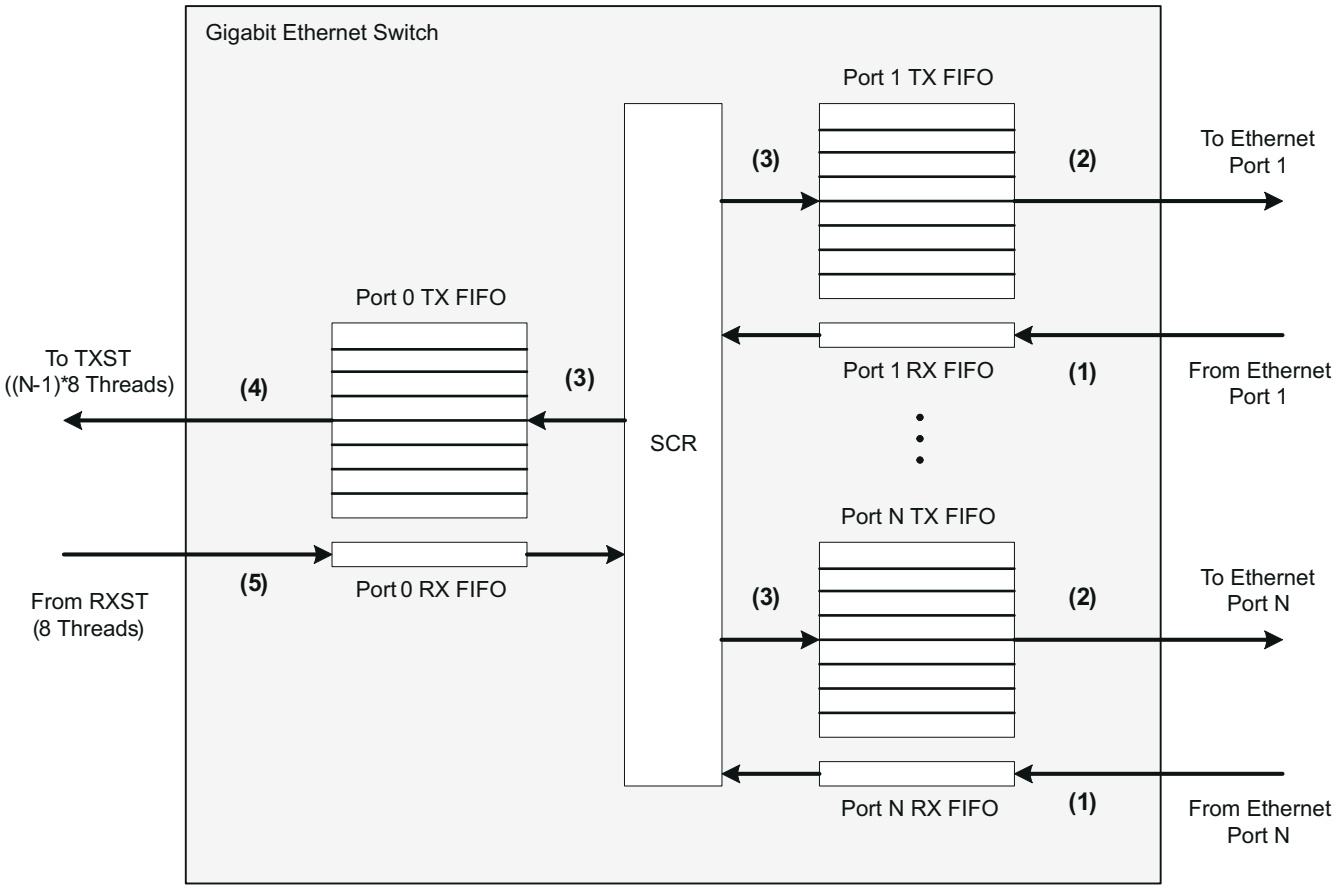


Figure 12-96. Gigabit Ethernet Switch Priority Mapping and Transmit VLAN Processing

Copyright © 2026 Texas Instruments Incorporated

From Figure 12-96 above:

- (1) is the ingress process that occurs at the external Ethernet ports
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the ingress port's priority. This **packet priority** is then mapped to a **header packet priority** using the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG register where N is the port where the packet entered the switch. This process is explained in further detail in Section 12.2.1.4.6.2.
- (2) is the egress process that occurs at the external Ethernet ports
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1) or (5). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in Section 12.2.1.4.6.4.1.
- (3) is the process by which it is decided which priority TX queue to place the packet on in the Port N TX FIFO during egress
  - Each Port's TX FIFO has 8 queues that each correspond to a priority that is used when determining which packet will egress from the switch next at that port. The **header packet priority** (Ethernet port ingress, process (1)) or the receive packet thread (host port 0 ingress, process (5)) gets mapped through the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG register (where N is the egress port number) to determine the **switch priority** of the packet. The **switch priority** determines which TX FIFO queue to place the packet in. The FIFO architecture is described in Section 12.2.1.4.6.10.5. The header packet priority to switch priority mapping is discussed in Section 12.2.1.4.6.2.
- (4) is the egress process that occurs at Host Port 0 toward the transmit streaming interface (TXST)
  - The TXST has 8 egress threads for each external Ethernet port (32 threads in a 5-port switch and 64 threads in a 9-port switch). The TX thread that is selected for a packet is determined by the Ethernet port of ingress and the **switch priority** of the packet that was determined in process (3).
  - If the switch is in VLAN Aware mode then the VLAN header may be added, replaced, or removed during the egress process. If the VLAN header is to be added or replaced, the VLAN priority will come from the **header packet priority** that was determined in process (1). Transmit VLAN processing is the same for both the host port and the external Ethernet ports and is described in Section 12.2.1.4.6.4.1.
- (5) is the ingress process that occurs at Host Port 0
  - The incoming packet is assigned a **packet priority** based on either its VLAN priority, IPv4 or IPv6 DSCP value, or the host port's priority. This packet priority is then mapped to a **header packet priority** using the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG register.
  - Host port 0 also has a received packet thread. The receive packet thread is based on either the packet's VLAN priority, IPv4 or IPv6 DSCP value, or the streaming interface (RXST) thread that the packet entered host port 0 on.

12.2.1.4.6.3 CPPI Port Ingress

Packets received on the CPPI host port have a received packet priority (0 to 7 with 7 being the highest priority).

The received packet priority is determined as follows:

1. If the first packet LTYPE = VLAN_LTYPE_SEL then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. Else if the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP_IPV4_EN is set in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG or CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CONTROL_REG register, then the received packet priority is the 6-bit TOS field in byte 15 (upper 6 bits) mapped through the port's DSCP priority mapping registers (IPv4 packet).
3. Else if the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP_IPV6_EN is set in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG or CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CONTROL_REG register, then the received packet priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPv6 packet).
4. Else the received packet priority is the source (ingress) port priority

Copyright © 2026 Texas Instruments Incorporated

The CPPI Port (port 0) also has a received packet thread. The received packet thread is determined exactly as the received packet priority except for untagged packets. For untagged packets, the received packet thread is the packet streaming interface thread that the packet was received on (instead of the port VLAN priority or DSCP priority). The received packet thread is the hardware switch priority. The received packet thread only determines which hardware switch priority the packet should be sent to, the egress VLAN rules are identical to packets that were received on Ethernet ports (as determined by the header packet priority).

For CPPI ingress packets, the destination port hardware switch priority is the below selected value remapped through CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG:

1. If the ingress packet is priority tagged or vlan tagged:
   - If RX_REMAP_VLAN in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
   - If RX_REMAP_VLAN in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is set then the the destination hardware switch priority is the packet priority value. Port transmit remapping (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TX_PRI_MAP_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
2. Else if the ingress packet has the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x4X, and DSCP_IPV4_EN is set in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register:
   - If RX_REMAP_DSCP_V4 bit in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
   - If RX_REMAP_DSCP_V4 bit in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is set then the destination hardware switch priority is the 6-bit TOS field in byte 15 (upper 6-bits) mapped through the port's DSCP priority mapping registers (IPV4 packet). Port 1 transmit remapping (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TX_PRI_MAP_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
3. Else if the ingress packet has the first packet LTYPE = 0x86DD and the most significant nibble of byte 14 (following the LTYPE) is equal to 0x6, and DSCP_IPV6_EN is set in P0_CONTROL_REG register:
   - If RX_REMAP_DSCP_V6 bit in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is clear then the destination hardware switch priority is the CPPI receive thread number.
   - If RX_REMAP_DSCP_V6 bit in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register is set then the destination hardware switch priority is the 6-bit priority (in the 6-bits following the upper nibble 0x6) mapped through the port's DSCP priority mapping registers (IPV6 packet). Port 1 transmit remapping (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TX_PRI_MAP_REG should remain the default value) is not compatible with this bit being set, but remapping can be configured on port 0 receive.
4. Else the ingress packet is non-tagged and the destination hardware switch priority is the CPPI receive thread number.

12.2.1.4.6.4 Packet CRC Handling

The P0_TX_CRC_REMOVE bit in the CPSW3_CONTROL_REG register determines if host port egress packets have CRC included or not. If P0_TX_CRC_REMOVE is set to 1h then all packets that are transmitted from port 0 do not contain CRC. If P0_TX_CRC_REMOVE bit is cleared to 0h then all packets that are transmitted from port 0 contain CRC. The CRC type, if present, is determined by the CRC_TYPE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register. If the CRC_TYPE bit is cleared to 0h then the CRC present in each packet after host port egress is Ethernet CRC. If the CRC_TYPE bit is set to 1h then the CRC present in each packet after host port egress is Castagnoli CRC.

Note

The CRC type present in the packet after host port egress is determined solely by the CRC_TYPE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register regardless of the CRC type present in the packet during Ethernet port ingress.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.4.1 Transmit VLAN Processing

Transmit packets are NOT modified during switch egress when the VLAN_AWARE bit in the CPSW3_CONTROL_REG register is cleared to 0h. This means that the switch is not in VLAN-aware mode.

The next three sections cover transmit processing when the switch is in VLAN-aware mode for different packet types. The Gigagibit Ethernet switch is in VLAN-aware mode when the VLAN_AWARE bit is set in the CPSW3_CONTROL_REG register. While in VLAN-aware mode, VLAN is added, removed, or replaced according to the type of packet as well as the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit in the packet header as explained below.

12.2.1.4.6.4.1.1 Untagged Packets (No VLAN or Priority Tag Header)

Untagged packets are all packets that are not a VLAN packet or a priority tagged packet. According to the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit in the packet header the packet may exit the switch with a VLAN tag inserted or the packet may leave the switch unchanged. The two cases are discussed below.

- Insert VLAN Case:

  Untagged input packets have the header packet VLAN inserted when the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit in the transmit packet header is de-asserted. For untagged packets, the VLAN EtherType = 0x8100 is inserted after the source address followed by the two byte header packet VLAN. The header packet VLAN is composed of the header packet priority along with the PORT_CFI and and PORT_VID values from the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_PORT_VLAN_REG register (where N is the port that the untagged packet entered the switch) through. The packet length/type field is output four bytes later than it is input and is not removed or replaced. If the CRC is present in the packet data (PASS_CRC is asserted), then the packet CRC is replaced with a hardware generated CRC.

- No Change Case:

  Untagged input packets are output unchanged when the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS transmit packet header bit is asserted.

12.2.1.4.6.4.1.2 Priority Tagged Packets (VLAN VID == 0 && EN_VID0_MODE ==0h)

Priority tagged packets are packets that contain a VLAN header with VID = 0. According to the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit in the packet header, priority tagged packets may exit the switch with their VLAN ID and priority replaced or they may have their priority tag completely removed. The two cases are discussed below.

Note

In order for a priority tagged packet to fall into this category the ENABLE_VID0_MODE bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register must also be set to 0h. If the ENABLE_VID0_MODE bit in the CPSW3_CPSW_NU_ALE_ALE_CONTROL register is set to 1h, then packets with a VLAN VID of 0 will fall into the VLAN Tagged Packets category in Section 12.2.1.4.6.4.1.3 below.

- Replace Priority and VLAN ID Case:

  Priority tagged input packets have the packet VLAN ID and the packet priority replaced with the header packet VLAN ID and the header packet priority when the transmit packet header CPSW_FORCE_UNTAGGED_EGRESS_REG[1-0] MASK bit is de-asserted. The header packet VLAN ID comes from the PORT_VID bits in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_PORT_VLAN_REG register (where N is the port where the packet entered the switch). The header packet priority is based on the packet priority to header packet

Copyright © 2026 Texas Instruments Incorporated

priority mapping in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG register (where N is the port where the packet entered the switch).

- Remove VLAN Header Case:

  Priority tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority-tagged packets use all four input CRC bytes as pad, input 65-byte priority-tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS_CRC input bit is not asserted - input 64-67 byte (on the wire) priority-tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

12.2.1.4.6.4.1.3 VLAN Tagged Packets (VLAN VID != 0 || (EN_VID0_MODE ==1h && VLAN VID ==0))

VLAN tagged packets are packets that contain a VLAN header specifying the VLAN the packet belongs to (VID), the packet priority (PRI), and the drop eligibility indicator (CFI). According to the CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit in the packet header, VLAN tagged packets may exit the switch with their VLAN priority replaced or they may have their VLAN header completely removed. The two cases are discussed below.

- Replace Priority Case:

  VLAN tagged input packets are output with the packet priority replaced with the header packet priority when the transmit packet header CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit is deasserted. The header packet priority is based on the packet priority to header packet priority mapping in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_PRI_MAP_REG register (where N is the port where the packet entered the switch). If the CRC is present in the packet data (PASS_CRC is asserted), then the packet CRC is replaced with a generated CRC.

- Remove VLAN Header Case:

  VLAN tagged input packets have the 4-byte packet VLAN information removed when the transmit packet header CPSW3_CPSW_NU_ALE_ALE_UVLAN_UNTAG[1-0] UVLAN_FORCE_UNTAGGED_EGRESS bit is asserted. The 0x8100 EtherType is removed as is the two byte packet VLAN. Input 64-67 byte priority tagged packets go out with the VLAN removed and padded to 64-bytes if the PASS_CRC input bit is asserted. The input CRC bytes are used as the pad data. Input 64-byte priority tagged packets use all four input CRC bytes as pad, input 65-byte priority tagged packets use three of the input CRC bytes as pad, and so on. No pad is performed if the PASS_CRC input bit is not asserted - input 64-67 byte (on the wire) priority tagged packets go out as 60-63 byte packets. The output CRC is replaced with a generated CRC when the VLAN is removed.

12.2.1.4.6.4.2 Ethernet Port Ingress Packet CRC

All Ethernet ports check the ingress packet CRC in all modes/speeds. The receive port can check either Ethernet CRC or Castagnoli CRC as determined by the CRC_TYPE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register.

12.2.1.4.6.4.3 Ethernet Port Egress Packet CRC

Ethernet ports transmit each egress packet with the CRC selected by the CRC_TYPE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register, regardless of the type of CRC that the packet had on ingress to the switch. At the egress port after passing through the switch, the packet CRC is checked for correctness and if the CRC is correct then the packet is output with the generated selected output CRC. If the packet CRC is incorrect, due either to a bit flip in a memory or an error CRC passed in on host ingress, then the generated egress CRC type is used with at least a single byte of the CRC inverted to indicate the error. If the packet length including CRC is divisible by 4 then all 4 CRC bytes will be inverted on

Copyright © 2026 Texas Instruments Incorporated

error. If there are three bytes remainder after dividing the packet length by 4 then three bytes will be inverted (and so on down to one byte remainder).

12.2.1.4.6.4.4 CPPI Port Ingress Packet CRC

CPPI port ingress packets can be passed in with or without a CRC. The host port is Ethernet CRC only. CPPI ingress packets are not checked for CRC correctness on CPPI ingress, however they are checked for correctness on Ethernet egress and are output with a CRC error if they came in with a CRC error. If a CPPI ingress packet does not have the INFO0 PASSED_CRC bit set then a CRC will be generated for the packet on CPPI ingress. If the PASSED_CRC bit is set then the packet is received and forwarded unchanged. The CRC type is input in the INFO0 word CRC_TYPE bit and must be Ethernet CRC.

12.2.1.4.6.4.5 CPPI Port Egress Packet CRC

The P0_TX_CRC_REMOVE bit in the CPSW3_CONTROL_REG register determines if CPPI egress packet have an Ethernet CRC included or not.

12.2.1.4.6.5 FIFO Memory Control

Each of the two CPSW_3G ports has an identical associated FIFO. Each FIFO contains a single logical receive queue and eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each FIFO memory contains 20,480 bytes (20k) total organized as 2560 by 64-bit words contained in a single memory instance. The FIFO memory is used for the associated port transmit and receive queues. The TX_MAX_BLKS field in the FIFOs associated CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG register determines the maximum number of 1k FIFO memory blocks to be allocated to the eight logical transmit queues (transmit total). The RX_MAX_BLKS field in the FIFO's associated CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG register determines the maximum number of 1k memory blocks to be allocated to the logical receive queue. The TX_MAX_BLKS value plus the RX_MAX_BLKS value must sum to 20 (the total number of blocks in the FIFO). If the sum were less than 20, then some memory blocks would be unused. The default is 17 (decimal) transmit blocks and three receive blocks. The FIFOs follow the naming convention of the Ethernet ports. Host Port is Port0 and External Ports is Port1.

Each transmit FIFO contains a configurable number of blocks (20 max) that are either 1k or 4k byte blocks that can be allocated to any priority.

12.2.1.4.6.6 FIFO Transmit Queue Control

There are eight transmit queues in the Ethernet port transmit FIFO. Software has some flexibility in determining how packets are loaded into the queues and on how packet priorities are selected for transmission (how packets are removed and transmitted from queues).

12.2.1.4.6.6.1 CPPI Port Receive Rate Limiting

Port 0 receive operations can be configured to rate limit the host ingress data for each receive thread (priority). CPPI Receive has 8 threads for QOS. There is a committed information rate (CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_CIR_REG_y, where y = 0 to 7) and an excess information rate for each thread (CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_EIR_REG_y, where y = 0 to 7). Rate limiting is enabled for a thread when the committed information rate for the thread is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. No bulk thread will be enabled to send unless there are CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_PRI_CTL_REG[15-12] TX_HOST_BLKS_REM number of unused blocks remaining in each of the Ethernet port transmit FIFOs. The “blocks remaining check” ensures that bulk traffic from the host will not block rate-limited traffic from the host. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then threads 7 and 6 should be configured for committed information (and excess information if desired). When any channels are configured

Copyright © 2026 Texas Instruments Incorporated

to be rate-limited, the priority type must be fixed for receive. Round-robin priority type is not allowed when rate-limiting is configured for any thread. The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to receive is controlled by the below equation. If the configured excess information rate is zero, then only the committed information rate is transferred:

Priority Transfer rate [Mbit/s] = ((((Frequency in MHZ) * CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_CIR_REG_y) / 32768) + (((Frequency in MHZ) * CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_EIR_REG_y) / 32768))

Where the *frequency* is the CPPI_ICLK frequency (in MHz) and priority 0 to 7.

For example, 10Mbps on priority 7 would give the below:

10Mbps = ~ ((350 * 936) / 32768), at 350Mhz and CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_CIR_REG_y[27-0] PRI_CIR value = 936 (no excess information rate)

12.2.1.4.6.6.2 Ethernet Port Transmit Rate Limiting

Ethernet port transmit operations can be configured to rate limit egress data for each egress priority. There is a committed information rate (CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_CIR_REG_y, where y = 0 to 7) and an excess information rate for each priority (CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_EIR_REG_y, where y = 0 to 7). Rate limiting is enabled for a priority when the committed information rate for the priority is non-zero. The excess information rate for a priority is enabled when the excess information rate for the priority is non-zero. The committed information rate must be non-zero if the excess information rate is configured to be non-zero. That is, there must be a configured non-zero committed information rate for there to be a configured non-zero excess information rate. Bulk traffic on other non-rate limited priorities does not impact the committed information traffic on a priority. However, bulk traffic on other non-rate limited threads does impact the excess information rates. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then priorities 7 and 6 should be configured for committed information (and excess information if desired). The configured transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is configured to send is controlled by the below equation. If the excess information rate is disabled then the committed information rate only is transferred:

Priority Transfer rate [Mbit/s] = ((((Frequency in MHZ) * CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_CIR_REG_y) / 32768) + (((Frequency in MHZ) * CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_PRI_EIR_REG_y) / 32768))

Where the *frequency* is the CPPI_ICLK frequency (in MHz) and priority 0 to 7.

12.2.1.4.6.7 Intersperced Express Traffic (IET – P802.3br/D2.0)

When IET is enabled through CPSW3_CONTROL_REG[17] IET_ENABLE = 1h and configured, IET allows premptable traffic on selected transmit priorities to be prempted by express traffic on selected express priorities. Received traffic is separated onto express and prempt receive queues. When IET is disabled (IET_ENABLE = 0h), all Ethernet traffic is express traffic and switch operation is as if IET does not exist. Traffic is only intended to be moved to the prempt queue after premption is verified and enabled.

12.2.1.4.6.7.1 IET Configuration

1. Using the prempt receive queue requires more blocks to be allocate to the ports receive FIFO. Write a value of decimal 7 to the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG[7-0] RX_MAX_BLKS bit field, and a value of decimal 13 to the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG [15-8] TX_MAX_BLKS register for every port to be enabled for IET.
2. Write the [23-0]MAC_VERIFY_CNT bit field in the Ethernet port CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_VERIFY_REG register to set the verify/response timeout count. The default is 10ms for Gigabit mode. For other time values or link speeds the verify

Copyright © 2026 Texas Instruments Incorporated

count should be updated. If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG[2] MAC_DISABLEVERIFY bit is to be set (this is forced mode) then this step is unneeded.

3. The receive FIFO block allocation is insufficient if CPSW3_CPSW_NU_CPSW_NU_STAT_RX_BOTTOM_OF_FIFO_DROP[31-0] COUNT is nonzero, indicating that receive packets are being dropped due to FIFO block allocation.
4. Write the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG register in the Ethernet port as below:
   - Set the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CONTROL_REG[16] IET_PORT_EN bit. The port will not actually be enabled until bit IET_ENABLE is set in the CSL_CPSW3_CONTROL_REG.
   - The CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG[0] MAC_PENABLE bit can be set as desired. No effect will occur until IET_ENABLE is set. This bit enables premptable packets to be prempted by express traffic but does not preclude packets from being sent to the prempt queue.
   - If verify/response is desired then CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG[3] MAC_LINKFAIL should be cleared by software to enable verify and response packets. Otherwise, MAC_DISABLEVERIFY bit should be set for forced mode. Verification and response will occur immediately after clearing this bit.
   - Configure the remaining CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG register bits as desired.
5. Set the IET_ENABLE bit in the CSL_CPSW3_CONTROL_REG register to enable IET operations.
6. After preemption has been verified, the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG[23-16] MAC_PREMPT field is written to configure the FIFO priorities to be sent to the prempt queue (the other priorities with cleared bits go to the express queue). The hardware switch for each queue from express to prempt happens only when there are no packets queued on the priority.

12.2.1.4.6.8 Enhanced Scheduled Traffic (EST – P802.1Qbv/D2.2)

12.2.1.4.6.8.1 Enhanced Scheduled Traffic Overview

- When enabled and configured, EST allows express queue traffic to be scheduled (placed) on the wire at specific repeatable time intervals.
- EST operates on a repeating time interval generated by the CPTS EST function generator. For example, a 125us repeating time interval can be configured.
- Each Ethernet port has 128 EST fetch commands maximum in the global EST fetch RAM.
- Each 22-bit fetch command consists of a 14-bit fetch count (14 MSB's) and an 8-bit priority fetch allow (8 LSB's) that will be applied for the fetch count time in wireside clocks.
- The configured port fetch commands are executed in sequence, beginning at port address zero each time through the time interval beginning at cycle start.
- EST allows non-scheduled express and prempt queue traffic to be cleared from the wire to ensure that the scheduled traffic is transmitted at the proper time (zero allow).
- EST can be used with or without premption. The CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG[23-16] MAC_PREMPT value determines whether the priority is enabled on the express or prempt queue. Whether a priority is on the express or prempt queue only effects the wire clear time from an EST operation perspective.
- Software should not move priorities to the prempt queue unless preemption is configured, enabled, and verified allowing preemption to occur.
- Express packet time stamp events can be enabled to assist software in configuring and timing EST operations.

12.2.1.4.6.8.2 Enhanced Scheduled Traffic Fetch RAM

- The EST fetch RAM is read/writable in the CPSW configuration address space.
- The Ethernet transmit port has 128 locations in the global EST fetch RAM.
  - Ethernet port 1 has EST fetch RAM addresses 0x000-0x07F.

Copyright © 2026 Texas Instruments Incorporated

- **One buffer operation:** When CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[0] EST_ONEBUF is set to 1h, the 128 port locations operate as one buffer. The EST_BUFACT bit in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_FIFO_STATUS_REG register is the upper address bit of the port's fetch RAM address indicating whether operation is currently in the upper or lower 64 locations of the port's fetch RAM.
- **Two buffer operation:** When CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[0] EST_ONEBUF is cleared there are two 64-location buffers with CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[1] EST_BUFSEL selecting the buffer to be used. When the buffer is switched by changing the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[1] EST_BUFSEL value, the actual switch occurs on cycle start. The actual buffer being used is indicated by the EST_BUFACT bit in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_FIFO_STATUS_REG. Software should avoid writing the switched out buffer fetch RAM locations until it detects that the actual switch has occurred.
- The first address location in the port's fetch RAM space (location zero) is read at the beginning of each EST time interval (cycle start). Addresses are then read in ascending order for the duration of the interval. The port address zero is then read again at the beginning of the next cycle repeating the time interval packet operations.

12.2.1.4.6.8.3 Enhanced Scheduled Traffic Time Interval

- Each Ethernet port has an Enhanced Scheduled Traffic Function (ESTF) generator in the CPTS submodule.
- The EST function generator generates the EST time interval as a configured number of CPTS reference clocks (CPTS_RFT_CLK).
- The EST function generator rising edge is the cycle start time and the cycle repeats (cycle start occurs) after every time interval.
- The first fetch allowed value is at the port base address zero in the EST fetch RAM and is actually applied 16 wireside clocks after cycle start. The 16 clock cycle delay allows the first fetch value time to be fetched from the EST fetch RAM (prefetch time at cycle start).
- Each successive fetch allow is applied for the associated fetch count thereafter. The minimum non-zero fetch count is 16. The minimum value of 16 guarantees that the next fetch value has time to be fetched before the current fetch count is over. There are 64 maximum fetch values when CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[0] EST_ONEBUF = 0h, and 128 maximum fetch values when CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[0] EST_ONEBUF = 1h.
- The next cycle start then causes the fetch to once again start at the port address zero.

12.2.1.4.6.8.4 Enhanced Scheduled Traffic Fetch Values

- The 22-bit fetch value is made up of the 14-bit fetch count and the 8-bit fetch allow.
- The fetch time indicates the number of wireside clocks that the fetch allow will be active.
- The fetch count is in Ethernet wireside clocks which is bytes in Gigabit mode (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[7] GIG = 1h) and nibbles in 10/100Mbps mode.
- When a fetch allow bit is set, the corresponding priority is enabled to begin packet transmission on an allowed priority subject to rate limiting. The actual packet transmission on the wire may carry over into the next fetch count and is the reason for the wire clear time in the fetch zero allow.
- When a fetch allow bit is cleared, the corresponding priority is not enabled to transmit for the fetch count time.
- A non-zero fetch allow value with a non-zero fetch count causes the fetch allow value to be applied for the fetch count number of wireside clocks.
- A zero fetch count causes the associated fetch allow to be held for the duration of the cycle (until the next cycle start).
- A zero fetch allow with a non-zero fetch count is intended to clear the wire for a scheduled (timed) express packet in the next fetch. A zero fetch allow indicates that no packet can be started for transmission for the associated fetch count. The associated fetch count must be sufficient to guarantee that the wire is cleared given that a packet on an allowed priority in the previous fetch could have been started on the previous clock

Copyright © 2026 Texas Instruments Incorporated

and that there is hardware latency in the clear time. The timed packet should be sent on a priority that is enabled in the next fetch but disabled in the current zero allow fetch. The fetch allow previous to a zero allow should have only prempt priorities enabled or only express priorities enabled but not both.
- The number of clocks required to clear the wire varies depending Ethernet wire speed and on whether express or prempt priorities were allowed in the previous fetch command.

12.2.1.4.6.8.5 Enhanced Scheduled Traffic Packet Fill

Packet fill can (should) be configured and enabled to occur in the fetch count time associated with a fetched zero allow that preceeds a timed express packet. The intention with fill is that a smaller packet on a non-timed priority might be able to be inserted on the wire during the wire clear time which would increase wire utilization. Fill must be configured to ensure that any fill packet does not conflict with the timed express packet allowed in the next fetch. Incorrect configuration might push out in time any express timed packet which indicates that the fill margin needs to be increased.

Fill Comfiguration:
- The **pn_est_fill_margin** value in **Enet_Pn_EST_Control** should be written with a 0x100 value.
- The **pn_est_prempt_comp** value in **Enet_Pn_EST_Control** should be written with a 0x12 value (if IET is to be configured and enabled). This value times eight is the number of wireside clocks required to clear prempt packets off the wire at the end of a zero allow.
- The **pn_est_fill_en** bit in **Enet_Pn_EST_Control** should be set.

12.2.1.4.6.8.6 Enhanced Scheduled Traffic Time Stamp

The EST can be configured to generate CPTS timestamp events for selected express traffic. The EST timestamp events use the CPTS host event type (CPSW3_CPSW_NU_CPTS_EVENT_1_REG[23-20] EVENT_TYPE = 7 decimal. The EST timestamps will not override host sent timestamps for packets that were sent from the host with an enabled host timestamp.

- EST Events (host events) contain the below information:
  - Time Stamp of the selected express packet.
  - The event CPSW3_CPSW_NU_CPTS_EVENT_1_REG[28-24] PORT_NUMBER indicates the transmit port number.
  - The event CPSW3_CPSW_NU_CPTS_EVENT_1_REG[23-20] EVENT_TYPE is decimal 7 (host event).
  - The event CPSW3_CPSW_NU_CPTS_EVENT_1_REG[23-20] MESSAGE_TYPE indicates the packet transmit hardware switch priority.
  - The event CPSW3_CPSW_NU_CPTS_EVENT_1_REG[15-0] SEQUENCE_ID upper nibble indicates the packet receive port number.
  - The event CPSW3_CPSW_NU_CPTS_EVENT_1_REG[15-0] SEQUENCE_ID lower byte indicates the sequence number of the express packet in numerical order. The first event is event one, the second is event two and so on. The sequence ID rolls over to zero after 0xFF (8-bits).
  - The event domain is the value from the CPSW3_CPSW_NU_EST_TS_DOMAIN_REG[7-0] EST_TS_DOMAIN register.
- When CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[2] EST_TS_EN is set, timestamp events will be generated on selected express traffic.
- When CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[3] EST_TS_FIRST is also set, events will be generated only on the first express packet in each time interval. If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[4] EST_TS_ONEPRI is also set then the event will only be on the first CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[7-5] EST_TS_PRI express packet in the time interval. If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[4] EST_TS_ONEPRI is clear then the event will be generated on the first express packet in the time interval on any priority.
- When CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[3] EST_TS_FIRST is clear, events will be generated on every express packet. If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[4] EST_TS_ONEPRI

Copyright © 2026 Texas Instruments Incorporated

is set then the event will be generated on every CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[7-5] EST_TS_PRI express packet. If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_EST_CONTROL_REG[4] EST_TS_ONEPRI is clear then event will be generated on every express packet on any priority.

12.2.1.4.6.8.7 Enhanced Scheduled Traffic Packets Per Priority

The number of packets allowed in a transmit FIFO priority can be selected by writing a non-zero value to CPSW_P0_RX_PKTS_PRI_REG register (packet priority 0 to 7). Then port 0 receive gap should then be enabled by setting the corresponding priority through CPSW_P0_RX_GAP_REG[7-0] RX_GAP_EN bit field. The receive gap allows a packet to land in the transmit FIFO before another packet is allowed in which guarantees that only the selected number (max) of packets is allowed in on the specified priority. If the receive gap is not enabled, then there might be one or two more packets allowed in on the priority/thread than the packets per priority value has selected (through CPSW_P0_RX_PKTS_PRI_REG register).

12.2.1.4.6.9 Audio Video Bridging

Audio Video Bridging is an ongoing project of IEEE 802.1 concerned with enabling low-latency streaming of time-sensitive audiovisual data over networks. Devices are designated as talkers (transmitters), bridges, or listeners (receivers). It is suggested that the maximum latency could be 2 ms over 7 hops for Class A devices and 20 ms over 7 hops for Class B devices. A hop is essentially a single local area network stage in the journey of a packet. Every time a bridge is encountered between one network section and another a hop is involved. One of the performance goals is that AVB streams will not use more than 75 percent of a link's bandwidth, leaving the remaining capacity for non-AVB streams.

The goal of developing AVB is simply--extend Ethernet's data-networking capabilities to the realm of reliable real-time audio/video networking.

An "Audio Video Bridging" network is one that implements a set of protocols being developed by the IEEE 802.1 Audio/Video Bridging Task Group. There are four primary differences between the proposed Audio Video Bridging architecture and existing 802 architectures (from now on the term "AVB" will be used instead of "Audio Video Bridging"):

1. Precise synchronization - IEEE 802.1AS: "*Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.* "a.k.a Precision Time Protocol (PTP).
2. Traffic shaping for media streams - IEEE 802.1Qav: "*Virtual Bridged Local Area Networks:* Forwarding and Queuing for Time-Sensitive Streams."
3. Admission controls - IEEE 802.1Qat: "*Virtual Bridged Local Area Networks - Amendment 9:* Stream Reservation Protocol (SRP)."
4. Identification of non-participating devices - IEEE 802.1BA: "Audio/Video Bridging (AVB) Systems"

Copyright © 2026 Texas Instruments Incorporated

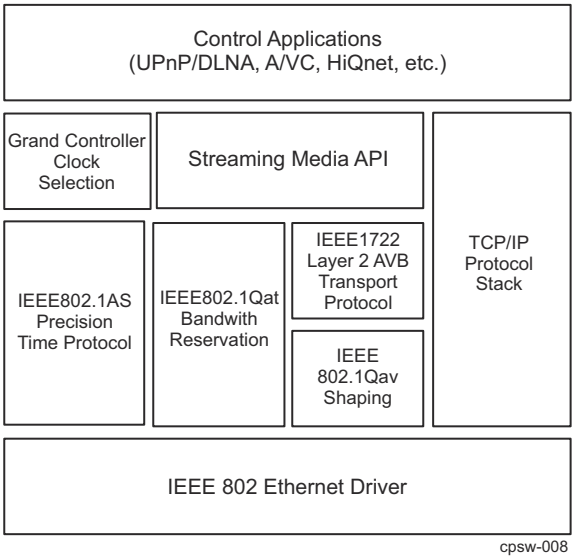


Figure 12-97. The Network Static with AVB

The following sections describe the media transport protocols that work within the AVB framework.

12.2.1.4.6.9.1 IEEE 802.1AS: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks (Precision Time Protocol (PTP))

The protocol defined by 802.1AS automatically selects a device to be the controller clock, and then distributes this clock throughout the bridged LAN / IP subnet to all other network devices using link-specific transmit/receive time-stamping. However, we only use a two-step solution only on transmit. That is, we do not modify a packet with the timestamp on the way out. The timestamp packet is sent out and then a separate message with the timestamp is sent by the host afterward. Receive can be one or two step.

Note

The 802.1AS-distributed clock is not used as a media clock. Rather, the shared 802.1AS clock reference is used to regenerate the media clock at the listener/renderer. Such a reference removes the need to force the latency of the network to be constant, or compute long running averages in order to estimate the actual media rate of the transmitter in the presence of substantial network jitter. IEEE 802.1AS is based on the ratified IEEE 1588 standard.

Based on IEEE 1588:2002,A PTP devices exchange standard Ethernet messages that synchronize network nodes to a common time reference by defining clock controller selection and negotiation algorithms, link delay measurement and compensation, and clock rate matching and adjustment mechanisms.

Designed as a simplified profile of IEEE 1588, a primary difference between 1588 and IEEE 802.1AS is that PTP is a layer 2-in other words, a non-IP routable protocol. Like IEEE 1588, PTP defines an automatic method for negotiating the network clock controller, the Best Controller Clock Algorithm (BCCA). PTP nodes can be assigned one of eight priority levels, presumably based on clock quality. BCCA defines the underlying negotiation and signaling mechanism whose purpose is to identify the AVB LAN Grandcontroller. Once a Grandcontroller has been selected, synchronization automatically begins.

At the core of 802.1AS synchronization is time-stamping. In short, during PTP message ingress/egress from the 802.1AS-capable MAC, the PTP Ether type triggers the sampling of the value of a local real-time counter (RTC). Target nodes compare the value of their RTC against the PTP Grandcontroller and, by use of link delay measurement and compensation techniques, match their RTC value to the time of the AVB LAN PTP domain. After network time throughout the AVB LAN has converged, periodic SYNC and FOLLOW_UP messages provide the information that enables the PTP rate matching adjustment algorithms. The result is all PTP nodes are then synchronized to the same "Wall Clock" time. PTP assures 1-µs accuracy over seven network hops.

Copyright © 2026 Texas Instruments Incorporated

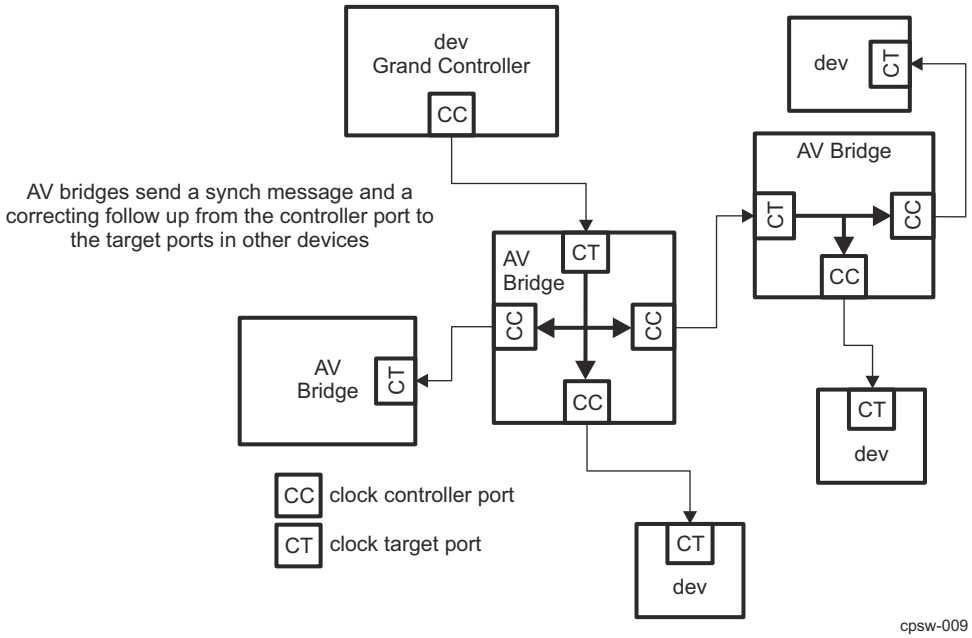


Figure 12-98. AVB Network & PTP Clock Entities

The media transport protocols that work within the AVB framework are:

12.2.1.4.6.9.1.1 IEEE 1722: "Layer 2 Transport Protocol for Time-Sensitive Streams"

AVBTP or 1722 sits above the IEEE 802.1 AVB plumbing and below the application layer. It acts as the conduit between an Ethernet MAC and a streaming application. AVBTP abstracts the underlying network transmission channel to enable the virtual connection of distributed audio and video CODECs over reliable Ethernet networks. A complete AVBTP Ethernet packet is shown in Figure 12-99 and illustrates how IEC 61883-6 AM824 uncompressed audio samples are encapsulated in an Ethernet frame.

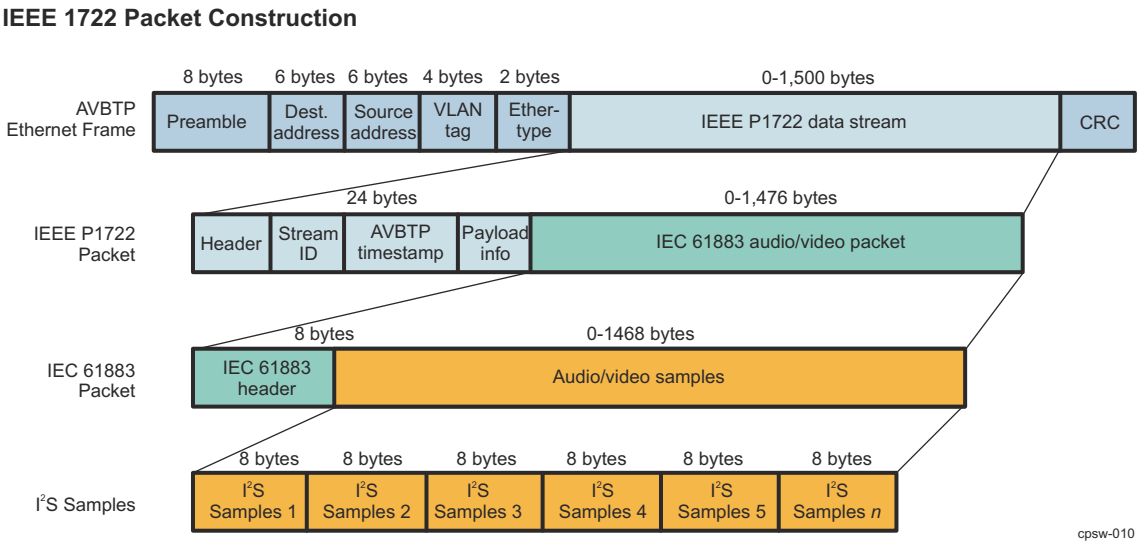


Figure 12-99. IEEE 1722 Packets

1722 or AVBTP Presentation Time and Synchronization:

Synchronization in an AVB network starts with the Precision Time Protocol but ends with synchronized media clocks. PTP is responsible for synchronizing all nodes in an AVB network to identical wall clock time; not for

Copyright © 2026 Texas Instruments Incorporated

synchronizing media clocks. In other words, PTP does not actually transport synchronized media clocks but instead provides a low-level building block crucial for managing a distributed media synchronization system.

A crucial benefit of this approach is coexistence of multiple, independent media clock domains on an AVB network. Unrelated audio and video streams can simultaneously exist in the same LAN.

12.2.1.4.6.9.1.1.1 Cross-timestamping and Presentation Timestamps

AVBTP assumes that AVB node media clocks are clocked by free-running oscillators. It is also assumed that the node's internal concept of wall clock time has been synchronized to the PTP Grandcontroller. AVBTP media clock sources embed "AVBTP Presentation Timestamps" in AVBTP streaming packets. Figure 12-100 illustrates the relationship between PTP network time and AVBTP Presentation Timestamps.

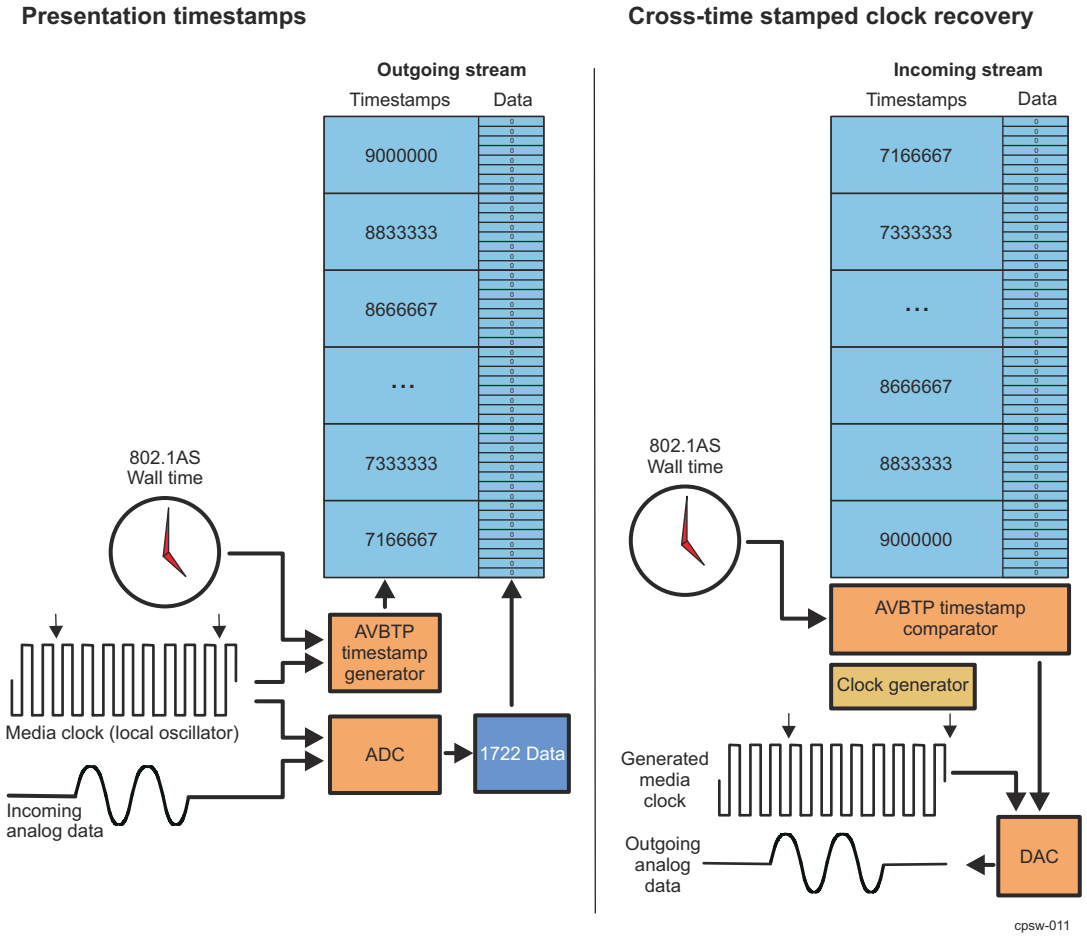


Figure 12-100. Cross Time Stamping and Presentation Timestamps

12.2.1.4.6.9.1.2 IEEE 1733: Extends RTCP for RTP Streaming over AVB-supported Networks

This standard specifies the protocol, data encapsulations, connection management and presentation time procedures used to ensure interoperability between audio and video based end stations that use standard networking services provided by all IEEE 802 networks meeting QoS requirements for time-sensitive applications by leveraging the Real-time Transport Protocol (RTP) family of protocols and IEEE 802.1 Audio/ Video Bridging (AVB) protocols.

12.2.1.4.6.9.2 IEEE 802.1Qav: "Virtual Bridged Local Area Networks: Forwarding and Queuing for Time-Sensitive Streams"

This standard allows bridges to provide guarantees for time-sensitive (that is, bounded latency and delivery variation), loss-sensitive real-time audio video (AV) data transmission (AV traffic). It specifies per priority ingress

Copyright © 2026 Texas Instruments Incorporated

metering, priority regeneration, and timing-aware queue draining algorithms. This standard uses the timing derived from IEEE 802.1AS. Virtual Local Area Network (VLAN) tag encoded priority values are allocated, in aggregate, to segregate frames among controlled and non-controlled queues, allowing simultaneous support of both AV traffic and other bridged traffic over and between wired and wireless Local Area Networks (LANs).

Such a guarantee in bandwidth is provided by two functional entities:

- A registration protocol, which registers the service and its maximum network utilization with a device or switch (IEEE 802.1Qat: "Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)")
- A hardware bandwidth management service.
  - Receive policing
  - Transmit rate control.

End Station Behavior

In order for an end station to successfully participate in the transmission and reception of time-sensitive streams, it is necessary for their behavior to be compatible with the operation of the forwarding and queuing mechanisms employed in bridges.

The requirements for end stations that participate as "talkers" i.e., sources of time-sensitive streams are different from the requirements that apply to "listeners", the destination station(s) for the streams.

Talker Behavior

In order for Talker-originated data streams to make use of the credit-based shaper behavior in Bridges, it is a requirement for a Talker to use the priorities that the Bridges in the network recognize as being associated with SR classes exclusively for transmitting stream data.

It is also necessary for the Talker and the Bridges in the path to the Listener(s), to have a common view of the bandwidth required in order to transmit the Talker's streams, and for that bandwidth to be reserved along the path to the Listener(s). This latter requirement can be met by means of stream reservation mechanisms, such as defined in SRP, or by other management means.

End stations that are Talkers shall exhibit transmission behavior for frames that are part of "time-sensitive streams" that is consistent with the operation of the credit-based shaper algorithm, both in terms of the way they transmit frames that are part of an individual data stream, and in terms of the way they transmit stream data frames from a Port.

In effect, the queuing model for a Talker Port (and a Listener port), and for given priorities, can be considered to look like Figure 12-101.

Listener Behavior

The primary requirement for a listener station is that it is capable of buffering the amount of data that could be transmitted for a stream during a time period equivalent to the accumulated maximum jitter that could be experienced by stream data frames in transmission between Talker and Listener.

From the point of view of the specification of the forwarding and queuing requirements for time-sensitive streams, it is assumed that the listener will assess the buffering required for a stream as part of the stream bandwidth reservation mechanisms employed by the implementation.

The credit-based shaper's operation details are beyond the scope of this document.

Copyright © 2026 Texas Instruments Incorporated

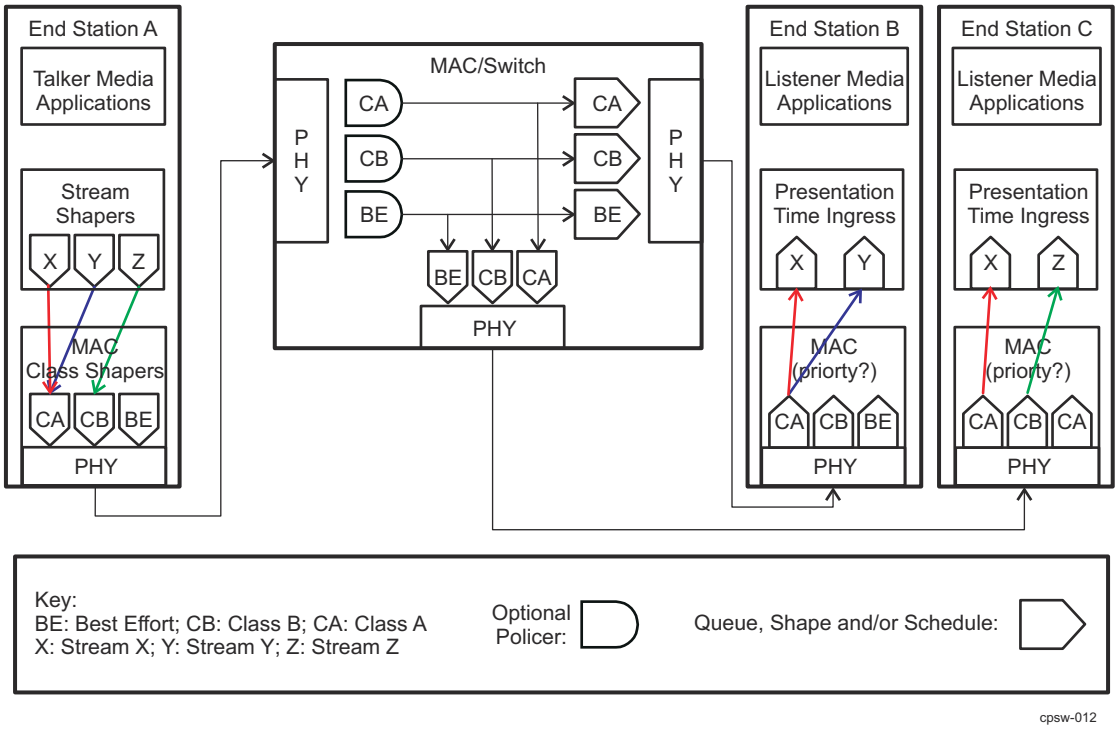


Figure 12-101. AV Stream Queuing/Policing

12.2.1.4.6.9.2.1 Configuring the Device for 802.1Qav Operation

There is no dedicated register-set to be configured for the time-sensitive stream handling. The list of functional features of CPSW_3G that will have to be configured are:

- DESCRIPTORS and CHANNEL CONFIGURATIONS:
  - CPPI TX and RX descriptors
  - VLAN and Priority tags

Table 12-123. Example of TX Configuration

TX DMA CHANNEL	Packet Priority	Switch Queue Priority
7	7	3
6	5	2
5	3	1
4	1	0

Table 12-124. Example of RX Configuration

RX DMA CHANNEL	Packet Priority	Switch Queue Priority
0	7	0
0	5	0
0	3	0
0	1	0

- ALE Configuration:
  - ALE in VLAN-ware mode, Non-ALE in bypass mode.

12.2.1.4.6.10 Ethernet MAC Sliver

The Ethernet port peripheral is compliant to the IEEE Std 802.3 Specification. Half-duplex mode is supported in 10/100 Mbps mode, but not in 1000 Mbps (gigabit) mode.

Copyright © 2026 Texas Instruments Incorporated

Features:

- Synchronous 10/100/1000 Mbit operation
- RMII/RGMII Interface
- Hardware Error handling including CRC
- Full-Duplex Gigabit operation (half-duplex gigabit is not supported)
- EtherStats and 802.3Stats RMON statistics gathering support for external statistics collection module
- Transmit CRC generation selectable on a per channel basis
- Emulation Support
- VLAN Aware Mode Support
- Hardware flow control
- Programmable Inter Packet Gap (IPG).

12.2.1.4.6.10.1 Ethernet MAC Sliver Overview

12.2.1.4.6.10.1.1 CRC Insertion

The MAC generates and appends a 32-bit Ethernet CRC onto the transmitted data, if the transmit packet header PASS_CRC bit is 0h. For the Ethernet port generated CRC case, a CRC at the end of the input packet data is not allowed.

If the header word PASS_CRC bit is set, then the last four bytes of the TX data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the packet byte count value. The MAC performs no error checking on the outgoing CRC when the PASS_CRC bit is set.

12.2.1.4.6.10.1.2 MTXER

The MTXER signal is only used for EEE. If an underflow condition occurs on a transmitted frame, the frame CRC will be inverted to indicate the error to the network. Underflow is a hardware error.

12.2.1.4.6.10.1.3 Adaptive Performance Optimization (APO)

The Ethernet MAC port incorporates Adaptive Performance Optimization (APO) logic that may be enabled by setting the TX_PACE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register. Transmission pacing to enhance performance is enabled when set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions) thereby increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision or excessive collision) the pacing counter is decremented by one, down to zero.

With pacing enabled, a new frame is permitted to immediately (after one IPG) attempt transmission only if the pacing counter is zero. If the pacing counter is non zero, the frame is delayed by the pacing delay, a delay of approximately four inter-packet gap delays. APO only affects the IPG preceding the first attempt at transmitting a frame. It does not affect the back-off algorithm for re-transmitted frames.

12.2.1.4.6.10.1.4 Inter-Packet-Gap Enforcement

The measurement reference for the IPG of 96-bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision, and MCRS is de-asserted within approximately 48-bit times of MTXEN being de-asserted, then 96-bit times is measured from MTXEN. If the frame suffered a collision, or if MCRS is not de-asserted until more than approximately 48-bit times after MTXEN is de-asserted, then 96-bit times (approximately, but not less) is measured from MCRS.

The Ethernet port transmit inter-packet gap (IPG) may be shortened by eight bit times when short gap is enabled and triggered. Setting the [10] TX_SHORT_GAP_ENABLE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register enables the gap to be shortened when triggered. The condition is triggered when the ports associated transmit packet FIFO has a user defined number of FIFO blocks used. The associated transmit FIFO blocks used value determines if the

Copyright © 2026 Texas Instruments Incorporated

gap is shortened, and so on. The CPSW3_CPSW_NU_GAP_THRESH_REG register value determines the short gap threshold. If the FIFO blocks used is greater than or equal to the GAP_THRESH value then short gap is triggered.

12.2.1.4.6.10.1.5 Back Off

The Gigabit Ethernet Mac Sliver implements the 802.3 binary exponential back-off algorithm.

12.2.1.4.6.10.1.6 Programmable Transmit Inter-Packet Gap

The transmit inter-packet gap (IPG) is programmable through the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register. The default value is decimal 12. The transmit IPG may be increased to the maximum value of 1FFh. Increasing the IPG is not compatible with transmit pacing. The short gap feature will override the increased gap value, so the short gap feature may not be compatible with an increased IPG.

12.2.1.4.6.10.1.7 Speed, Duplex and Pause Frame Support Negotiation

The Ethernet port can operate in half duplex or full duplex in 10/100 Mbit modes, and can operate in full duplex only in 1000 Mbit mode. Pause frame support is included in 10/100/1000 Mbit modes as configured by the host.

12.2.1.4.6.10.2 RMII Interface

The CPRMII peripheral is compliant to the RMII specification document.

12.2.1.4.6.10.2.1 Features

- Source Synchronous 10/100 Mbit operation
- Full and Half Duplex support

12.2.1.4.6.10.2.2 RMII Receive (RX)

The CPRMII receive (RX) interface converts the input data from the external RMII PHY (or switch) into the required MII (CPGMAC) signals. The carrier sense and collision signals are determined from the RMII input data stream and transmit inputs as defined in the RMII specification.

An asserted RMII_RXER on any di-bit in the received packet will cause an MII_RXER assertion to the CPGMAC during the packet. In 10Mbps mode, the error is not required to be duplicated on 10 successive clocks. Any di-bit which has an asserted RMII_RXER during any of the 10 replications of the data will cause the error to be propagated.

Any received packet that ends with an improper nibble boundary aligned RMII_CRS_DV toggle will issue an MII_RXER during the packet to the CPGMAC. Also, a change in speed or duplex mode during packet operations will cause packet corruption.

The CPRMII can accept receive packets with shortened preambles, but 0x55 followed by a 0x5D is the shortest preamble that will be recognized (1 preamble byte with the start of frame byte). At least one byte of preamble with the start of frame indicator is required to begin a packet. An asserted RMII_CRS_DV without at least a single correct preamble byte followed by the start of frame indicator will be ignored.

12.2.1.4.6.10.2.3 RMII Transmit (TX)

The CPRMII transmit (TX) interface converts the CPGMAC MII input data into the RMII transmit format. The data is then output to the external RMII PHY.

The CPGMAC does not source the transmit error (MII_TXERR) signal. Any transmit frame from the CPGMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be de-asserted at all times and is not an input into the CPRMII module. Zeroes are output on RMII_TXD[1:0] for each clock that RMII_TXEN is de-asserted.

12.2.1.4.6.10.3 RGMII Interface

The CPRGMII peripheral is compliant to the RGMII specification document.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.10.3.1 Features

- Supports 1000/100/10 Mbps speed
- Full and Half Duplex support (CPGMAC supports only Full duplex in Gigabit mode).
- MII mode support
- Energy Efficient Ethernet Support

12.2.1.4.6.10.3.2 RGMII Receive (RX)

The CPRGMII receive (RX) interface converts the source synchronous DDR input data from the external RGMII PHY into the required G/MII (CPGMAC) signals.

12.2.1.4.6.10.3.3 In-Band Mode of Operation

The CPRGMII is operating in the in-band mode of operation when the RGMII_RX_INBAND input is asserted. RGMII_RX_INPUT is asserted by configuring the CTL_EN bit to 1h of the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register. The link status, duplexity, and speed are determined from the RGMII input data stream RXD[3:0] when RX_CTL is deasserted, as defined in the RGMII specification. The PHY might need to be configured beforehand to output in-band data. The in-band data is indicated as shown in Table 12-125.

Table 12-125. In-Band Data

RXD3	RXD[2:1]		RXD0
Duplex status:	Link Speed:	RXC_CLK Speed:	Link Status:
0h: half-duplex	0h: 10-Mbps mode	2.5 MHz	0h: Link is down
1h: full-duplex	1h: 100-Mbps mode	25 MHz	1h: Link is up
	2h: 1000-Mbps mode	125 MHz	
	3h: Reserved	Reserved	

12.2.1.4.6.10.3.4 Forced Mode of Operation

The CPRGMII is operating in the forced mode of operation when the RGMII_RX_INBAND input is deasserted by setting to 0h bit CTL_EN of the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register. In the forced mode of operation, the in-band data is ignored if present.
The link status is forced high, and the duplexity and speed are determined from the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[0] FULLDUPLEX and CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[7] GIG bits. If bit [7] GIG = 1h, then CPRGMII is operating in Gigabit mode. If bit [7] GIG is cleared (0h), then CPRGMII is operating in 100 Mbps mode.

12.2.1.4.6.10.3.5 RGMII Transmit (TX)

The CPRGMII transmit (TX) interface converts the CPGMAC G/MII input data into the DDR RGMII format. The DDR data is then output to the external PHY.

The CPGMAC does not source the transmit error (TXERR) signal. Any transmit frame from the CPGMAC with an error (underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRGMII module.

In 10/100 Mbps mode, the TXD[7:0] data bus uses only the lower nibble. The CPRGMII will output the lower nibble twice in 10/100 Mbps mode to avoid unnecessary signal switching.

Packets will be precluded from transmission through the CPRGMII module for 4096 transmit clocks after the rising edge of RGMII_LINK. Packet transmission will begin on the first TX_CTL rising edge after the 4096 transmit clock count has expired.

12.2.1.4.6.10.4 Frame Classification

Received frames are proper (good) frames if they are between 64 and CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_RX_MAXLEN_REG[13-0] RX_MAXLEN in length (inclusive) and contain no errors (code/align/CRC).

Copyright © 2026 Texas Instruments Incorporated

Received frames are long frames if their frame count exceeds the value in the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_RX_MAXLEN_REG/ CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_MAXLEN_REG register. The register reset (default) value is 1518 (decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames if their frame count is less than 64 bytes. Short frames that contain no errors are undersized frames. Short frames with CRC, code, or alignment errors are fragment frames. If RX_CSF_EN bit in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG is set to 1h, undersized frames from 33 to 63 bytes will be forwarded only to the host on a best effort basis (meaning that the ALE may or may not be able to keep up with the packet rate and the short packet may be dropped due to bandwidth limitations). If RX_CSF_EN and RX_CEF_EN in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG are set, fragment frames from 33 to 63 bytes will also be forwarded only to the host on a best effort basis. Ethernet port received frames shorter than 33 bytes are dropped in all cases.

A received long packet will always contain RX_MAXLEN number of bytes transferred to memory (if CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[22]RX_CEF_EN = 1h). An example with RX_MAXLEN = 1518 is:

- If the frame length is 1518, then the packet is not a long packet and there will be 1518 bytes transferred to memory.
- If the frame length is 1519, there will be 1518 bytes transferred to memory. The last three bytes will be the first three CRC bytes.
- If the frame length is 1520, there will be 1518 bytes transferred to memory. The last two bytes will be the first two CRC bytes.
- If the frame length is 1521, there will be 1518 bytes transferred to memory. The last byte will be the first CRC byte.

If the frame length is 1522, there will be 1518 bytes transferred to memory. The last byte will be the last data byte.

12.2.1.4.6.10.5 Receive FIFO Architecture

This section describes the architecture of the Ethernet port's receive FIFOs. Internal to the Gigabit Ethernet switch, all Ethernet ports have an identical associated packet FIFO. Each transmit packet FIFO contains eight logical transmit queues (priority 0 through 7 with 7 the highest priority). Each transmit FIFO memory contains 81,920 bytes total organized as 2560 by 256-bit words. Each FIFO also contains a single memory for the receive queue. Each receive FIFO memory contains a total of 32768 bytes total organized as 1024 by 256-bit words.

12.2.1.4.6.11 Embedded Memories

Table 12-126. Embedded Memories

Memory Type Description	Number of Instances	
Single-port 3072-word × 64 RAM	1	(Combined FIFO RAM)
Single-port 128-word × 28-bit RAM	1	1 (EST)

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.12 Memory Error Detection and Correction

The CPSW_3G error detection and correction logic uses the ECC Aggregator Module.

The CPSW3_ECC_ECC_VECTOR register is used to select which ECC RAM's status and control registers are currently being read or written as shown in Table 12-127. The CPSW FIFO RAMs implement ECC only on packet headers. The packet data is protected by Ethernet CRC. The ALE and EST RAMs have complete ECC as normal.

Table 12-127. ECC RAM to CPSW RAM Mapping

ECC RAM Number	CPSW RAM
0	ALE RAM
1	Port 0 FIFO RAM

12.2.1.4.6.12.1 Packet Header ECC

Only packet headers bits are protected by ECC in the FIFO RAMs. The ECC_ERR_CTRL1[31-0] ECC_ROW bit is used to activate a row address where force single-bit error or force double-bit error needs to be applied. ECC_ERR_CTRL2 [15-0] ECC_BIT1 is implemented to determine which bit of the header is flipped for an SEC error when the ECC_CRC_MODE bit is cleared in the CPSW3_CONTROL_REG register. The ECC status registers return the RAM row address where the single or double-bit error has occured (ECC_ERR_STAT2[31-0] ECC_ROW) along with the bit position in the RAM data that is in error (ECC_ERR_STAT1[31-16] ECC_BIT1 value). Forcing double-bit errors in testing can cause indeterminate operation if multiple used packet header bits are flipped given that only single-bit errors are fixed by the ECC logic. Header bits 207 down to 200 are not currently used in the CPSW and may be used to test double bit errors without the possibility of requiring a reset for the switch to recover from the double bit error. No header bits are flipped when ECC_CRC_MODE is set to 1h. Either the RX_ECC_ERR_EN (enable receive ECC error operations) or the TX_ECC_ERR_EN (enable transmit ECC error operations) bits must be set in the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register to test ECC header errors.

The header ECC code is stored in bits 255 down to 208. If any bit is flipped in the ECC code, the flipped bit will be corrected, but the index of the flipped bit will be reported as bit zero. This implies that when the aggregator reports that there is a SEC on bit 0, it can mean two things: either SEC on data bit 0 or SEC somewhere inside the ECC code. Any packet header with ECC error issues a pulse interrupt (ECC_PULSE_INTR) as does an ALE RAM ECC error.

12.2.1.4.6.12.2 Packet Protect CRC

Each packet received without error is passed through the CPSW_3G memories with a generated Ethernet protect CRC. The protect CRC is checked on egress for correctness and removed. If the CRC is correct (no RAM bit errors), then the packet is output with the selected port CRC type. If a protect CRC error is detected on host egress then the TXST_DROP signal will be asserted so that the packet is dropped to the host. If a protect CRC error is detected on Ethernet egress then the egress CRC will be generated on the packet and at least one byte of the CRC will be inverted on output. CRC memory protect errors do not assert the ECC_PULSE_INTR signal. CRC memory protect errors are counted in the associated port statistics registers and issue an interrupt on STAT_PEND_INTR if any CRC memory protect error occurs (and the statistics for that port are enabled). When the ECC_CRC_MODE bit in the CSL_CPSW3_CONTROL_REG register is set, the ECC_ERR_CTRL2 [15-0] ECC_BIT1 bit field will flip the associated column bit in any FIFO memory read operation, inducing a CRC protect error when the protect CRC is checked. No header bits are flipped when ECC_CRC_MODE is set. Either the RX_ECC_ERR_EN or the TX_ECC_ERR_EN bits must be set in the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register to test packet CRC errors.

12.2.1.4.6.12.3 Aggregator RAM Control

The ECC logic for each FIFO RAM (receive and transmit) is divided into eight separate ECC encoders/decoders that encode/decode 26-bits of data each. Each of the 8 encoders (0 to 7) generates 6-bits of ECC code (48 code bits total), and each of the eight decoders (0 to 7) checks 6-bits of ECC code across the 26-bits of data (208 data bits total). The 48-bits of ECC code are passed through the RAM in the upper 48 unused bits in the header word.

Copyright © 2026 Texas Instruments Incorporated

The header data bits and ECC code bits are shown in Table 12-128. The [15-0] ECC_BIT1 value returned on error is a 16-bit value that is the concatenation of 5 bits of zero, 3 bits of the encoder/decoder number (0 to 7), 3 bits of zero, and 5 bits of index into the indicated 26-bit encoder/decoder.

For example, an ECC_BIT1 value of 0x0308 is bit 8 of encoder/decoder 3, which is header bit 86 (that is, (26×3) + 8).

Table 12-128. ECC Submodule Header Data Bit to Encoder/Decoder Mapping

Header Data Bits	Encoder/Decoder
25:0	Encoder/Decoder 0 Data
51:26	Encoder/Decoder 1 Data
77:52	Encoder/Decoder 2 Data
103:52	Encoder/Decoder 3 Data
129:104	Encoder/Decoder 4 Data
155:130	Encoder/Decoder 5 Data
181:156	Encoder/Decoder 6 Data
207:182	Encoder/Decoder 7 Data
213:208	Encoder/Decoder 0 ECC
219:214	Encoder/Decoder 1 ECC
225:220	Encoder/Decoder 2 ECC
231:226	Encoder/Decoder 3 ECC
237:232	Encoder/Decoder 4 ECC
243:238	Encoder/Decoder 5 ECC
249:244	Encoder/Decoder 6 ECC
255:250	Encoder/Decoder 7 ECC

12.2.1.4.6.13 Ethernet Port Flow Control

The Ethernet port have flow control available for transmit and receive. Transmit flow control stops the Ethernet port from transmitting packets to the wire (switch egress) in response to a received pause frame. Transmit flow control does not depend on FIFO usage.

The Ethernet port have flow control available for receive operations (packet ingress). Ethernet port receive flow control is initiated when enabled and triggered. Packets received on an Ethernet port can be sent to the CPPI port. The destination port can trigger the receive Ethernet port flow control. An Ethernet destination port triggers another Ethernet receive flow control when the destination port is full.

When a packet is received on an Ethernet port interface with enabled flow control the below occurs:

- The packet will be sent to all ports that currently have room to take the entire packet.
- The packet will be retried until successful to all ports that indicate they don't have room for the packet.

The flow control trigger to the Ethernet port will be asserted until the packet has been sent, and there is room in the logical receive FIFO for packet runout from another flow control trigger (RX_BLK_CNT = 0h). Ethernet port receive flow control is disabled by default on reset. Ethernet port receive flow control requires that the RX_FLOW_EN bit in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG be set to 1h. When receive flow control is enabled on a port, the port's associated FIFO block allocation must be adjusted. The port RX allocation must increase from the default three blocks to accommodate the flow control runout. A corresponding decrease in the TX block allocation is required. If a sending port ignores a pause frame then packets may overrun on receive (and be dropped) but will not be dropped on transmit.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.13.1 Ethernet Receive Flow Control

For every Ethernet port to be configured for fullduplex receive flow control, write a value of decimal 7 to the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG[7-0] RX_MAX_BLKS bit field, and a value of decimal 13 to the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAX_BLKS_REG[15-8] TX_MAX_BLKS register. This re-allocation allows for flow control runout on the receive FIFO at the expense of FIFO memory on the Ethernet transmit side. 10/100Mbps half-duplex collision based receive flow control does not need this re-allocation. Receive flow control is enabled by the RX_FLOW_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register.

12.2.1.4.6.13.1.1 Collision Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the port is operating in half-duplex mode (FULLDUPLEX is cleared in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG). When receive flow control is enabled and triggered, the port will generate collisions for received frames. The jam sequence transmitted will be the twelve byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3 (hex). The jam sequence will begin no later than approximately as the source address starts to be received. Note that these forced collisions will not be limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm. Receive flow control does not depend on the value of the incoming frame destination address. A collision will be generated for any incoming packet, regardless of the destination address.

12.2.1.4.6.13.1.2 IEEE 802.3X Based Receive Flow Control

IEEE 802.3x based receive flow control provides a means of preventing frame reception when the port is operating in full-duplex mode (FULLDUPLEX bit is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register). When receive flow control is enabled and triggered, the port will transmit a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The Ethernet port will transmit a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle, or following the completion of the frame currently being transmitted). The pause frame will contain the maximum possible value for the pause time (FFFFh). The MAC will count the receive pause frame time (decrements FF00h down to 0) and retransmit an outgoing pause frame if the count reaches zero. When the flow control request is removed, the MAC will transmit a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval will be received normally (provided the RX FIFO is not full at which time the receive FIFO will overrun and CPSW3_CPSW_NU_CPSW_NU_STAT_RX_BOTTOM_OF_FIFO_DROP[31-0] COUNT value will increment).

Pause frames will be transmitted if enabled and triggered regardless of whether or not the port is observing the pause time period from an incoming pause frame.

The Ethernet port will transmit pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01.
- The 48-bit source address - from SL_SA[47-0] input.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit pause time value FF.FF. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request will have a pause time value of 00.00.
- Zero padding to 64-byte data length (The Ethernet port will transmit only 64 byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

Copyright © 2026 Texas Instruments Incorporated

If CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[3] RX_FLOW_EN is cleared to 0h while the pause time is nonzero, then the pause time will be cleared to 0h and a 0 count pause frame will be sent.

12.2.1.4.6.13.2 Flow Control Trigger

Receive flow control is triggered (when enabled), when the number of words in the receive FIFO is greater than or equal to the value writen in the CPSW_PN_RX_FLOW_THRESH_REG[8-0] COUNT bit field. The flow control packet runout is then contained in the remainder of the receive FIFO.

12.2.1.4.6.13.3 Ethernet Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the Ethernet port from transmitting any further frames. Incoming pause frames are only acted upon when the [0] FULLDUPLEX and [4] TX_FLOW_EN bits in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register are set. Pause frames are not acted upon in half-duplex mode. Pause frame action will be taken if enabled, but normally the frame will be filtered and not transferred to memory. MAC control frames will be transferred to memory if the [24] RX_CMF_EN (RX Copy MAC Control Frames Enable) bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register is set. The [4] TX_FLOW_EN and [0] FULLDUPLEX bits effect whether or not MAC control frames are acted upon, but they have no effect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC Control Frames with an opcode field = 0001h. Incoming pause frames will only be acted upon by the port if:

- [4] TX_FLOW_EN is set in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register, and
- the RX maximum frame length is 64 bytes inclusive (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_MAXLEN_REG[13-0] RX_MAXLEN), and
- the frame contains no CRC error or align/code errors.

The pause time value from valid frames will be extracted from the two bytes following the opcode. The pause time will be loaded into the port's transmit pause timer and the transmit pause time period will begin.

If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- if the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer will immediately expire, or
- if the new pause time value is zero then the transmit pause timer will immediately expire, else the port transmit pause timer will immediately be set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame will be discarded).

If [4] TX_FLOW_EN in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register is cleared, then the pause-timer will immediately expire.

The port will not start the transmission of a new data frame any sooner than 512-bit times after a pause frame with a non-zero pause time has finished being received (MRXDV going inactive). No transmission will begin until the pause timer has expired (the port may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received will be completed and unaffected.

Incoming pause frames consist of the below:

- A 48-bit destination address equal to:
  - The reserved multicast destination address 01.80.C2.00.00.01, or the Ethernet port SL_SA [47:0] input.
- The 48-bit source address of the transmitting device.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_TX_PAUSETIMER_REG[15-0] TX_PAUSETIMER. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

Copyright © 2026 Texas Instruments Incorporated

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 Bytes. The standard allows pause frames longer than 64 Bytes to be discarded or interpreted as valid pause frames. The Ethernet port will recognize any pause frame between 64 Bytes and CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_MAXLEN_REG[13-0] RX_MAXLEN bytes in length.

12.2.1.4.6.14 Energy Efficient Ethernet Support (802.3az)

Energy Efficient Ethernet (EEE) allows the LPSC to turn off the module clock during inactive periods as determined by network and host traffic. The module can then be awakened by host queued transmit packet(s) or by a port's external Ethernet PHY. The module EEE clock stop interface is used by the external controller to control module EEE operations. EEE operations are configured as shown below:

1. The 12-bit EEE clock pre-scale value is written to the CPSW3_CPSW_NU_EEE_PRESCALE_REG register. The pre-scaler is used to clock all EEE-related counters
2. The port Idle to LPI count values (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IDLE2LPI_REG[23-0] COUNT) are written with the desired values
3. The port LPI to Wake count values (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_LPI2WAKE_REG[23-0] COUNT) are written with the desired values
4. The [0] EEE_EN bit is set in the switch CSL_CPSW3_CPSW_NU_CONTROL_REG register

EEE operation can begin after configuration. The host allows (through LPSC) the CPSW to enter a low power state by asserting the EEE_CLKSTOP_REQ signal. There are no requirements on host queues or traffic in order for the host to assert or de-assert EEE_CLKSTOP_REQ to the CPSW.

Each Ethernet port has a transmit and a receive LPI (low power indicate) state. The PHY indicates LPI by asserting MRXER with a MRXD[7:0] value of 0x01 while MRXDV is deasserted (inter-packet gap). The Ethernet transmit port indicates LPI after the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IDLE2LPI_REG value has been counted (the transmit port has gone idle for the configured amount of time). If another packet is received for transmit during the count then the count is restarted. When the transmit port has been idle for the Idle to LPI time, the transmit port enters the LPI state and indicates LPI to the associated PHY. The LPI is indicated to the external PHY by an asserted MTXER with a MTXD[7:0] while MTXEN is deasserted (inter-packet gap). The CPPI (port 0) LPI state includes transmit and receive. The CPPI LPI state is entered when the CPPI transmit and receive have both been idle for the Idle to LPI time (CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_IDLE2LPI_REG). The Idle to LPI time value for all ports must be large relative to the switch latency to ensure that the count is not able to complete between successive packets.

Note

External PHY signaling has the following conditions:

- RGMII is a DDR interface. TXEN and TXER are the sampled values of TX_CTL at the rising and the falling TXC edges, respectively. RXDV and RXER are the sampled values of RX_CTL at the rising and the falling RXC clock edges, respectively
- In RMII mode, EEE is not supported.

When all transmit and receive ports are in the LPI state (CPSW LPI state), the EEE_CLKSTOP_ACK signal is asserted, and the LPSC is allowed to stop the module clock. When EEE_CLKSTOP_ACK is asserted, the clock may be turned on and off as desired by the host. The host is allowed to restart the clock, perform target read/write operations to the CPSW memory address space, and then turn off the clock again while EEE_CLKSTOP_ACK is asserted.

The software can remove and disable from re-entering the CPSW LPI state by restarting the module clock and then de-asserting EEE_CLKSTOP_REQ. There must be at least one rising edge of the clock before EEE_CLKSTOP_REQ is de-asserted. The module EEE_CLKSTOP_ACK output signal will be deasserted on the

Copyright © 2026 Texas Instruments Incorporated

clock after the de-assertion of EEE_CLKSTOP_REQ. The host may queue CPPI receive packets at any time without regard to the CPSW module LPI state. The Host must deassert EEE_CLKSTOP_REQ on wakeup for a minimum of two clock periods. If EEE_CLKSTOP_REQ is deasserted for less than 5 clock periods for a wakeup event from the host to a particular Ethernet port (or visa versa), then the wakeup event will not cause the other Ethernet port port to awaken.

The external Ethernet PHY's can also wakeup the LPSC by removing the Ethernet receive LPI indication. If the CPSW module is in Idle state with EEE_CLKSTOP_ACK asserted and the receive LPI indication is removed, the EEE_CLKSTOP_WAKEUP signal will be asynchronously asserted. On wakeup, the LPSC restarts the clock and de-assert the EEE_CLKSTOP_REQ signal. The EEE_CLKSTOP_WAKEUP signal will be synchronously deasserted with EEE_CLKSTOP_ACK. Upon the deassertion of EEE_CLKSTOP_REQ, the Ethernet ports will count the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_LPI2WAKE_REG time for each port at which time the port is available for transmit.

12.2.1.4.6.15 Ethernet Switch Latency

When CPSW is configured as a store and forward switch, the switch latency is defined as the amount of time between the end of packet reception of the received packet to the start of the output packet transmit.

The store and forward latency is shown in Table 12-129:

Table 12-129. Switch Latency

Mode	Latency
Gig (1000)	880 ns
100	1.3 µs
10	6.5 µs

12.2.1.4.6.16 MAC Emulation Control

The emulation control input (EMUSUSP) and submodule emulation control registers allow CPSW operation to be completely or partially suspended. Each Ethernet port has associated emulation control registers (CPSW3_CPSW_NU_EM_CONTROL_REG and CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_EMCONTROL_REG). The submodule emulation control registers must be accessed to facilitate CPSW emulation control. The CPSW module enters the emulation suspend state if all three submodules are configured for emulation suspend and the emulation suspend input is asserted. A partial emulation suspend state is entered if one or two submodules is configured for emulation suspend and the emulation suspend input is asserted. Emulation suspend occurs at packet boundaries. The emulation control feature is implemented for compatibility with other peripherals.

Ethernet port Emulation Control

The emulation control input (TBEMUSUP) and register bits (SOFT and FREE bits in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_EMCONTROL_REG register) allow Ethernet port operation to be suspended. When the emulation suspend state is entered, the Ethernet port will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For receive, frames that are detected by the Ethernet port after the suspend state is entered are ignored.

Table 12-130 shows the operations of the emulation control input and register bits.

Table 12-130. Emulation Control Input

EMUSUSP	SOFT	FREE	Description
0	X	X	Normal Operation
1	0	0	Normal Operation
1	1	0	Emulation Suspend
1	X	1	Normal Operation

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.17 MAC Command IDLE

The CMD_IDLE bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register allows MAC operation to be suspended. When the idle state is commanded, the MAC will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the MAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Commanded idle is similar in operation to emulation control and clock stop.

12.2.1.4.6.18 CPSW Network Statistics

The CPSW has a set of statistics that record events associated with frame traffic on selected switch ports. The statistics values are cleared to zero 38 clocks after the rising edge of CPSW0_RST. When one ore more port enable (Pn_STAT_EN) bits in the CPSW3_CPSW_NU_STAT_PORT_EN_REG register are set, all statistics registers are write to decrement. The value written will be subtracted from the register value with the result being stored in the register. If a value greater than the statistics value is written, then zero will be written to the register (writing 0xFFFF FFFF clears a statistics location). When all port enable bits are cleared to zero, all statistics registers are read/write (normal write direct, so writing 0x0000 0000 clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STAT_PEND0) will be issued if enabled when any statistics value is greater than or equal to 0x8000 0000. The statistics interrupt is removed by writing to decrement any statistics value greater than 0x8000 0000. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from 0xFFFF FFFF to 0x0000 0000.

Table 12-131 and Table 12-132 summarize network statistics.

12.2.1.4.6.18.1 Rx-only Statistics Descriptions

12.2.1.4.6.18.1.1 Good Rx Frames (Offset = 3A000h)

All ports

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Had a length of 64 to RX_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.2 Broadcast Rx Frames (Offset = 3A004h)

All ports

The total number of good broadcast frames received on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for address FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error.

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.3 Multicast Rx Frames (Offset = 3A008h)

All ports

Copyright © 2026 Texas Instruments Incorporated

The total number of good multicast frames received on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Had a length of 64 to RX_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.4 Pause Rx Frames (Offset = 3A00Ch)

Ethernet port N where N = 1 to 2

The total number of IEEE 802.3X pause frames received by the port (whether acted upon or not). Such a frame:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08 (hex) and the opcode 0x0001
- Was of length 64 to RX_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error
- Pause-frames had been enabled on the port (TX_FLOW_EN = 1h).

The port could have been in either half or full-duplex mode.

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.5 Rx CRC Errors (Offset = 3A010h)

All ports

The total number of frames received on the port that experienced a CRC error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX_MAXLEN bytes inclusive
- Had no code/align error
- Had a CRC error

Overruns have no effect upon this statistic.

A CRC error is defined to be:

- A frame containing an even number of nibbles, and
- Failing the Frame Check Sequence test.

12.2.1.4.6.18.1.6 Rx Align/Code Errors (Offset = 3A014h)

Ethernet port N where N = 1 to 2

The total number of frames received on the port that experienced an alignment error or code error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to RX_MAXLEN bytes inclusive
- Had either an alignment error, or a code error.

Overruns have no effect upon this statistic.

An alignment error is defined to be:

- A frame containing an odd number of nibbles

Copyright © 2026 Texas Instruments Incorporated

- Failing the Frame Check Sequence test if the final nibble is ignored

A code error is defined to be a frame which has been discarded because the port's MRXER pin driven with a one for at least one bit-time's duration at any point during the frame's reception.

Note

RFC 1757 etherStatsCRCAlignErrors Ref. 1.5 can be calculated by summing Rx Align/Code Errors and Rx CRC errors.

12.2.1.4.6.18.1.7 Oversize Rx Frames (Offset = 3A018h)

All ports

The total number of oversized frames received on the port. An oversized frame is defined to be:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX_MAXLEN in bytes
- Had no CRC error, alignment error, or code error.

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.8 Rx Jabbers (Offset = 3A01Ch)

All ports

The total number of jabber frames received on the port. A jabber frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than RX_MAXLEN in bytes
- Had no CRC error, alignment error or code error

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.9 Undersize (Short) Rx Frames (Offset = 3A020h)

All ports

The total number of undersized frames received on the port. An undersized frame is defined to be:

- Was any data frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes
- Had no CRC error, alignment error, or code error

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.10 Rx Fragments (Offset = 3A024h)

Ethernet port N where N = 1 to 2

The total number of frame fragments received on the port. A frame fragment is defined to be:

- Any data frame (address matching does not matter)
- Less than 64 bytes long

Copyright © 2026 Texas Instruments Incorporated

- Having a CRC error, an alignment error, or a code error
- Not the result of a collision caused by half-duplex, collision-based flow control

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.11 RX IPG Error (Offset = 3A05Ch)

The total number of 10G frames received on a port that had a correct preamble but did not have at least five bytes of IDLE preceding the frame. This does not indicate if the frame with the IPG error was kept or ignored.

12.2.1.4.6.18.1.12 ALE Drop (Offset = 3A028h)

All ports

The total number of frames received on a port such that the destination address was not equal to the source address and the packet was not destined to the port it was received on, but the frame was not forwarded to any port (the PORT_MASK was zero).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was not equal to the source address
- the packet was not destined for the port it was receive on
- had a zero PORT_MASK

12.2.1.4.6.18.1.13 ALE Overrun Drop (Offset = 3A02Ch)

All ports (non cut-thru mode)

The total number of frames received on a port that were dropped (zero PORT_MASK) due to exceeding the maximum ALE lookup rate (Port 0 should not have ALE Overrun Drops because the ingress rate is controlled to prevent it). This statistic should be zero and when non-zero indicates a system clock issue or indicates that short packets were sent with RX_CSF_EN at a rate that exceeded the maximum lookup rate.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- The port has no receive priorities enabled for cut-thru, and
- the maximum ALE lookup rate was exceeded so the lookup was aborted and the packet was dropped.

Ethernet Ports (cut-thru mode)

The total number of received frames on a port that were dropped due to packet errors. This statistic does not count error frames that are sent to the host with a set CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[22] RX_CEF_EN or CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG[24] RX_CMF_EN.

- was any data or MAC control frame, and
- the port has at least one receive priority enabled for cut thru, and
- the received packet was long, mac_control, code, align, crc error, and
- was dropped on receive (not transferred to the host due to error enable bits).

12.2.1.4.6.18.1.14 Rx Octets (Offset = 3A030h)

All ports

The total number of bytes in all good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode

Copyright © 2026 Texas Instruments Incorporated

- Of length 64 to RX_MAXLEN bytes inclusive
- Had no CRC error, alignment error or code error

See the Section 12.2.1.4.6.18.1.6, *Rx Align/Code Errors* and Section 12.2.1.4.6.18.1.5, *Rx CRC errors* statistic descriptions for definitions of alignment, code and CRC errors.

Overruns have no effect upon this statistic.

12.2.1.4.6.18.1.15 Rx Bottom of FIFO Drop (Offset = 3A084h)

Ethernet port N where N = 1 to 2

The total number of frames received on a port that overran the port's receive FIFO and were dropped (bottom of receive FIFO). Port 0 (CPPI receive port) should not drop packets on receive because port 0 receive flow control should be enabled. The Ethernet ports will only drop packets in the receive FIFO when receive flow control is enabled and the sending port ignores sent pause frame and then overruns the receive FIFO. An overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- Was dropped on port 0 due to a lack of memory space in the receive FIFO.

Note

This statistic should be zero if proper flow control is being followed.

Host port 0

This statistic also counts frames dropped on port 0 that were 17 to 63 bytes (only for port 0). For Ethernet ports, the drop count for frames shorter than 33 bytes is included in the undersized or fragment count. Port 0 simply gives an indication that a packet was dropped. No other statistics are counted for frames shorter than 33 bytes.

12.2.1.4.6.18.1.16 Portmask Drop (Offset = 3A088h)

All ports

The total number of frames received on a port that were dropped by the ALE (the ALE did not forward the packet to any port). Port mask drop frame is defined to be:

- Any data or MAC control frame
- Any length greater than 32 bytes
- Was dropped by the ALE due to PORT_MASK=0 (was not sent to any destination port)
- Was not a long or error packet with the port in cut-thru mode
- The frame could have been dropped due to error or other counted reason, so it could be counted elsewhere also.

Note

This statistic does not count in the overall total as it includes every packet received greater than 32 bytes that had a zero PORT_MASK.

12.2.1.4.6.18.1.17 Rx Top of FIFO Drop (Offset = 3A08Ch)

All ports

The total number of frames received on a port that had a start-of-frame (SOF) overrun on any destination port egress (when attempting to load the packet from the top of the ingress port receive FIFO into any other port's transmit FIFO). If a multicast/broadcast packet is dropped by multiple destination ports then this statistic will increment by the number of ports that dropped the packet. Rx Top Of FIFO Drop is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode

Copyright © 2026 Texas Instruments Incorporated

- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error or code error
- had a SOF of frame overrun on another port egress.

12.2.1.4.6.18.1.18 ALE Rate Limit Drop (Offset = 3A090h)

All ports

The total number of frames received on a port that were dropped (zero PORT_MASK) due to receive rate limiting on this port or due to transmit rate limiting on any destination port (not sent to all expected destination ports if transmit rate limiting).

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the receive rate was exceeded and the packet was dropped, or the transmit rate was exceeded to any destination port and the packet was dropped to one or more expected destination ports (indicates that the destinations were reduced due to rate limiting).

12.2.1.4.6.18.1.19 ALE VLAN Ingress Check Drop (Offset = 3A094h)

All ports

The total number of frames received on a port that were dropped (zero PORT_MASK) due to VLAN ingress check failure.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the VLAN ID ingress check failed (the receive port was not in the group)
- The address lookup did not return a match with the SUPER bit set.

12.2.1.4.6.18.1.19.1 ALE DA=SA Drop (Offset = 3A098h)

All ports

The total number of frames received on a port that were dropped (zero PORT_MASK) due to destination address equal to source address.

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error
- the destination address was equal to the source address
- the source address was not an entry in the table.

12.2.1.4.6.18.1.19.2 Block Address Drop (Offset = 3A09Ch)

The total number of frames received on a port that were dropped (zero PORT_MASK) due to the destination or source address being blocked.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source or destination address matched a table entry with the block bit set.

12.2.1.4.6.18.1.19.3 ALE Secure Drop (Offset = 3A0A0h)

The total number of frames received on a port that were dropped (zero port_mask) due to a secure violation (the source address is owned by a different receive port).

Copyright © 2026 Texas Instruments Incorporated

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- the source address is an entry in the table with the SECURE bit set and a port number for a different receive port.

12.2.1.4.6.18.1.19.4 ALE Authentication Drop (Offset = 3A0A4h)

The total number of frames received on a port that were dropped (zero port_mask) due to authentication failure.

- was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode, and
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)
- had no CRC error, alignment error, or code error, and
- CPSW3_CPSW_NU_ALE_ALE_CONTROL[1] ENABLE_AUTH_MODE is set to 1h, and
- the source address is not equal to the destination address, and
- the source address is not a table entry, and
- the destination address is not a table entry with the SUPER bit set.

12.2.1.4.6.18.1.19.5 ALE Unknown Unicast (Offset = 3A0A8h)

All ports

The total number of frames received on a port that had a unicast destination address with an unknown source address.

- was any data frame with a unicast destination address
- the source address was not a table entry
- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

Note: The ALE Unknown Unicast Bytecount statistic is the number of bytes contained in the ALE Unknown Unicast frames.

12.2.1.4.6.18.1.19.6 ALE Unknown Unicast Bytecount (Offset = 3A0ACh)

The total number of bytes received on a port that had a unicast destination address with an unknown source address.

12.2.1.4.6.18.1.19.7 ALE Unknown Multicast (Offset = 3A0B0h)

The total number of frames received on a port that had a multicast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and
- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Multicast Bytecount statistic is the number of bytes contained in the ALE Unknown Multicast frames.

12.2.1.4.6.18.1.19.8 ALE Unknown Multicast Bytecount (Offset = 3A0B4h)

The total number of bytes received on a port that had a multicast destination address with an unknown source address.

12.2.1.4.6.18.1.19.9 ALE Unknown Broadcast (Offset = 3A0B8h)

The total number of frames received on a port that had a broadcast destination address with an unknown source address. The frame is defined to be:

- was any data frame with a unicast destination address
- the source address was not a table entry, and

Copyright © 2026 Texas Instruments Incorporated

- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error or code error

Note: The ALE Unknown Broadcast Bytecount statistic is the number of bytes contained in the ALE Unknown Broadcast frames.

12.2.1.4.6.18.1.19.10 ALE Unknown Broadcast Bytecount (Offset = 3A0BCh)

The total number of bytes received on a port that had a broadcast destination address with an unknown source address.

12.2.1.4.6.18.1.19.11 ALE Policer/Classifier Match (Offset = 3A0C0h)

All ports

The total number of frames received on a port that matched a policer. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

12.2.1.4.6.18.2 ALE Policer Match Red (Offset = 3A0C4h)

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

12.2.1.4.6.18.3 ALE Policer Match Yellow (Offset = 3A0C8h)

The total number of frames received on a port that had matched a policer and the condition was red. The frame is defined to be:

- was any data frame
- matched a condition on a policer,
- was of length 64 to RX_MAXLEN bytes inclusive
- had no CRC error, alignment error, or code error

12.2.1.4.6.18.4 IET Receive Assembly Error (Offset = 3A140h)

The total number of premptable received frames with IET assembly errors.

- any frame received
- was any size, and
- was a non-initial fragment that mismatched the frame count or fragment count (went to the assembly error state in the IET receive state machine)

12.2.1.4.6.18.5 IET Receive Assembly OK (Offset = 3A144h)

The total number of correctly received and re-assembled premptable frames.

- any premptable frame received
- was any size, and
- was correctly received and re-assembled without error

12.2.1.4.6.18.6 IET Receive SMD Error (Offset = 3A148h)

The total number of received frames rejected due to an unknown SMD value or received frames rejected with an SMD-C when no frame is in progress.

- any frame received
- was any size, and

Copyright © 2026 Texas Instruments Incorporated

- was rejected because of an unknown SMD value or SMD-C with no frame in progress.

Note: If IET_ENABLE is not set, this statistic counts any received frame with any non express SMD.

12.2.1.4.6.18.7 IET Receive Merge Fragment Count (Offset = 3A14Ch)

The total number of received non-initial fragments that did not have an assembly error. The IET stat CPSW3_CPSW_NU_CPSW_NU_STAT_RXFRAGMENTS_k is derived by adding the Receive Assembly Error count to this value.

- any frame received
- was any size, and
- was a non-initial fragment that did not contain an assembly error

12.2.1.4.6.18.8 Rx Cut Thru with No Delay

The total number of frames received on the port and subsequently sent cut-thru to all internal destination port FIFOs with no delay. Such a frame:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address, and
- was any size greater than or equal to 128 bytes, and
- was sent cut-thru by the receive port to all destination ports without delay.

12.2.1.4.6.18.9 Rx Cut Thru with Delay

The total number of frames received on the port and subsequently sent cut-thru to all destination ports with some amount of delay. Such a frame:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address, and
- was any size greater than or equal to 128 bytes, and
- was sent cut-thru by the receive port to all destination ports with some amount of delay.

12.2.1.4.6.18.10 Rx Cut Thru Store-and-Forward

The total number of frames received on the port were attempted to be sent cut-thru to all destination ports but were held off due to configuration or traffic conditions. Such a frame:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address, and
- was any size greater than or equal to 128 bytes, and
- was attempted to be sent cut-thru by the receive port to all destination ports but was sent store-and-forward.

12.2.1.4.6.18.11 Tx-only Statistics Descriptions

The maximum and minimum transmit frame size is software controllable.

12.2.1.4.6.18.11.1 Good Tx Frames (Offset = 3A034h)

All ports

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

Note

A nonzero CPSW3_CPSW_NU_CPSW_NU_STAT_TXDEFERREDFRAMES_k value should be subtracted from this value to obtain the actual good frames value (the good frames statistic also increments on a transmitted CRC error).

12.2.1.4.6.18.11.2 Broadcast Tx Frames (Offset = 3A038h)

All ports

Copyright © 2026 Texas Instruments Incorporated

The total number of good broadcast frames transmitted on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for only address FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

12.2.1.4.6.18.11.3 Multicast Tx Frames (Offset = 3A03Ch)

All ports

The total number of good multicast frames transmitted on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than FF.FF.FF.FF.FF.FF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

12.2.1.4.6.18.11.4 Pause Tx Frames (Offset = 3A040h)

Ethernet port N where N = 1 to 2

This statistic indicates the number of IEEE 802.3X pause frames transmitted by the port.

Pause frames cannot contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect upon the statistic. Pause frames sent by software will not be included in this count.

Since pause frames are only transmitted in full duplex, carrier loss and collisions have no effect upon this statistic.

Transmitted pause frames are always 64-byte multicast frames so will appear in the *Tx Multicast Frames* and *64octet Frames* statistics.

12.2.1.4.6.18.11.5 Deferred Tx Frames (Offset = 3A044h)

Ethernet port N where N = 1 to 2

The total number of frames transmitted on the port that first experienced deferment. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect upon this statistic.

12.2.1.4.6.18.11.6 Collisions (Offset = 3A048h)

Ethernet port N where N = 1 to 2

This statistic records the total number of times that the port experienced a collision. Collisions occur under two circumstances.

1. When a transmit data or MAC control frame:
   - Was destined for any unicast, broadcast or multicast address
   - Was any size
   - Had no carrier loss and no underrun
   - Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic will increment on each occasion if a frame experiences multiple collisions (and increments on late collisions)

   CRC errors have no effect upon this statistic.
2. When the port is in half-duplex mode, flow control is active, and a frame reception begins.

12.2.1.4.6.18.11.7 Single Collision Tx Frames (Offset = 3A04Ch)

Ethernet port N where N = 1 to 2

Copyright © 2026 Texas Instruments Incorporated

The total number of frames transmitted on the port that experienced exactly one collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect upon this statistic.

12.2.1.4.6.18.11.8 Multiple Collision Tx Frames (Offset = 3A050h)

Ethernet port N where N = 1 to 2

The total number of frames transmitted on the port that experienced multiple collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect upon this statistic.

12.2.1.4.6.18.11.9 Excessive Collisions (Offset = 3A054h)

Ethernet port N where N = 1 to 2

The total number of frames for which transmission was abandoned due to excessive collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect upon this statistic.

12.2.1.4.6.18.11.10 Late Collisions (Offset = 3A058h)

Ethernet port N where N = 1 to 2

The total number of frames on the port for which transmission was abandoned because they experienced a late collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions which had previously required the transmission to be re-attempted. The *Late Collisions* statistic dominates over the single-, multiple- , and excessive- collision statistics. If a late collision occurs, the frame will not be counted in any of these other three statistics.

CRC errors have no effect upon this statistic.

12.2.1.4.6.18.11.11 Carrier Sense Errors (Offset = 3A060h)

Ethernet port N where N = 1 to 2

The total number of frames on the port that experienced carrier loss. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted). This is a transmit only statistic. Carrier Sense is a don’t care for received frames. Transmit frames with carrier sense errors are sent until completion and are not aborted.

CRC errors have no effect upon this statistic.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.6.18.11.12 Tx Octets (Offset = 3A064h)

All ports

The total number of bytes in all good frames transmitted on the port. A good frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Was any size
- Had no late or excessive collisions, no carrier loss and no underrun.

12.2.1.4.6.18.11.13 Transmit Priority 0-7 (Offset = 3A180h to 3A1A8h)

The total number of frames transmitted on the port from transmit FIFO priority 0-7. Collision retries do not affect this statistic. Pause frames do not affect this statistic.

- Any frame transmitted from priority 0-7, and
- Was less than or equal to CPSW3_CPSW_NU_TX_PRI0_MAXLEN_REG to CPSW3_CPSW_NU_TX_PRI7_MAXLEN_REG
- Collision retries are not counted in this statistic.
- Pause frames are not counted in this statistic.
- Carrier sense errors do not affect this statistic.

Note: The Transmit Priority 0-7 Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 statistic.

12.2.1.4.6.18.11.14 Transmit Priority 0-7 Drop (Offset = 3A1C0h to 3A1E8)

The total number of transmit frames on the port that overran the transmit FIFO priority 0-7 and were dropped. This count includes frames dropped due to CPSW3_CPSW_NU_TX_PRI0_MAXLEN_REG to CPSW3_CPSW_NU_TX_PRI7_MAXLEN_REG.

- Any frame destined to be transmitted from priority 0-7, and
- Was any size, and
- Was dropped due to priority 0-7 FIFO overrun (Start of packet overrun).
- Was dropped due to frame size larger than CPSW3_CPSW_NU_TX_PRI0_MAXLEN_REG to CPSW3_CPSW_NU_TX_PRI7_MAXLEN_REG.

Note: The Transmit Priority 0-7 Drop Bytecount statistic is the number of bytes contained in the frames of the Transmit Priority 0-7 Drop statistic.

12.2.1.4.6.18.11.15 Tx Memory Protect Errors (Offset = 3A17Ch)

All ports

The total number of transmit frames on the port that had a memory protect CRC error on egress:

- Any frame destined to be transmitted,
- Was any size
- Had a memory protect CRC error on egress.

Note

1. Frames to the host with memory protect errors are dropped via TXST_DROP. Ethernet frames will have at least one byte of the generated port type CRC inverted on egress.
2. This statistic is 8-bits wide only and will not rollover but will limit at 0xFF.
3. A non-zero value in this statistic will issue a STAT_PEND0 interrupt for the associated port.

12.2.1.4.6.18.11.16 IET Transmit Merge Hold Count (Offset = 3A150h)

The total number of premptable frames that were prempted and reassembled by the assertion of MAC_HOLD in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_IET_CONTROL_REG register or were prempted by Enhanced Scheduled Traffic (EST). The IET statistic can be derived and maintained by software.

Copyright © 2026 Texas Instruments Incorporated

- Any frame destined to be transmitted on the premptable port, and
- was any size, and
- was prempted by the assertion of MAC_HOLD.
- was prempted by Enhanced Scheduled Traffic (EST).

12.2.1.4.6.18.11.17 IET Transmit Merge Fragment Count (Offset = 3A14Ch)

The total number of non-initial premptable transmit fragments on premptable transmit.

- Any frame destined to be transmitted on the premptable port, and
- Was any size, and
- was a non-initial fragment.

12.2.1.4.6.18.11.18 Tx CRC Errors

The total number of frames transmitted on the port with a CRC error. Such a frame:

- was any data frame destined for any unicast, broadcast or multicast address, and
- was any size, and
- was sent cut-thru by the receive port and was received with a CRC error, or
- was a transmitted frame that also had a memory protect error (counted also in CPSW3_CPSW_NU_CPSW_NU_STAT_TX_MEMORY_PROTECT_ERROR_k).

Note

For port0 this statistics location (CPSW_STAT0_TXDROP) counts the number of drops to the host due to a memory protect error, or due to a cut-thru packet having an error on reception but was forwarded with the error. The Ethernet port receive error could be long, CRC, jabber, or code. For Ethernet ports, the receive error could be the same but the packet is transmitted with at least one byte of the actual outgoing packet CRC inverted to indicate the error.

Note

A nonzero CPSW3_CPSW_NU_CPSW_NU_STAT_TXDEFERREDFRAMES_k value should be subtracted from the CPSW3_CPSW_NU_CPSW_NU_STAT_TXGOODFRAMES_k value to obtain the actual good frames value (the good frames statistic also increments on a transmitted CRC error).

12.2.1.4.6.18.11.19 Tx Cut Thru

The total number of cut-thru frames transmitted on the port with or without delay. Such a frame:

- was any data frame destined for any unicast, broadcast or multicast address, and
- was any size greater than or equal to 128 bytes, and
- was sent cut-thru by the receive port.
- was transmitted cut-thru with or without delay.

12.2.1.4.6.18.11.20 Tx Cut Thru Store-and-Forward

The total number of cut-thru frames transmitted store and forward. Such a frame:

- was any data frame destined for any unicast, broadcast or multicast address, and
- was any size greater than or equal to 128 bytes, and
- was sent cut-thru by the receive port.
- was transmitted store-and-forward due to traffic congestion.

12.2.1.4.6.18.12 Rx- and Tx (Shared) Statistics Descriptions

12.2.1.4.6.18.12.1 Rx + Tx 64 Octet Frames (Offset = 3A068h)

All ports

The total number of 64-byte frames received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address

Copyright © 2026 Texas Instruments Incorporated

- Did not experience late collisions, excessive collisions, or carrier sense error
- Was exactly 64 bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame will be recorded in this statistic).

CRC errors, code/align errors and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.2 Rx + Tx 65–127 Octet Frames (Offset = 3A06Ch)

All ports

The total number of frames of size 65 to 127 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 65 to 127 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.3 Rx + Tx 128–255 Octet Frames (Offset = 3A070h)

All ports

The total number of frames of size 128 to 255 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 128 to 255 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.4 Rx + Tx 256–511 Octet Frames (Offset = 3A074h)

All ports

The total number of frames of size 256 to 511 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 256 to 511 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.5 Rx + Tx 512–1023 Octet Frames (Offset = 3A078h)

All ports

The total number of frames of size 512 to 1023 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 512 to 1023 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.6 Rx + Tx 1024_Up Octet Frames (Offset = 3A07Ch)

All ports

The total number of frames of size 1024 to RX_MAXLEN bytes for receive or 1024 up for transmit on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 1024 to RX_MAXLEN bytes long on receive, or any size on transmit

Copyright © 2026 Texas Instruments Incorporated

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

12.2.1.4.6.18.12.7 Net Octets (Offset = 3A080h)

All ports

The total number of bytes of frame data received and transmitted on the port. Each frame counted:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address (address match does not matter)
- Any length (including less than 64 bytes and greater than RX_MAXLEN bytes)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced, (that is, multiple retries are counted each time)
- Every byte received if the port is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence was not counted to prevent double-counting)

Error conditions such as alignment errors, CRC errors, code errors, overruns and underruns do not affect the recording of bytes by this statistic.

The objective of this statistic is to give a reasonable indication of Ethernet utilization.

12.2.1.4.6.18.13

Table 12-131. Rx Statistics Summary

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data[5]															
			Pause frame	Non-pause[4]	Multi cast	Broad cast	Uni cast	<64	64	65-127	128-255	256-511	512-1023	1024-rx_ maxlen	>rx_ maxlen	Flow Coll.[8]	CRC Error	Align/ Code	Overrun	Addr. Disc.
Good Rx Frames	F	Rx	(y|[1]	y|	y|	y|	y)	n	(y|	y|	y|	y|	y|	y)	n	-[2]	n	n	-	n
Broadcast Rx Frames	F	Rx	(%|[6]	%|	n	y)	n	n	(y|	y|	y|	y|	y|	y)	n	-	n	n	-	n
Multicast Rx Frames	F	Rx	(%|	%|	y)	n	n	n	(y|	y|	y|	y|	y|	y)	n	-	n	n	-	n
Pause Rx Frames	F	Rx	y	n	n	n	n	n	(y|	y|	y|	y|	y|	y)	n	-	n	n	-	-
Rx CRC Errors	F	Rx	(y|	y|	y|	y|	y)	n	(y|	y|	y|	y|	y|	y)	n	-	y	n	-	n
Rx Align/Code Errors	F	Rx	(y|	y|	y|	y|	y)	n	(y|	y|	y|	y|	y|	y)	n	-	-	y	-	n
Oversized Rx Frames	F	Rx	(y|	y|	y|	y|	y)	n	n	n	n	n	n	n	y	-	n	n	-	n
Rx Jabbers	F	Rx	(y|	y|	y|	y|	y)	n	n	n	n	n	n	n	y	-	(y|	y)	-	n
Undersized Rx Frames	F	Rx	n	n	(y|	y|	y)	y	n	n	n	n	n	n	n	-	n	n	-	n
Rx Fragments	F	Rx	n	n	(y|	y|	y)	y^[7]	n	n	n	n	n	n	n	-	(y|	y)	-	-
Rx Overruns[9]	F	Rx	(y|	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y|	y)	-	-	-	y	n
64octet Frames	F	Rx+ Tx[3]	(y|	y|	y|	y|	y)	n	y	n	n	n	n	n	n	-	-	-	-	n
65-127octet Frames	F	Rx+ Tx	(y|	y|	y|	y|	y)	n	n	y	n	n	n	n	n	-	-	-	-	n
128-255octet Frames	F	Rx+ Tx	(y|	y|	y|	y|	y)	n	n	n	y	n	n	n	n	-	-	-	-	n

Copyright © 2026 Texas Instruments Incorporated

Table 12-131. Rx Statistics Summary (continued)

Rx Statistic	Frame/ Oct	Rx/ Rx+ Tx	Frame Type					Frame Size (bytes)								Event				
			MAC control		Data[5]															
			Pause frame	Non-pause[4]	Multicast	Broadcast	Unicast	<64	64	65-127	128-255	256-511	512-1023	1024-rx_maxlen	>rx_maxlen	Flow Coll.[8]	CRC Error	Align/ Code	Overrun	Addr. Disc.
256-511octet Frames	F	Rx+ Tx	(y|	y|	y|	y|	y)	n	n	n	n	y	n	n	n	-	-	-	-	n
512-1023octet Frames	F	Rx+ Tx	(y|	y|	y|	y|	y)	n	n	n	n	n	y	n	n	-	-	-	-	n
1024-UPoctet Frames	F	Rx+ Tx	(y|	y|	y|	y|	y)	n	n	n	n	n	n	y	n	-	-	-	-	n
Rx Octets	O	Rx	(y|	y|	y|	y|	y)	n	(y|	y|	y|	y|	y|	y)	n	-	n	n	-	n
Net Octets	O	Rx+ Tx	(y|	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y|	y|	y)	-	-	-	-

(1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
(2) "-" indicates conditions which are ignored in the formations of the statistic.
(3) Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
(4) The non-pause column refers to all MAC control frames (for example, frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
(5) The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
(6) "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
(7) "y^" Frame fragments are not counted if less than 8 bytes.
(8) Flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
(9) The rx_overruns stat is for RX_MOF_OVERRUNS and RX_SOF_OVERRUNS added together.

Table 12-132. Tx Statistics Summary

Tx Statistic[9]	Frame/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)								Event								
			MAC control[4]		Data												Collision Type								
			Pause-MAC	Any-CPU	Multicast	Broadcast	Unicast	64	65-127	128-255	256-511	512-1023	1024-1535	>1535	CRC Error	Flow[8]	1	2-15	16	Late	No Carrier	Queued	Deferred	Underrun
Good Tx Frames	F	Tx	(y| [1]	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-[2]	-	-	-	-	n	n	n	-	-	n
Broadcast Tx Frames	F	Tx	n	(%| [5]	n	y)	n	(y|	y|	y|	y|	y|	y|	y)	-	-	-	-	-	n	n	n	-	-	n
Multicast Tx Frames	F	Tx	(y|	%|	y)	n	n	(y|	y|	y|	y|	y|	y|	y)	-	-	-	-	-	n	n	n	-	-	n
Pause Tx Frames	F	Tx	y	n	n	n	n	y	n	n	n	n	n	n	-	-	-	-	-	-	-	-	-	-	-
Collisions	F	Tx	n	(y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	(+ [6]	+	+	+	+)	n	-	-	-
Single Collision Tx Frames	F	Tx	n	(y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	-	y	n	n	n	n	-	-	-
Multiple Collision Tx Frames	F	Tx	n	(y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	-	n	y	n	n	n	-	-	-
Excessive Collisions	F	Tx	n	(y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	-	n	n	y	n	n	-	-	-

Copyright © 2026 Texas Instruments Incorporated

Table 12-132. Tx Statistics Summary (continued)

Tx Statistic[9]	Frame/ Oct	Tx/ Rx +Tx	Frame Type					Frame Size (bytes)								Event								
			MAC control [4]	Data				64	65-127	128-255	256-511	512-1023	1024-1535	>1535	CRC Error	Collision Type					No Carrier	Queued	Deferred	Underrun
			Pause-MAC	Any-CPU	Multicast	Broadcast	Unicast									Flow[8]	1	2-15	16	Late				
Late Collisions	F	Tx	n	(y|	y|	y|	y)	n	(y|	y|	y|	y|	y|	y)	-	-	-	-	-	y	-	-	-	-
Deferred Tx Frames	F	Tx	n	(y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	n	n	n	n	n	-	y	n
Carrier Sense Errors	F	Tx	(y|	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	-	-	-	-	y	-	-	-
64octet Frames	F	Rx+Tx [3]	(y|	y|	y|	y|	y)	y	n	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-
65-127octet Frames	F	Rx+Tx	(y|	y|	y|	y|	y)	n	y	n	n	n	n	n	-	-	-	-	n	n	n	-	-	-
128-255octet Frames	F	Rx+Tx	(y|	y|	y|	y|	y)	n	n	y	n	n	n	n	-	-	-	-	n	n	n	-	-	-
256-511octet Frames	F	Rx+Tx	(y|	y|	y|	y|	y)	n	n	n	y	n	n	n	-	-	-	-	n	n	n	-	-	-
512-1023octet Frames	F	Rx+Tx	(y|	y|	y|	y|	y)	n	n	n	n	y	n	n	-	-	-	-	n	n	n	-	-	-
1024-UPoctet Frames	F	Rx+Tx	(y|	y|	y|	y|	y)	n	n	n	n	n	y	y	-	-	-	-	n	n	n	-	-	-
Tx Octets	O	Tx	(y|	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	-	-	n	n	n	-	-	n
Net Octets	O	Rx+Tx	(y|	y|	y|	y|	y)	(y|	y|	y|	y|	y|	y|	y)	-	-	$[7]	$	$	$	$	-	-	-

(1) "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning “OR” is indicated. Parentheses are significant.
(2) “-“ indicates conditions which are ignored in the formations of the statistic.
(3) Statistics marked “Rx+Tx” are formed by summing the Rx and Tx statistics, each of which is formed independently.
(4) Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
(5) “%” If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
(6) “+” indicates collisions which are “summed” (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
(7) “$” Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
(8) The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the ‘-‘s in this column might in reality be ‘n’s. To prevent double-counting, Net Octets are unaffected by the jam sequence – the ‘received’ bytes, however, are counted. (See Table 12-131.)
(9) When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

12.2.1.4.7 Common Platform Time Sync (CPTS)

The Common Platform Time Sync (CPTS) module is used to facilitate host control of time sync operations. It enables compliance with the IEEE 1588-2008 standard for a precision clock synchronization protocol.

Main features of CPTS module are:

- Supports the selection of multiple external clock sources

Copyright © 2026 Texas Instruments Incorporated

- Software control of time sync events via interrupt or polling
- Supports up to 8 hardware timestamp push inputs
- Supports timestamp counter compare output (CPTS_COMP)
- Supports timestamp counter bit output (CPTS_SYNC)
- Supports a configurable number of timestamp Generator bit outputs (CPTS_GENFn).
- Supports Ethernet Enhanced Scheduled Traffic Operations (CPTS_ESTFn).
- 32-bit and 64-bit timestamp modes with PPM and nudge adjustment.

12.2.1.4.7.1 CPSW0 CPTS Integration

This section describes CPTS module integration in the CPSW0 module, including information about clocks, resets, and hardware requests.

Figure 12-102 shows CPTS integration in the device CPSW0 module.

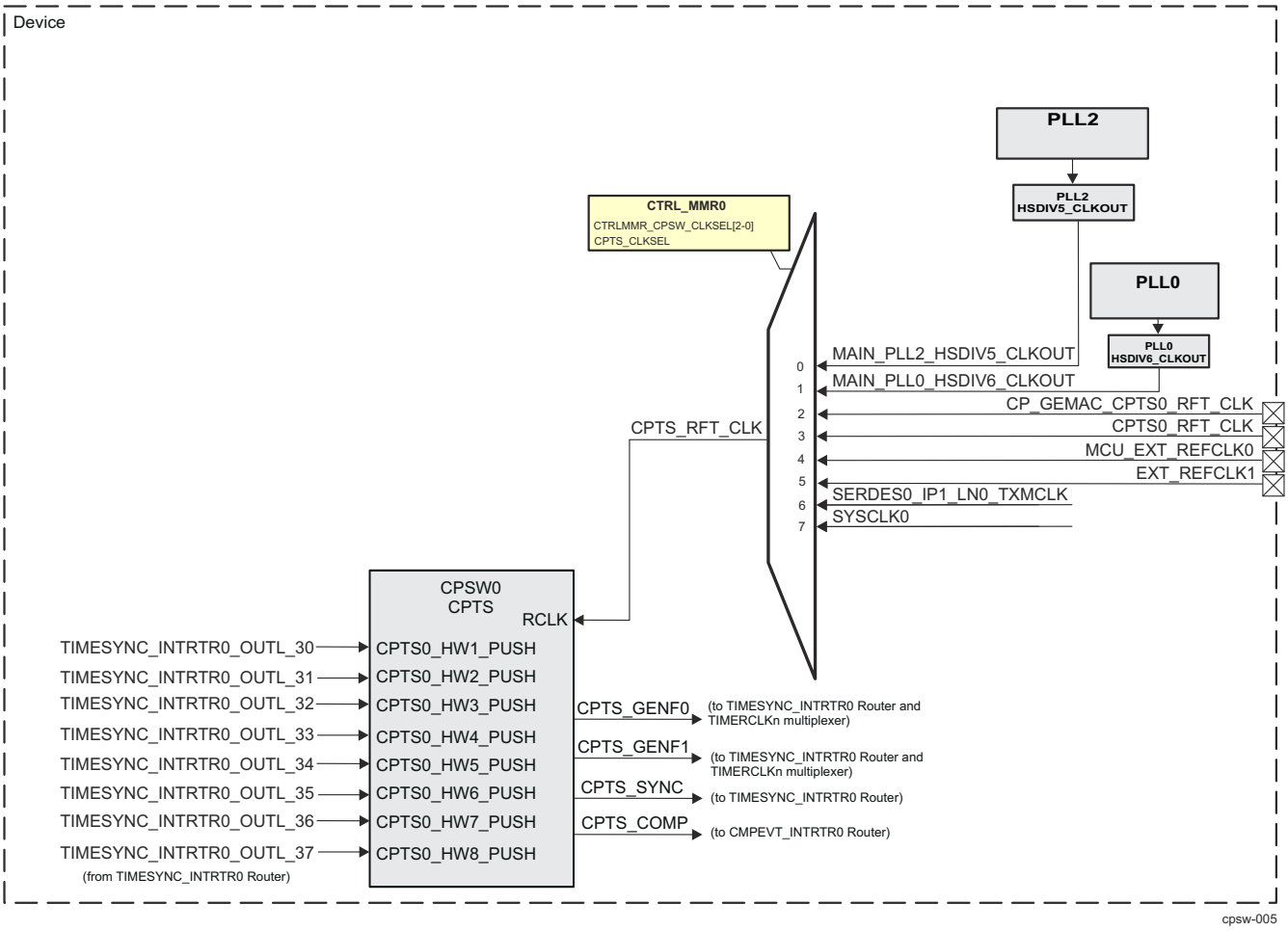


Figure 12-102. CPSW0 CPTS Integration

CPTS IEEE 1588 clock (RCLK) is selected through the CPTS_CLKSEL bit.

Note

For more information about CPTS clocks and resets, see *CPWS0 Clocks* in *CPSW0 Integration*.

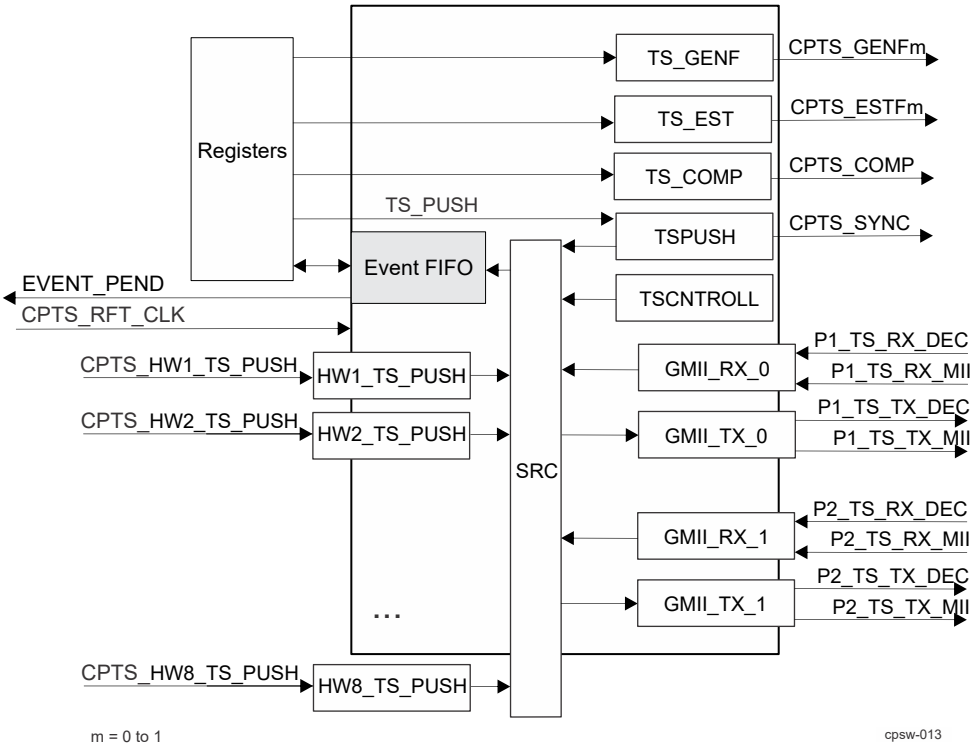
Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.7.2 CPTS Architecture

Figure 12-103 shows the architecture of the CPTS module inside the CPSW Ethernet Subsystem. Time stamp values for every packet transmitted or received on external port of the CPSW are recorded. At the same time, each packet is decoded to determine if it is a valid time sync event. If so, an event is loaded into the Event FIFO for processing containing the recorded time stamp value when the packet was transmitted or received.

In addition, both hardware (HWn_TS_PUSH) and software (TS_PUSH) can be used to read the current time stamp value though the Event FIFO. The reference clock used for the time stamp (CPTS_RFT_CLK) can be derived from several sources.

Figure 12-103. CPTS Block Diagram



Note

See Section 12.2.1.4.7.1, *CPSW0 CPTS Integration* for CPTS integration in the device CPSW0 module.

12.2.1.4.7.3 CPTS Initialization

The CPTS module should be configured as follows:

1. Reset the CPTS module.
2. Write the CPTS_CLKSEL value with the desired reference clock selection. This value is allowed to be written only when the CPTS_EN bit in the CPSW3_CPSW_NU_CPTS_CONTROL_REG register is cleared to zero.
3. Set the CPTS_EN bit in the CPSW3_CPSW_NU_CPTS_CONTROL_REG register.
4. If using interrupts and not polling, enable the interrupt by setting the TS_PEND_EN bit in the CPSW3_CPSW_NU_CPTS_INT_ENABLE_REG register.

12.2.1.4.7.4 32-bit Time Stamp Value

The time stamp value is a 32-bit value that increments on each CPTS_RFT_CLK rising edge when CPTS_EN bit is set to 1h. When CPTS_EN bit is cleared to 0h, the time stamp value is reset to 0h.

Copyright © 2026 Texas Instruments Incorporated

If more than 32-bits of time stamp are required by the application, the host software must maintain the necessary number of upper bits. The upper time stamp value should be incremented by the host when the rollover event is detected.

For test purposes, the time stamp can be written via the time stamp load function (CPSW3_CPSW_NU_CPTS_TS_LOAD_VAL_REG / CPSW3_CPSW_NU_CPTS_TS_LOAD_HIGH_VAL_REG and CPSW3_CPSW_NU_CPTS_TS_LOAD_EN_REG registers).

12.2.1.4.7.5 64-bit Time Stamp Value

The time stamp value is a 64-bit value that increments on each CPTS_RFT_CLK rising edge when CPTS_EN bit is set to 1h. When CPTS_EN bit is cleared to 0h, the time stamp value is reset to 0h.

64-bit mode is selected when CPSW3_CPSW_NU_CPTS_CONTROL_REG[5] MODE bit set to 1h.

For test purposes, the time stamp value can be written via the time stamp load function (CPSW3_CPSW_NU_CPTS_TS_LOAD_EN_REG, CPSW3_CPSW_NU_CPTS_TS_LOAD_VAL_REG, and CPSW3_CPSW_NU_CPTS_TS_LOAD_HIGH_VAL_REG registers). The CPSW3_CPSW_NU_CPTS_TS_ADD_VAL_REG feature is included to allow 1ns timestamp operations with an CPTS_RFT_CLK rate less than 1Ghz. Table 12-133 shows the CPTS_RFT_CLK and CPSW3_CPSW_NU_CPTS_TS_ADD_VAL_REG values for 1ns operations.

Table 12-133. ADD_VAL feature

CPTS_RFT_CLK (MHz)	CPSW3_CPSW_NU_CPTS_TS_ADD_VAL_REG[2-0] ADD_VAL
1 GHz	0
500 MHz	1
333.33 MHz	2
250 MHz	3
200 MHz	4
166.66 MHz	5
142.85714 MHz	6
125 MHz	7

12.2.1.4.7.6 64-Bit Timestamp Nudge

The 64-bit TIME_STAMP value can be adjusted by writing the CPSW3_CPSW_NU_CPTS_TS_NUDGE_VAL_REG[7-0] TS_NUDGE_VAL bit field value which is a two's complement value. A value of FFh will subtract 1 clock cycle from the next incremented 64-bit time stamp value (CPSW3_CPSW_NU_CPTS_EVENT_0_REG[31-0] TIME_STAMP and CPSW3_CPSW_NU_CPTS_EVENT_3_REG[31-0] TIME_STAMP value). A nudge value of 1h will add 1 clock cycle to the next incremented TIME_STAMP[63-0] value. For example, if the current TIME_STAMP value is F06h, and CPSW3_CPSW_NU_CPTS_TS_ADD_VAL_REG[2-0] ADD_VAL = 3h, the next incremented timestamp value would be F0Ah without a nudge and F0Ah +/- [7-0] TS_NUDGE_VAL with a nudge. The [7-0] TS_NUDGE_VAL value is cleared to zero when the nudge has occurred.

12.2.1.4.7.7 64-bit Timestamp PPM

The 64-bit TIME_STAMP can be adjusted by parts per million or by parts per hour. Writing a non-zero value to the CPSW3_CPSW_NU_CPTS_TS_PPM_LOW_VAL_REG[31-0] TS_PPM_LOW_VAL (Time stamp PPM Low value) and CPSW3_CPSW_NU_CPTS_TS_PPM_HIGH_VAL_REG[9-0] TS_PPM_HIGH_VAL (Time stamp PPM High value) enables PPM operations. The adjustment is up or down depending on the [7] TS_PPM_DIR bit in the CPSW3_CPSW_NU_CPTS_CONTROL_REG register. The TIME_STAMP value is increased by the PPM value when [7] TS_PPM_DIR bit is cleared. The TIME_STAMP value is decreased by the PPM value when [7] TS_PPM_DIR bit is set.

Parts Per Million example:

Copyright © 2026 Texas Instruments Incorporated

To adjust for 100 parts per million the configured value for TS_PPM[41-0] (through CPSW3_CPSW_NU_CPTS_TS_PPM_LOW_VAL_REG[31-0] TS_PPM_LOW_VAL and CPSW3_CPSW_NU_CPTS_TS_PPM_HIGH_VAL_REG[9-0] TS_PPM_HIGH_VAL) is:
1,000,000/100 = 10,000(decimal)

Parts Per Hour example:
To adjust for 1 part per hour at 1 GHz CPTS_RFT_CLK the configured value for TS_PPM[41-0] (through CPSW3_CPSW_NU_CPTS_TS_PPM_LOW_VAL_REG[31-0] TS_PPM_LOW_VAL and CPSW3_CPSW_NU_CPTS_TS_PPM_HIGH_VAL_REG[9-0] TS_PPM_HIGH_VAL) is:
(1,000,0000,000Hz/1pph) * (3600 seconds/hour) = 34630B8A000 (hex)

12.2.1.4.7.8 Event FIFO

All time sync events are pushed onto the Event FIFO. There are 32 locations in the event FIFO with no overrun indication supported. Software must service the event FIFO in a timely manner to prevent FIFO overrun.

12.2.1.4.7.9 Timestamp Compare Output

CPTS features one Time Stamp Compare (CPTS_COMP) output. The CPTS_COMP function is a software oriented feature that is intended to be replaced going forward by the hardware oriented GENF function. CPTS_COMP is not compatible with timepstamp PPM or a non-zero CPSW3_CPSW_NU_CPTS_TS_ADD_VAL_REG[2-0] ADD_VAL value.

12.2.1.4.7.9.1 Non-Toggle Mode: 32-bit

The CPTS_COMP output is asserted for CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG[31-0] TS_COMP_LENGTH periods when the CPSW3_CPSW_NU_CPTS_EVENT_0_REG[31-0] TIME_STAMP value (lowe 32-bits) compares with the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG[31-0] TS_COMP_VAL and the length value is non-zero. The CPTS_COMP rising edge occurs three CPTS_RFT_CLK clock periods after the values compare. A timestamp compare event is pushed into the event FIFO when CPTS_COMP is asserted. The polarity of the CPTS_COMP output is determined by the CPSW3_CPSW_NU_CPTS_CONTROL_REG[2] TS_COMP_POLARITY bit. The output is asserted low when the polarity bit is 0h.

12.2.1.4.7.9.2 Non-Toggle Mode: 64-bit

64-bit mode operation is identical to 32-bit mode except that all 64-bits of the TIME_STAMP are used (CPSW3_CPSW_NU_CPTS_EVENT_0_REG and CPSW3_CPSW_NU_CPTS_EVENT_3_REG). In 32-bit mode only the lower 32-bits (CPSW3_CPSW_NU_CPTS_EVENT_0_REG) are used.

12.2.1.4.7.9.3 Toggle Mode: 32-bit

The CPTS_COMP output is asserted (CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG[31-0] TS_COMP_LENGTH) for CPTS_RFT_CLK clock periods when the TIME_STAMP[31:0] value compares with the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG and the length value is non-zero. The CPTS_COMP toggles thereafter on CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG[31-0] TS_COMP_LENGTH for CPTS_RFT_CLK periods. The length high or low can be adjusted by writing the CPSW3_CPSW_NU_CPTS_TS_COMP_NUDGE_REG[7-0] NUDGE bit field value which is a two's complement value. A value of FFh will subtract one CPTS_RFT_CLK period from the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG[31-0] TS_COMP_LENGTH value. A value of 0x01h will add one CPTS_RFT_CLK period to the CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG[31-0] TS_COMP_LENGTH value. Only a single high or low time is adjusted (nudged) and the CPSW3_CPSW_NU_CPTS_TS_COMP_NUDGE_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred. The CPTS_COMP output is asserted low when the CPSW3_CPSW_NU_CPTS_CONTROL_REG[2] TS_COMP_POLARITY bit is 0h. No compare events and no CPTS_EVNT interrupts are generated in toggle mode. The CPSW3_CPSW_NU_CPTS_CONTROL_REG[6] TS_COMP_TOG bit must be set for toggle mode (value 1h). Note this bit must be set before writing a non-zero value to CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG register.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.7.9.4 Toggle Mode: 64-bit

64-bit mode operation is identical to 32-bit mode except that all 64-bits of the TIMESTAMP are used (CPSW3_CPSW_NU_CPTS_EVENT_0_REG and CPSW3_CPSW_NU_CPTS_EVENT_3_REG). In 32-bit mode only the lower 32-bits (CPSW3_CPSW_NU_CPTS_EVENT_0_REG) are used.

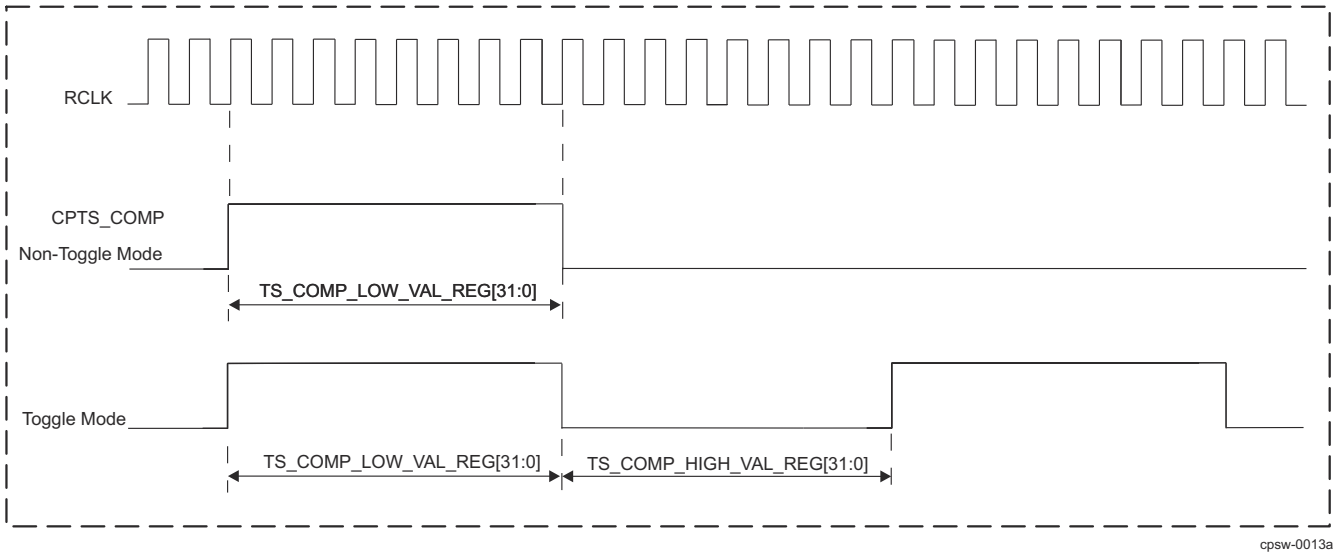


Figure 12-104. CPTS_COMP Output in Toggle and Non-Toggle Mode

12.2.1.4.7.10 Timestamp Sync Output

The CPTS_SYNC output is a selected bit of the [31-0]TIME_STAMP counter value. One of bits 17-31 can be selected in CPSW3_CPSW_NU_CPTS_CONTROL_REG[31-28] TS_SYNC_SEL. The CPTS_SYNC output is disabled when CPSW3_CPSW_NU_CPTS_CONTROL_REG[31-28] TS_SYNC_SEL is zero.

If the selected counter bit is 1 at the time when TS_SYNC_SEL value is written then a rising edge will not occur on the CPTS_SYNC output. A rising edge will occur on the CPTS_SYNC output upon the next transition to 1 of the selected counter bit. The TS_SYNC_SEL value must be written to zero before changing to a different non-zero value. No events are generated due to the CPTS_SYNC operation. The CPTS_SYNC output is two CPTS_RFT_CLK periods after the actual count value.

12.2.1.4.7.11 Timestamp GENFn Output

The CPTS_GENFn outputs have a programmable cycle (frequency) with a PPM feature and software nudge feature. The CPTS_GENFn output cycle is CPSW3_CPSW_NU_CPTS_TS_GENF_LENGTH_REG_L[31-0] CPTS_RFT_CLK periods (which is different than CPTS_COMP operation). Figure 12-105 represents the CPTS_GENFn output signal.

The CPTS_GENFn output cycle is CPSW3_CPSW_NU_CPTS_TS_GENF_LENGTH_REG_L[31-0] CPTS_RFT_CLK periods beginning when the 64-bit TIME_STAMP value compares with the 64-bit GENFn_COMP value (CPSW3_CPSW_NU_CPTS_TS_GENF_COMP_LOW_REG_L and CPSW3_CPSW_NU_CPTS_TS_GENF_COMP_HIGH_REG_L registers) and the length value is non-zero. The CPTS_GENFn output cycle repeats thereafter every CPSW3_CPSW_NU_CPTS_TS_GENF_LENGTH_REG_L[31-0] CPTS_RFT_CLK periods. The upper 32-bit word should be written first for 64-bit values. The length should be zero while the comparison value and other configuration parameters are being configured. The length should be written non-zero to enable operations last. The first cycle after comparison is active high when the CPSW3_CPSW_NU_CPTS_CONTROL_REG[2] TS_COMP_POLARITY bit is low. No compare events and no CPTS_EVNT interrupts are generated.

Copyright © 2026 Texas Instruments Incorporated

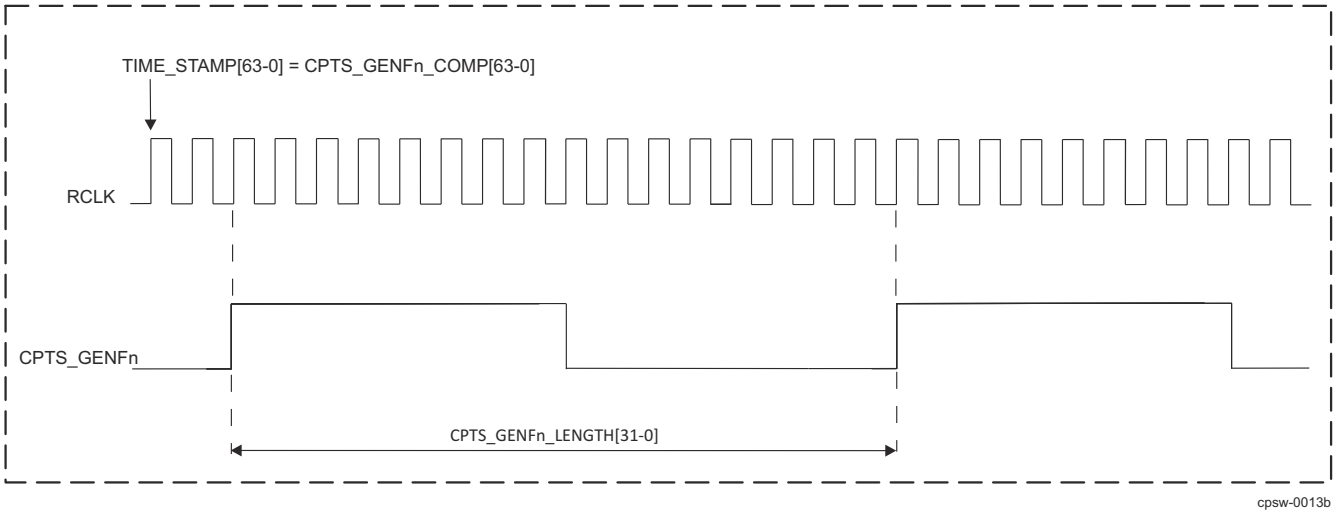


Figure 12-105. CPTS_GENFn Output Signal Diagram

12.2.1.4.7.11.1 GENFn Nudge

The cycle length can be adjusted by writing the CPSW3_CPSW_NU_CPTS_TS_COMP_NUDGE_REG[7-0] NUDGE register value which is a two's complement value. A value of FFh will subtract 1 CPTS_RFT_CLK from theCPSW3_CPSW_NU_CPTS_TS_GENF_LENGTH_REG_L[31-0] value. A value of 1h will add 1 CPTS_RFT_CLK to theCPSW3_CPSW_NU_CPTS_TS_GENF_LENGTH_REG_L[31-0] value. The CPSW3_CPSW_NU_CPTS_TS_COMP_NUDGE_REG[7-0] NUDGE value is cleared to zero when the nudge has occurred.

12.2.1.4.7.11.2 GENFn PPM

The CPTS_GENFn output cycle can be adjusted by parts per million or by parts per hour. Writing a non-zero value to CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_LOW_REG_L/ CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_HIGH_REG_L enables PPM operations. The PPM counter continually loads and decrements to zero and then loads again. A single CPTS_RFT_CLK adjustment is made when the PPM counter decrements to zero. The adjustment is up or down depending on the CPSW3_CPSW_NU_CPTS_TS_GENF_CONTROL_REG[0] PPM_DIR bit. When PPM_DIR bit is set a single CPTS_RFT_CLK time is subtracted from the generate function counter which has the effect of increasing the generate function frequency by the PPM amount. When PPM_DIR bit is cleared a single CPTS_RFT_CLK time is added to the generate function counter which has the effect of decreasing the generate function frequency by the PPM amount.

Parts Per Million example:
To adjust for 100 parts per million the configured value for GENF_PPM[41-0] (through CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_LOW_REG_L and CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_HIGH_REG_L) is:
1,000,000/100 = 10,000(decimal)

Parts Per Hour example:
To adjust for 1 part per hour at 1 GHz CPTS_RFT_CLK the configured value for GENF_PPM[41-0] (through CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_LOW_REG_L and CPSW3_CPSW_NU_CPTS_TS_GENF_PPM_HIGH_REG_L) is:
(1,000,0000,000Hz/1pph) * (3600 seconds/hour) = 34630B8A000 (hex)

12.2.1.4.7.12 Timestamp ESTFn

Each Ethernet port has a dedicated ESTFn generator which operates identically to the GENFn function.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.7.13 Time Sync Events

Time Sync events are 96-bit values that are pushed onto the event FIFO and read by software in 32-bit reads. Four 32-bit registers, CPSW3_CPSW_NU_CPTS_EVENT_0_REG through CPSW3_CPSW_NU_CPTS_EVENT_3_REG hold the data of a time sync event. There are eight types of sync events:

- Time Stamp Push Event
- Time Stamp Counter Rollover Event (32-bit mode only)
- Time Stamp Counter Half-rollover Event (32-bit mode only)
- Hardware Time Stamp Push Event
- Ethernet Receive Event
- Ethernet Transmit Event
- Time Stamp Compare Event
- Host Transmit Event

12.2.1.4.7.13.1 Time Stamp Push Event

Software can obtain the current time stamp value (at the time of the write) by initiating a time stamp push event. The push event is initiated by setting the [0]TS_PUSH bit of the CPSW3_CPSW_NU_CPTS_TS_PUSH_REG register. The time stamp value is returned in the event, along with a time stamp push event code. The upper 32-bits (CPSW3_CPSW_NU_CPTS_EVENT_3_REG register) of the timestamp are zero in 32-bit mode.

12.2.1.4.7.13.2 Time Stamp Counter Rollover Event (32-bit mode only)

The CPTS module contains a 32-bit time stamp value (CPSW3_CPSW_NU_CPTS_EVENT_0_REG). The counter upper bits are maintained by host software. The rollover event indicates to software that the time stamp counter has rolled over from 0xFFFF FFFF to 0x0000 0000 and the software-maintained upper count value should be incremented. This event occurs only in 32-bit mode.

12.2.1.4.7.13.3 Time Stamp Counter Half-rollover Event (32-bit mode only)

The CPTS includes a time stamp counter half-rollover event. The half-rollover event indicates to software that the time stamp value (CPSW3_CPSW_NU_CPTS_EVENT_0_REG[31-0] TIME_STAMP) has incremented from 0x7FFF FFFF to 0x8000 0000. The half-rollover event is included to enable software to correct a misaligned event condition. This event occurs only in 32-bit mode.

The half-rollover event is included to enable software to determine the correct time for each event that contains a valid time stamp value, such as an Ethernet event. If an Ethernet event occurs around a counter rollover (full rollover), the rollover event could possibly be loaded into the event FIFO before the Ethernet event, even though the Ethernet event time was actually taken before the rollover. Figure 12-106 shows a misalignment condition. This misaligned event condition arises because an Ethernet event time stamp occurs at the beginning of a packet and time passes before the packet is determined to be a valid synchronization packet. The misaligned event condition occurs if the rollover occurs in the middle, after the packet time stamp has been taken, but before the packet has been determined to be a valid time sync packet.

Host software must detect and correct for misaligned event conditions. For every event time stamp after a rollover and before a half-rollover, software must examine the time stamp most significant bit. If bit 31 of the time stamp value is low (0x0000 0000 through 0x7FFF FFFF), then the event time stamp was taken after the rollover and no correction is required. If the value is high (0x8000 0000 through 0xFFFF FFFF), the time stamp value was taken before the rollover and a misalignment is detected. The misaligned case indicates to software that it must subtract one from the upper count value stored in software to calculate the correct time for the misaligned event. The misaligned event occurs only on the rollover boundary and not on the half-rollover boundary. Software only needs to check for misalignment from a rollover event to a half-rollover event.

When a rollover occurs, software increments the software time stamp upper value. The misaligned case indicates to software that the misaligned event time stamp has a valid upper value that is pre-increment, so one must be subtracted from the upper value to allow software to calculate the correct time for the misaligned event.

Copyright © 2026 Texas Instruments Incorporated

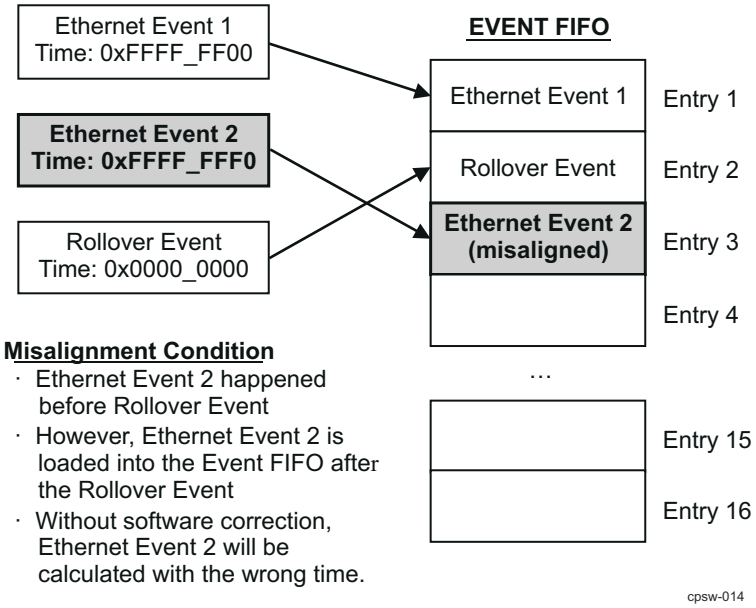


Figure 12-106. Event FIFO Misalignment Condition

12.2.1.4.7.13.4 Hardware Time Stamp Push Event

There are four hardware time stamp inputs (CPTS_HW[1:4]_TS_PUSH events) that can cause hardware time stamp push events to be loaded into the Event FIFO. Each time stamp input is mapped in the device as shown in *CPTS Integration*. The event is loaded into the event FIFO on the rising edge of the timer, and the PORT_NUMBER field in the CPSW3_CPSW_NU_CPTS_EVENT_1_REG register indicates the hardware push input that caused the event (encoded).

The hardware time stamp inputs are asynchronous and are low frequency signals. The CPTS logic synchronizes and performs a rising edge detect on the incoming asynchronous input.

Each hardware time stamp input must be asserted for at least 10 periods of the selected CPTS_RFT_CLK clock. Each input can be enabled or disabled by setting the respective bits in the CPSW3_CPSW_NU_CPTS_CONTROL_REG register.

Hardware time stamps are intended to be an extremely low frequency signals, such that the event FIFO does not overrun. Software must keep up with the event FIFO and ensure that there is no overrun, or events will be lost.

12.2.1.4.7.13.5 Ethernet Port Events

Packets transmitted or received on each Ethernet port can generate Ethernet Transmit Events or Ethernet Receive Events, respectively. The CPTS hardware will decode each packet to determine if it is a valid CPTS time sync event.

According to the IEEE 802.3 Ethernet standard, each Ethernet frame contains a 2-octet EtherType field to indicate which protocol is encapsulated in the PayLoad field, as shown in Figure 12-107. For standard time sync packets, this will contain the EtherType for the Precision Time Protocol (IEEE 1588), which is defined as 0x88F7. The CPTS hardware will compare this field to the TS_LTYPE1 field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG or the TS_LTYPE2 field in CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register (depending on which enable bit was set) , which should also be programmed to 88F7h.

When a virtual LAN is used, an additional 4-octet 802.1Q tag is inserted in the Ethernet frame before the EtherType field, as shown in Figure 12-107. To indicate to the CPTS hardware that a

Copyright © 2026 Texas Instruments Incorporated

virtual LAN is in use, the TS_TX_VLAN_LTYPE1_EN (or TS_TX_VLAN_LTYPE2_EN) enable bit must be set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register. The EtherType for the 802.1Q tag is defined as 0x8100, and the CPTS hardware will compare this value to the TS_VLAN_LTYPE1 (or TS_VLAN_LTYPE2 depending on which enable bit was set) field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register, which should also be programmed to 0x8100.

When two stacked VLANs are used, two additional 4-octet 801.Q tags are inserted in the Ethernet frame before the EtherType field, as shown in Figure 12-107. In this case, both TS_VLAN_LTYPE1 and TS_VLAN_LTYPE2 must be enabled. The outer tag must match the value of the TS_VLAN_LTYPE1 field, and the inner tag must match the value of the TS_VLAN_LTYPE2 field.

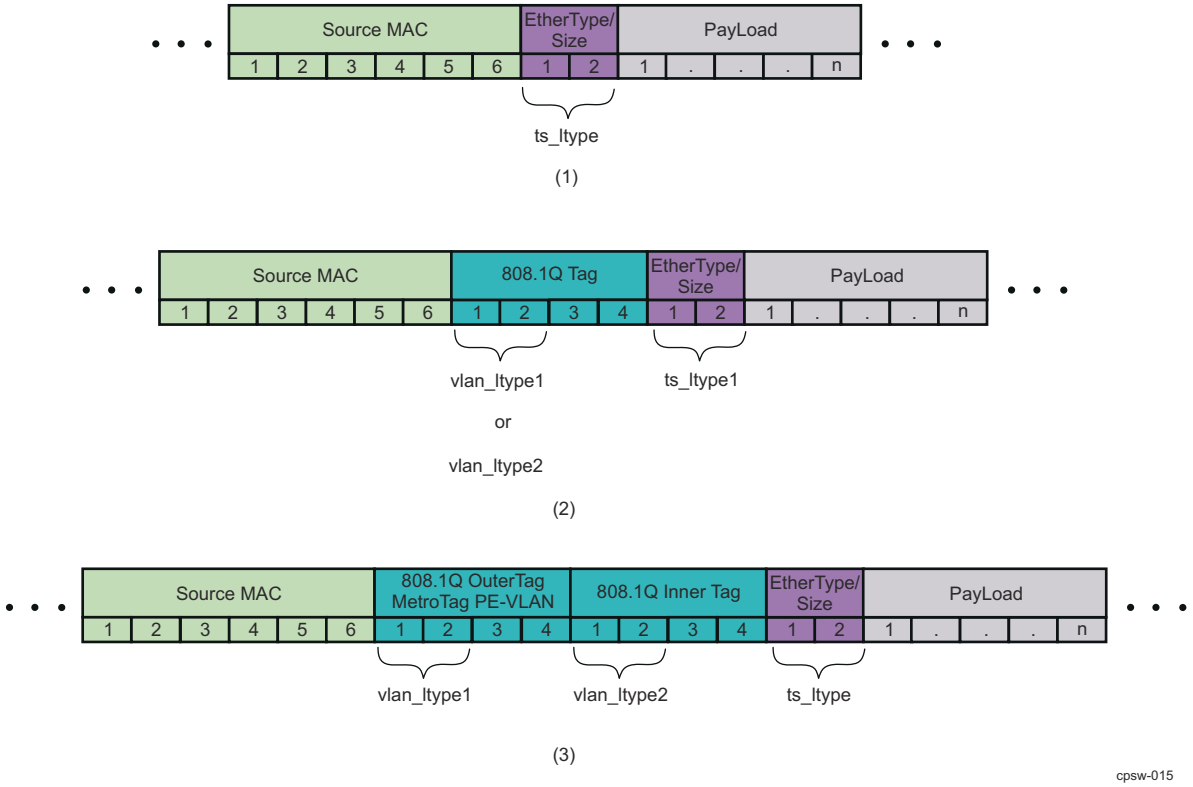


Figure 12-107. Partial Ethernet-II Frames Showing Register Mapping of EtherTypes for a Simple Frame (1), a Single 1Q Tag Added (2), and Two 1Q Tags Added (3)

12.2.1.4.7.13.5.1 Ethernet Port Receive Event

This section describes Ethernet port receive events. Ethernet port generates time synchronization events for valid received time sync packets. For every packet received on the Ethernet port, a timestamp will be captured by the receive module inside the CPTS for the corresponding port. The time stamp will be captured by the receive module regardless of whether or not the packet is a time synchronization packet to make sure that the time stamp is captured as soon as possible. The packet is sampled on both the rising and falling edges of the CPTS_RFT_CLK, and the time stamp will be captured once the start of frame delimiter for the receive packet is detected.

After the time stamp has been captured, the receive interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPSW decoder determines if the packet is a valid Ethernet receive time synchronization event. The receive interface for the port will use the following criteria to determine if the packet is a valid Annex D, Annex E, or Annex F time synchronization Ethernet receive event:

Annex D (IPv4)

Copyright © 2026 Texas Instruments Incorporated

1. Receive annex D time sync is enabled (TS_RX_ANNEX_D_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true.
   a. The first packet LTYPE matches 0x0800
   b. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x0800
   c. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x0800
   d. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches 0x0800
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).

Note

The byte numbering assumes that there are no VLANs. The byte number is intended to show the relative order of the bytes.

4. Byte 20 contains 0bXXX00000 (5 lower bits zero) and Byte 21 contains 0x00 (fragment offset zero)
5. Byte 22 contains 0x01 (HOP Limit = 1) if the TS_TTL_NONZERO bit in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h, or byte 22 contains any value if CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG is set to 1h. Byte 22 is the TTL/HOP field.
6. Byte 23 contains 0x11 (Next Header UDP Fixed).
7. The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h and Bytes 30 through 33 contain:
   a. Decimal 224.0.1.129 and the TS_129 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. Decimal 224.0.1.130 and the TS_130 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   c. Decimal 224.0.1.131 and the TS_131 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   d. Decimal 224.0.1.132 and the TS_132 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   e. Decimal 224.0.0.107 and the TS_107 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set

   -OR-

   The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set and Bytes 30 through 33 contain any values.
8. Bytes 36 and 37 contain:
   a. Decimal 0x01 and 0x3F respectively and the TS_319 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set -OR-

Copyright © 2026 Texas Instruments Incorporated

b. Decimal 0x01 and 0x40 respectively and the TS_320 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set.

9. The PTP message begins in byte 42.
10. The packet message type is enabled in the TS_MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
11. The packet was received without error (not long/short/mac_ctl/CRC/code/align).

Annex E (IPv6)

1. Receive annex E time sync is enabled (TS_RX_ANNEX_E_EN bit is set in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true.
   a. The first packet LTYPE matches 0x86dd.
   b. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x86dd
   c. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x86dd
   d. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches 0x86dd
3. Byte 14 (the byte after the LTYPE) contains 0x6X (IPv6).
4. Byte 20 contains 0x11 (UDP Fixed Next Header).
5. Byte 21 contains 0x01 (Hop Limit = 1) if the TS_TTL_NONZERO bit in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h, or byte 21 contains any value if TS_TTL_NONZERO is set to 1h. Byte 21 is the TTL/HOP field.
6. The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0 and Bytes 38 through 53 contain:
   a. FF0M:0:0:0:0:0:0:0181 and the TS_129 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. FF0M:0:0:0:0:0:0:0182 and the TS_130 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   c. FF0M:0:0:0:0:0:0:0183 and the TS_131 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   d. FF0M:0:0:0:0:0:0:0184 and the TS_132 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   e. FF0M:0:0:0:0:0:0:006B and the TS_107 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set

Copyright © 2026 Texas Instruments Incorporated

Note

All values above are 16-bit hex numbers where M is enabled in the TS_MCAST_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL2_REG register.

-OR-

The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set to 1h and Bytes 38 through 53 contain any value.

7. Bytes 56 and 57 contain (UDP Header in bytes 54 through 61):
   a. Decimal 0x01 and 0x3F respectively and the TS_319 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL2_REG register is set, or
   b. Decimal 0x01 and 0x40 respectively and the TS_320 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL2_REG register is set.
8. The PTP message begins in byte 62.
9. The packet message type is enabled in the MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL2_REG register.
10. The packet was received without error (not long/short/mac_ctl/CRC/code/align).

Annex F (IEEE 802.3)

1. Receive Annex F time sync is enabled (TS_RX_ANNEX_F_EN is set in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true:
   a. The first packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG/ CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register. LTYPE 1 should be used when only one time sync LTYPE is to be enabled.
   b. The first packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
   c. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register
   d. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
   e. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register.
   f. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the

Copyright © 2026 Texas Instruments Incorporated

CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.

g. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register.
h. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_RX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register

3. The PTP message begins in the byte after the LTYPE.
4. The packet message type is enabled in the TS_MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
5. The packet was received without error (not long/short/mac_ctl/CRC/code/align).

If all of the criteria described above are met for either Annex D, Annex E, or Annex F, and the packet is determined to be a valid time synchronization packet, then the RX interface will push an Ethernet receive event into the event FIFO.

12.2.1.4.7.13.5.2 Ethernet Port Transmit Event

This section describes Ethernet port transmit events. For every packet transmitted on the Ethernet ports, the port transmit interface will begin parsing the packet to determine if it is a valid Ethernet time synchronization packet. The CPTS transmit interface for the port will use to the following criteria to determine if the packet is a valid time synchronization Ethernet transmit event. The CPSW decoder determines if the packet is a valid ethernet receive time synchronization event. To be a valid Ethernet transmit time synchronization event, the conditions listed below must be true for either Annex D, Annex E, or Annex F:

Annex D (IPv4)

1. Transmit time sync is enabled (TS_TX_ANNEX_D_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true.
   a. The first packet LTYPE matches 0x0800
   b. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x0800

Copyright © 2026 Texas Instruments Incorporated

c. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x0800
d. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches 0x0800

3. Byte 14 (the byte after the LTYPE) contains 0x45 (IPv4).

Note

The byte numbering assumes that there are no VLANs. The byte number is intended to show the relative order of the bytes.

4. Byte 20 contains 0bXXX00000 (5 lower bits zero) and Byte 21 contains 0x00 (fragment offset zero)
5. Byte 22 contains 0x01 (HOP Limit = 1) if the TS_TTL_NONZERO bit in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h, or byte 22 contains any value if TS_TTL_NONZERO is set to 1h. Byte 22 is the TTL/HOP field.
6. Byte 23 contains 0x11 (Next Header UDP Fixed).
7. The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h and Bytes 30 through 33 contain:
   a. Decimal 224.0.1.129 and the TS_129 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. Decimal 224.0.1.130 and the TS_130 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   c. Decimal 224.0.1.131 and the TS_131 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   d. Decimal 224.0.1.132 and the TS_132 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   e. Decimal 224.0.0.107 and the TS_107 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   f. The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set and Bytes 30 through 33 contain any values.
8. Bytes 36 and 37 contain:
   a. Decimal 0x01 and 0x3F respectively and the TS_319 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. Decimal 0x01 and 0x40 respectively and the TS_320 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set.
9. The PTP message begins in byte 42.
10. The packet message type is enabled in the TS_MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
11. The packet was sent by host port 0.

Annex E (IPv6)

1. Transmit annex E time sync is enabled (TS_TX_ANNEX_E_EN bit is set in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true.

Copyright © 2026 Texas Instruments Incorporated

a. The first packet LTYPE matches 0x86dd.
b. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x86dd
c. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches 0x86dd
d. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches 0x86dd

3. Byte 14 (the byte after the LTYPE) contains 0x6X (IPv6).
4. Byte 20 contains 0x11 (UDP Fixed Next Header).
5. Byte 21 contains 0x01 (Hop Limit = 1) if the TS_TTL_NONZERO bit in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0h, or byte 21 contains any value if TS_TTL_NONZERO is set to 1h. Byte 21 is the TTL/HOP field..
6. The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is cleared to 0 and Bytes 38 through 53 contain:
   a. FF0M:0:0:0:0:0:0:0181 and the TS_129 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. FF0M:0:0:0:0:0:0:0182 and the TS_130 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   c. FF0M:0:0:0:0:0:0:0183 and the TS_131 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   d. FF0M:0:0:0:0:0:0:0184 and the TS_132 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   e. FF0M:0:0:0:0:0:0:006B and the TS_107 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set

   Note

   All values above are 16-bit hex numbers where M is enabled in the TS_MCAST_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL2_REG register.

   -OR-

   The TS_UNI_EN bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set to 1h and Bytes 38 through 53 contain any value.

7. Bytes 56 and 57 contain (UDP Header in bytes 54 through 61):
   a. Decimal 0x01 and 0x3F respectively and the TS_319 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set, or
   b. Decimal 0x01 and 0x40 respectively and the TS_320 bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register is set.
8. The PTP message begins in byte 62.

Copyright © 2026 Texas Instruments Incorporated

9. The packet message type is enabled in the TS_MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
10. The packet was sent by host port 0.

Annex F (IEEE 802.3)

1. Transmit time sync is enabled (TS_TX_ANNEX_F_EN is set in the switch CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register).
2. One of the sequences below is true:
   a. The first packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register. LTYPE 1 should be used when only one time sync LTYPE is to be enabled.
   b. The first packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
   c. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register.
   d. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_SEQ_LTYPE_REG register.
   e. The first packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register.
   f. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
   g. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register.
   h. The first packet LTYPE matches TS_VLAN_LTYPE1 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE1_EN is set in the

Copyright © 2026 Texas Instruments Incorporated

CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the second packet LTYPE matches TS_VLAN_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_VLAN_LTYPE_REG register and TS_TX_VLAN_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register and the third packet LTYPE matches TS_LTYPE2 in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_LTYPE2_REG register and TS_LTYPE2_EN is set in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register

3. The packet message type is enabled in the TS_MSG_TYPE_EN field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_TS_CTL_REG register.
4. The packet was sent by host port 0.

If all of the criteria described above are met, and the packet is determined to be a valid time synchronization packet, then the time stamp for the transmit event will not be generated until the start of frame delimiter of the packet is actually transmitted. The start of frame delimiter will be sampled on every rising and falling edge of the CPTS_RFT_CLK. Once the packet is transmitted, then the TX interface will push an Ethernet transmit event into the event FIFO.

12.2.1.4.7.13.5.3 Time Sync Message Types

Table 12-134. Values of Message Type Field

Message Type	Value (hex)
Sync	0
Delay_Req	1
Pdelay_Req	2
Pdelay_Resp	3
Reserved	4:7
Follow_Up	8
Delay_Resp	9
Pdelay_Resp_Follow_Up	A
Announce	B
Signaling	C
Management	D
Reserved	E:F

Once a transmitted or received packet is determined to be a valid time sync packet, the Ethernet Transmit Event or Ethernet Receive Event is loaded onto the Event FIFO.

The CPSW3_CPSW_NU_CPTS_EVENT_1_REG register contains the Message Type and Sequence ID values from the original time sync packet. The CPSW3_CPSW_NU_CPTS_EVENT_0_REG (and CPSW3_CPSW_NU_CPTS_EVENT_3_REG) register contains the time stamp value when the packet arrived at the corresponding port.

12.2.1.4.7.14 Timestamp Compare Event

Note

Timestamp compare events are generated for non-toggle mode only.

The CPTS can generate an event for a time stamp comparison in 32-bit or 64-bit mode.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.7.14.1 32-Bit Mode

The CPTS_COMP output is also asserted when the event is generated. The event is generated when the 32-bit time stamp value (CPSW3_CPSW_NU_CPTS_EVENT_0_REG) compares with the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG register and the CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG value is non-zero. The CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG value should be written by software after the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG register is written and should be zero when the comparison value is written.

12.2.1.4.7.14.2 64-Bit Mode

The CPTS_COMP output is also asserted when the event is generated. The event is generated when the 64-bit time stamp value (CPSW3_CPSW_NU_CPTS_EVENT_0_REG and CPSW3_CPSW_NU_CPTS_EVENT_3_REG) compares with the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG and CPSW3_CPSW_NU_CPTS_TS_COMP_HIGH_VAL_REG registers and the CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG value is non-zero. The CPSW3_CPSW_NU_CPTS_TS_COMP_LEN_REG value should be written by software after the CPSW3_CPSW_NU_CPTS_TS_COMP_VAL_REG register is written and should be zero when the comparison value is written.

12.2.1.4.7.15 Host Transmit Event

The host can send a packet to be transmitted on an Ethernet port that will generate a time synchronization event. The host sets the TSTAMP_EN bit and sends the DOMAIN, MESSAGE_TYPE, and SEQUENCE_ID in the additional control information that resides in the protocol specific section of the descriptor that is transmitted to the CPSW_3G. An event is then generated and placed on the event FIFO once the packet is transmitted. Host events allow the user to timestamp exactly when a software generated packet exits the device.

12.2.1.4.7.16 CPTS Interrupt Handling

When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred. The following steps should be taken to process time sync events using interrupts:

1. Enable the TS_PEND interrupt by setting the TS_PEND_EN bit of the CPSW3_CPSW_NU_CPTS_INT_ENABLE_REG register.
2. Upon interrupt, read the CPSW3_CPSW_NU_CPTS_EVENT_0_REG through CPSW3_CPSW_NU_CPTS_EVENT_3_REG registers values.
3. Set the CPSW3_CPSW_NU_CPTS_EVENT_POP_REG[0] EVENT_POP bit to 1h to pop the previously read value off of the event FIFO.
4. Process the interrupt as required by the application software.

Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way:

1. Enable the TS_PEND interrupt by setting the TS_PEND_EN bit of the CPSW3_CPSW_NU_CPTS_INT_ENABLE_REG
2. Upon interrupt, enter the CPTS service routine.
3. Read the CPSW3_CPSW_NU_CPTS_EVENT_0_REG through CPSW3_CPSW_NU_CPTS_EVENT_3_REG registers values.
4. Set the CPSW3_CPSW_NU_CPTS_EVENT_POP_REG[0] EVENT_POP bit to 1h to pop the previously read value off of the event FIFO.
5. Wait for an amount of time greater than four CPTS_RFT_CLK periods plus four CPPI_ICLK periods.
6. Read the TS_PEND_RAW bit in the CPSW3_CPSW_NU_CPTS_INTSTAT_RAW_REG register to determine if another valid event is in the event FIFO. If bit TS_PEND_RAW is asserted, go to step 3. If bit TS_PEND_RAW is not asserted proceed with step 7.
7. Process the interrupt(s) as required by the application software.

Copyright © 2026 Texas Instruments Incorporated

Software also has the option of disabling the interrupt and polling the TS_PEND_RAW bit of the CPSW3_CPSW_NU_CPTS_INTSTAT_RAW_REG register to determine if a valid event is on the event FIFO.

12.2.1.4.8 CPPI Streaming Packet Interface

The receive streaming interface on port 0 of the CPSW is responsible for receiving packet for Ethernet egress data from the packet streaming switch in the . The CPPI receive port is equivalent to an Ethernet port with the difference being that the data is provided to the CPSW in the 128-bit streaming interface data format instead RGMII data format.

In addition to the packet data, the receive streaming interface also can provide additional control information that resides in the information words of the descriptor that was transmitted to the CPSW.

The tables below show the information that may be passed along with which descriptor information word to put it in.

12.2.1.4.8.1 Port 0 CPPI Transmit Packet Streaming Interface (CPSW_3G Egress)

The CPSW_3G has a single transmit packet streaming interface. All Ethernet packet data destined for the host (Port 0) is transferred on the transmit packet streaming interface. The transmit packet streaming interface is equivalent to an Ethernet MAC output with the difference being that 128-bit streaming interface data is output. Egress packet data is packed on the 128-bit data bus with all words having 16-bytes except possibly the last packet data word which is the word previous to the EOP word. INFO Word 0–3 is transferred on SOP. The EOP word contains the packet status 0–3 (data type 24) which includes the timestamp and checksum data. Packets are not dropped on the transmit streaming interface due to pushback, but packets may be dropped in the associated priority FIFO.

INFO Word 0–3 and Status Data Word 0–3 (on EOP) are the only non-payload data word types that are transferred. Long packets are truncated at the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_MAXLEN_REG_k[13-0] RX_MAXLEN byte value of the ingress port (only the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_RX_MAXLEN_REG_k number of bytes are kept if long packets are transferred due to CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k register copy error frames set - RX_CEF_EN). MAC control frames are only transferred if the receiving Ethernet port has the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k[24] RX_CMF_EN bit set.

The error encoding on the TXST_PKT_ERR[3:0] output is shown in Table 12-135:

Table 12-135. Error Encoding on the TXST_PKT_ERR[3:0] Output

TXST_PKT_ERR[3:0]	Description
0000	No Error
0001	CRC Error
0010	Code or alignment error
0011	Short (no code/align/crc error)
0100	FragCRC (short with crc error)
0101	Frag Code/Align (short with code/align error)
0110	Long
0111	Jabber CRC (long with crc error)
1000	Jabber Code/Align (long with code/align error)
1001	Mac ontrol packet (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k[24] RX_CMF_EN set on ingress Ethernet port)
1010	Mac control CRC
1011	Mac control Code/Align
1100	Mac control short/frag (short MAC control frame with CRC/Code/Align)
1101	Mac control long/jabber (long MAC control frame with CRC/Code/Align)
1110	Reserved

Copyright © 2026 Texas Instruments Incorporated

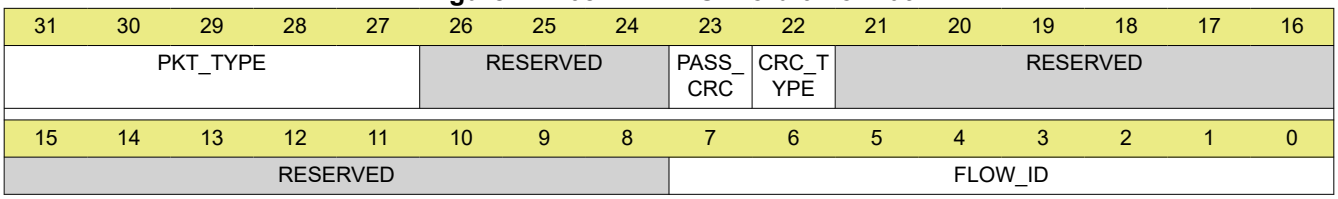
Table 12-135. Error Encoding on the TXST_PKT_ERR[3:0] Output (continued)

TXST_PKT_ERR[3:0]	Description
1111	Reserved

The CPPI Port 0 transmit packet streaming interface has a single output thread. If a packet transmission has begun then the entire packet will be transmitted before the next packet is sent (packet data from multiple packets are not interleaved on the transmit streaming interface). The default egress flow is the port number minus 1, concatenated with the 3-bit Port 0 transmit FIFO hardware switch priority. For example, a packet that was received on port 4 with a Port 0 transmit FIFO hardware switch priority of 5 would be sent on flow 29 (decimal) or flow 0b0011101 (binary). Priority remapping on ingress and Port 0 transmit egress effects the output flow.

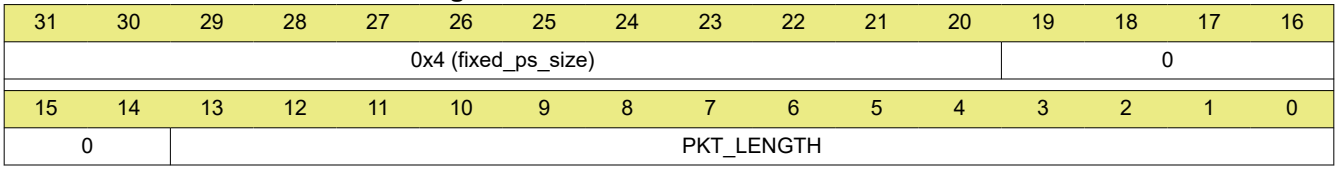
INFO Words are a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

Figure 12-108. TX INFO Word 0 Format



Bit	Field	Description
31-27	PKT_TYPE	Always set to 0b00111. Host PD (Packet Descriptor) Word 2. Packet Type: bits[31-27].
26-24	RESERVED	Reserved.
23	PASS_CRC	This bit is cleared to zero (no CRC passed) when the P0_TX_CRC_REMOVE bit in the CPSW3_CONTROL_REG register is set (and the egress packet has no errors). When the remove bit is cleared to zero then this bit is cleared and no CRC is passed with the output packet. The packet length includes the CRC if it is present.
22	CRC_TYPE	The packet CRC type. The type of CRC passed is determined by CRC_TYPE field in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k register (not by the type of CRC the packet had on Ethernet port ingress). Host PD Word 1. Protocol Specific Flags: bits[27-24]. 0h: Ethernet CRC 1h: Castagnoli CRC
21-8	RESERVED	Reserved.
7-0	FLOW_ID	This is the packet output transmit streaming interface flow. The default flow ID can be overridden by ALE classification (Thread mapping). The switch default flow is the 3-bit “From Port” value concatenated with the 3-bit “Switch Priority” {From_Port[2:0], Switch_Priority[2:0]} as shown below: Host PD (Packet Descriptor) Word 1. Flow ID: bits[13-0]. 0h: The packet was received on Ethernet port 1 Switch Priority – The actual hardware switch priority that the packet was stored in on the CPPI transmit FIFO.

Figure 12-109. TX INFO Word 1 Format



Copyright © 2026 Texas Instruments Incorporated

Bit	Field	Description
31-20	FIXED_PS_SIZE	Fixed packet size: 0x4
19-14	RESERVED	Reserved.
13-0 (Host PD (Packet Descriptor) Word 1. Packet Length: bits[21-0])	PKT_LENGTH	Specifies the number of bytes in the entire packet. Offset bytes are not included. Valid only on SOP. The packet length must be greater than zero. The packet data will be truncated to the packet length if the packet length is shorter than the sum of the packet buffer descriptor buffer lengths. A host error occurs if the packet length is greater than the sum of the packet buffer descriptor buffer lengths.

Figure 12-110. TX INFO Word 2 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFFF															

Figure 12-111. TX INFO Word 3 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SRC_ID							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															

Bit	Field	Description
31-24	RESERVED	Reserved.
23-16	SRC_ID	The packet SRC_ID value comes from the PORT1/PORT2 field in the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_SRC_ID_A_REG register. (src_tag) PD (Packet Descriptor) Word 3. Source Tag Low bits[23-16] if RFLOW[a]_RFC.rx_src_tag_lo_sel = 0x4 or (src_tag) PD (Packet Descriptor) Word 3. Source Tag High bits[31-24] if RFLOW[a]_RFC.rx_src_tag_hi_sel = 0x4
15-0	RESERVED	Reserved.

Note

TX Status Data Word [0..3] are mapped to Host Packet Descriptor Protocol Specific Words if RFLOW[a]_RFA.rx_psinfo_present = 1 and RFLOW[a]_RFA.rx_ps_location = 0

Figure 12-112. TX Status Data Word 0 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP[31:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[31:0]															

Bit	Field	Description
31-0	TIMESTAMP[31:0]	Contains the lower 32-bits of the time stamp value.

Figure 12-113. TX Status Data Word 1 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Copyright © 2026 Texas Instruments Incorporated

Figure 12-113. TX Status Data Word 1 Format (continued)

TIMESTAMP[63:32]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMESTAMP[63:32]															

Bit	Field	Description
31-0	TIMESTAMP[63:32]	Contains the upper 32-bits of the time stamp value.

Figure 12-114. TX Status Data Word 2 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											IPV4_VALID	IPV6_VALID	TCP_UDP_N	FRAGMENT	CHECKSUM_ERROR
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM_ADD															

Bit	Field	Description
31-21	RESERVED	Reserved.
20	IPV4_VALID	An IPV4 TCP or UDP Packet was detected.
19	IPV6_VALID	An IPV6 TCP or UDP Packet was detected.
18	TCP_UDP_N	Valid only when either the IPV4_VALID or IPV6_VALID bits are set. 0h: Indicates UDP packet was detected. 1h: Indicates TCP packet was detected.
17	FRAGMENT	Indicates that an IP fragment was detected. Valid only when when either the IPV4_VALID or IPV6_VALID bits are set.
16	CHECKSUM_ERROR	Valid only when either the IPV4_VALID or IPV6_VALID bits are set.
15-0	CHECKSUM_ADD	This is the value that was summed during the checksum computation. This value is FFFFh for IPV4/6 UDP/TCP packets with no checksum error.

Figure 12-115. TX Status Data Word 3 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															

Bit	Field	Description
31-0	RESERVED	Reserved.

12.2.1.4.8.2 Port 0 CPPI Receive Packet Streaming Interface (CPSW_3G Ingress)

Info Word 0/1/2/3 (INFO1 and INFO3 are ignored) are transferred on SOP. If a timestamp word (Extended Packet Info Word 0/1/2/3) is to be transferred it must be after SOP and before any packet data is transferred. Any following non-data type words will be dropped. The EOP word must be payload data.

Input receive packets cannot be aborted by the host. The INFO Word bit descriptions and Extended Packet INFO Word bit descriptions are shown below. The PASS_CRC bit indicates that the CRC is passed with the packet data. Packets that have a passed CRC that is an error CRC will be output on the Ethernet port with at least one CRC byte inverted to indicate the error if P0_RX_PASS_CRC_ERR bit is set, otherwise they are dropped. The packet is a directed packet when any of the TO_PORT bits are nonzero. A packet may be directed

Copyright © 2026 Texas Instruments Incorporated

only to a single port. The packet will be sent to the port number indicated. For directed packets the lookup process is skipped to determine the destination. However, in vlan aware mode (when VLAN_AWARE bit in the CPSW3_CONTROL_REG register is set to 1h) the lookup is performed to determine untagged egress. Packets longer then the value in CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_RX_MAXLEN_REG[13-0] RX_MAXLEN bit field are dropped. Packets shorter than 60-Bytes are padded to 64-Bytes (after adding pad and CRC) if P0_RX_PAD bit in the CPSW3_CONTROL_REG register is set and if PASS_CRC is clear, otherwise they are dropped. This means that packets shorter than 64-Bytes are dropped if the PASS_CRC info bit is set regardless of P0_RX_PAD bit (packets are padded only if they are short and do not have CRC).

A RX INFO word is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

Note

RX Control Data Words [0..2] are mapped to Host Packet Descriptor Protocol Specific Words if (TCHAN[a]_TCFG.tx_filt_pswords = 0) and (Host PD Word 1.Protocol Specific Region Location.bit[28] = 0h) and (Host PD Word 1.Protocol Specific Valid Word Count.bits[22-27] = 4h)

Figure 12-116. RX INFO Word 0 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								PASS_CRC	CRC_TYPE	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-24	RESERVED	Reserved.
23	PASS_CRC	The PASS_CRC bit indicates that the CRC is passed with the packet data. 0h: CRC is not passed with packet (CRC_TYPE is don't care) 1h: CRC of type CRC_TYPE is passed with the packet.
22 (Host PD (Packet Descriptor) Word 1. Protocol Specific Flags: bits[27-24])	CRC_TYPE	CRC Type 0h: Ethernet CRC 1h: Castagnoli CRC
21-0	RESERVED	Reserved.

Figure 12-117. RX INFO Word 2 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED											TO_PORT				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Bit	Field	Description
31-21	RESERVED	Reserved.

Copyright © 2026 Texas Instruments Incorporated

Bit	Field	Description
20-16 (Host PD (Packet Descriptor) Word 3. Dest Tag Low bits[8-0])	TO_PORT	Port number to send the directed packet to. This field is set by the host. This field is valid on SOP. Directed packets go to the directed port, but an ALE lookup is performed to determine untagged egress in VLAN_AWARE mode. 0h: Not directed 1h: Send the packet to port 1. 2h: Send the packet to port 2.
15-0	RESERVED	Reserved.

Figure 12-118. RX Control Data Word 1 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMESTAMP_EN	RESERVED			DOMAIN								MSG_TYPE			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEQUENCE_ID															

Bit	Field	Description
31	TIMESTAMP_EN	When set, this bit indicates that the packet will generate a timesync event on Ethernet egress (if the CPTS is configured properly) with the associated DOMAIN, MSG_TYPE, and SEQUENCE_ID.
30-28	RESERVED	Reserved.
27-20	DOMAIN	Timesync domain.
19-16	MSG_TYPE	Timesync message type.
15-0	SEQUENCE_ID	Timesync sequence ID.

Figure 12-119. RX Control Data Word 2 Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHECKSUM_RESULT								CHECKSUM_START_BYTE							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHECKSUM_INV	RESERVED	CHECKSUM_BYTECOUNT													

Bit	Field	Description
31-24	CHECKSUM_RESULT	This is the packet byte number where the checksum result will be placed in the egress packet. The first packet byte which is the first byte of the destination address is Byte 1 (not byte zero).
23-16	CHECKSUM_START_BYTE	This is the packet byte number to start the checksum calculation on. The first packet byte is Byte 1.
15	CHECKSUM_INV	When set, a zero checksum value will be inverted and sent as FFFFh.
14	RESERVED	Reserved.
13-0	CHECKSUM_BYTECOUNT	This is the number of bytes to calculate the checksum on. The outgoing Ethernet packet will have a checksum inserted when this value is non-zero.

Other INFO words are not taken into account.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.8.3 Cut-Thru

An Ethernet port received packet can be sent cut-thru to all destination ports when the following is true:

- The CPSW3_CONTROL_REG[19] CUT_THRU_ENABLE bit is set, and
- The receive port is full duplex, and
- The Ethernet receive port speed is non-zero, and
- The packet was received on an express priority, and
- The Ethernet receive port has the remapped received packet priority enabled via the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CUT_THRU_REG[15-8] RX_PRI_CUT_THRU_EN field, and
- All Ethernet destination ports have the remapped packet priority enabled via the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_CUT_THRU_REG[7-0] TX_PRI_CUT_THRU_EN field, and
- All Ethernet destination ports are full duplex, and
- All Ethernet destination ports are express priorities, and
- All Ethernet destination ports have a non-zero CPSW_PN_SPEED_REG_k[3-0] SPEED, and
- All Ethernet destination ports are equal to or slower in speed than the Ethernet receive port, and
- The host port is not in the destination port list unless the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG[19] CUT_MODE_ETH bit is set.

Whether or not a packet actually egresses cut-thru (with or without delay) on an Ethernet destination port depends on traffic congestion on that port. Cut-thru is compatible with Intersperced Express Traffic (IET) and Enhanced Scheduled Traffic (EST).

12.2.1.4.8.3.1 Host Port Cut-Thru Operations

Packets received (CPSW ingress) on the host port are sent store-and-forward to all Ethernet destination ports. Setting the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG[19] CUT_MODE_ETH bit enables Ethernet received packets to be sent cut-thru to a destination port mask that includes the host port. The packet can egress cut-thru on Ethernet ports, but the packet will not actually egress cut-thru on the host port. Setting CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG[19] CUT_MODE_ETH can cause head of line blocking on the host priority if multiple Ethernet ports are sending to the same host port priority. The head of line blocking issue is worsened if there are differing Ethernet port speeds. Clearing the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG[19] CUT_MODE_ETH bit causes packets with the host port in the destination mask to egress store-and-forward on all egress ports.

12.2.1.4.8.3.2 Cut-Thru Error Packets

Any received packets with errors sent cut-thru from an Ethernet receive port to any Ethernet transmit port(s) will egress with at least one byte of the generated outgoing packet CRC inverted to indicate the error. This is due to the fact that cut-thru operations begin before the end of packet when the receive port determines that the packet had an error (long/code/align/CRC).

Any packet also sent cut-thru with the host included in the destination port mask will be dropped to the host (TXST_PKT_DROP) with the packet error indicated on TXST_PKT_ERR[3:0].

Any long/Jabber/Code/Align/CRC errored Ethernet received packet that is not actually sent cut-thru by the receive port will dropped unless CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k[22] RX_CEF_EN is set (which if set will cause the packet to be sent to the host port with the error indicated on TXST_PKT_ERR[3:0]. If a long/Jabber/Code/Align/CRC errored Ethernet received packet is sent cut-thru by the received port then a set CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG_k[22] RX_CEF_EN will not cause the errored packet to be sent to the host port since the packet has already been transferred to the destination ports.

The following cut-thru notes are valid:

- Any Ethernet received packet and that is sent cut-thru to any destination port will not be dropped at any destination port due to priority maximum lengths (CPSW3_CPSW_NU_TX_PRIx_MAXLEN_REG).

Copyright © 2026 Texas Instruments Incorporated

- Any Ethernet received packet that is decoded as a timesync packet will be sent store-and-forward to all destination ports.
- Cut-Thru is full duplex only.
- Cut-Thru is not compatible with any form of flow control.
- Cut-Thru should not be enabled for any receive port that is in ALE bypass mode.

12.2.1.4.8.4 Port Speed

Cut-thru operations require that the Ethernet port speeds be known. Port speed can be set manually by software or be automatically determined by the hardware. A zero port speed causes the port to be disabled for cut-thru operations, but does not disable the port. The port speed values are as follows:

- 0000 – Port speed operations disabled (not a port disable)
- 0001 – 10Mpbs
- 0010 – 100Mbps
- 0011 – 1G
- 0100 – 2.5G
- 0101 – 5G
- 0110 – 10G
- 0111 through 1111 Reserved

When the CPSW_PN_SPEED_REG_k[8] AUTO_SPEED_EN bit is cleared, the port speed is written by software. When the CPSW_PN_SPEED_REG_k[8] AUTO_SPEED_EN bit is set, the port speed is automatically detected by the hardware.

12.2.1.4.8.5 CPPI Checksum Offload

The CPPI host port can be enabled to perform checksum offload on host port packet ingress and egress. UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) over IPV4 and IPV6 are supported. For the purposes of checksum description, the first packet byte (the first byte of the destination address) is byte 1 (not byte 0). That is, a 64 byte packet goes from byte 1 to byte 64. For all packet types, the S_CN_SWITCH bit in the CPSW3_CONTROL_REG register must be set for the Outer VLAN L type to be supported.

12.2.1.4.8.5.1 CPPI Transmit Checksum Offload

IPV4 and IPV6 UDP and TCP packets that are received on any Ethernet port and destined for port 0 egress are checked for correct checksum as described below. The byte counts below are shown for packets with no VLAN's. The byte counts vary with one or two packet VLAN's. Packets received on an Ethernet port with errors are not checked for a correct checksum if they are passed to the host.

12.2.1.4.8.5.1.1 IPV4 UDP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).
- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no UDP header).
- Byte 24 = 0x11 for UDP protocol.
- Received packet UDP checksum of zero means that there is no IPV4 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

12.2.1.4.8.5.1.2 IPV4 TCP

- Byte 15 Upper Nibble = 4 for IPV4
- Byte 15 Lower Nibble = IHL - Nibble with number of 32-bit words in IPV4 header (5 to 15 supported).

Copyright © 2026 Texas Instruments Incorporated

- Bytes 20-21 = fragment[15-0] – Bit 13 is the MF bit and bits [12-0] are the Fragment offset. A packet is a fragment if the MF bit is set or if the fragment offset is non-zero. The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets have MF=0 and a zero offset. A count is output for packet fragments but no errors are reported. First fragments have the UDP header included in the count. Middle and last fragments have only data included in the count (there is no TCP header).
- Byte 24 = 0x06 for TCP protocol.

12.2.1.4.8.5.1.3 IPV6 UDP

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x11 for UDP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x11). Middle and last fragments have a fragment extension header followed by data only (no UDP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.
- Received packet UDP checksum of zero means that there is no IPV6 checksum sent with the packet so no error will be issued.
- Received packet UDP checksum of 0xFFFF means that the checksum was calculated to be 0xFFFF or 0x0000 but was sent in the transmitted packet as 0xFFFF by the sending originating entity.

12.2.1.4.8.5.1.4 IPV6 TCP

- Byte 15 upper nibble = 6 for IPV6.
- Byte 21 = 0x06 for TCP protocol as next header.
- Fragment extension headers are supported. First fragments have a fragment extension header (byte 21 = 0x2C) followed by a UDP header (byte 55 = 0x06). Middle and last fragments have a fragment extension header followed by data only (no TCP header). The first packet fragment has MF=1 with a zero offset. Middle fragments have MF=1 with a nonzero offset. The last packet fragment has MF=0 with a nonzero offset. Non-fragmented packets do not have a fragment extension header. A count is output for packet fragments but no errors are reported.

12.2.1.4.8.6 CPPI Receive Checksum Offload

Packets sent from host port 0 (switch ingress) to any Ethernet port can have a checksum calculated and inserted into the Ethernet egress packet. The RX_CHECKSUM_EN bit in the CPSW3_CPSW_NU_CPSW_NU_CPPI_P0_CONTROL_REG register must be set for receive checksum operation to be enabled. When bit RX_CHECKSUM_EN is enabled, Control Data Word 2 input on CPPI receive PSI interface determines how the checksum is calculated. The CHECKSUM_RESULT field in Control Data Word 2 determines where the checksum is inserted. The checksum result location is adjusted by the egress port if a VLAN is to be inserted or removed on Ethernet port egress.

12.2.1.4.8.7 Egress Packet Operations

Each CPSW egress port (Ethernet and Host) is capable of performing egress packet processing operations (CPSW3_CPSW_NU_ALE_EGRESSOP). IntraVLAN processing either adds, removes, or replaces VLAN information or does nothing. InterVLAN routing allows hardware routing between a limited number of VLANs - thereby allowing high-bandwidth or other routing operations to be offloaded from software to the CPSW (hardware). IntraVLAN processing and InterVLAN routing operations are mutually exclusive. In addition, the packet source and destination addresses can be swapped on egress to facilitate OAM or generic testing operations.

Copyright © 2026 Texas Instruments Incorporated

12.2.1.4.9 MII Management Interface (MDIO)

The MII Management interface module implements the 802.3 serial management interface to interrogate and control external Ethernet PHY using a two-wire bus.

12.2.1.4.9.1 MDIO Frame Formats

Table 12-136 shows the address, Table 12-137 shows the read format and Table 12-138 shows the write format of the supported Clause 45 MII Management interface frames. Post-increment accesses are not supported.

Table 12-136. MDIO Clause 45 Address Frame Format

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	00	AAAAA	RRRRR	10	AAAA.AAAA.AAAA.AAAA

Table 12-137. MDIO Clause 45 Read Frame Format

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	11	AAAAA	RRRRR	Z0	DDDD.DDDD.DDDD.DDDD

Table 12-138. MDIO Clause 45 Write Frame Format

Pre-amble	Start Delimiter	Operation Code	PHY Address	MMD Number	Turnaround	Data
FFFF FFFFh	00	01	AAAAA	RRRRR	10	DDDD.DDDD.DDDD.DDDD

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the MDIO line to a logic 1. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic 1 bits on the MDIO line with 32 corresponding cycles on MDCLK to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on MDIO with 32 corresponding MDCLK cycles before it responds to any other transaction. The MDIO CPSW3_MDIO_USER_ADDR0_REG register must be written before a read or write operation is performed to set the address used in the operation. Each read or write operation has a preceeding address frame.

Preamble

The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a 1. This sequence provides the PHY a pattern to use to establish synchronization. The preamble is required in clause 45 operation.

Start Delimiter

The preamble is followed by the start delimiter which is indicated by a 00 pattern.

Operation Code

The operation code for an address transaction is 00. The operation code for a read is 11, while the operation code for a write is a 01.

PHY Address

The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

MMD Number

The MMD number is the 5 bits allowing 32 unique values. The first bit transmitted is the MSB.

Turnaround

An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall

Copyright © 2026 Texas Instruments Incorporated

drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

Address

The address field is 16 bits on address operations. The first bit transmitted is the MSB of the address word. Each read/write operation initiated has an automatic address operation initiated first that uses the MDIO CPSW3_MDIO_USER_ADDR0_REG/ CPSW3_MDIO_USER_ADDR1_REG register values as the 16-bit address.

Data

The Data field is 16 bits on read and write operations. The first bit transmitted and received is the MSB of the data word.

12.2.1.4.9.2 MDIO Functional Description

The MII Management I/F will remain idle until enabled by setting the ENABLE bit in the CPSW3_MDIO_CONTROL_REG register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the MDIO CPSW3_MDIO_LINK_REG register. Individual PHY's can be enabled or disabled for polling the associated bit in the CPSW3_MDIO_POLL_EN_REG register. The CPSW3_MDIO_LINK_REG and CPSW3_MDIO_ALIVE_REG register bit values are updated on the poll of each PHY. The LINKSEL bit in the CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the MDIO CPSW3_MDIO_LINK_INT_RAW_REG register and the MDIO CPSW3_MDIO_LINK_INT_MASKED_REG register, if enabled by the LINKINT_ENABLE bit in the CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k register.

The MDIO CPSW3_MDIO_ALIVE_REG register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the MDIO CPSW3_MDIO_ALIVE_REG register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the DATA, PHYADR, REGADR, and WRITE fields in a CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register. When the host sets the GO bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the GO bit and set the USERINTRAW field in the CPSW3_MDIO_USER_INT_RAW_REG register corresponding to the CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register being used. The corresponding bit in the CPSW3_MDIO_USER_INT_MASKED_REG register may also be set depending on the mask setting in the MDIO CPSW3_MDIO_USER_INT_MASK_SET_REG and CPSW3_MDIO_USER_INT_MASK_CLEAR_REG registers. A round-robin arbitration scheme is used to schedule transactions that may be queued by the host in different CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k registers. The host should check the status of the GO bit in the MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ACK bit in the MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the auto-negotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG register in the corresponding MAC.

12.2.1.5 CPSW0 Programming Guide

12.2.1.5.1 Initialization and Configuration of CPSW Subsystem

To configure the CPSW Ethernet Subsystem for operation, the host must perform the following:

1. Select the Interface (RMII, SGMII, or RGMII) Mode. See the ENET1_CTRL and ENET2_CTRL[2-0] PORT_MODE_SEL fields.

Copyright © 2026 Texas Instruments Incorporated

2. Configure pads (pin muxing), as per the interface selected. Refer to *Pad Configuration Registers* and the device-specific data sheet.
3. Enable the CPSW Ethernet Subsystem clocks. See *CPSW Integration*
4. Ensure that at least 2000 CPPI_ICLK periods are run after reset is de-asserted.
5. Configure the CPSW3_CONTROL_REG register
6. Configure the Ethernet Port Source Address registers (CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_SA_L_REG_k and CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_SA_H_REG_k)
7. Configure the CPSW statistic port enable register CPSW3_CPSW_NU_STAT_PORT_EN_REG
8. Configure the ALE (Section 12.2.1.4.6.1, *Address Lookup Engine*)
9. Configure the MDIO (Section 12.2.1.5.3.1, *Initializing the MDIO Module*)
10. Configure Ethernet port, as per the desired mode of operations

12.2.1.5.2 CPSW Reset

To reset the Ethernet port, the host must perform the following:

1. Set CMD_IDLE bit to 1h in the Ethernet port control register: CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_CONTROL_REG.
2. Wait for IDLE bit to be set to 1h, which is indicated in the Ethernet port status register: CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_STATUS_REG.
3. Set SOFT_RESET bit to 1h in the Ethernet port software reset register: CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_SOFT_RESET_REG.
4. Wait for SOFT_RESET bit in the CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_SOFT_RESET_REG registers to be cleared to confirm reset completion.
5. Configure the Ethernet ports.
6. Re-configure registers reset to default value by CPSW3_CPSW_NU_CPSW_NU_ETH_MAC_i_PN_MAC_SOFT_RESET_REG.

12.2.1.5.3 MDIO Software Interface

12.2.1.5.3.1 Initializing the MDIO Module

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO Control register (CPSW3_MDIO_CONTROL_REG).
2. Enable the MDIO module by setting the ENABLE bit in CPSW3_MDIO_CONTROL_REG.
3. The MDIO PHY alive status register (MDIO CPSW3_MDIO_ALIVE_REG) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIO CPSW3_MDIO_LINK_REG) can determine whether this PHY already has a link.
4. Set the appropriate PHY addresses in the MDIO user PHY select register (CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k, where k = 0 or 1), and set the LINKINT_ENABLE bit to enable a link change event interrupt if desirable.
5. Set the appropriate LINKSEL bit in the CPSW3_MDIO_USER_GROUP_USER_PHY_SEL_REG_k register (where k = 0 or 1).
6. Set the appropriate USERINTMASKSET bit field in the CPSW3_MDIO_USER_INT_MASK_SET_REG register.
7. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (MDIO CPSW3_MDIO_USER_INT_MASK_SET_REG) to use the MDIO user access register (MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k, where k = 0 or 1).

Copyright © 2026 Texas Instruments Incorporated

12.2.1.5.3.2 Writing Data To a PHY Register

The MDIO module includes a user access register (MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k, where k = 0 or 1) to directly access a specified PHY device.To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k corresponding to the PHY and PHY register SW wants to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW3_MDIO_USER_INT_RAW_REG) corresponding to MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW3_MDIO_USER_INT_MASK_SET_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW3_MDIO_USER_INT_MASKED_REG) and an interrupt is triggered on the host processor.

12.2.1.5.3.3 Reading Data From a PHY Register

The MDIO module includes a user access register (MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k, where k = 0 or 1) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k, where k = 0 or 1) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in the CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register corresponding to the PHY and PHY register SW wants to read.
3. The read data value is available in the DATA bit field in MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k register. After the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (CPSW3_MDIO_USER_INT_RAW_REG) corresponding to MDIO CPSW3_MDIO_USER_GROUP_USER_ACCESS_REG_k used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (CPSW3_MDIO_USER_INT_MASK_SET_REG), then the bit is also set in the MDIO user command complete interrupt register (CPSW3_MDIO_USER_INT_MASKED_REG) and an interrupt is triggered on the host processor.

Copyright © 2026 Texas Instruments Incorporated

12.2.2 Peripheral Component Interconnect Express (PCIe) Subsystem

This chapter describes the features and functions of the device Peripheral Component Interconnect Express (PCIe) subsystem.

12.2.2.1 PCIe Subsystem Overview 1395
12.2.2.2 PCIe Subsystem Environment 1397
12.2.2.3 PCIe Subsystem Functional Description 1399

Copyright © 2026 Texas Instruments Incorporated

12.2.2.1 PCIe Subsystem Overview

The Peripheral Component Interconnect Express (PCIe) subsystem is built around a single-lane dual-mode PCIe controller that provides low pin-count, high reliability, and high-speed data transfers at rates of up to 5.0 Gbps per lane for serial links on backplanes and printed wiring boards.

The device includes one instantiation the of PCIe subsystem - PCIE0. Table 12-139 shows the PCIe subsystem allocation across device domains.

Table 12-139. PCIe Subsystem Allocation Across Device Domains

Module Instance	Domain	
	MCU	MAIN
PCIE0	-	✓

Figure 12-120 provides PCIe subsystem overview.

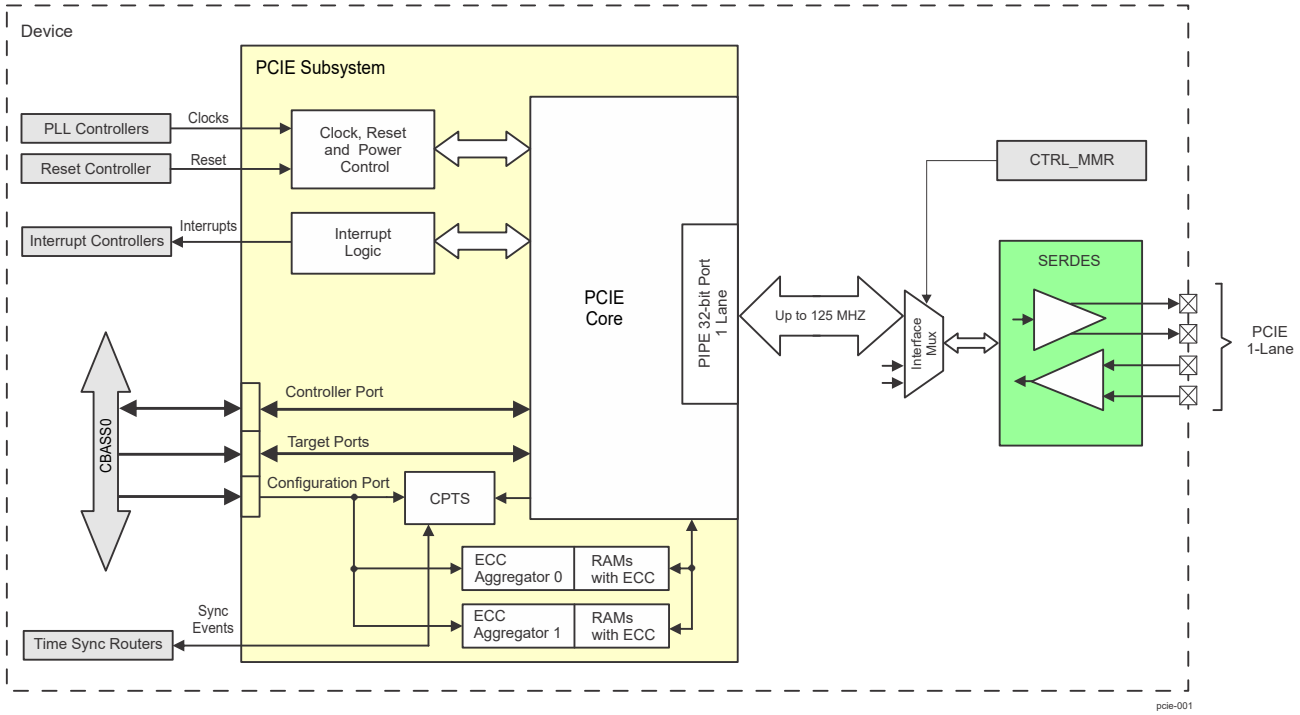


Figure 12-120. PCIe Subsystem Overview

12.2.2.1.1 PCIe Subsystem Features

Each PCIe subsystem supports the following main features:

- Compliance to PCIe® Base Specification, Revision 4.0 (Version 0.7)
- 1-lane configuration with up to 8.0 Gbps/lane (Gen3).
- Gen3 (8 Gbps 128/130-bit encoding), Gen2 (5 Gbps 8/10-bit encoding), and Gen1 (2.5 Gbps 8/10-bit encoding) with auto-negotiation
- 62.5/125 MHz operation on PIPE interface for Gen1/Gen2, respectively
- Constant 32-bit PIPE width for Gen1/2 modes
- Dual mode of operation: Root Complex (RC) or End Point (EP)
- Maximum payload size of 128 bytes
- Maximum remote read request size of 4K bytes
- Maximum Number of non-posted outstanding transactions: 8
- Single Physical Function in End Point (EP) mode
- Four virtual channels (VC)

Copyright © 2026 Texas Instruments Incorporated

- Four traffic classes (TC)
- PCI Power Management states:
  - L1 Power Management sub-state support
  - Device Power Management states D0, D1, D3Hot
- Resizable BAR capability
- Functional Level Reset (FLR) support
- Separate Reference Clock with Independent Spread (SRIS) support
- Legacy, MSI and MSI-X Interrupt Support
- 32 outbound address translation regions
- Precision Time Measurement (PTM) for both RC and EP modes in combination with internal Common Platform Time Sync (CPTS) module
- PCIe compliant PHY (PIPE 4.0) interface for connection to a SERDES-based PHY

12.2.2.1.2 PCIe Subsystem Not Supported Features

The PCIe subsystem does not support the following features:

- PCIe beacon for in-band wake
- Vendor Messaging
- I/O access in inbound direction in RC or EP mode
- Addressing modes other than incremental for burst transactions. As a result, the PCIe addresses cannot be in cacheable memory space.
- Single-root I/O virtualization (SR-IOV)
- Address Translation Services (ATS) capability to support virtualization in RC mode
- L2 power state
- PCI Device Power Management state D3Cold
- Hot-plug
- 48-bit address on VBUSM controller and target interfaces. Only 36-bit address is supported for these interfaces.
- Parity checking on VBUSM and VBUSP interfaces

Copyright © 2026 Texas Instruments Incorporated

12.2.2.2 PCIe Subsystem Environment

This section describes the PCIe subsystem and related application fields from an external system environment point of view.

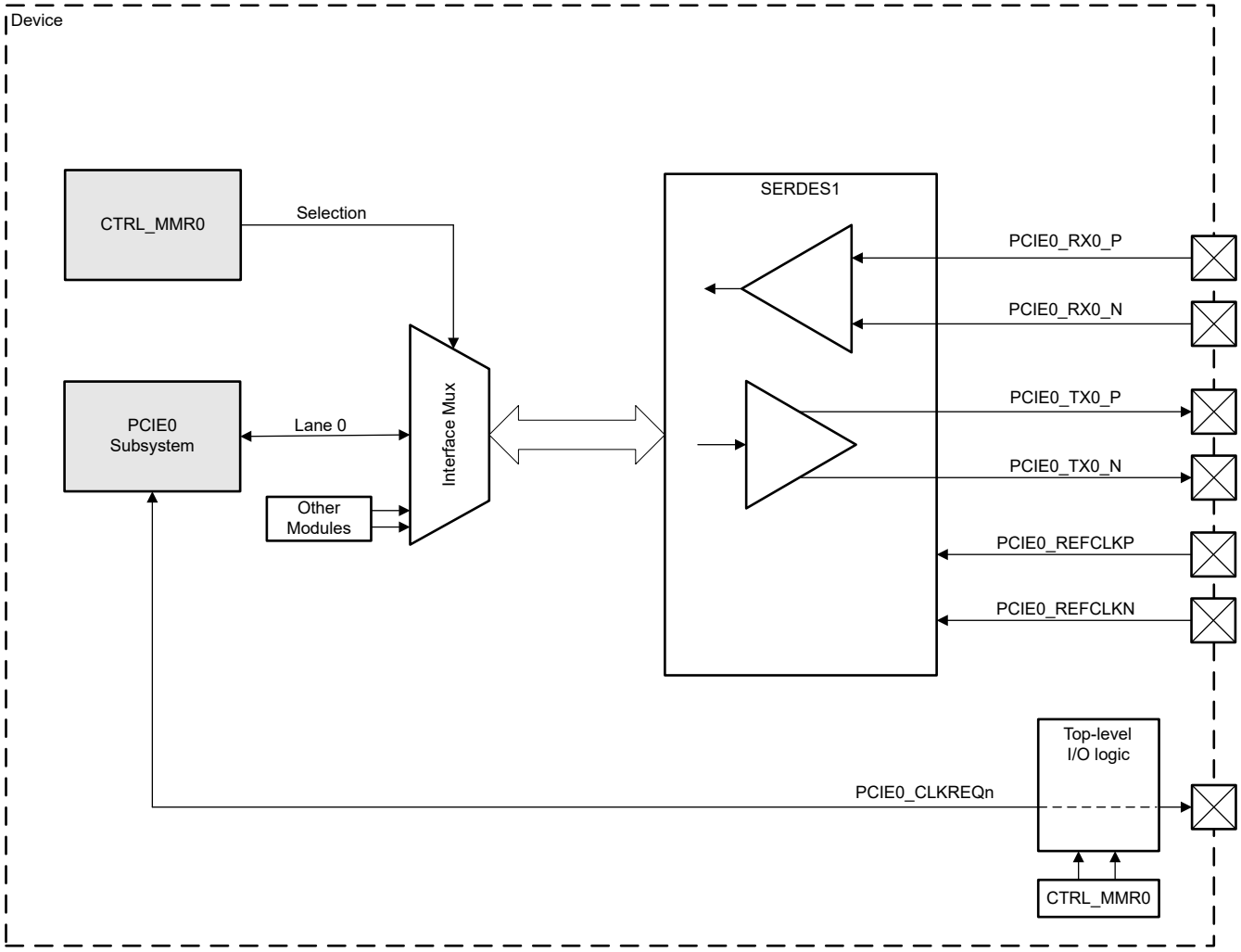


Figure 12-121. PCIe Subsystem Environment

Table 12-140 describes the SERDES signal names at device level related to PCIe subsystem and specifies their functions. For more information on the SERDES operation and interface signals, refer to Section 12.2.3, *Serializer/Deserializer (SerDes)*.

Copyright © 2026 Texas Instruments Incorporated

Table 12-140. PCIe Subsystem I/O Signals

Device Level Signal	I/O [1]	Description
PCIE1 Subsystem		
PCIE0_RX0_N	I	PCIe Lane 0 Receive Differential Data (-)
PCIE0_RX0_P	I	PCIe Lane 0 Receive Differential Data (+)
PCIE0_TX0_N	O	PCIe Lane 0 Transmit Differential Data (-)
PCIE0_TX0_P	O	PCIe Lane 0 Transmit Differential Data (+)
PCIE0_REFCLKP	I	PCIe Differential Reference Clock (+)
PCIE_REFCLKN	I	PCIe Differential Reference Clock (-)
PCIE0_CLKREQn	I/O[2]	PCIe sideband signal for negotiation of L1 Substate entry/exit. The CLKREQn pin operates as an active low open-drain bi directional reference clock request pin. 1 = no request for clock; 0 = request for clock

(1) I = Input; O = Output.
(2) 3.3-V LVCMOS input/open-drain output

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3 PCIe Subsystem Functional Description

12.2.2.3.1 PCIe Subsystem Block Diagram

The block diagram of PCIe subsystem is shown in Figure 12-122. The subsystem id comprised of these major components – the PCIe Core with AXI interfaces, bridges to connect to the system CBASS0 interconnect controller and target interfaces, bridges to connect the system CBASS0 configuration interfaces, additional logic to implement the Precision Time Measurement (PTM), user configuration and interrupt, and RAM to support the controller FIFOs.

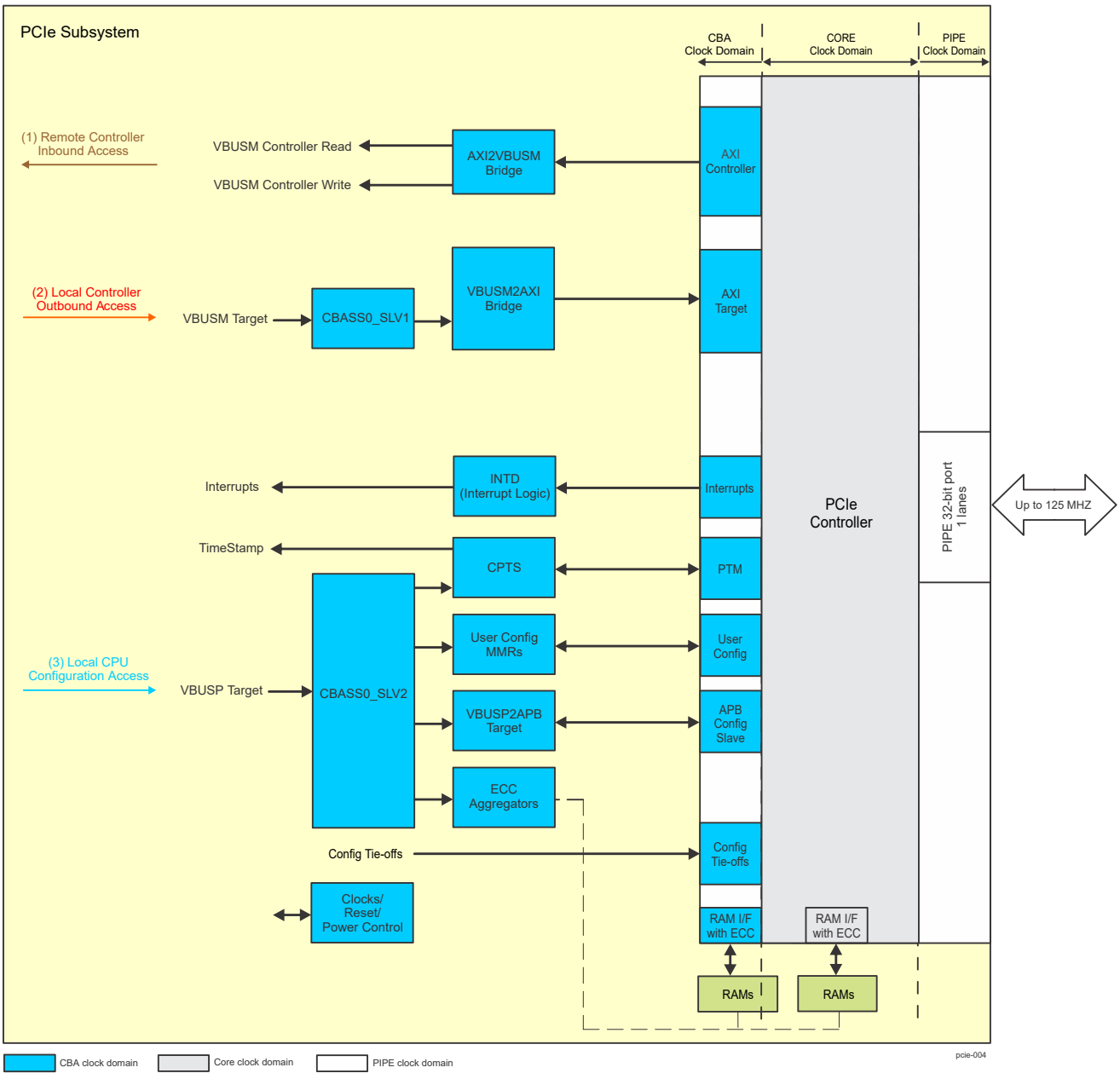


Figure 12-122. PCIe Subsystem Block Diagram

Figure 12-122 also shows some example data flows in the PCIe subsystem, such as:

1. A remote controller issues a read or write access over PCIe to the local device. This will create a command to be issued on the PCIe VBUSM controller read or write interface.

Copyright © 2026 Texas Instruments Incorporated

2. A controller in the local device issues a read or write access over PCIe to the remote device. This will create a command to be issued on the PCIe VBUSM target read or write interface.
3. A controller in the local device wants to access the local PCIe configuration registers, the ECC aggregator registers, or the CPTS registers. This will create a transaction over the PCIe VBUSP target interface.

12.2.2.3.1.1 PCIe PHY Interface

The PCIe subsystem incorporates a 1-lane PCIe compliant PHY (PIPE) interface to connect to a SERDES-based PHY. The PCIe PHY module consist of a SERDES module and a PCIe PCS (Physical Coding Sub-block) module. The SERDES module converts parallel data into PCIe serial signals and the PCIe PCS module provides an industry standard PIPE Interface to PCIe MAC. The frequency of the PIPE interface can be 62.5MHz or 125MHz depending on whether the system is operating in Gen1 or Gen2 mode. The width of the PIPE interface remains constant at 32-bits for all modes of operation. For more information on the SERDES module, see Section 12.2.3, *Serializer/Deserializer (SerDes)*.

12.2.2.3.1.1.1 PCIe Core Module

The PCIe Core module supports dual mode of operation - it can be configured as an End Point (EP) and also as a Root Complex (RC). The operational mode is selected with the CTRLMMR_PCIE0_CTRL[7] MODE_SEL register bit within the device Control Module (CTRL_MMR). It is expected that the MODE_SEL bit is programmed during initial power up based on settings in the SoC boot configuration or a non-volatile storage such as eFuse or Flash memory.

It is not expected that the MODE_SEL setting would have to change during a full power cycle of the device. It is more likely that the operational mode of a SoC will stay as EP or RC for a particular end product's life cycle. It is not expected to switch back and forth during operation without a reset cycle.

The PCIe core module supports four virtual channels (VC) and four transfer classes (TC). The VCs can be used to implement Quality-of-Service (QoS) mechanism by enabling priority or round-robin arbitration. Typically, the highest numbered enabled VC is assigned the highest priority.

There is one iinitiator port and one target port in the PCIe core. All ingress data traffic regardless of the VC assigned will be delivered on the initiator port. Similarly, all egress data that is presented on target port of the PCIe core will be assigned VCs through the outbound address translation registers.

12.2.2.3.1.2 Custom Logic

The PCIe subsystem includes custom logic to implement Precision Time Management (PTM), interrupts and user configuration registers.

12.2.2.3.2 PCIe Subsystem Reset Schemes

This section describes the reset schemes supported by PCIe Base Specification and the way these resets are supported by the PCIe Subsystem. PCIe Base Specification specifies two mechanisms for performing reset – Conventional Reset mechanism and Functional Level Reset mechanism.

12.2.2.3.2.1 PCIe Conventional Reset

There are three distinct types of Conventional Reset and each of these causes the hardware state machines, hardware logic, port states and configuration registers to be initialized to their default values.

1. Cold Reset – The reset occurring at power-up of the device is referred to as Cold Reset. Cold Reset is triggered by the PERSTn signal being asserted. PERSTn signal is an auxiliary signal in PCIe Specification and can be used at power on reset for the device that has PCIe as its primary bus interface to rest of the system. The device is allowed to generate its own power-on reset as long as the PCIe requirements for PERSTn are met. See PCIe Base Specifications for details.
2. Warm Reset – A reset can be triggered by the hardware without the removal of power from the device. This reset is referred to as Warm Reset. The hardware implementation is not specified by the PCIe Specification.
3. Hot Reset – The PCIe Specifications provides an in-band mechanism to propagate a Conventional Reset across a Link. This reset, called the Hot Reset, propagates via the transmission of TS1 Ordered Sets. In

Copyright © 2026 Texas Instruments Incorporated

general, Hot Reset is software controlled procedure and can only be issued by Root Complex in a PCIe network as the propagation is downstream only. PCIe subsystem translates the received in-band hot reset into an interrupt that can then be used by software to reset the PCIe subsystem.

After a conventional reset, the software must wait at least 100 ms before attempting any PCIe transaction on the device that has been reset. If the downstream device does not respond to transaction packets, it must not give up until 1 second plus an additional 50% (0.5 second) time is lapsed.

12.2.2.3.2.2 PCIe Function Level Reset

The Function Level Reset (FLR) is an optional in-band reset mechanism that is used to reset one particular function in a PCIe device. The PCIe core will initate the appropriate function level reset process when it receives the FLR message. The PCIe subsystem will raise an interrupt to the SoC, based on the FLR_IN_PROGRESS[0] status output from the PCIe core. Software can update the PCIE_USER_FLR_DONE[5-0] FLR_DONE register bits to signal to the PCIe core that the application layer FLR processing is complete.

12.2.2.3.2.3 PCIe Reset Isolation

It can be important to isolate reset of PCIe subsystem from reset of the rest of the device. The procedure to implement this depends upon what functionality the PCIe subsystem is providing – Root Complex or End Point.

12.2.2.3.2.3.1 Root Complex Reset with Device Not Reset

When PCIe subsystem is operating as a Root Complex, it is the controller controller on the PCIe subsystem. When PCIe subsystem is to be reset in this operating mode, the link to the downstream device will get disconnected. To accomplish this reset, the PCIe subsystem RC should issue a hot reset to End Point or Switch downstream. It should also stop any ongoing transactions. Then, a PCIe subsystem reset can be issued. The sequence of events is outlined below:

1. Stop transactions at system and application level. This is recommended for graceful suspension of activity at system level. It is not required by PCIe subsystem though. Not choosing to gracefully stop transaction would cause some of the outstanding transactions to complete in error and it may be harmful from software stability standpoint.
2. Disable “Bus Controller Enable” for End Points (see Section 12.2.2.3.6.2, *PCIe Transaction Limitations*). Note that a hand shake with EP Software may be required as not all End Point application software will automatically become aware of this disable action from RC.
3. Optionally, issue Hot Reset to End Point devices via PCIE_USER_RSTCMD[0] INIT_HOT_RESET bit . Note that the link will be getting disconnected. So, a reset may be happening regardless of the Hot Reset command from RC. It depends on the architecture of the other (external) device. In devices with PCIe subsystem, link disconnection automatically results in a reset request interrupt. A PCIe subsystem reset is required to get the memory buffers out of internal flush modes.
4. Initiate Clock Stop sequence and wait for Acknowledgement. For more information, see Section 12.2.2.3.3.1, *CBA Power Management*.
5. Issue reset to PCIe subsystem (local reset).
6. Reinitialize PCIe subsystem and downstream devices. PCIe bus enumeration must be performed again.

Note that it is possible for Root Complex to occasionally see the downstream device going down for some reason and disconnecting. Such events typically occur due to an internal error condition in the End Point. When such an event occurs, the sequence of events for the Root Complex will be as follows:

1. The PCIe subsystem in RC mode will issue the “Reset Request” interrupt when it detects link getting disconnected unexpectedly.
2. The PCIe subsystem will flush all controller transactions and any completion data from pending transactions. It will also start completing target transactions with error completions.
3. After servicing the interrupt and disabling further interrupts, the system software must reset PCIe subsystem Root Complex (see steps #4 and later specified previously in this section).

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3.2.3.2 Device Reset with Root Complex Not Reset

As a Root Complex, PCIe core is the controller of PCIe subsystem. If the device hardware and the software is reset and re-initialized, the system does not get additional benefit of keeping the PCIe subsystem alive as all context information is lost. So, this mode is not supported.

12.2.2.3.2.3.3 End Point Device Reset with Root Complex Not Reset

It may be desirable to isolate the reset only to PCIe subsystem when it is operating as an End Point. Such resets may be implemented when a Hot Reset command is received from upstream device (Root Complex or Switch). The sequence to be followed is as outlined below.

1. Hot Reset is received. The “Request Reset” interrupt is triggered.
2. The PCIe subsystem will automatically enter Flush Mode. The PCIe Link Training and Status State Machine (LTSSM) will be disabled automatically.
3. PCIe subsystem will complete all controller transactions and any completion data from pending transactions will be accepted and discarded. It will also start issuing error response for new target transactions.
4. Upon receipt of the reset interrupt from PCIe subsystem, the system software must immediately start preparing for shut down of PCIe subsystem. The DMA or software process accessing PCIe subsystem should gracefully suspend operations. Otherwise, step #3 may continue for unduly long time and Root Complex may assume that the EP is non-responsive to its link re-training attempts.
5. Read the Reset Command register (PCIE_USER_RSTCMD) to check if the bridge activity flush bit is zero. This indicates that it is safe to issue warm reset to PCIe subsystem.
6. Initiate Clock Stop sequence. This will ensure no outstanding transactions exist.
7. Issue a PCIe subsystem Reset.
8. Resume initialization sequence.

12.2.2.3.2.3.4 Device Reset with End Point Device Not Reset

As an End Point, the PCIe subsystem does not need to stay operational when the device reset is happening. Upon detecting the link disconnection, the upstream port will re-train the link when Root Port issues a link retrain command to itself or to a switch port next to the EP.

If it is required that the PCIe subsystem be operational (without any transactions though), the End Point must negotiate with the other devices (primarily Root Port) to stop activity. Otherwise, if the End Point goes into force idle/standby or clock stop modes without properly managing the process, there will be timeouts or errors on incoming read transactions and writes will be completely lost. In such situation, the Root Port will encounter excessive errors and may issue a hot reset to the End Point anyway.

12.2.2.3.2.4 PCIe Reset Limitations

Once a reset situation occurs, the PCIe subsystem prepares for reset assertion from the device level reset controller. In this mode, all outbound write transactions are discarded and all outbound reads are returned with error. Similarly, all inbound reads/writes are discarded and any pending reads are completed but data is discarded. Additionally, any register reads on local or remote registers regions are returned in error even through data may be correct. Register writes on local registers are executed correctly.

Whenever the reset interrupt is asserted by PCIe subsystem, the host controller should clear the interrupt, perform a clock stop request/ack sequence and issue a local reset to PCIe subsystem.

12.2.2.3.2.5 PCIe Reset Requirements

Whenever link goes from connected to disconnected state, the PCIe subsystem requires re-initialization based on the events occurring in PCIe core. A request for reset is typically going to be issued and will manifest as a reset request interrupt. Since loopback requires transitions through link connect and disconnect, it will generate reset request interrupts. Typically, issuing the warm reset (module reset for CBA) would be sufficient.

12.2.2.3.3 PCIe Subsystem Power Management

PCIe has multiple power management protocols. Some of them are invoked by the hardware, such as Active State Power Management (ASPM), while others are activated at higher levels via software.

Copyright © 2026 Texas Instruments Incorporated

The PCIe core supports D0, D1 and D3-Hot power states.

Link power states L0, L0s, L1, L1s are supported.

L0s entry and exit is managed by the PCIe core if ASPM L0s is enabled.

L1 entry and exit is also managed by the PCIe core if ASPM L1 is enabled. Software can force the exit from L1 by writing to the PCIE_USER_PMCMD[0] CLIENT_REQ_EXIT_L1 bit in the PCIe subsystem. Entry into L1 can also be blocked by setting the PCIE_USER_PMCMD[0] CLIENT_REQ_EXIT_L1 bit.

L1s support requires that the CLKREQ pin be connected to the remote peer. L1s power state is entered when the link is in L1 and CLKREQ pin is de-asserted. Software can force the exit from L1s by writing to either the PCIE_USER_PMCMD[0] CLIENT_REQ_EXIT_L1 or PCIE_USER_PMCMD[1] CLIENT_REQ_EXIT_L1_SUBSTATE bits in the PCIe subsystem.

12.2.2.3.3.1 CBA Power Management

The power management for PCIe subsystem with CBA interface is primarily accomplished through the clock stop protocol. Any time the clock stop protocol is initiated, the PCIe subsystem will acknowledge after making sure that there are no outstanding transactions. If there are any pending transactions in the subsystem, then the clock stop acknowledgement procedure cannot be completed. Therefore, it is necessary that the system software first suspend activity on the serial link as well as on the CBA interface. To do so, it is expected that the devices communicating with the device on which clock stop procedure is to be performed agree to stop transactions targeted to the device in question. Similarly, it is required that the outgoing transactions also be stopped to successfully complete the clock stop sequence. The PCIe subsystem will guarantee that in the event there are pending transactions that are still in process of draining will prevent the clock-stop acknowledgement to be issued to CBA subsystem power management logic.

Note

The PCIe subsystem does not have ability to terminate the clock stop state. Therefore, any wakeup sequence can only be initiated through other means such as software driven timers or software detected events occurring outside of PCIe subsystem.

12.2.2.3.4 PCIe Subsystem Interrupts

The PCIe subsystem provides a mix of interrupts from the PCIe controller interrupt, CPTS interrupts, and interrupts generated in the subsystem using the internal Interrupt Distributor (CP_INTD) module.

The CP_INTD module is used to generate and aggregate interrupts from various status signals of the PCIe core. Interrupts from the PCIe core and CPTS module are ported out directly to the PCIe subsystem boundary without any aggregation. Table 12-141 shows the details on the PCIe subsystem interrupt outputs.

Table 12-141. PCIe Subsystem Interrupt Events

Interrupt Name	Clock Dependency	Description
PCIE_LEGACY_PULSE	CBA_CLK	PCIe legacy interrupt. INTA_OUT, INTB_OUT, INTC_OUT and INTD_OUT status outputs from the PCIe core are aggregated. Note: Valid in RC mode only.
PCIE_ERROR_PULSE	CBA_CLK	PCIe error interrupt. FATAL_ERROR_OUT, NON_FATAL_ERROR_OUT and CORRECTABLE_ERROR_OUT status outputs from the PCIe core are aggregated. Note: Valid in both RC and EP modes.
PCIE_FLR_PULSE	CBA_CLK	PCIe Function Level interrupt. FLR_IN_PROGRESS[0] status outputs from the PCIe core is used to generate this interrupt. Note: Valid in EP mode only.
PCIE_DOWNSTREAM_PULSE	CBA_CLK	PCIe downstream interrupt. • F0_VSEC_INTERRUPT_OUT, status output from the PCIe controller is used to generate this interrupt. Note: Valid in EP mode only

Copyright © 2026 Texas Instruments Incorporated

Table 12-141. PCIe Subsystem Interrupt Events (continued)

Interrupt Name	Clock Dependency	Description
PCIE_HOT_RESET_PULSE	CBA_CLK	PCIe hot reset interrupt. HOT_RESET_OUT status output from the PCIe controller is aggregated. Note: Valid in EP mode only.
PCIE_LINK_STATE_PULSE	CBA_CLK	PCIe link state interrupt. LINK_DOWN_RESET_OUT status from the PCIe core is is used to generate this interrupt. Note: Valid in both RC and EP modes.
PCIE_PWR_STATE_PULSE	CBA_CLK	PCIe power state interrupt. POWER_STATE_CHANGE and DPA_INTERRUPT[0] status output from the PCIe core are used to generate this interrupt Note: Valid in both EP and RC modes.
PCIE_DPA_PULSE	CBA_CLK	PCIe dynamic power allocation interrupt. DPA_INTERRUPT[0] status output from the PCIe core is used to generate this interrupt. Note: Valid in both EP and RC modes.
PCIE_LOCAL_LEVEL	CBA_CLK	PCIe local interrupt. The LOCAL_INTERRUPT from the PCIe controller is brought out directly without any aggregation. Note: Valid in both RC and EP modes.
PCIE_PHY_LEVEL	CBA_CLK	PCIe PHY interrupt. The PHY_INTERRUPT_OUT from the PCIe controller is brought out directly without any aggregation. Note: Valid in both RC and EP modes.
PCIE_PTM_VALID_PULSE	CBA_CLK	PCIe PTM valid interrupt. PTM_LOCAL_TIMER_OUT_VALID status output from the PCIe controller is used to generate this interrupt. Note: Valid only in EP mode.
PCIE_ECC0_UNCORR_PULSE	CBA_CLK	PCIe ECC Aggregator0 uncorrected pulse interrupt.
PCIE_ECC0_UNCORR_LEVEL	CBA_CLK	PCIe ECC Aggregator0 uncorrected level interrupt.
PCIE_ECC0_CORR_PULSE	CBA_CLK	PCIe ECC Aggregator0 corrected pulse interrupt.
PCIE_ECC0_CORR_LEVEL	CBA_CLK	PCIe ECC Aggregator0 corrected level interrupt.
PCIE_ECC1_UNCORR_PULSE	LANE0_TXMCLK	PCIe ECC Aggregator1 uncorrected pulse interrupt.
PCIE_ECC1_UNCORR_LEVEL	LANE0_TXMCLK	PCIe ECC Aggregator1 uncorrected level interrupt.
PCIE_CPTS_PEND_INTR	CBA_CLK	PCIe CPTS level interrupt.

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3.4.1 Interrupts Aggregation

The PCIe subsystem includes the CP_INTD module to aggregate some of the PCIe controller signals into the subsystem interrupts indicated in Table 12-141. The interrupt aggregator takes both level and pulse signals from the PCIe core and produces the aggregated pulse interrupt outputs.

The aggregator also supports the End of Interrupt (EOI) feature. EOI can be used to re-trigger a pulse interrupt output, if a PCIe controller level signal is still asserted at the end of interrupt processing. The re-triggering of the the pulse interrupt can be achieved by writing the specified EOI vector value to the PCIE_USER_EOI_VECTOR register.

Table 12-142. PCIe Controller Interrupts Aggregation - 1

Aggregated Interrupt	CP_INTD Registers and Bits Mapping		EOI_VECTOR	PCIe Controller Signal	Description
	PCIE_INTD_ENABLE_REG_SYS_0 PCIE_INTD_ENABLE_CLR_REG_SYS_0 PCIE_INTD_STATUS_REG_SYS_0 PCIE_INTD_STATUS_CLR_REG_SYS_0(1)				
PCIE_DOWNSTREAM_PULSE	0	pcie_downstream_0	0	down_pf0_intr_out	Downstream PF0 interrupt (EP mode only)
-	31-1	Reserved	-	-	-

(1) Register is not used since all PCIe controller signals being aggregated are of type 'level'.

Table 12-143. PCIe Controller Interrupts Aggregation - 2

Aggregated Interrupt	CP_INTD Registers and Bits Mapping		EOI_VECTOR	PCIe Controller Signal	Description
	PCIE_INTD_ENABLE_REG_SYS_1 PCIE_INTD_ENABLE_CLR_REG_SYS_1 PCIE_INTD_STATUS_REG_SYS_1 PCIE_INTD_STATUS_CLR_REG_SYS_1				
PCIE_FLR_PULSE	0	pcie_flr_0	1	flr_in_progress	PF0 function-level reset (EP mode only)
-	21-1	Reserved	-	-	-
PCIE_LEGACY_PULSE	22	pcie_legacy_0	2	inta_out	Legacy interrupt A (RC mode only)
	23	pcie_legacy_1		intb_out	Legacy interrupt B (RC mode only)
	24	pcie_legacy_2		intc_out	Legacy interrupt C (RC mode only)
	25	pcie_legacy_3		intd_out	Legacy interrupt D (RC mode only)
PCIE_PWR_STATE_PULSE	26	pcie_pwr_state	3	power_state_change_intr	Power state change to D1 or D3
-	31-27	Reserved	-	-	-

Copyright © 2026 Texas Instruments Incorporated

Table 12-144. PCIe Controller Interrupts Aggregation - 3

Aggregated Interrupt	CP_INTD Registers and Bits Mapping		EOI_VECTOR	PCIe Controller Signal	Description
	PCIE_INTD_ENABLE_REG_SYS_2 PCIE_INTD_ENABLE_CLR_REG_SYS_2 PCIE_INTD_STATUS_REG_SYS_2 PCIE_INTD_STATUS_CLR_REG_SYS_2				
PCIE_DPA_PULSE	0	pcie_dpa	N/A(1)	dpa_intr	PF0 DPA power state change (EP mode only)
-	5-1	Reserved	-	-	-
PCIE_ERROR_PULSE	6	pcie_error_0	N/A(1)	correctable_error	Correctable error
	7	pcie_error_1		non_fatal_error	Non-Fatal error
	8	pcie_error_2		fatal_error	Fatal error
PCIE_HOT_RESET_PULSE	9	pcie_hot_reset	N/A(1)	c_hot_reset_out_sync	Hot reset (EP mode only)
PCIE_LINK_STATE_PULSE	10	pcie_link_state	N/A(1)	link_down_reset_out	Link down reset
PCIE_PTM_VALID_PULSE	11	pcie_ptm	N/A(1)	ptm_local_timer_out_valid_sync	PTM local timer valid (EP mode only)
-	31-12	Reserved	-	-	-

(1) EOI is not used since the input signal is of type 'pulse'.

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3.4.2 Interrupt Generation in EP Mode

When PCIe subsystem is operating as an End Point (EP), either the legacy interrupts, MSI or MSI-X interrupts can be triggered to the upstream ports (eventually leading to an interrupt in RC device). As per PCIe Specifications, each PCIe function may generate only one of the Legacy or MSI interrupt types as decided during configuration period.

12.2.2.3.4.2.1 Legacy Interrupt Generation in EP Mode

The End Point (EP) can trigger generation of a PCI Legacy Interrupt at the Root Complex via an in-band Assert_INTx / Deassert_INTx messages (where x = A, B, C, or D). Software can write to the PCIE_USER_LEGACY_INTR_SET register to trigger the required INTx (where x = A, B, C, or D) assert and de-assert message from the PCIe core. Once an assert message has been generated, it cannot be generated again until a deassert message is generated. Thus, only one interrupt can be pending at a time.

There is no hardware input port provided that will allow generation of legacy interrupts on the EP port.

Note

The interrupt messaging mechanism makes it unfeasible to guarantee a time of delivery of the interrupt unlike in conventional designs where the interrupt line is often electrically connected to the final destination.

12.2.2.3.4.2.2 MSI and MSI-X Interrupt Generation

In the case of MSI interrupts signaling, the interrupt conditions are communicated from the EP to the RC via messages. Thus, upon the occurrence of an interrupt condition, a message is sent by the EP with information that identifies the origin of the interrupt. Each message is in the form of a memory write request, containing an address and a data value to be written. Each PCI function supported by a device can be assigned a separate memory address, thus providing separate virtual channels for signaling interrupts generated by each function. In addition, the MSI mechanism allows a maximum of 32 distinct data patterns in the messages generated by each PCI function, and each pattern can be assigned to an interrupt condition within the function.

In the case of MSI-X interrupts signaling, the operation is similar to the MSI mode, except that the mechanism allows a much larger number of distinct interrupt conditions as many as 2048 per function to be communicated, and enables a distinct address to be defined for each of these conditions. This mechanism requires two tables to be stored in the EP memory. The MSI-X table contains the address/data patterns to be used for each interrupt condition (as many as 2048 per function) as well as individual enable/mask bits, and the Pending Bit Array (PBA) stores the status of each interrupt condition. Interrupt conditions are communicated from the EP to the RC via messages (write requests), as in the case of the MSI mode.

12.2.2.3.4.3 Interrupt Reception in EP Mode

The PCIe specification does not have provision for End Points to receive legacy interrupts. As a result, only events other than these are used to map to interrupts.

12.2.2.3.4.3.1 PCIe Core Downstream Interrupts

The Vendor Specific Capability signal of the PCIe core, F0_VSEC_INTERRUPT_OUT, is used to generate interrupts to the EP from the RP. The F0_VSEC_INTERRUPT_OUT represents the interrupt for the EP Physical Function 0. This signal is used to generate the PCIE_DOWNSTREAM_PULSE interrupt to the local host.

The RP can write to the Vendor Specific Control registers (PCIE_CORE_PFn_I_VENDOR_SPECIFIC_CONTROL_REG) to assert this signal at the EP and this will trigger the PCIE_DOWNSTREAM_PULSE interrupt to the EP host.

12.2.2.3.4.3.2 PCIe Core Function Level Reset Interrupts

The PCIE_FLR_PULSE interrupt is asserted to indicate to the host that the PCIe controller has received a function-level reset request from the remote side. The PCIE_FLR_PULSE interrupt is an aggregation of the

Copyright © 2026 Texas Instruments Incorporated

FLR_IN_PROGRESS[0] signal from the PCIe core. The FLR_IN_PROGRESS[0] represents Physical Function 0.

Upon assertion of the function level reset interrupt, software will need to write to the PCIE_USER_FLR_DONE[5-0] FLR_DONE bit field within 100ms to acknowledge to the PCIe core that all the application level processing related to the function level reset is complete.

12.2.2.3.4.3.3 PCIe Core Power Management Event Interrupts

The PCIE_PWR_STATE_PULSE interrupt is generated to let the software know of the power management events. The PCIE_PWR_STATE_PULSE interrupt is generated by the POWER_STATE_CHANGE_INTERRUPT output of the PCIe core. This interrupt is asserted when the power state of a physical is being changed to D1 or D3 state by writing into their Power Management Control register (PCIE_CORE_PFn_I_PWR_MGMT_CTRL_STAT_REP).

Software can check the PCIE_USER_LINKSTATUS[23-16] POWER_STATE_CHANGE_FUNCTION_NUM register field to determine the physical function for which power state change occurred. The PCIE_USER_PMCMD[2] POWER_STATE_CHANGE_ACK register bit can be used to acknowledge the POWER_STATE_CHANGE_INTERRUPT.

The PCIE_DPA_PULSE interrupt is generated by aggregating the PCIe controller DPA_INTR0 interrupt status output. This interrupt is asserted in EP mode when there is a configuration write to the dynamic power allocation control register (PCIE_CORE_PFn_I_DPA_CTRL_STATUS_REG) to modify the DPA power state of the device. The DPA_INTR0 is asserted for such an event for PF0.

12.2.2.3.4.3.4 PCIe Core Hot Reset Request Interrupt

When the link is down, the Root Complex may request reset of the End Point. This request is terminated as a PCIE_HOT_RESET_PULSE interrupt to the End Point host software. All outstanding transactions are completed in error on target port and further transactions are not generated on the controller port. Once the transactions are completely stopped, the software should issue a local reset to PCIe subsystem. The re-initialization process may then be started.

The PCIE_HOT_RESET_PULSE interrupt is generated from the HOT_RESET_OUT output of the PCIe core.

12.2.2.3.4.3.5 PTM Valid Interrupt

In EP mode, the PCIe subsystem will generate the PCIE_PTM_VALID_PULSE interrupt to the local CPU after a PTM dialog between the PTM requester (EP) and the PTM responser (RP). The PTM valid interrupt indicates to the CPU in the EP that the local timers have been updated with the timestamp data obtained from the RP. The CPU can read the timer registers inside the EP to determine the current timebase.

The PCIE_PTM_VALID_PULSE interrupt is generated from the PTM_LOCAL_TIMER_OUT_VALID signal from the PCIe core.

12.2.2.3.4.4 Interrupt Generation in RC Mode

As per PCIe Base Specifications, Root Complex ports only receive interrupts. There is no mechanism to generate interrupts from RC port to EP mode as per PCIe specification.

However, PCIe subsystem does support generation of interrupts from RC to EP. The downstream interrupts described in Section 12.2.2.3.4.3.1, *PCIe Core Downstream Interrupts* provides a mechanism for the RC to generate software triggered interrupts to the EP.

12.2.2.3.4.5 Interrupt Reception in RC Mode

12.2.2.3.4.5.1 PCIe Legacy Interrupt Reception in RC Mode

The RC can receive any of the four legacy INTx interrupts from the EP. These will trigger the PCIE_LEGACY_PULSE interrupt to the host CPU.

The PCIE_LEGACY_PULSE interrupt is an aggregation INTA_OUT, INTB_OUT, INTC_OUT and INTD_OUT signals from the PCIe core.

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3.4.5.2 MSI/MSI-X Interrupt Reception in RC Mode

The PCIe core decodes all MSI and MSI-X messages received from the link and forwards them as a 32-bit write transaction on the AXI controller interface.

The destination address for MSI/MSI-X write transactions should be the ARM GIC. The GIC will process the write transaction and create an interrupt to the ARM Ax CPU core in the device.

12.2.2.3.4.5.3 Advanced Error Reporting Interrupt

If enabled by software at the time of enumeration, PCIe subsystem will generate this interrupt to indicate occurrence of errors of various levels of severity. The software can choose not to enable Advanced Error Reporting but if it enables, it must process the interrupt as per PCIe Specifications.

The advanced error reporting interrupt is signaled by the PCIE_ERROR_PULSE interrupt. The PCIE_ERROR_PULSE is an aggregation of the FATAL_ERROR_OUT, NON_FATAL_ERROR_OUT and CORRECTABLE_ERROR_OUT from the PCIe core.

12.2.2.3.4.6 Common Interrupt Reception in RC and EP Modes

These interrupts are common to both RC and EP modes of operation.

12.2.2.3.4.6.1 PCIe Local Interrupt

The LOCAL_INTERRUPT from the PCIe controller is ported out directly to the PCIe subsystem boundary. For more details, see the PCIe specification

12.2.2.3.4.6.2 PHY Interrupt

The PHY_INTERRUPT_OUT from the PCIe controller is ported out directly to the PCIe subsystem boundary. For more details, see the PCIe specification

12.2.2.3.4.6.3 Link down Interrupt

If the PHY link is disconnected, the PCIE_LINK_STATE_PULSE interrupt will be generated. The expected course of action is to reset the entire PCIe subsystem and restart. All application states must also be initialized so that the operations can resume following the reset and renegotiation of the link.

The PCIE_LINK_STATE_PULSE interrupt is generated from the LINK_DOWN_RESET_OUT output of the PCIe core.

12.2.2.3.4.6.4 Transaction Error Interrupts

If there is a timeout on PCIe or an abort, PCIE_PHY_LOCAL_LEVEL interrupt will be issued. The PCIE_PHY_LOCAL_LEVEL interrupt is generated by the aggregation of the LOCAL_INTERRUPT and PHY_INTERRUPT_OUT signals from the PCIe core. For more details, see the PCIe specification.

12.2.2.3.4.6.5 Power Management Event Interrupt

This PCIE_PWR_STATE_PULSE interrupt is generated to let the software know of the power management events.The PCIE_PWR_STATE_PULSE interrupt is generated by aggregating the POWER_STATE_CHANGE_INTERRUPT and DPA_INTERRUPT outputs of the PCIe core.

In EP mode, software can check the PCIE_USER_LINKSTATUS[23-16] POWER_STATE_CHANGE_FUNCTION_NUM bit field to determine the physical function for which power state change occurred.

The PCIE_USER_PMCMD[2] POWER_STATE_CHANGE_ACK bit can be used to acknowledge the POWER_STATE_CHANGE_INTERRUPT.

12.2.2.3.4.7 ECC Aggregator Interrupts

The PCIE_ECC0_UNCORR_PULSE/PCIE_ECC0_UNCORR_LEVEL and PCIE_ECC0_CORR_PULSE/ PCIE_ECC0_CORR_LEVEL interrupts are asserted by the CBA clock domain ECC Aggregator. The PCIE_ECC1_UNCORR_PULSE/PCIE_ECC1_UNCORR_LEVEL interrupts are asserted by the Core clock domain ECC Aggregator. The Core clock domain ECC Aggregator correctable interrupts are not exported since

Copyright © 2026 Texas Instruments Incorporated

this aggregator is only connected to the error injection logic and the correctable interrupts for this module will never fire.

12.2.2.3.4.8 CPTS Interrupt

The PCIE CPTS_PEND_INT interrupt is asserted by the CPTS module in the PCIe subsystem to signal a TimeStamp event.

12.2.2.3.5 PCIe Subsystem DMA Support

The PCIe Subsystem has different requirements based upon whether it is operated in Root Complex (RC) or End Point (EP) mode. The PCIe subsystem does not have DMA capabilities built into it. It has internal target and controller ports connected to the device-level interconnect.

An external DMA engine can make burst data read/writes on the target port and the controller port on PCIe subsystem can initiate reads/writes to memory on behalf of a remote PCIe device.

The PCIe subsystem does not specify the DMA protocol that is used for data transfers. The software implementations on the two ends of the PCIe link implement a data transfer protocol that is compatible with each other.

As a result, there will be several software drivers required – one driver that will manage the PCIe subsystem in RC mode and another set of drivers that will run on the RC side and will manage each of the End Points connected to the PCIe subsystem RC.

Similarly, when PCIe subsystem is operating as an End Point, there are two drivers – one to manage the PCIe subsystem from the device side and another driver that will run on the device operating as the Root Complex.

12.2.2.3.5.1 PCIe DMA Support in RC Mode

When operating in Root Complex mode, a DMA controller internal to the device (outside PCIe subsystem) can perform DMA to and from any remote device located on the PCIe subsystem. The memory address of such devices is available to the software via the PCIe bus enumeration procedure. In addition, the PCIe subsystem has a provision to perform memory address translation on outbound requests. Thus, the software is able to map different memory regions in its memory map to correspond to different addresses (and different access types) on the PCIe side.

12.2.2.3.5.2 PCIe DMA Support in EP Mode

When operating as a PCIe End Point, the device will be located in PCIe memory map at location programmed in the Base Address Registers by the PCIe Root device. Any PCIe transactions destined for the device from the upstream ports will get transferred to the controller port on the PCIe subsystem. Similarly, any transactions originating from the software will be sent over to PCIe link.

In End Point mode, the PCIe subsystem provides address translation functionality. It is possible to map memory accesses originating on PCIe side to memory accesses with different address on the CBA bus side. These address ranges are configurable through application registers.

12.2.2.3.6 PCIe Subsystem Transactions

12.2.2.3.6.1 PCIe Supported Transactions

PCIe subsystem supports the following transactions:

1. Memory Read/Write in inbound/outbound direction in Root Complex (RC) and End Point (EP) modes.
2. I/O Read/Write in outbound direction in RC mode.
3. Configuration Read/Write in outbound direction in RC mode.
4. Configuration Read/Write in inbound direction in EP mode.
5. Message Transactions for interrupts and power management in RC and EP modes.

The following transactions are not supported:

1. Locked Read transactions and associated messages.
2. Inbound I/O Transaction Layer Packets (TLPs).

Copyright © 2026 Texas Instruments Incorporated

3. User defined messages.

12.2.2.3.6.2 PCIe Transaction Limitations

There are some limitations that must be adhered to while issuing transactions to PCIe subsystem internal bus interface.

- System Initialization

The PCIe subsystem operation is completely dependent upon the availability of the clock from the PLL that is inside the SERDES. All registers in the PCIe controller located in the core clock domain need the PLL to be in lock and properly configured. Register access transactions that are initiated before ensuring that the PLL is locked will result in a status error.

- Remote Configuration and I/O Requests

Since the remote configuration and I/O transaction windows are directly mapped to internal bus space, the software must care to not access these spaces when there is no operational PCIe link. No response may be generated for such transactions. It is recommended that checks be built into software to avoid remote accesses in the absence of an operational link.

- Byte Strobe Limitations

For any type of write transactions, the byte enables can only have a single unbroken string of 1s. In other words, in a transaction, if a byte's write strobe is set, then all following bytes must have write strobe set until the last byte with write enabled. "Holes" or "Zeros" in between the byte enables are not allowed.

Since the internal bus width is greater that 32-bit, the TLP (Transaction Layer Packets) size will not be 1 (PCIe counts in 32-bit units) and therefore, it is through the FBE/LBE (First/Last Byte Enable) that the actual data transfer size is controlled.

- Burst Type Limitations

The PCIe core and respectively the PCIe subsystem does not support 'fixed' or 'wrap' burst types on its Target or Controller port. Transactions that require any burst type other than incremental burst type will result in unspecified behavior, possibly bus lock up.

- Transaction Address Alignment

The PCIe subsystem imposes a limitation of a maximum of 128-byte outbound read/write command. However, if the starting address is not aligned to an 8-byte boundary, then the maximum transaction size is reduced to 120 bytes. This limitation is placed to avoid arithmetic overflow in computing transaction length from CBA to AXI. Unspecified behavior will occur, if misaligned transactions in outbound direction are not limited to a maximum of 120 bytes.

- Read Interleaving

Read interleaving refers to the process of returning split read responses from multiple transactions. This implies that read data is not guaranteed to be sent in sequential order (data for one transaction to be sent completely before the next). The PCIe core will not interleave read responses, if the outbound read command/transaction size does not exceed the maximum transaction size configured in the PCIe core.

12.2.2.3.7 PCIe Subsystem Address Translation

The PCIe subsystem uses the address translation registers in the PCIe core to translate outbound system addresses to PCIe space and inbound PCIe address to a valid system address.

12.2.2.3.7.1 PCIe Inbound Address Translation

12.2.2.3.7.1.1 Root Complex Inbound PCIe to AXI Address Translation

The Root Complex inbound PCIe to AXI address translation is performed on memory and IO TLPs. The selection of which address translation registers to use in the translation process is dependent on the BAR match of the incoming TLP. In Root Complex mode there are 2 bars, so BAR0 and BAR1 registers are implemented. There is a BAR7 register which is used as a no match BAR address translation register. In Root Complex mode there are 2 bars but a BAR value of 7 will be indicated by HAL2AXI when BAR matching is disabled.

Copyright © 2026 Texas Instruments Incorporated

Any address that does not match the Root Complex BARs will be sent out as a BAR7 TLP. Each BAR register is implemented as two 32-bit registers which are named addr0 and addr1. The 'Root Complex Inbound PCIe to AXI Address Translation Logic' takes the upper bits from the 'Root Complex Inbound PCIe to AXI Address Translation Registers' and the lower bits are taken from the inbound PCIe address to form the AXI address. An addr0 [5:0] + 1 number of lower bits are passed from the inbound PCIe address to AXI address. In other words, the number of bits taken from inbound PCIe address is given by the addr0[5:0] + 1 value.

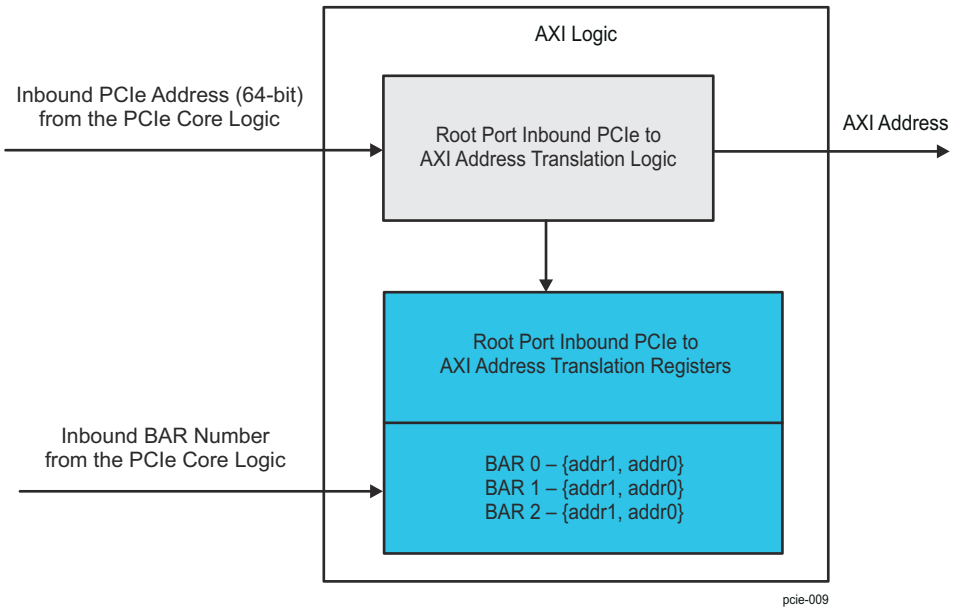


Figure 12-123. PCIE Root Complex Inbound PCIe to AXI Address Translation

A set of registers corresponding to one Root Complex BAR is shown in Table 12-145.

Table 12-145. PCIE Root Complex Inbound PCIe to AXI Address Translation Registers for one BAR

Register Name	Bits	Description	Default Value
addr1	31:0	Upper [63:32] bits of the AXI address.	32'd0
addr0	31:8	Lower [31:8] bits of the AXI address.	24'd0
	7:6	Reserved	2'd0
	5:0	Number of address bits passed through from PCIe to AXI. The PCIe controller passes the programmed value + 1 bits from PCIe to AXI. Minimum value to be programmed into this field is 7 as the lower 8 bits of the base address programmed in these registers (AXI) are replaced by zeros by the 'Root Complex Inbound PCIe to AXI Address Translation Logic'.	6'd0

12.2.2.3.7.1.2 End Point Inbound PCIe to AXI Address Translation

The End Point Inbound PCIe to AXI address translation is performed on memory and IO TLPs. The selection of which address translation registers to use in the translation process is dependent on the function number and BAR match of the incoming TLP. In End Point mode there are 7 bars per function, so 7 sets of registers are implemented per function, each BAR having two 32-bit registers (addr0 and addr1). The "End Point Inbound PCIe to AXI Address Translation Logic' takes the upper bits from the 'End Point Inbound PCIe to AXI Address Translation Registers' and the lower bits are taken from the Inbound PCIe Address to form the AXI address. The number of bits to pass from Inbound PCIe Address to AXI is decided by the Inbound BAR aperture.

Copyright © 2026 Texas Instruments Incorporated

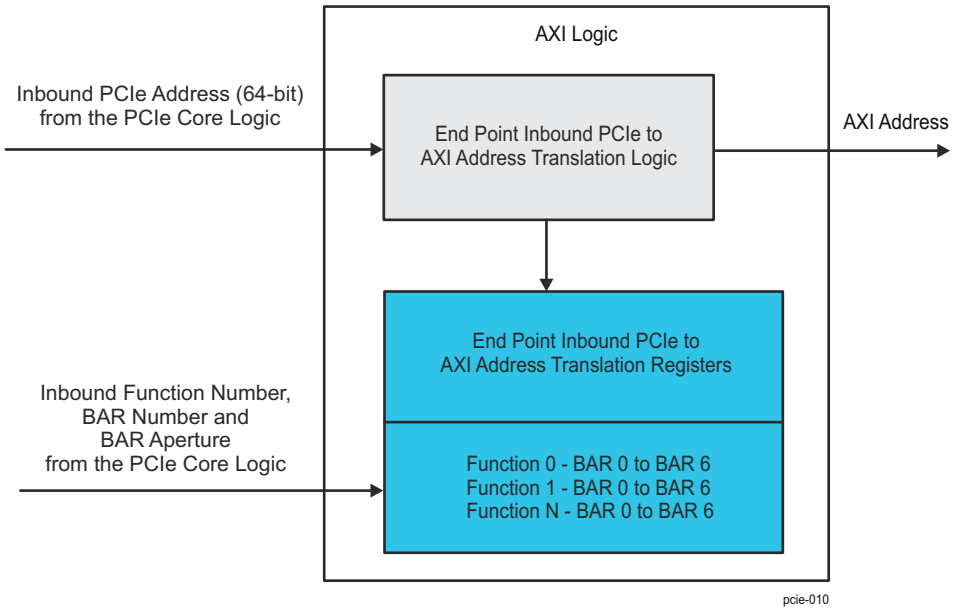


Figure 12-124. PCIE End Point Inbound PCIe to AXI Address Translation

A set of registers corresponding to one End Point BAR is shown in Table 12-146.

Table 12-146. PCIE End Point Inbound PCIe to AXI Address Translation Registers for one BAR

Register Name	Bits	Description	Default Value
[PF]_ib_ep_[BAR]_addr1 (where BAR can be bar0, bar1, bar2,...bar7, and PF can be pf0, pf1, pf2,... pf21)	31:0	Upper [63:32] bits of the AXI address.	32'd0
[PF]_ib_ep_[BAR]_addr0 (where BAR can be bar0, bar1, bar2,...bar7, and PF can be pf0, pf1, pf2,... pf21)	31:0	Lower [31:0] bits of the AXI address.	32'd0

12.2.2.3.7.2 PCIe Outbound Address Translation

The PCIe Subsystem allows mapping of PCIe addresses to/from the internal bus addresses of the device. This is accomplished by using internal address translation unit (iATU) in the PCIe core. For each outbound read/write request, the address translation module within PCIe subsystem can convert a VBUSM address to a PCIe address of memory Read/Write type.

If a transaction is large enough that it goes past the address translation region, unspecified behavior may occur. The address translation only works at the time a command is issued. So, a memory write, for example, will not automatically go to next translation region, if it starts in the previous one and is bigger than the remaining size in the starting translation region.

The “Dynamic Method: Sideband Descriptor Based” outbound address translation mechanism is used to bypass the outbound address translation unit under certain conditions. For more details on the bypass operation, see Section 12.2.2.3.7.2.1, *PCIe Outbound Address Translation Bypass*.

12.2.2.3.7.2.1 PCIe Outbound Address Translation Bypass

The PCIe subsystem supports bypassing the outbound address translation in the PCIe controller when the *casel* value on the VBUSM target interface is non-zero.

The device DMA has the ability to send transactions to alternate address maps which are external to the local device. This is achieved by using casel identifier in the DMA descriptors. There are three requirements that need to be met in the PCIe subsystem when using this mode in the DMA, namely:

Copyright © 2026 Texas Instruments Incorporated

- Block transactions based on initiator credentials to support FFI (Freedom from Interference)
- Bypass the Address Translation Unit (ATU) inside the PCIe controller since the transaction address is external to the local device
- Assign the correct Bus, Device, Function (BDF) and Traffic Class (TC) values to the transaction

To meet these requirements, the PCIe subsystem uses the *cvirtid* values on the VBUSM target interface to filter transactions based on initiator credentials. The VMAP block in the PCIe subsystem has a *virtid* match register (PCIE_VMAP_OB_VIRTID_MATCH) and a set of 32 registers that can be used to program the descriptor fields when the ATU logic in the PCIe controller is bypassed.

The PCIe subsystem will filter outbound VBUSM target transactions that have a non-zero casel value. If the *cvirtid[11:5]* attribute of the VBUSM target transaction matches the value programmed in the PCIE_VMAP_OB_VIRTID_MATCH register, the credentials of the transaction initiator are verified. The *cvirtid[11:5]* value should also be non-zero for a successful match.

When the credentials match, the ATU of the PCIe controller is bypassed . The *cvirtid[4:0]* is used to select one of the 32 PCIE_VMAP_DESC_j descriptor registers. These registers can be programmed to assign the BDF and TC values to the outbound PCIe transaction. If the PCIE_VMAP_DESC_j[16] BD_EN0 register bit is set, then the bus, device and function numbers are all set from the external descriptor registers. If the BD_EN bit is not set, then the bus and device numbers are assigned by the PCIe controller using the values captured during enumeration. If ARI mode is enabled (in the PCIE_CORE_PFn_I_SRIOV_CTRL_STATUS_REG register), the function number uses the PCIE_VMAP_DESC_j[7:0] DEV_FUNC_NUM register field. If ARI mode is not enabled, the device number uses PCIE_VMAP_DESC_j[7:4] DEV_FUNC_NUM field and the function number uses the [3:0] DEV_FUNC_NUM field. The ARI mode is a PCIe controller hardware configuration option that is enabled for PCIe endpoints that support SR-IOV.

If the credentials of the initiator fail to match the programmed value in the PCIE_VMAP_OB_VIRTID_MATCH register, the VBUSM target transaction is then flushed by the bridge and is not submitted to the PCIe controller. The VBUSM2AXI2 bridge will return a protection error status for both read and write transactions. In addition, the write data is discarded and the correct number of read data phases with data value of zero are returned.

12.2.2.3.8 PCIe Subsystem Quality-of-Service (QoS)

The Virtual Channel/Transfer Class (VC/TC) feature in the PCIe core, in conjunction with the device system CBASS Quality-of-Service (QOS) capabilities, provide a mechanism for supporting differentiated QOS within the PCIe subsystem. The policy for traffic differentiation is determined by the Transfer Class (TC) and Virtual Channel (VC) mapping and VC-based arbitration mechanism.

A Virtual Channel is established when one or more TCs are associated with a physical resource designated by a VC ID. Every Traffic Class that is supported on a given path within the subsystem must be mapped to one of the enabled Virtual Channels. Every Port must support the default TC0/VC0 pair – this is “hardwired.” Any additional TC mapping or additional VC resource enablement is optional. The number of VC resources provisioned within a component or enabled within a given subsystem may vary due to implementation and usage model requirements.

The PCIe core in the PCIe subsystem is configured to support four virtual channels (4VC/4TC). For both ingress and egress traffic, VC3, the highest numbered virtual channel, has the highest priority.

For ingress traffic, the TC information from each transaction on the AXI controller information is mapped to the CCHANID signal of the VBUSM controller interface. The system level interconnect can use the CCHANID along with the ORDERID for QoS purposes. Table 12-147 shows the TC mapping to the 12-bit CCHANID of each VBUSM controller interface.

Table 12-147. PCIe Subsystem QoS Ingress CCHANID Mapping

TC Value	CCHANID[11:0]
0	{9’d0, 3’b000}
1	{9’d0, 3’b001}
2	{9’d0, 3’b010}

Copyright © 2026 Texas Instruments Incorporated

Table 12-147. PCIe Subsystem QoS Ingress CCHANID Mapping (continued)

TC Value	CCHANID[11:0]
3	{9'd0, 3'b011}
4	{9'd0, 3'b110}
5	{9'd0, 3'b101}
6	{9'd0, 3'b110}
7	{9'd0, 3'b111}

For egress traffic requiring the highest priority it should be mapped to VC=3 in the PCIe controller. The VC mapping is done using the AXI outbound descriptor registers in the PCIe controller.

12.2.2.3.9 PCIe Subsystem Precision Time Measurement (PTM)

The Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Controller Time: an independent time domain associated with a PTM Root.

The PTM defines the following components:

- PTM Requester - A Function capable of using PTM as a consumer associated with an Endpoint.
- PTM Responder - A Function capable of using PTM to supply PTM Controller Time associated with a RC.
- Time Source - A local clock associated with a PTM Responder.
- PTM Root – The source of PTM Controller Time for a PTM hierarchy. A PTM Root must also be a Time Source and is typically also a PTM Responder.

When using PTM between two components on a Link, the EP sends PTM requests to the RC on the same link. During each dialog, the RC populates the PTM Response message based on timestamps stored during previous PTM dialogs. Once each component has historical timestamps from the preceding dialog, the EP can combine its timestamps with those passed in the PTM Response message to calculate the PTM Controller Time.

The PCIe core implements all of the features required to handle the PTM conversation between the requestor and responder in hardware. In addition, an internal TimeStamp module (CPTS) is connected to the timestamp interface of the PCIe core so that events can be logged.

The Precision Time Measurement (PTM) support in the PCIe subsystem is handled by the combination of the PCIe core and the CPTS module.

The PCIe core controller handles the PCIe conversation to exchange timestamps between the RC and EP. It also includes logic to calculate the timestamp differences as specified in the PCIe standard. A 64-bit timer that is the controller clock is included in the PCIe core. The PCIe controller can be programmed to initiate the PTM conversation automatically by setting the [0] PTMRQM bit in the PCIE_CORE_LM_I_PTM_LOCAL_CONTROL_REG register.

In EP mode, there are two methods to register a CPTS HW1 push timestamp that is needed to transfer the PCIe PTM timebase to the local system time base.

1. The first method involves the PTM_LOCAL_TIMER_OUT_VALID and PTM_EP_TIMER. The PTM_LOCAL_TIMER_OUT_VALID output from the PCIe controller will be asserted when the PTM conversation is complete and the EP local timer has been updated. The adjusted value of the EP 64-bit timer is available in the PTM_LOCAL_TIMER_OUT output of the controller. A second 64-bit free running timer, PTM_EP_TIMER, is used to shadow the value of the EP timebase. The PTM_EP_TIMER is loaded with the value of the PTM_LOCAL_TIMER_OUT[63:0] when PTM_LOCAL_TIMER_OUT_VALID is high. When PTM_LOCAL_TIMER_OUT_VALID is low, PTM_EP_TIMER will be free running based on the same PCIE CORE_CLK that drives the PTM_LOCAL_TIMER_OUT. The increment value of PTM_EP_TIMER can be adjusted by writing to the PCIE_USER_PTM_CFG[10-8] PTM_EP_TIMER_ADJ register field. The increment value is one by default. The value of the PTM_EP_TIMER will not be updated to the RC timebase until the first PTM dialog has occurred. Software can set the PCIE_USER_PTM_CFG[6-0] PTM_CLK_SEL register

Copyright © 2026 Texas Instruments Incorporated

field value between 0 and 63 to choose a particular bit of the PTM_EP_TIMER as the source for the CPTS HW1 time stamp push event.

2. The second method to register the CPTS HW1 push timestamp involves only the PTM_LOCAL_TIMER_OUT_VALID output from the PCIe controller. Software can set the PCIE_USER_PTM_CFG[6-0] PTM_CLK_SEL register field value to 64 to select the PTM_LOCAL_TIMER_OUT_VALID as the source of the CPTS HW1 time stamp push event. This will create a CPTS HW1 timestamp when there is a PTM dialog between the PCIe End Point and Root Complex. The actual value of the EP PTM local timer can be read from the PCIE_USER_PTM_TIMER_LOW and PCIE_USER_PTM_TIMER_HIGH registers.

In RC mode, the CPTS can take control of the PTM controller time. The 64-bit CPTS timer is continuously loaded into the PCIe PTM controller clock by tying high the PTM_TIMER_IN_VALID input of the PCIe controller. This allows for software to adjust the PCIe PTM controller clock as necessary using the CPTS module and maintain the PTM controller time by monotonically increasing it as required by the PCIe standard. The CPTS HW1_PUSH timestamp input is driven by the particular bit of the PTM_LOCAL_TIMER_IN counter, as selected by the value in the PCIE_USER_PTM_CFG[6-0] PTM_CLK_SEL register field.

The reference clock (RCLK) for the CPTS can be controlled via the CTRLMMR_PCIE0_CLKSEL[2-0] CPTS_CLKSEL register field in the device Control Module. This enables the SoC flexibility in choosing the CPTS reference clock input. In RC mode, the RCLK needs to be connected to the SERDES PIPE clock since the CPTS will be driving the PTM controller clock. For more information on the RCLK source clock selection, see *PCIe Subsystem Integration*.

Table 12-148. PHY Latency Parameters for PTM

	PHY Logic Latency	PHY Analog Latency	Total PHY Latency
Gen1 Transmit Latency	112ns	26.8ns	138.8ns
Gen2 Transmit Latency	56ns	13.4ns	69.4ns
Gen1 Receive Latency	160ns	25.2ns	185.2ns
Gen2 Receive Latency	80ns	12.6ns	92.6ns

Figure 12-125 shows the PTM support logic in the PCIe subsystem.

Copyright © 2026 Texas Instruments Incorporated

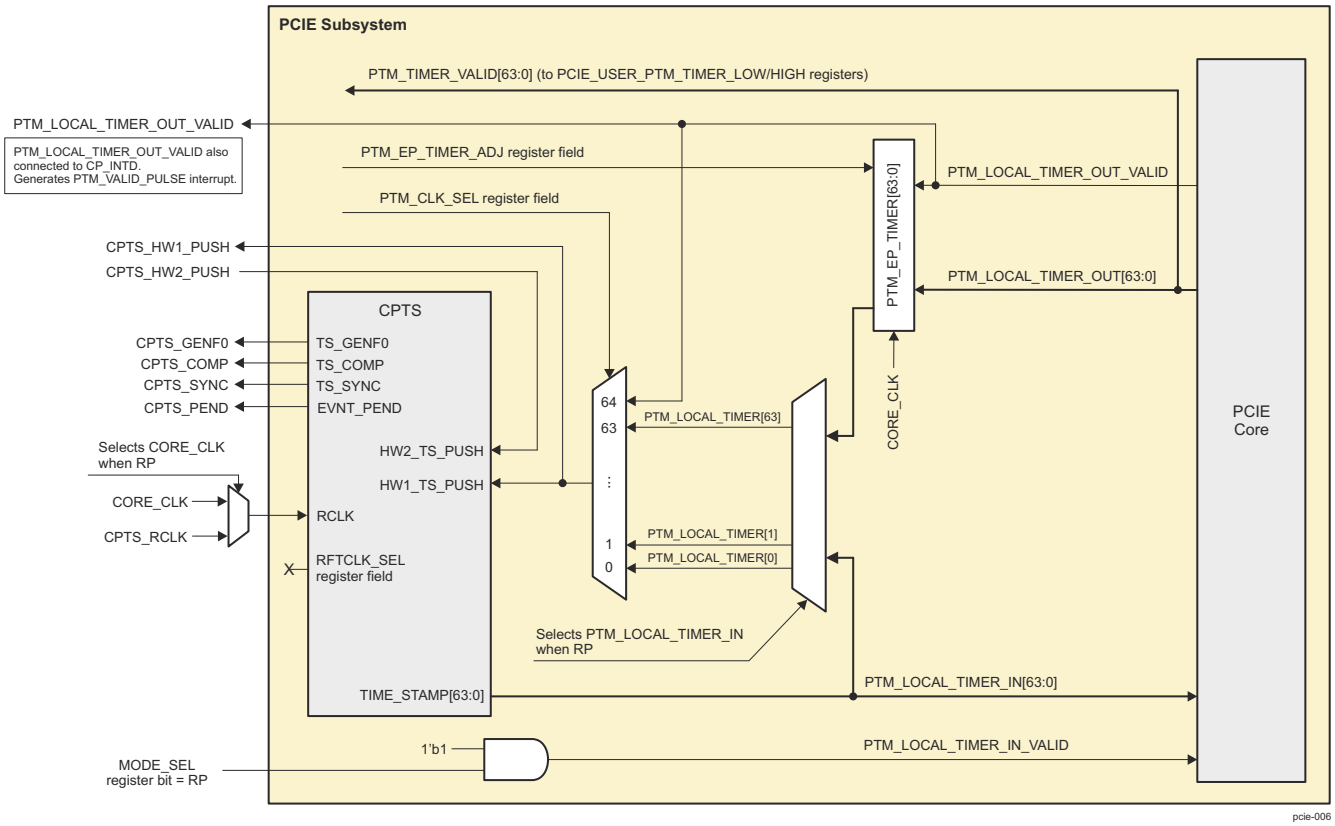


Figure 12-125. PCIe Subsystem PTM Implementation

12.2.2.3.10 PCIe Subsystem Loopback

The PCIe subsystem provides loopback support through PIPE interface (Link Layer).

12.2.2.3.10.1 PCIe Loopback

The procedure depends upon whether the device is operating in RC or EP Mode. In either case, the PCIe subsystem can be loopback initiator or loopback target as outlined in PCIe specifications. A loopback initiator is the component requesting loopback and transmitting the data. A loopback target is the component looping back the data.

12.2.2.3.10.1.1 PCIe Loopback Initiator Mode

The loopback path when PCIe subsystem is loopback initiator is:

PCIe subsystem (loopback initiator) -> PIPE (TX) -> PCIe Link -> Loopback -> PCIe Link -> PIPE (RX) -> PCIe subsystem

12.2.2.3.10.1.2 PCIe Loopback Target Mode

The loopback path when PCIe subsystem is loopback target is:

Remote device -> PCIe Link -> PIPE (RX) -> Loopback -> PIPE (TX) -> PCIe Link -> Remote device

If the PCIe subsystem is a loopback target, then the incoming serial data is routed back to the originating device from the PIPE interface as per PCIe loopback requirements. Typically, PCIe test equipment will be used as loopback initiator and it will transition PCIe subsystem into loopback target state following which, the inbound transactions will be looped back to the test equipment.

Copyright © 2026 Texas Instruments Incorporated

12.2.2.3.11 PCIe Subsystem Error Handling

As an End Point, PCIe subsystem reports errors to root complex so that corrective action may be taken. These messages become error interrupts on the Root Complex side. In addition, the errors that are detected by the PCIe subsystem locally are also reported to software via interrupts.

Note that for several errors of the same type that are detected in close succession, the PCIe subsystem will not necessarily send as many error messages to Root Complex. It is guaranteed to send at least one such error message.

12.2.2.3.12 PCIe Subsystem Internal Diagnostics Features

12.2.2.3.12.1 ECC Aggregators

The PCIe subsystem instantiates two ECC Aggregators: AXI_ECC_AGGR and CORE_ECC_AGGR.

AXI_ECC_AGGR is connected to the RAMs in PCIe CBA Clock Domain and CORE_ECC_AGGR is connected to the RAMs in the PIPE Clock Domain.

The ECC Aggregator modules integrated within the PCIe subsystem provide a mechanism to control and monitor the ECC RAMs via Single Error Correction (SEC) and Double Error Detection (DED) functions. Each ECC Aggregator module gathers level pending status from the ECC RAMs into two interrupts to system level. One interrupt is for correactable errors (SEC) and the other one for uncorrectable errors (DED). The ECC Aggregator supports software readable status of ECC single/double-bit errors and associated information such as RAM address and data bit(s) that are in error. Write back correction for async read RAMs is not supported.

For more information on the ECC Aggregator operation, see *ECC Aggregator*.

12.2.2.3.12.2 RAM ECC Inversion

The RAM ECC logic inside the PCIe controller can be tested using the ECC inversion logic.

Copyright © 2026 Texas Instruments Incorporated

12.2.3 Serializer/Deserializer (SerDes)

This section describes the Serializer/Deserializer (SerDes) in the device.

12.2.3.1 SerDes Overview 1420
12.2.3.2 SerDes Environment 1423
12.2.3.3 SerDes Integration 1424
12.2.3.4 SerDes Functional Description 1427

Copyright © 2026 Texas Instruments Incorporated

12.2.3.1 SerDes Overview

SerDes'es goal is to convert device (SoC) parallel data into serialized data that can be output over a high-speed electrical interface. In the opposite direction, SerDes converts high-speed serial data into parallel data that can be processed by the device. To this end, the SerDes contains a variety of functional blocks to handle both the external analog interface as well as the internal digital logic.

Most important building blocks of SerDes are:

- Physical Media Attachment (PMA):
  - Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer, and Clock and Data Recovery (CDR) unit.
  - Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes, and muxes SerDes to peripherals.

Figure 12-126 shows SERDES0 highlights.

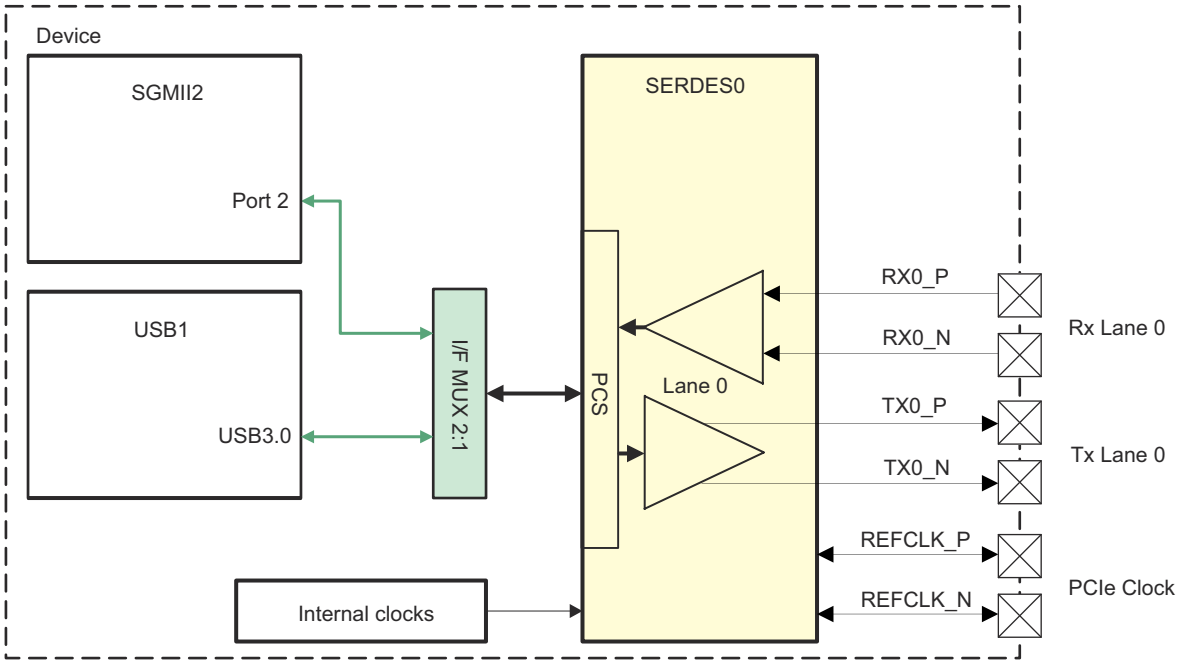


Figure 12-126. SERDES0 Overview

Copyright © 2026 Texas Instruments Incorporated

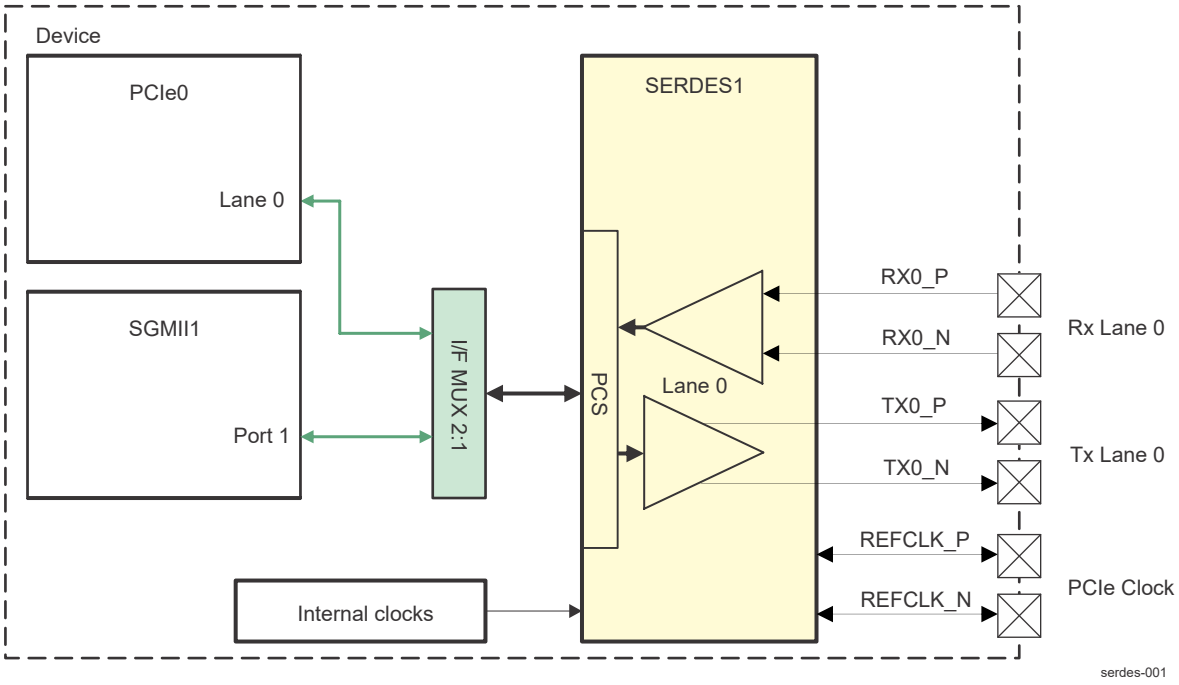


Figure 12-127. SERDES1 Overview

12.2.3.1.1 SerDes Features

The SERDES module features include:

- Single lane PHY containing:
  - Transmit and Receive I/Os
  - Serializer
  - Deserializer
  - Clock and data recovery (CDR) unit
- Common Module (CMN)
  - PLLs
  - Controller bias
  - Automatic calibration of pin termination resistors
  - Reference clock input buffers
  - Reset and startup management
- Physical Coding Sub-block (PCS)
  - USB3.1 Gen 1 (5 Gbps)
  - PCIe Gen 1 (2.5 Gbps), Gen 2 (5 Gbps), and PCIe Gen 3 (8 Gbps)
  - Symbol alignment
  - Selectable serial pin polarity reversal for both transmit and receive paths
  - Bit stream reordering
- Physical Media Attachment (PMA) layer
  - Transmit equalization
  - Receive equalization
  - Supports on-the-fly eye and bathtub curve diagramming with 8-bit voltage amplitude resolution and up to 1/64 UI time resolution
  - Data path BIST with programmable pattern generation and error detection
  - Serial bit stream and parallel word loopback for both line and parallel side
  - 8-bit ADC provides digitized ATB measurement results
  - Supports DC and AC JTAG (boundary scan) per IEEE 1149.6

Copyright © 2026 Texas Instruments Incorporated

The SERDES mux (WIZ) module supports the following features:

- Multiplexes device interfaces onto a single SERDES lane (Tx and Rx)
- Provides registers to implement SERDES control and status functions and alignment delays
- Clock generator block for providing MAC transmit clock
- Rx comma align block
  - Performs de-stuffing the Rx data stream in the event that the Rx rate is different from the Tx rate
  - Supports comma detection that is not sensitive to false commas using all 8b/10b character combinations

12.2.3.1.2 Not Supported Features

All features described in this chapter are supported.

12.2.3.1.3 Industry Standards Compatibility

All SerDes interfaces are configured as point-to-point connections. It is assumed that the connection is made between the SoC and another device that is compliant to the appropriate industry standard. The list of supported standards is given below.

This chapter deals with the physical layer and, therefore, it is the electrical specifications in these standards that are relevant. For more information regarding protocol compliance [1], see the device-specific Datasheet.

- PCIe is electrically compliant with Base specification revision 4.0.
- USB is electrically compliant with USB 3.1.
- SGMII: Electrically compliant with IEEE 802.3 Clause 70-7

Table 12-149. SerDes Supported Standards

Standard	Bit Rate (Gbps)	Reference Clock Frequency (MHz)
PCI Express 4.0 Gen1	2.5	100
PCI Express 4.0 Gen2	5.0	100
PCI Express 4.0 Gen3	8.0	100
USB 3.1 Gen1 SuperSpeed	5.0	100
SGMII	1.0	100

[1] Electrical compatibility does not guarantee interoperability with devices

Copyright © 2026 Texas Instruments Incorporated

12.2.3.2 SerDes Environment

12.2.3.2.1 SerDes I/Os

Figure 12-128 shows the I/O interface signals of SERDES.

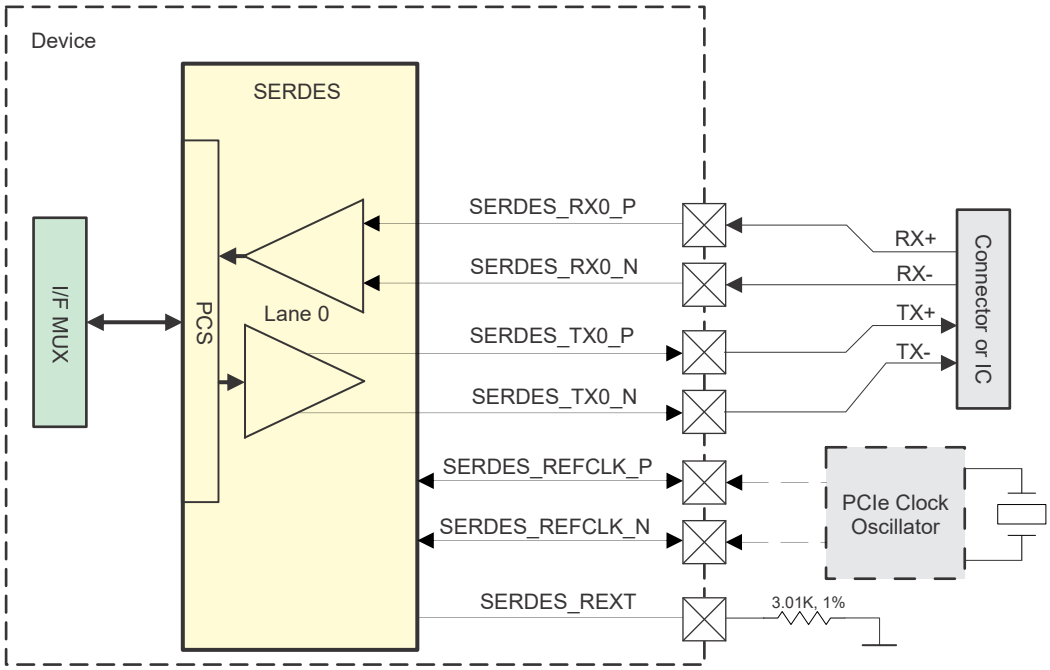


Figure 12-128. SerDes Environment

Note

Although containing some of the basic external components, Figure 12-128 must not be considered as an exhaustive guide for the PCB designer. TI provides additional documents for those who are willing to design PCBs and/or fine tune the SerDes.

Table 12-150 describes the external signals of SERDES0 module.

Table 12-150. SERDES0/1 Input/Output Description

Device Pin(2)	Module Signal	I/O(1)	Description	Value at Reset
SERDESx_RX0_P	RX_P_LN0	I	SerDes differential data receive pins	HiZ
SERDESx_RX0_N	RX_M_LN0	I		HiZ
SERDESx_TX0_P	TX_P_LN0	O	SerDes differential data transmit pins	HiZ
SERDESx_TX0_N	TX_M_LN0	O		HiZ
SERDESx_REFCLK0P	CMN_REFCLK_P	I/O	SerDes external PCIe reference clock	HiZ
SERDESx_REFCLK0N	CMN_REFCLK_N	I/O		HiZ
SERDESx_REXT	CMN_REXT	A/I	PMA external calibration resistor. Requires a 3.01 kOhm ±1% accurate off-chip resistor connected from this pin to ground.	HiZ

(1) I = Input; O = Output; A = Analog
(2) x = 0 or 1

Copyright © 2026 Texas Instruments Incorporated

12.2.3.3 SerDes Integration

Figure 12-129 shows the integration of the SERDES0 module in the device.

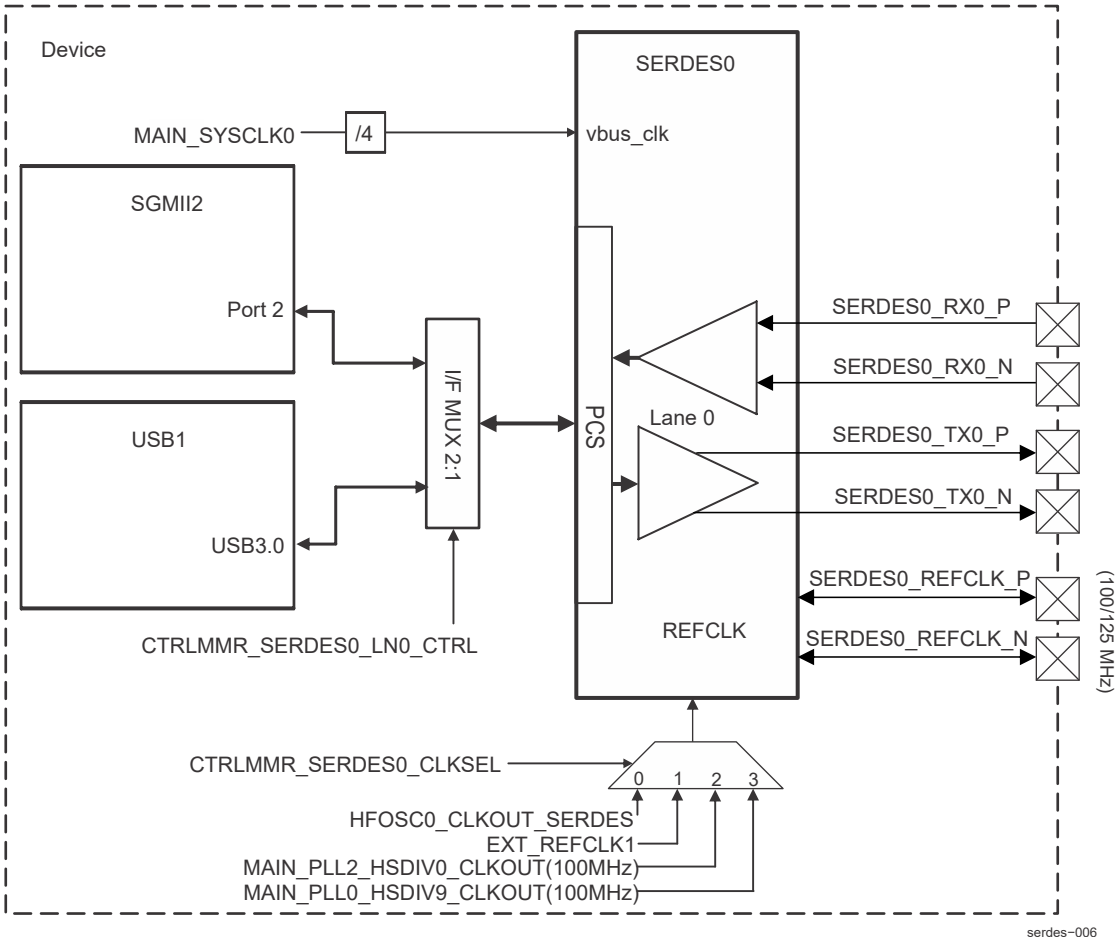


Figure 12-129. SERDES0 Integration

Copyright © 2026 Texas Instruments Incorporated

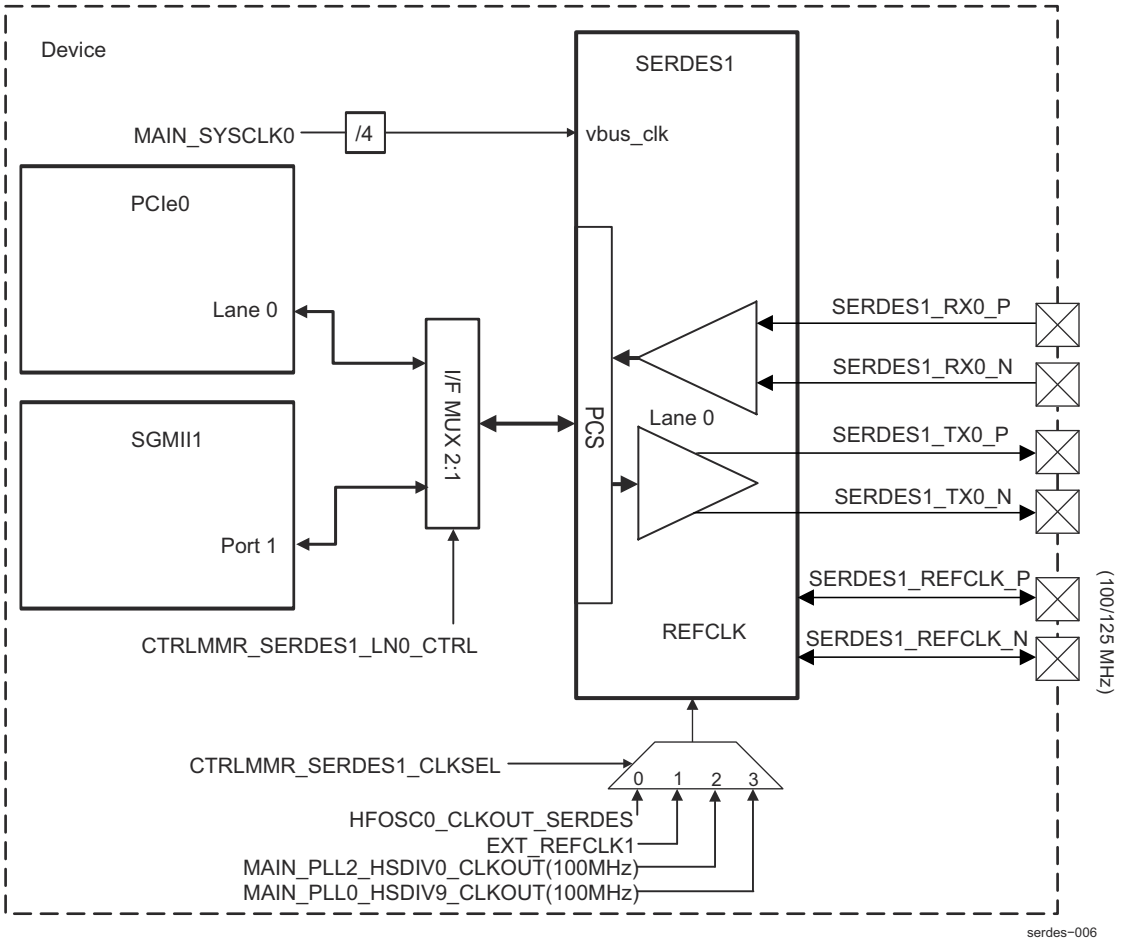


Figure 12-130. SERDES1 Integration

The following table summarizes the integration of SerDes in device MAIN domain.

Table 12-151. SerDes Clocks

Module Instance	Module Clock Input	Source Clock Signal	Source	Description
SERDES0/1	SERDES0/1_ICLK	MAIN_SYSCLK0/4	PLL_CTRL0	VBUS interface clock
	CMN_REFCLK_INT	HFOSC0_CLKOUT_SERDES	HFOSC0	Internal reference clock from device sources. Software selectable. See Section 12.2.3.3.1.2
		EXT_REFCLK1	External REFCLK	
		MAIN_PLL2_HSDIV0_CLKOUT(100MHz)	PLL2	
		MAIN_PLL0_HSDIV9_CLKOUT(100MHz)	PLL9	

Copyright © 2026 Texas Instruments Incorporated

12.2.3.3.1 WIZ Settings

This section describes the quazi-static settings that software must perform after global resets. These settings are made via registers in the CTRL_MMR0 module. For CTRL_MMR0 description, please refer to *Control Module (CTRL_MMR)*.

12.2.3.3.1.1 Interface Selection

SerDeses provide PHY functions for the following high-speed interfaces:

- PCIe0 (one lane)
- USB1 SS
- SGMII (1G)

Table 12-152 describes the interface combinations supported by SERDES0.

Table 12-152. SERDES0 Supported Configurations

Interface Alias	CTRLMMR_SERDES0_LN0_CTRL [1:0]	
	LANE_FUNC_SEL	Interface on Lane 0
IP1	0x0	USB1 SS
IP2	0x1	SGMII2

Table 12-153. SERDES1 Supported Configurations

Interface Alias	CTRLMMR_SERDES1_LN0_CTRL [1:0]	
	LANE_FUNC_SEL	Interface on Lane 0
IP1	0x0	PCIe0
IP2	0x1	SGMII1

12.2.3.3.1.2 Internal Reference Clock Selection

Table 12-154 describes the internal reference clock options for SERDES0.

Table 12-154. SerDes Internal Reference Clock Selection

CTRLMMR_SERDES0/1_CLKSEL	
[1:0] CLK_SEL	Internal Clock to SERDES Core
0x0	HFOSC0_CLKOUT_SERDES
0x1	EXT_REFCLK1
0x2	MAIN_PLL2_HSDIV0_CLKOUT(100MHz)
0x3	MAIN_PLL0_HSDIV9_CLKOUT(100MHz)

Copyright © 2026 Texas Instruments Incorporated

12.2.3.4 SerDes Functional Description

12.2.3.4.1 SerDes Block Diagram

Figure 12-131 is a top-level, non-exhaustive diagram of SerDes and WIZ wrapper. Note that only one (Tx + Rx) lane is shown.

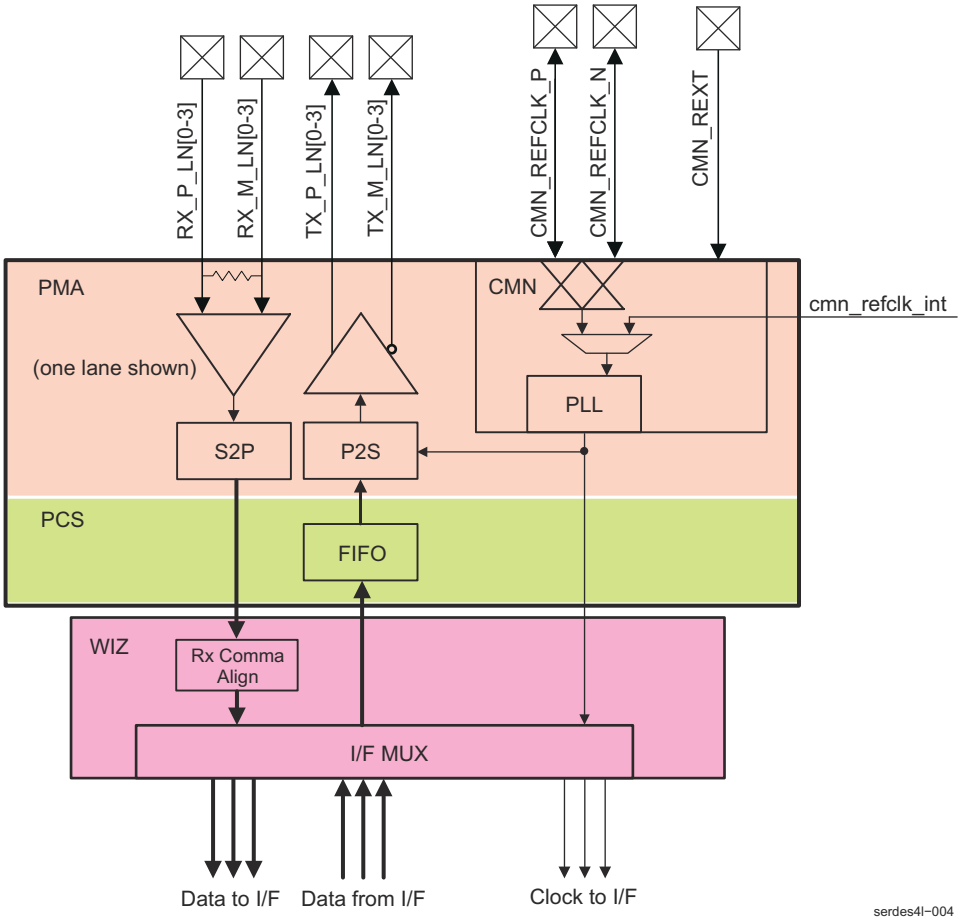


Figure 12-131. SerDes and WIZ Block Diagram

Building blocks of SerDes include:

- Lanes: The lanes handle all inputs and outputs from the serial interface, and contain the Tx/Rx I/Os, serializer/deserializer (P2S/S2P), and Clock and Data Recovery (CDR) unit. Each SerDes contains one Tx and Rx lane.
- Common module (CMN): The CMN handles peripheral and Tx clocking of the SerDes. It consists of internal PLLs and external reference clock input buffer, reset, and startup circuitry.
- Lanes and CMN are parts of the Physical Media Attachment (PMA) layer.
- Physical Coding Sub-block (PCS): The PCS is responsible for translating data from/to the parallel interface, as well as data encoding/decoding and symbol alignment.
- WIZ: The WIZ acts as a wrapper for the SerDes, and can both send control signals to and report status signals from the SerDes (register interface), and muxes SerDes to peripherals (USB, PCIe, and others).

Copyright © 2026 Texas Instruments Incorporated

12.2.4 Universal Serial Bus Subsystem (USBSS)

This section describes the Universal Serial Bus Subsystem (USBSS) in the device.

12.2.4.1 USB Overview

USB (Universal Serial Bus) provides a low-cost connectivity solution for numerous consumer portable devices by implementing a mechanism for data transfer between USB devices.

The device instantiates two independent instances of a third-party USB subsystem (USB2SS) operating at up to USB2.0 speeds (480Mb/s), either of which can be independently configured to act as a USB Host or a USB Device. SuperSpeed (5.0 Gb/s) operation is not supported in either operational mode.

12.2.4.1.1 USB Features

The USB 2.0 subsystem supports the following USB Features:

- Operational modes:
  - Supports USB 2.0 Host mode at High-Speed (HS, 480 Mbps), Full-Speed (FS, 12 Mbps), and Low-Speed (LS, 1.5 Mbps)
  - Supports USB 2.0 Device mode at High-Speed (HS, 480 Mbps), and Full-Speed (FS, 12 Mbps). Low-Speed is not supported in Device mode.
  - Supports all modes of transfers - Control, Bulk, Interrupt, and Isochronous.
- A DRD (Dual-Role-Device - Host or Device) USB controller with the following features:
  - Compatible to the xHCI 1.0 specification in Host mode
  - Compatible with the USB 2.0 specification in Device mode
  - Supports 15 IN (Receive), 15 OUT (Transmit) endpoints (EPs), and one EP0 endpoint which is bidirectional
  - Internal DMA controller
  - Descriptor caching and data pre-fetching ensures high performance
  - Dynamic FIFO memory allocation for all endpoints
- Operation flexibility
  - Same programming model for HS, FS, and LS operation
  - Each controller instance can provide either USB Host or USB Device functionality

12.2.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

12.2.4.2 USB Environment

12.2.4.2.1 USB Pin List

The names of the pins used by the USB2SS with descriptions are shown in Table 12-155 and Table 12-156.

Table 12-155. USB Signal Pins Description

Pin Name	I/O(1)	Description
USBn_DP	I/O	USB 2.0 and 1.1 Specification-compliant signal pins. They are HS/FS/LS bidirectional differential data pins.
USBn_DM	I/O	

(1) I = Input, O = Output, A = Analog

Table 12-156. USB Control, Configuration, and Monitor Signal Pins

Pin Name	I/O(1)	Description
USBn_DRVVBUS	O	An active-high digital output signal for VBUS power supply. Used to enable an external charge pump to supply +5V power to the VBUS receptable when operating in USB Host mode.
USBn_VBUS(2)	A/I	An analog input for monitoring the voltage on VBUS.

Copyright © 2026 Texas Instruments Incorporated

Table 12-156. USB Control, Configuration, and Monitor Signal Pins (continued)

Pin Name	I/O(1)	Description
USBn_ID	I	USB operational mode identifier. This functionality is supported via GPIO.

(1) I = Input, O = Output, A = Analog
(2) This pin is not 5V tolerant. Refer to the device-specific datasheet for implementation requirements.

12.2.4.2.2 Typical Pin Connections of Device

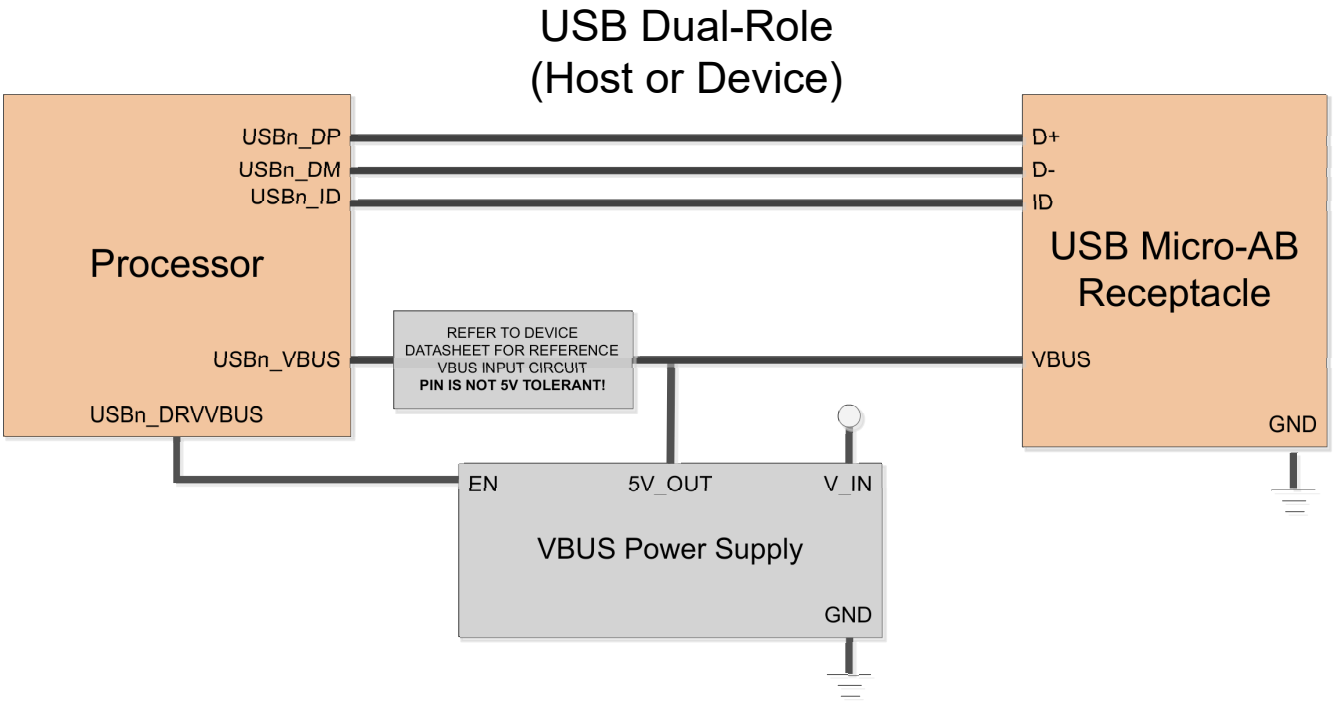


Figure 12-132. USB Dual-Role Connections

Copyright © 2026 Texas Instruments Incorporated

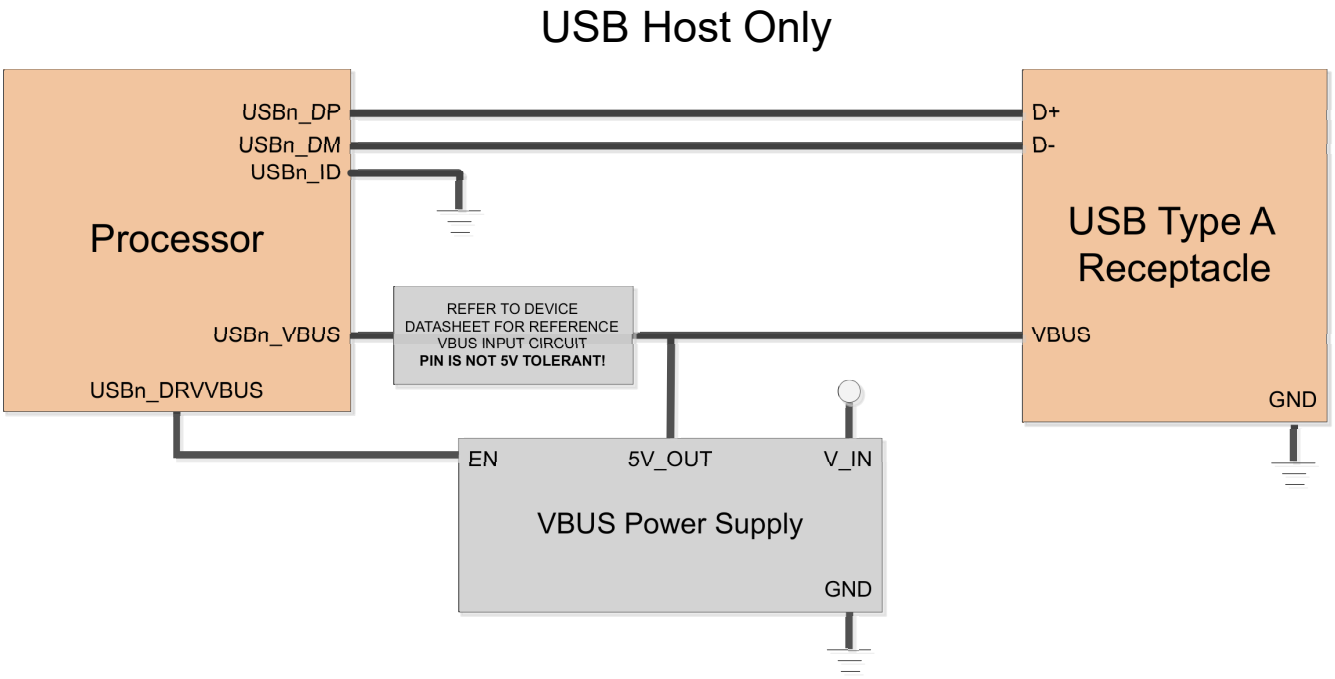


Figure 12-133. USB Host-Only Connections

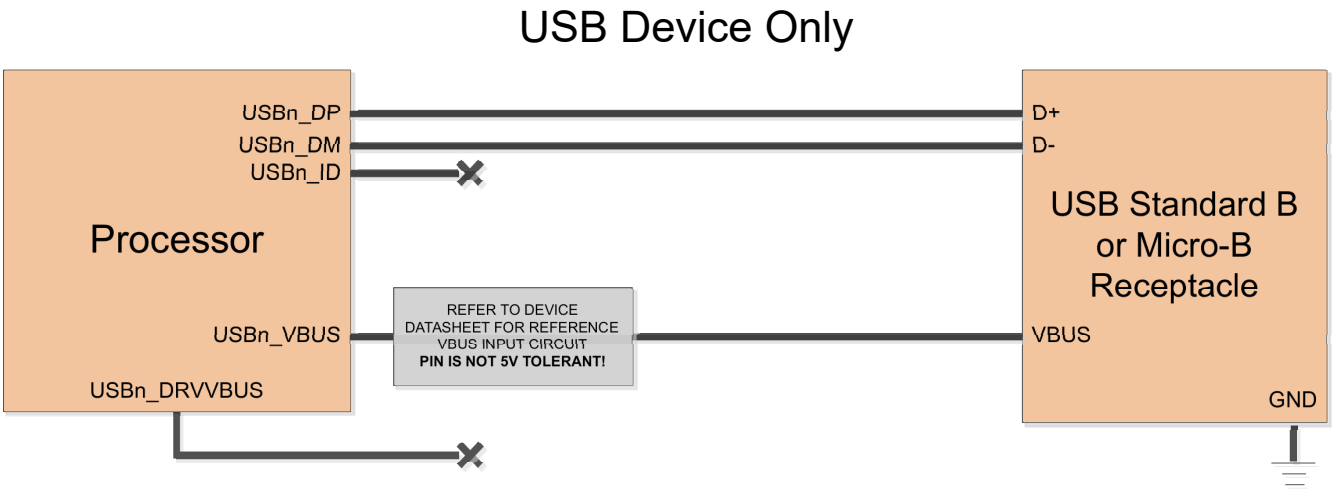


Figure 12-134. USB Device-Only Connections

12.2.4.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

12.2.4.4 Use Cases

This is a standard USB 2.0 module, and is optimized for following applications and systems:

- Portable electronic devices
- High-bandwidth applications

It supports all typical USB connections, and Table 12-157 shows some examples.

Copyright © 2026 Texas Instruments Incorporated

Table 12-157. Typical Use Cases In Terms of Connections

Connectors (Receptacle)	Signals to Use	SS	HS/FS	LS (Host only)	Comments
USB 2.0 Micro-AB	DP, DM, VBUS, ID	N	Y	Y	Support Host or Device operation, depending on state of the ID pin.
USB 2.0 Type-A	DP, DM, VBUS	N	Y	Y	Support HS/FS/LS Host
USB 2.0 Type-B	DP, DM, VBUS	N	Y	Y	Only used for USB2.0 Device (no LS)

12.2.4.4.1 USB Operational Mode Determination

USB[n]_ID Pin	USB[n]_VBUS Pin(1)	USB Role
FLOAT	FLOAT or GROUND	INDETERMINATE
GROUND	UNUSED	HOST
FLOAT	RAIL VOLTAGE	DEVICE

(1) PHY VBUS pin voltage is dependent on the divider circuit used on the board. Refer to the device datasheet for VBUS implementation requirements.

12.2.4.4.2 VBUS Voltage Sourcing Control

When either of the USB controllers assumes the role of a host, the controller is required to supply 5V to an attached device through its VBUS line. In order to achieve this task, the USB controller requires the use of external power logic (or a charge pump) capable of sourcing 5V power. The USB_DRVVBUS pin is used as a control signal to enable/disable this external power logic to either source or disable power on the VBUS line. The control on the USB_DRVVBUS is handled by the controller based on host software programming. The control should be transparent to the user so long as the proper hardware connection and software initialization are in place. The USB controller drives the USB_DRVVBUS signal high when it assumes the role of a host while the controller is in session. When assuming the role of a device, the controller drives the USB_DRVVBUS signal low disabling the external charge pump/power logic; hence, no power is driven on the VBUS line (in this case, power is expected to be provided by the external host).

Note that both USB controllers are self-powered and the device does not rely on the voltage on the VBUS line sourced by an external host for controller operation when assuming the role of a device. The voltage present on the VBUS line is used to identify the presence of a Host. The USB PHY continually monitors the voltage on the VBUS and reports the status to USB controller.

12.2.4.4.3 VBUS Detection

VBUS detection is supported by the USB2SS via the USB_VBUS pin, but **it is important that this pin not be connected directly to the connector 5V or any other 5V source.** Instead, an external circuit must be used to ensure that the recommended operating conditions of the pin are not exceeded. Refer to the device datasheet for requirements and example VBUS-sense circuitry.

The PHY includes session valid and VBUS valid comparators and provides corresponding status signals.

The Controller uses the session valid output from the PHY in device mode. One of the reasons is because the USB specification requires that the DP pull-up (implemented inside the PHY) must be be enabled only after VBUS is turned on by host. This PHY output also in general indicates that a connection is active from host and it can also be used to detect disconnects.

The USB2SS wrapper also includes logic to detect a change on session valid and VBUS valid outputs from PHY.

12.2.4.4.4 Pull-up/Pull-down Resistors

As the USB controllers are dual role controllers, capable of assuming a role of a host or device, the required pull-up/pull-down resistors cannot exist external to the device. These pull-up/pulldown resistors exist internal to the device, within the PHY to be more specific, and are enabled or disabled based on the role the controller assumes allowing for dynamic hardware configuration. When assuming the role of a host, the data lines are

Copyright © 2026 Texas Instruments Incorporated

pulled low by the PHY enabling the internal 15KΩ resistors. When assuming the role of a device the required 1.5KΩ pull-up resistor on the D+ line is enabled automatically to signify the USB capability to the external host as a FS device (HS operation is negotiated during reset bus condition). Low-Speed Device functionality is not supported.

Copyright © 2026 Texas Instruments Incorporated

12.2.5 Universal Serial Bus Subsystem (USBSS)

This section describes the Universal Serial Bus Subsystem (USBSS) in the device.

Note

This chapter serves to describe the integration of the third-party USB controller and should not be considered sufficient for those wishing to modify the existing Linux or RTOS USB driver(s) or create a new driver to support this controller implementation. For those who do wish to substantially modify the existing Linux or RTOS USB driver(s), or create new drivers, contact TI for more information on how to obtain the third-party documentation under NDA.

12.2.5.1 USB1 Overview 1434
12.2.5.2 USB Environment 1437
12.2.5.3 USB Functional Description 1439

Copyright © 2026 Texas Instruments Incorporated

12.2.5.1 USB1 Overview

The USB 3.1 Gen1 Interface is a general-purpose cable bus, supporting data exchange between a host device and a wide range of simultaneously accessible peripherals, similar to previous versions of USB.

The device supports one USB subsystem integrated in the MAIN domain instantiated as USB1:

- USB3SS is SuperSpeed (SS) USB 3.0 Dual-Role-Device (DRD) subsystem with on-chip SS (USB3.0) PHY and HS/FS/LS [2](USB2.0) PHY

Table 12-158. USB Allocation Within Device Domains

Module Instance	Domain	
	MCU	MAIN
USB3SS	-	✓

Figure 12-135 shows the USB subsystem highlights.

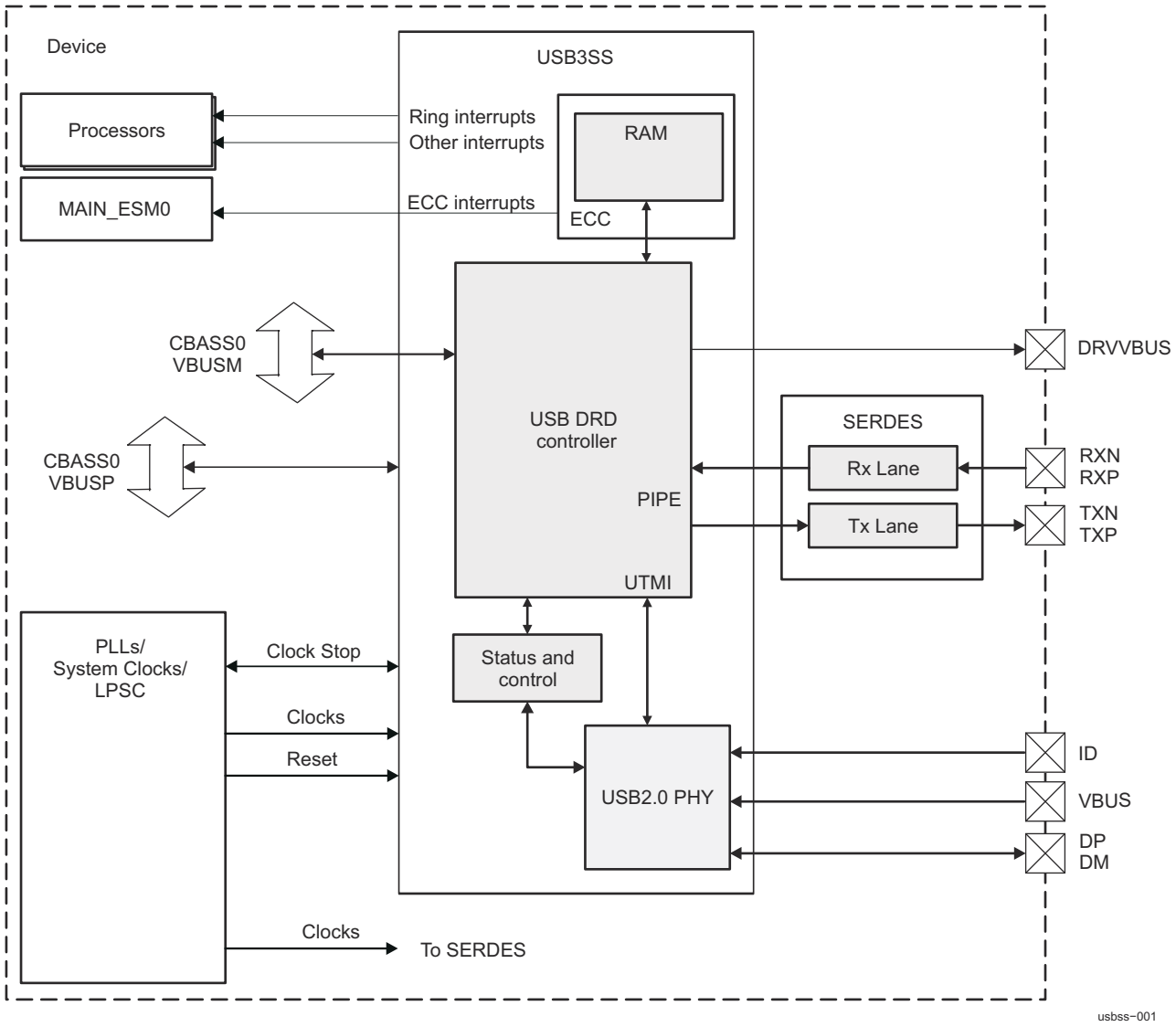


Figure 12-135. USB Subsystem Overview

[2] LowSpeed (LS) is supported only in host mode.

Copyright © 2026 Texas Instruments Incorporated

12.2.5.1.1 USB Features

The USB subsystem supports the following USB Features:

- USB interface:
  - Compliant with USB 3.1 Gen1 specification
  - Compliant with xHCI 1.0 specification
  - Limited USB 2.0 on-the-go support
  - High speed (480 Mbps), and full speed (12Mbps) Device
  - SuperSpeed (5 Gbps), high speed (480 Mbps), full speed (12Mbps), and low speed (1.5 Mbps) Host
  - Single USB2.0 port
  - Single USB3.1 port
  - USB2 L1/L2 support
  - USB3 U1/U2/U3 support
- Dual mode operation:
  - Programmable runtime mode change
  - OTG 2.0 host negotiation protocol (HNP) support
- Host mode:
  - 64 slots supported
  - Up to 32 endpoints per slot
  - 256 primary streams supported
  - Root hub functionality
- USB2 PHY:
  - Fully compliant with UTMI+ Level 3 specification revision 1.0
  - Supports high-speed (480 Mbps), full-speed (12 Mbps) and low-speed (1.5 Mbps) data rates
  - Supports battery charging BC1.2v specification
  - Supports host, peripherals and OTG 2.0 (dual role device) applications
  - Supports D+/D- lane reversal for flexible board integration
  - Supports USB low-power states; namely, suspend and link power management (LPM)
  - Supports internal comparators for monitoring OTG voltage thresholds
  - Supports multiple PLL reference clocks
  - Supports internal PLL for high-speed (480 MHz) clock and data recovery (CDR) operation
  - Integrated termination resistors (45 Ω, 1.5 KΩ, and 15 KΩ)
  - Supports built-in self-test (BIST) for production testing
  - 3.3-V ESD support on VBUS

12.2.5.1.2 USB Not Supported Features

The following are USB features which are not supported by the hardware integration of the current device:

Table 12-159. USBSS Unsupported Features

Feature	Reason
Virtualization (SRIOV or xHCI-IOV or ARM SMMU)	No Support
SuperSpeed (5 Gbps) Device Mode	No Support
Debug Trace for USB	No Support
Attach Detection Protocol (ADP)	This function is not possible because the USB2 PHY implements 3.3v ESD on the VBUS pin.
5 volt compatible VBUS pin	This function is not possible because the USB2 PHY implements 3.3v ESD on the VBUS pin. Note: The VBUS signal should be applied to a 20k/10k voltage divider which divides the potential by 3. This allows the VBUS signal to go up to 10.75 volts before exceeding the VBUS pin maximum of 3.63 volts.
NRDY Event TRB, when operating in host mode	NRDY Event TRB is not supported when operating in host mode.
Halt Endpoint Command TRB, when operating in host mode	Halt Endpoint Command TRB is not supported when operating in host mode.
OTG Protocol, for USB Type-C applications	OTG Protocol is not supported for USB Type-C applications.

Copyright © 2026 Texas Instruments Incorporated

Table 12-159. USBSS Unsupported Features (continued)

Feature	Reason
Light HC Reset Capability (LHRC)	No Support
xHCI Debug Capability (Debug Port)	No Support
xHCI Save and Restore Operations	No Support
HIMD, when operating in host mode	No Support
Latency Tolerance Messaging	Set Latency Tolerance Value Command is not supported
Negotiate Bandwidth Command	No Support
Number of slots limited to 64	Total for USB2 and USB3
Number of active periodic endpoints limited to 96, when operating in host mode	No Support
Smart ISO	Deprecated in USB 3.2 specification
Secondary Streams	No Support
PTM, when operating in device mode	No Support

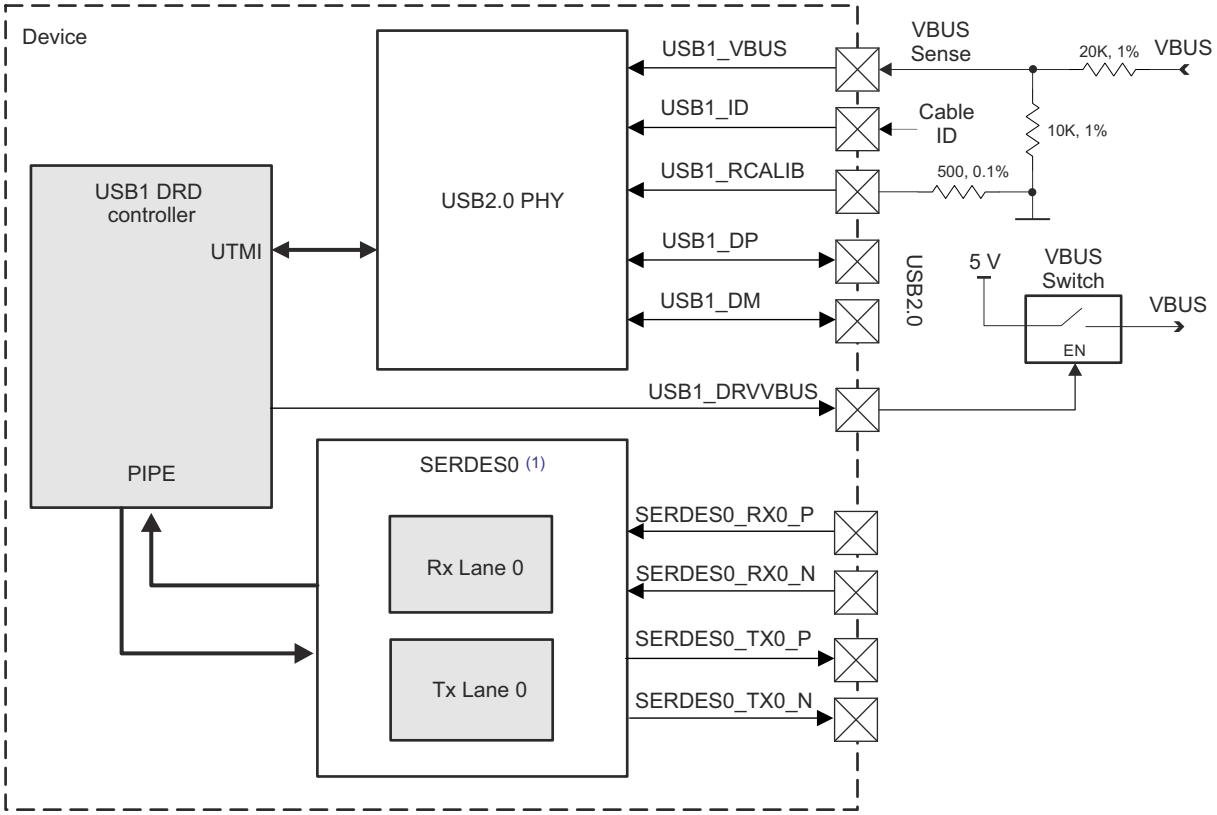
12.2.5.1.3 USB Terminology

The following acronyms and abbreviations are related to USB.

Term	Definition
ADP	Attach Detection Protocol: detects USB OTG attach/detach events.
DRD	Dual Role Device: USB Host and Peripheral capable.
DS	Down Stream (A USB port facing from a Host or Hub to a Device/Peripheral).
EP	Endpoint: USB communication channel between the USB link partners carrying a single transfer type (BULK, ISOCH, INT, or CONTROL) and bus sharing arbitration scheme.
FS	Full-Speed USB data rate (12 Mbps).
HNP	Host Negotiation Protocol, OTG extension to swap USB host and peripheral roles.
HS	High-Speed USB data rate (480 Mbps).
ITP	Isochronous Timestamp Packet: USB SS micro-frame boundary packets.
LMP	Link Management Packet.
LS	Low-Speed USB data rate (1.5 Mbps).
OTG	On-The-Go extension to USB protocol.
PHY	Physical Layer Device.
SS	Super-Speed USB data rate. 5 Gbps (USB3.0 or USB3.1 Gen1)
US	Upstream facing from a device (or hub) to the host (or a hub).
USB IF	USB Implementers Forum. The governing organization that develops and maintains the USB specifications and compliance standard. http://www.usb.org
xHC	Host Controller, designates the USB hardware that implements the xHCI specification.
xHCI	eXtensible Host Controller Interface. The specification that defines the register level interface for host controller.

Copyright © 2026 Texas Instruments Incorporated

12.2.5.2 USB Environment



(1) See section *SerDes* for details

Figure 12-136. USBSS Subsystem Environment

12.2.5.2.1 USB I/Os

Table describes the external signals of the USB3SS module.

Table 12-160. USBSS Input/Output Description

Device Pin	Module Signal	I/O	Description	Value at Reset
USB1_DP	DP	I/O	DP to USB connector	HiZ
USB1_DM	DM	I/O	DM to USB connector	HiZ
USB1_ID	ID	A/I	ID to USB connector	HiZ
USB1_VBUS	VBUS	A/I	VBUS to USB connector	HiZ
USB1_RTRIM			USB2 PHYreference resistor	
USB1_DRVVBUS	DRVVBUS	O	3.3 Volt LVCMOS Output. Controls VBUS power switch	0
The following pins are implemented through SERDES				
USB1_SSRXN		I	N of differential receive data	HiZ
USB1_SSRXP		I	P of differential receive data	HiZ
USB1_SSTXN		O	N of differential transmit data	HiZ
USB1_SSTXP		O	P of differential transmit data	HiZ

Copyright © 2026 Texas Instruments Incorporated

Table 12-160. USBSS Input/Output Description (continued)

Device Pin	Module Signal	I/O	Description	Value at Reset
USB1_REFCLKN		I	N of differential reference clock	HiZ
USB1_REFCLKP		I	P of differential reference clock	HiZ
USB1_REXT			SERDES reference resistor	

12.2.5.2.2 USB Subsystem Application

As the USB controller modules present in this device are DRD (Dual-Role Devices), they can support operation as either a USB Host or a USB Device. The operational mode determination is made based on the state of the USBn_ID pin; when this pin is grounded, the controller will operate as a USB Host, when this pin is left floating, the controller assumes the role of a USB Device. For implementations that do not require DRD functionality, the USBn_ID pin can either be left floating or can be grounded in the board design depending on the static role required. For implementations that do require DRD functionality, the USBn_ID pin should be connected directly to the corresponding ID pin on a USB Micro-AB socket. In doing so, the USBn_ID pin will be correctly terminated (open or grounded) depending on the cable attached and the controller will enter Host or Device mode accordingly.

12.2.5.2.3 VBUS Sense

The VBUS voltage needs to be monitored for detection of the various valid signals (AVALID, BVALID, and SESSEND).

The voltage of the VBUS signal is scaled down using an external resistor divider (as shown in Figure 12-1796), to limit the voltage applied to the VBUS pin. A zener diode is used to clamp the maximum intermediate voltage to 6.8 volts. The tolerance of resistors used must be less than 1%. The leakage current of the zener diode must be less than 100 nA at 5 V.

Copyright © 2026 Texas Instruments Incorporated

12.2.5.3 USB Functional Description

Note

This chapter serves to describe the integration of the third-party USB controller and should not be considered sufficient for those wishing to modify the existing Linux or RTOS USB driver(s) or create a new driver to support this controller implementation. For those who do wish to substantially modify the existing Linux or RTOS USB driver(s), or create new drivers, contact TI for more information on how to obtain the third-party documentation under NDA.

The USB3SS subsystem contains the USB3.0 Dual Role Device (DRD) controller module and a USB2.0 PHY module and is instantiated as USB1 in the device. A wrapper module is controlling some top-level functions like Host and Device role switch and reset release.

The USB controller uses USB2PHY for USB2.0 operation and SERDES0 for USB3.0 speeds.

12.2.5.3.1 USB Controller Reset

USB controller has three resets going in, called preset_n, pwrup_rst_n, and aresetn.

- preset_n comes from the mod_g_rst_n input coming from LPSC
- pwrup_rst_n is PWRUP_RST_N register bit in the USB wrapper module (USB3P0SS_W1). This reset has to be deasserted before accessing controller registers. The default value for pwrup_rst_n is 0 (reset asserted). This reset goes to the default asserted state when mod_g_rst_n reset is asserted.
- areset_n comes from PWRUP_RST_N register bit, but the deassertion is synchronized to ACLK.

12.2.5.3.2 Overcurrent Detection

The overcurrent detection circuit is external to the device. Software must specify an overcurrent condition to the controller whenever the overcurrent signal is received from the detection circuit. Software must set the OVERCURRENT_N bit to 0 in the USB3P0SS_W1 register when overcurrent was detected.

12.2.5.3.3 Top-Level Initialization Sequence

The following initialization sequence has to be followed before configuring the controller for USB operation:

1. Software must select and configure the SERDES for USB operation. Refer to *Serializer/Deserializer (SerDes)* for more details. USB2.0 PHY contains default configuration that normally does not need change
2. Software must configure the pseudo-static settings in USB3P0SS_W1 register
3. Software must set PWRUP_RST_N to 1 in order to deassert controller reset

After the above sequence, software can access controller registers.

12.3 Memory Interfaces

Copyright © 2026 Texas Instruments Incorporated

12.3.1 Flash Subsystem (FSS)

This section describes the Flash Subsystem (FSS) in the device.

12.3.1.1 FSS Overview

The Flash Subsystem (FSS) provides access to external Flash devices via Octal Serial Peripheral Interface (OSPI).

The FSS includes one OSPI. For more information, see *Octal Serial Peripheral Interface (OSPI)*.

Figure 12-137 shows the FSS overview.

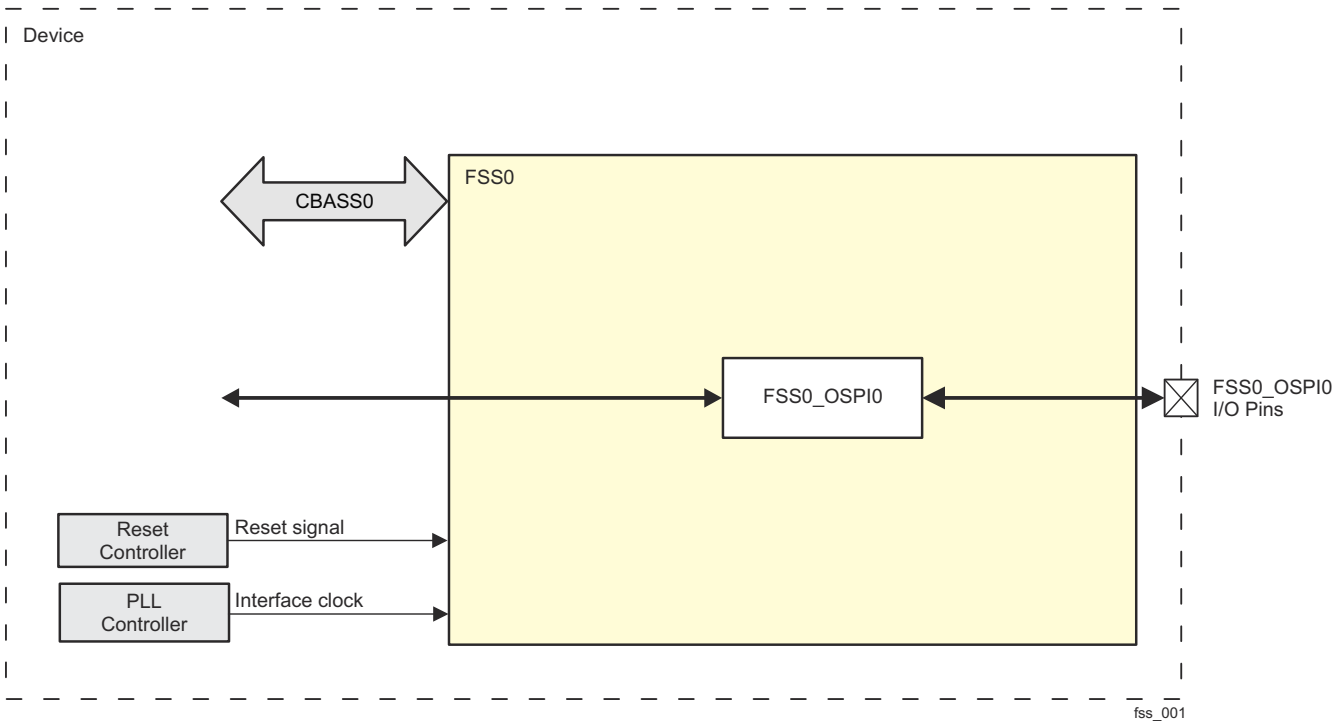


Figure 12-137. FSS Overview

12.3.1.1.1 FSS Features

For more information, see Section 12.3.2.1.1, *OSPI Features*.

12.3.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.3.1.2 FSS Environment

The FSS is hereinafter also referred to as FSS0.

12.3.1.2.1 FSS Typical Application

Figure 12-138 shows typical FSS application.

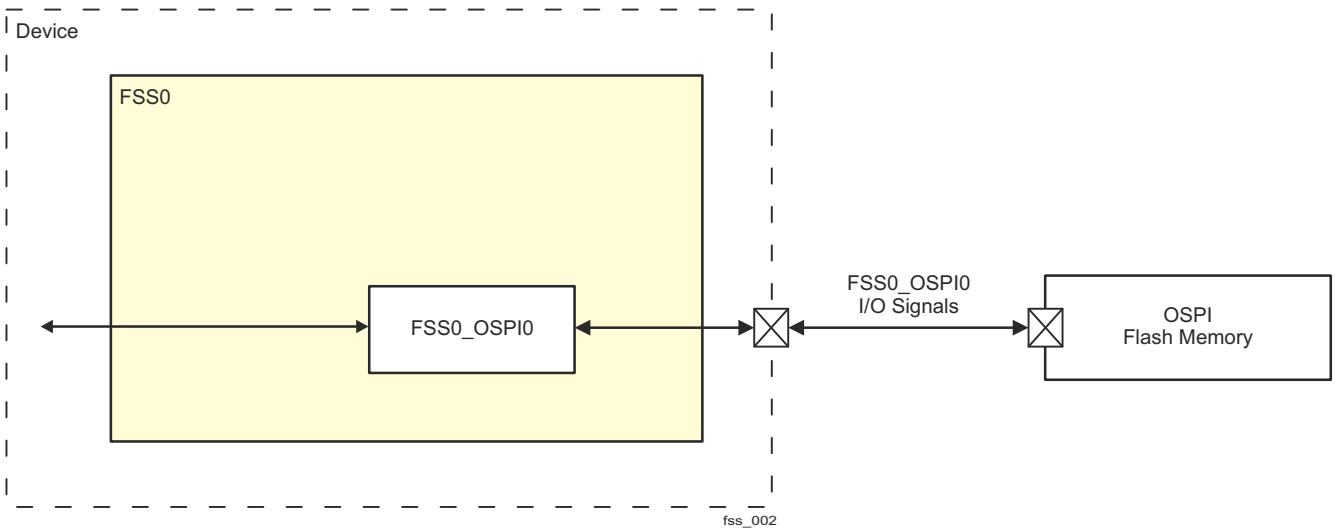


Figure 12-138. FSS Typical Application

Copyright © 2026 Texas Instruments Incorporated

Table 12-161 describes the FSS I/O signals.

Table 12-161. FSS I/O Signals

FSS0 Interface	I/O Signals
FSS0_OSPI0	For more information about OSPI I/O signals, see *OSPI I/O Signals*.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.3.1.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.1.4 FSS Functional Description

12.3.1.4.1 FSS Block Diagram

Figure 12-139 shows the FSS block diagram.

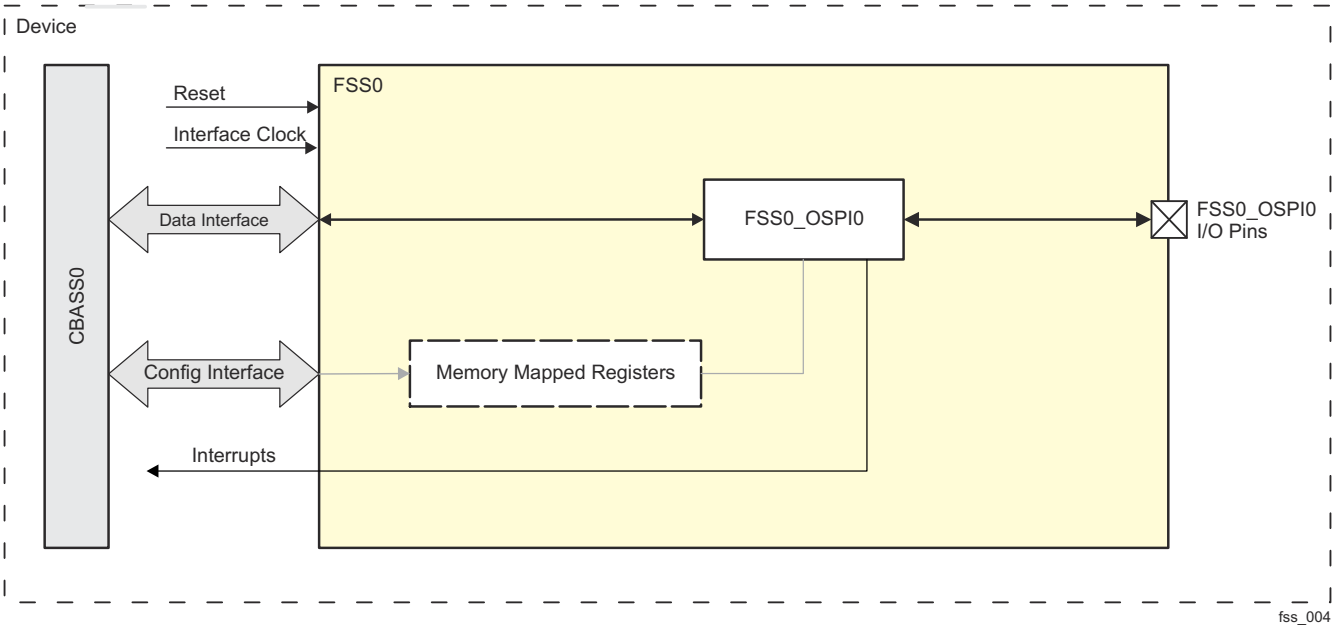


Figure 12-139. FSS Block Diagram

FSS Blocks:

- CBASS0: The CBASS0 interconnect allows FSS to communicate with the device modules and subsystems.
- Data Interface: It is 64-bit data/32-bit address multi issue data interface with coherent in-band bypass. It provides accessibility to the FSS0_OSPI0.
- Config Interface: It is used for configuration of the memory mapped registers within the FSS.
- Interface Clock and Reset:
  - For more information, see *FSS0 Clocks and Resets*.
  - For more information, see *FSS0_OSPI Clocks and Resets*.
- Interrupts: For more information, see *FSS0_OSPI Hardware Requests*.
- Memory Mapped Registers: This block includes the FSS registers. The configuration of these registers defines which FSS features are used. For more information, see *FSS Registers*.
- FSS0_OSPI0: For more information about OSPI, please see *Octal Serial Peripheral Interface (OSPI)*.
- FSS0_OSPI0 I/O Pins: For more information, see *OSPI I/O Signals*.

12.3.1.4.2 FSS Regions

12.3.1.4.2.1 FSS Regions Boot Size Configuration

The boot size for FSS defaults to 64 MB but can be configured to be 128 MB. Selection of boot block which will be used is also configurable. For more information see CTRLMMR_FSS_CTRL register in *Control Module (CTRL_MMR)*.

Copyright © 2026 Texas Instruments Incorporated

12.3.1.4.3 FSS Memory Regions

Table 12-162 shows the FSS memory regions.

Table 12-162. FSS Memory Regions

Address Range	Size	Description
0x400000000 - 0x4FFFFFFFF	4 GB	External Memory Space (Region 0)
0x060000000 - 0x067FFFFFF	128 MB	Boot Space (Region 1)
0x500000000 - 0x5FFFFFFFF	4 GB	External Memory Space (Region 3)

12.3.1.5 FSS Programming Guide

12.3.1.5.1 FSS Initialization Sequence

Initialization steps:

- Configure the main boot parameters for FSS (see Section 12.3.1.4.2.1):
  - Select the boot block to be used.
  - Select the size of the boot block to be used.
- Enable FSS in PSC.
- Configure the FSS0_OSPI0.
- Enable the FSS0_OSPI0 in PSC. For more information about OSPI configuration, please see *Octal Serial Peripheral Interface (OSPI)*.

12.3.1.5.2 FSS Power Up/Down Sequence

There are two PSC controls for the FSS: for FSS0 itself and for FSS0_OSPI0. The CPU enables the FSS0_OSPI0 before using the FSS. Software should ensure the FSS0_OSPI0 is enabled prior to FSS transactions.

Normal Power Down Sequence:

- Block any new transaction.
- Power down the FSS0_OSPI0.
- In the event when the FSS0_OSPI0 is powered down, the FSS can then be powered down.

Copyright © 2026 Texas Instruments Incorporated

12.3.2 Octal Serial Peripheral Interface (OSPI)

This section describes the Octal Serial Peripheral Interface (OSPI) module for the device.

12.3.2.1 OSPI Overview

The Octal Serial Peripheral Interface (OSPI) module is a kind of Serial Peripheral Interface (SPI) module which allows single, dual, quad or octal read and write access to external flash devices.

The OSPI module is used to transfer data, either in a memory mapped direct mode (for example a processor wishing to execute code directly from external flash memory), or in an indirect mode where the module is set-up to silently perform some requested operation, signaling its completion via interrupts or status registers. For indirect operations, data is transferred between system memory and external flash memory via an internal SRAM which is loaded for writes and unloaded for reads by a device controller at low latency system speeds. Interrupts or status registers are used to identify the specific times at which this SRAM should be accessed using user programmable configuration registers.

shows the OSPI allocation across device domains.

Figure 12-140 shows the OSPI module overview.

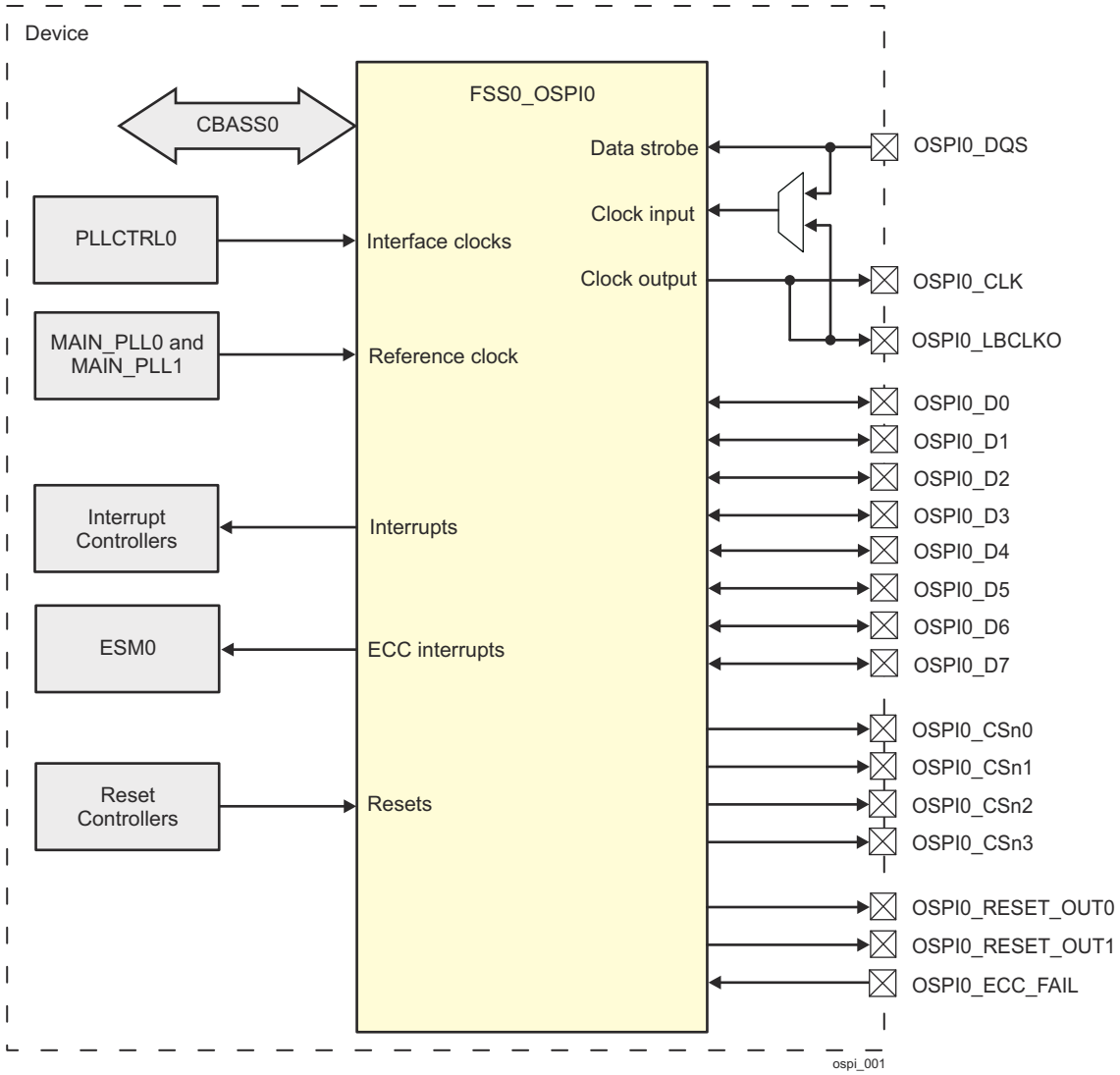


Figure 12-140. OSPI Overview

Copyright © 2026 Texas Instruments Incorporated

12.3.2.1.1 OSPI Features

The OSPI module has the following features:

- Support for single, dual, quad (QSPI mode) or octal I/O bus widths.
- Memory mapped ‘direct’ mode of operation for performing flash data transfers and executing code from flash memory.
- Software triggered 'indirect' mode of operation for performing low latency and non-processor intensive flash data transfers.
- Local SRAM of configurable size to reduce advanced high-performance bus overhead and buffer flash data during indirect transfers.
- Set of software advanced peripheral bus accessible flash control registers to perform any flash command, including data transfers up to 8-bytes at a time.
- Additional addressable memory bank to accommodate more than 8-bytes at a time.
- Support for XIP, sometimes referred to as continuous mode.
- Support for DDR Mode and DTR protocol (including Octal DDR protocol with DQS for Octal-SPI devices)
- Programmable device sizes.
- Programmable write protected regions to block system writes from taking effect.
- Programmable delays between transactions.
- Legacy mode allowing software direct access to low level transmit and receive FIFOs, bypassing the higher layer processes.
- An independent reference clock to decouple bus clock from SPI clock – allows slow system clocks.
- Programmable baud rate generator to generate OSPI clocks.
- Features included to improve high speed read data capture mechanism.
- Option to use adapted clocks or DQS to further improve read data capturing.
- Programmable interrupt generation.
- Up to four external device selects - OSPI and QSPI devices can be mixed
- Programmable data decoder, enables continuous addressing mode for each of the connected devices and auto-detection of boundaries between devices.
- Supports BOOT mode.
- Bidirectional CRC on Multiple-SPI interface.
- Handling ECC errors for flash devices with embedded correction engine.
- Full integration with PHY module dedicated to more flexible and power efficient transfers.
- Supports RESET_OUT[1-0] and ECC_FAIL pins for external flash devices where ECC is checked on the flash.

12.3.2.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.2 OSPI Environment

The FSS0_OSPI0 module is hereinafter referred to as OSPI module.

This section describes the OSPI external connections (environment).

The OSPI module is primarily intended for fast booting from Octal- and Quad-SPI flash memories. Figure 12-141 shows a typical connection of the OSPI module to an external Octal-SPI flash memory.

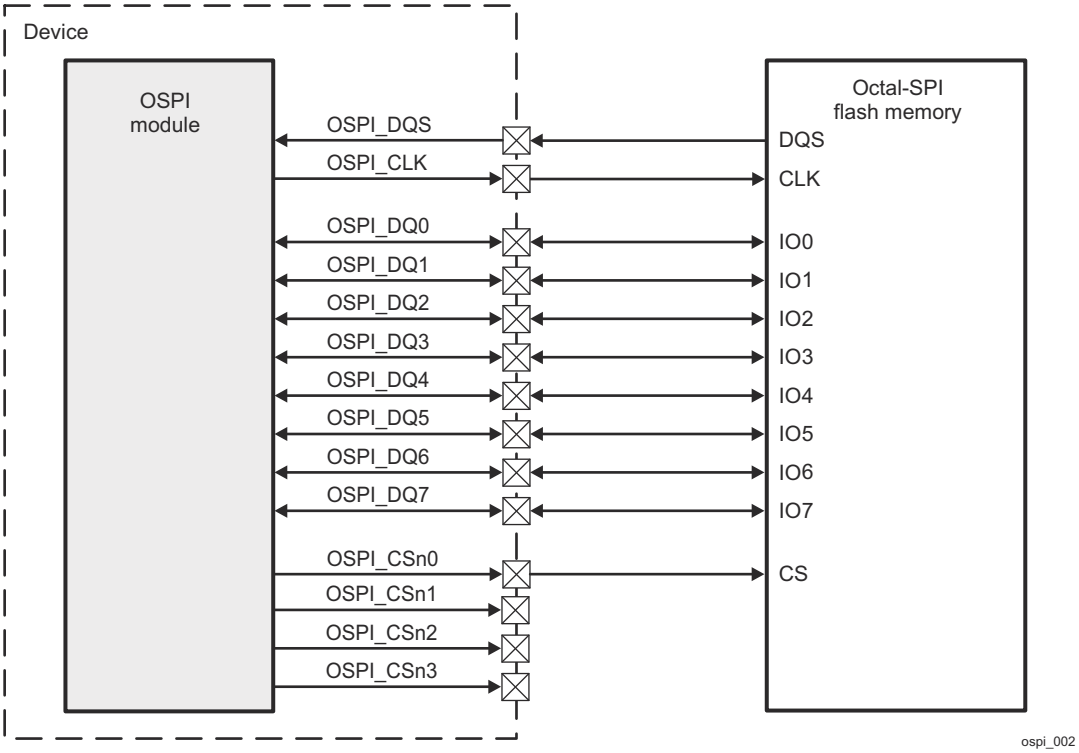


Figure 12-141. OSPI Connected to an External Octal-SPI Flash Memory

Table 12-163 lists and describes the FSS0_OSPI I/O signals.

Table 12-163. OSPI I/O Signals

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
			FSS0_OSPIi[4]	
DQ0	OSPIi[4]_D0	IO	FSS0_OSPIi[4] data input/output 0	HiZ
DQ1	OSPIi[4]_D1	IO	FSS0_OSPIi[4] data input/output 1	HiZ
DQ2	OSPIi[4]_D2	IO	FSS0_OSPIi[4] data input/output 2	HiZ
DQ3	OSPIi[4]_D3	IO	FSS0_OSPIi[4] data input/output 3	HiZ
DQ4	OSPIi[4]_D4	IO	FSS0_OSPIi[4] data input/output 4	HiZ
DQ5	OSPIi[4]_D5	IO	FSS0_OSPIi[4] data input/output 5	HiZ
DQ6	OSPIi[4]_D6	IO	FSS0_OSPIi[4] data input/output 6	HiZ
DQ7	OSPIi[4]_D7	IO	FSS0_OSPIi[4] data input/output 7	HiZ
N_SS_OUT0	OSPIi[4]_CSn0	O	FSS0_OSPIi[4] external flash device chip select 0	0x1
N_SS_OUT1	OSPIi[4]_CSn1	O	FSS0_OSPIi[4] external flash device chip select 1	0x1
N_SS_OUT2	OSPIi[4]_CSn2	O	FSS0_OSPIi[4] external flash device chip select 2	0x1
N_SS_OUT3	OSPIi[4]_CSn3	O	FSS0_OSPIi[4] external flash device chip select 3	0x1
OCLK	OSPIi[4]_CLK	O	FSS0_OSPIi[4] clock output for the external flash device	0x0
	OSPI0_LBCLKO	O	FSS0_OSPIi[4] external loopback output	0x0

Copyright © 2026 Texas Instruments Incorporated

Table 12-163. OSPI I/O Signals (continued)

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
DQS	OSPIi[4]_DQS	I[3]	FSS0_OSPIi[4] data strobe / external loopback input	Don't care
RESET_OUT0	OSPIi[4]_RESET_OUT0	O	FSS0_OSPIi[4] reset output 0 for the external flash device	0x1
RESET_OUT1	OSPIi[4]_RESET_OUT1	O	FSS0_OSPIi[4] reset output 1 for the external flash device	0x1
ECC_FAIL	OSPIi[4]_ECC_FAIL	I	FSS0_OSPIi[4] ECC status from the external flash device	0x1

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) When used as an external loopback input, the DQS signal can alternatively be referred to as LBCLKI. The LBCLKI clock input signal is a looped back version of the LBCLKO clock output signal and facilitates easier timing closure at higher speeds. The loopback has to be at board level in order to support higher OSPI speeds. The source of the loopback clock is defined by CTRLMMR_MCU_OSPIn_CLKSEL[4] LOOPCLK_SEL bit in *Control Module (CTRL_MMR)*.
(4) i represents an OSPI instance. See the device datasheet for available domains and OSPI instances.

Table 12-164 describes the OSPI I/O connectivity to external SPI devices.

Table 12-164. OSPI I/O Connectivity to External SPI Devices

Module Pin	I/O[1]	Description			
		4-pin[1] SPI - Single Read/Write (SIO) (DATA_XFER_TYPE_EXT_MODE_FLD=0x0)	4-pin[1] SPI - Dual Read/ Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x1)	6-pin[1] SPI - Quad Read/ Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x2)	11-pin[1] SPI - Octal Read/ Write (DATA_XFER_TYPE_EXT_MODE_FLD=0x3)
DQ0	IO	Used as SPI data output	Used as SPI data input 0 Used as SPI data output 0	Used as SPI data input 0 Used as SPI data output 0	Used as SPI data input 0 Used as SPI data output 0
DQ1	IO	Used as SPI data input	Used as SPI data input 1 Used as SPI data output 1	Used as SPI data input 1 Used as SPI data output 1	Used as SPI data input 1 Used as SPI data output 1
DQ2	IO	Not used	Not used	Used as SPI data input 2 Used as SPI data output 2	Used as SPI data input 2 Used as SPI data output 2
DQ3	IO	Not used	Not used	Used as SPI data input 3 Used as SPI data output 3	Used as SPI data input 3 Used as SPI data output 3
DQ4	IO	Not used	Not used	Not used	Used as SPI data input 4 Used as SPI data output 4
DQ5	IO	Not used	Not used	Not used	Used as SPI data input 5 Used as SPI data output 5
DQ6	IO	Not used	Not used	Not used	Used as SPI data input 6 Used as SPI data output 6
DQ7	IO	Not used	Not used	Not used	Used as SPI data input 7 Used as SPI data output 7
DQS	I[2]	Not used	Not used	Not used	Data strobe or loopback clock input
OCLK	O	Output clock or loopback clock output. For more information, see Table 12-163.			
N_SS_OUT0	O	External SPI device chip-select 0			
N_SS_OUT1	O	External SPI device chip-select 1			
N_SS_OUT2	O	External SPI device chip-select 2			
N_SS_OUT3	O	External SPI device chip-select 3			
RESET_OUT0	O	External SPI device reset 0			
RESET_OUT1	O	External SPI device reset 1			
ECC_FAIL	I	External SPI device ECC failure indication			

(1) This is the pin count at the external SPI flash memory side.
(2) When used as an external loopback input, the DQS signal can alternatively be referred to as LBCLKI. The LBCLKI clock input signal is a looped back version of the LBCLKO clock output signal and facilitates easier timing closure at higher speeds. The loopback has to be at board level in order to support higher OSPI speeds. The source of the loopback clock is defined by CTRLMMR_MCU_OSPIn_CLKSEL[4] LOOPCLK_SEL bit in *Control Module (CTRL_MMR)*.

Copyright © 2026 Texas Instruments Incorporated

Note

For OSPI0_CLK, OSPI0_LBCLKO, and OSPI0_DQS signals to work properly, the RXACTIVE bit of the appropriate registers should be set to 0x1 because of retiming purposes.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.3.2.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.4 OSPI Functional Description

12.3.2.4.1 OSPI Block Diagram

Figure 12-142 shows the OSPI module block diagram.

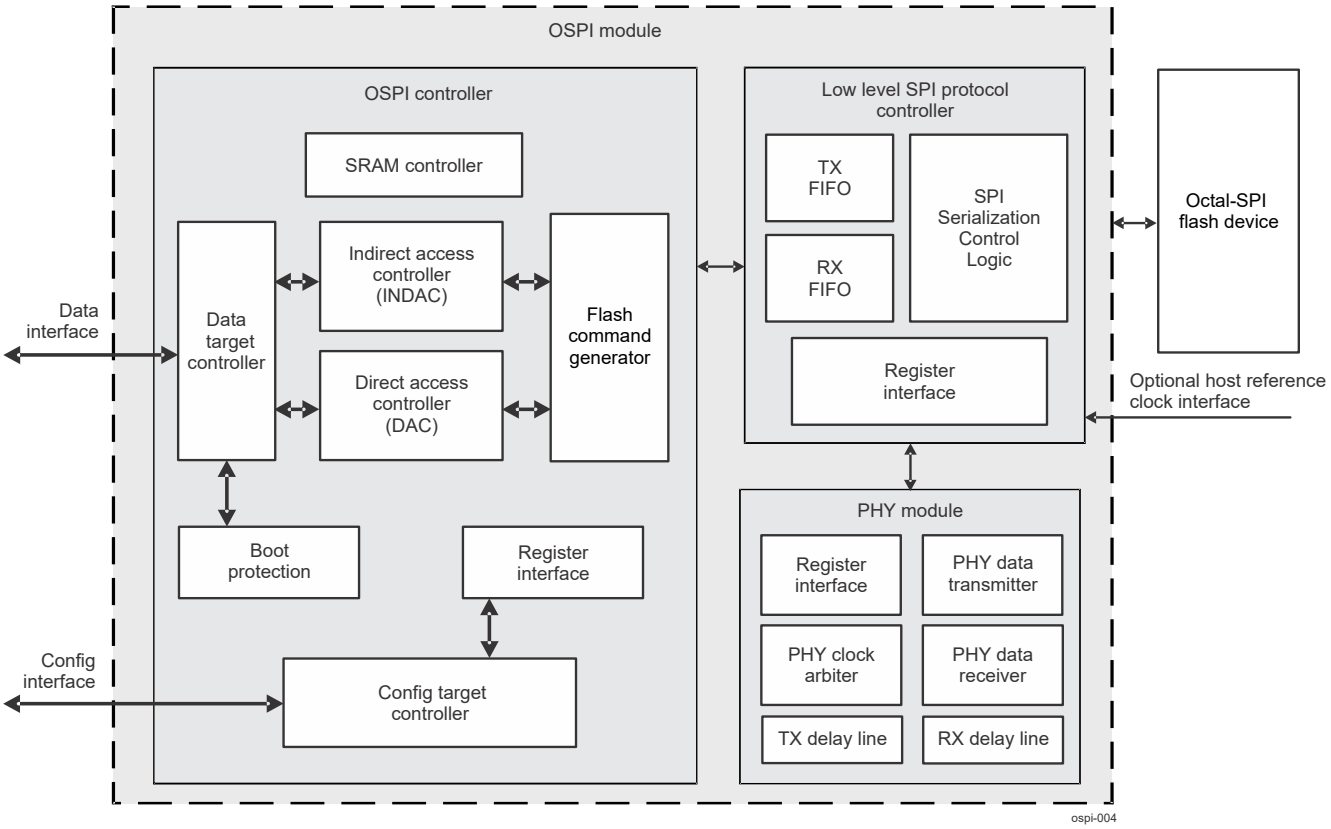


Figure 12-142. OSPI Block Diagram

The OSPI module is composed of three main blocks. The first one is the OSPI controller, the second one is the low level SPI protocol controller, and the third one is the integrated PHY.

The OSPI module has the following two target interfaces:

- Data target interface intended for data transfer.
- Configuration target interface intended for accessing the programmable set of registers.

12.3.2.4.1.1 Data Target Interface

The data interface is used for data transfer to external flash devices in direct and indirect mode of operation. The data target controller validates incoming data accesses, responds to invalid requests, performs any required byte and halfword reordering, blocks writes that violate the programmed write protection rules (only for direct access) and forwards the transfer request to either the direct access controller (DAC) or the indirect access controller (INDAC).

The data interface bus is 32-bits wide. Therefore only byte, halfword and word accesses are permitted. When the controller is configured to work in SPI Octal DDR Mode or Octal DDR Protocol (where 2 bytes are collected within single SPI clock cycle what exceeds the size of 1 byte transfer request), 8 bit transfer size is not allowed.

Note

Cache line wrap accesses over the data target port should be word aligned.

Data target port doesn't support cache line wrap bursts of 128 bytes.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.4.1.2 Configuration Target Interface

The configuration interface is used to configure the OSPI module and perform software controlled flash accesses using the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register (for more information refer to Section 12.3.2.4.11, *OSPI Software Triggered Instruction Generator (STIG)*). Depending on the address it routes the incoming interconnect transfer to the Low level SPI protocol controller or to the ECC aggregator. The configuration port is also used to interact with the OSPI configuration and SRAM ECC registers.

Note

The configuration interface supports only 32-bit accesses. For single byte or halfword manipulations software should perform read-modify-write operations.

12.3.2.4.1.3 OSPI Clock Domains

The OSPI module has two main clock sources for the Octal-SPI controller.

- For interface clocks
- For reference clock

The source for the interface clocks corresponds to the configuration and data buses. The data bus clock (OSPI_HCLK) is the main system clock used to transfer data over the data bus between a controller on the system interconnect and the OSPI module. The data bus clock also drives the internal OSPI SRAM. The configuration bus clock (OSPI_PCLK) is used to access the OSPI configuration register and perform basic configuration and for interrupt handling. The OSPI reference clock (OSPI_RCLK) drives the SPI transmit and receive logic in the OSPI module. It is also used to generate the output SPI protocol clock (OSPI_OCLK) and for oversampling of the input data. Using the reference clock (OSPI_RCLK) allows the OSPI module to decouple the frequency of the SPI flash device from the device system clocks, thereby providing more flexible clocking solution.

Note

There is no particular clock ratio requirement between configuration (OSPI_PCLK) and data bus (OSPI_HCLK) clocks.

12.3.2.4.2 OSPI Modes

Note

Some of the OSPI features described in this section may not be supported on this family of devices. For more information, see *OSPI Not Supported Features*.

The OSPI module supports four SPI modes. These modes are defined through the OSPI_FLASH_CFG_CONFIG_REG[1] SEL_CLK_POL_FLD and OSPI_FLASH_CFG_CONFIG_REG[2] SEL_CLK_PHASE_FLD bits. The SEL_CLK_POL_FLD bit defines the clock polarity and the SEL_CLK_PHASE_FLD bit defines the data launch and data capture relation to the OSPI clock edges. Table 12-165 gives a brief description of these modes.

Table 12-165. OSPI Modes

SPI Mode	SEL_CLK_POL_FLD	SEL_CLK_PHASE_FLD	Description
0	0	0	Clock inactive state: low
			Data launch edge: clock falling edge
			Data capture edge: clock rising edge
1	0	1	Clock inactive state: low
			Data launch edge: clock rising edge
			Data capture edge: clock falling edge

Copyright © 2026 Texas Instruments Incorporated

Table 12-165. OSPI Modes (continued)

SPI Mode	SEL_CLK_POL_FLD	SEL_CLK_PHASE_FLD	Description
2	1	0	Clock inactive state: high
			Data launch edge: clock rising edge
			Data capture edge: clock falling edge
3	1	1	Clock inactive state: high
			Data launch edge: clock falling edge
			Data capture edge: clock rising edge

Octal flash devices provide DQS signal which allows source synchronous capture, but for Quad flash devices the OSPI module has a loopback mode. In this loopback mode the clock, looped back at board level, is used for registering the input data, and the edge used is same as the launch edge, thus giving a full cycle path (for more information, see Section 12.3.2.4.2.1, *Read Data Capture*).

12.3.2.4.2.1 Read Data Capture

Figure 12-143 shows the Read Data Capture Logic in the OSPI module.

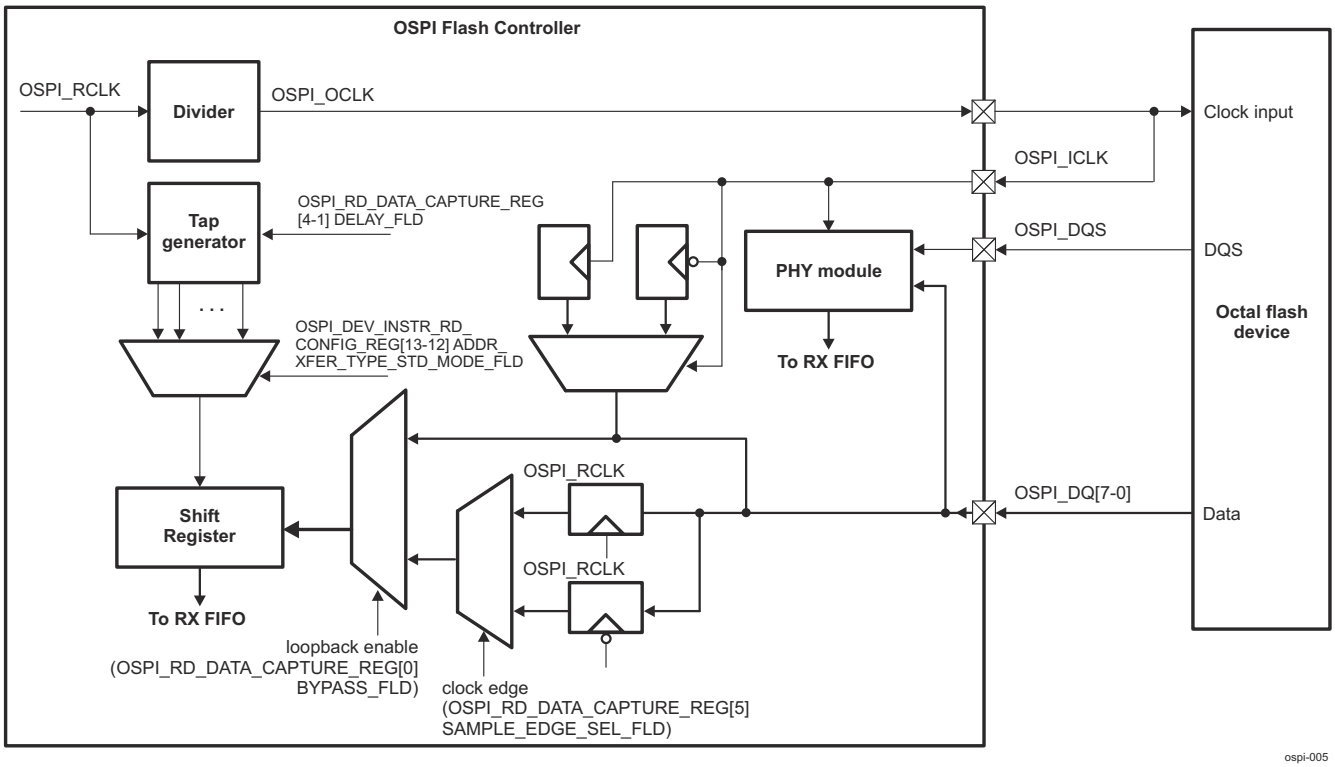


Figure 12-143. Read Data Capture Logic

The PHY module includes a DLL which allows adjustment of the sampling edge with respect to the incoming data to achieve maximum frequency. There are three sources for the sampling signal:

- The reference clock
- Output SPI clock external loopback
- The DQS (only available in Octal Flash devices)

The loopback mode (only for Quad flash devices) can work in two cases. The first one is when OSPI_FLASH_CFG_CONFIG_REG[2] SEL_CLK_PHASE_FLD=0. When SEL_CLK_PHASE_FLD=1 there aren't enough clock falling edges for the register pipeline to catch the last data driven, thus causing a functional failure. Additionally, since the capture edge is falling edge, it gives a full cycle input path only in SPI mode 0, that

Copyright © 2026 Texas Instruments Incorporated

is when SEL_CLK_POL_FLD=0 and SEL_CLK_PHASE_FLD=0. Thus SPI mode 0 is the first of two modes that support high MHz operation (greater than 50 MHz). The second mode is when SEL_CLK_PHASE_FLD=1 and SEL_CLK_PHASE_FLD=1 (SPI mode 3). In this case the missing clock falling edge is compensated inside the OSPI controller when using the incorporated PHY module by inverting the loopback clock.

The loopback mode is enabled by writing 0x0 to OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[0] BYPASS_FLD. The taps are selected by programming OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[4-1] DELAY_FLD field. The taps delay the read data capturing logic by the programmed number of OSPI_RCLK cycles.

12.3.2.4.2.1.1 Mechanisms of Data Capturing

There are two mechanisms of data capturing in the OSPI module. They can be combined in some parts to ensure reliable sampling solution independent on the system requirements and the controller configuration. The mechanisms are as follows:

- Data capturing mechanism using taps
- Data capturing mechanism using PHY module.

12.3.2.4.2.1.2 Data Capturing Mechanism Using Taps

This section describes the data capturing mechanism where sampling point is adjusted for one of the reference clock edges inside divided OSPI clock.

After POR, the adapted loopback clock circuit and the OSPI_RCLK delay register line both wake in a disabled state. The OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG register provides the control for the mechanism using taps.

OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[5] SAMPLE_EDGE_SEL_FLD bit selects the edge of the reference clock, on which data outputs from flash memory are sampled.

OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[4-1] DELAY_FLD bit field controls the additional number of read data capture cycles (this is the fast reference clock, running at least x4 of the device clock) that should be applied to the internal read data capture circuit. The large clock-to-out delay of the flash memory together with trace delays as well as other device delays may impose a maximum flash clock frequency which is less than the flash memory device itself can operate at. To compensate, software shall set this register to a value that guarantees robust data captures.

12.3.2.4.2.1.3 Data Capturing Mechanism Using PHY Module

PHY module is responsible for data capturing. More detailed description of all internal PHY sampling mechanisms is included in *Read Data Capturing by the PHY Module*.

12.3.2.4.2.1.4 External Pull Down on DQS

Per the OSPI protocol, the FLASH device drives DQS while CS is asserted. When CS is not asserted the FLASH device presents HiZ on DQS. When configured to use DQS, the controller uses the DQS as a clock, which samples the incoming data into a FIFO. Noise on the DQS when it is HiZ can cause spurious false triggering of the FIFO and filling it with invalid data. There is no way to clear this data except to reset the OSPI module.

To avoid this issue, it is recommended to add a pull down on the DQS line.

During device wakeup, before the IO ring is configured properly, the CS to the FLASH device is HiZ. Depending on the actual level of the CS line the FLASH device might drive the DQS High, Low or HiZ. A pull down on DQS forces the DQS input to Low, but the DQS might still be High or in the presence of noise there might be transitions between Low and High. This again can cause the same issue of capturing garbage data in the Controller FIFO.

To avoid this issue it is recommended to release the OSPI from reset only after the IO ring is configured properly.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.4.3 OSPI Power Management

Note

The OSPI module does not provide any hardware signal for busy or idle status. Software need to ensure that the OSPI module is idle before clocks can be shut off by reading the OSPI_FLASH_CFG_CONFIG_REG[31] IDLE_FLD bit.

OSPI_PCLK and OSPI_HCLK share the same clock stop request/acknowledge and clock enable/acknowledge interface.

12.3.2.4.4 Auto HW Polling

The OSPI controller is capable of automatically testing the Flash device busy bit to guarantee no reads or writes are ignored by the flash when it is busy burning in programmed data.

At the end of a programming transaction, the Flash device goes into a burn-in state and becomes busy.

When Auto HW Polling is enabled, the OSPI controller keeps track of programming transactions and will initiate a Flash status read polling transactions automatically, until Flash indicates it is not busy, before any additional data read or programming operations are sent to the flash device. See OSPI_FLASH_CFG_WRITE_COMPLETION_CTRL_REG register and the associated registers.

The OSPI controller requires that the OSPI_FLASH_CFG_WRITE_COMPLETION_CTRL_REG[23-16] POLL_COUNT_FLD field should always be set with values greater or equal to 3 (>=3).

12.3.2.4.5 Flash Reset

OSPI provides Flash reset out ports. These ports are active low and controlled thru OSPI_FLASH_CFG_CONFIG_REG register.

12.3.2.4.6 OSPI Memory Regions

Note

For more information about the memory space, see *FSS Memory Regions*.

12.3.2.4.7 OSPI Interrupt Requests

The OSPI module generates three interrupts. The ECC interrupts (FSS0_OSPI_0_OSPI_ECC_CORR_LVL_INTR_0 and FSS0_OSPI_0_OSPI_ECC_UNCORR_LVL_INTR_0) are generated by the OSPI ECC aggregator.

The other interrupt (FSS0_OSPI_0_OSPI_LVL_INTR_0) is generated by the OSPI module.

Table 12-166 lists the event flags and the corresponding mask bits of the sources which can cause interrupts.

Copyright © 2026 Texas Instruments Incorporated

Table 12-166. OSPI Events

Event Flag	Event Mask	Description
OSPI_FLASH_CFG_IRQ_STATUS_REG[0] MODE_M_FAIL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[0] MODE_M_FAIL_MASK_FLD	Event Flag and Event Mask for the OSPI Interrupts.
OSPI_FLASH_CFG_IRQ_STATUS_REG[1] UNDERFLOW_DET_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[1] UNDERFLOW_DET_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[2] INDIRECT_OP_DONE_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[2] INDIRECT_OP_DONE_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[3] INDIRECT_READ_REJECT_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[3] INDIRECT_READ_REJECT_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[4] PROT_WR_ATTEMPT_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[4] PROT_WR_ATTEMPT_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[5] ILLEGAL_ACCESS_DET_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[5] ILLEGAL_ACCESS_DET_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[6] INDIRECT_XFER_LEVEL_BREACH_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[6] INDIRECT_XFER_LEVEL_BREACH_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[7] RECV_OVERFLOW_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[7] RECV_OVERFLOW_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[8] TX_FIFO_NOT_FULL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[8] TX_FIFO_NOT_FULL_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[9] TX_FIFO_FULL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[9] TX_FIFO_FULL_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[10] RX_FIFO_NOT_EMPTY_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[10] RX_FIFO_NOT_EMPTY_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[11] RX_FIFO_FULL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[11] RX_FIFO_FULL_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[12] INDRD_SRAM_FULL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[12] INDRD_SRAM_FULL_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[13] POLL_EXP_INT_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[13] POLL_EXP_INT_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[14] STIG_REQ_INT_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[14] STIG_REQ_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[16] RX_CRC_DATA_ERR_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[16] RX_CRC_DATA_ERR_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[17] RX_CRC_DATA_VAL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[17] RX_CRC_DATA_VAL_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[18] TX_CRC_CHUNK_BRK_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[18] TX_CRC_CHUNK_BRK_MASK_FLD	
OSPI_FLASH_CFG_IRQ_STATUS_REG[19] ECC_FAIL_FLD	OSPI_FLASH_CFG_IRQ_MASK_REG[19] ECC_FAIL_MASK_FLD	

Copyright © 2026 Texas Instruments Incorporated

Table 12-166. OSPI Events (continued)

Event Flag	Event Mask	Description
OSPI_ECC_SEC_STATUS_REG0[0] SRAM_PEND	OSPI_ECC_SEC_ENABLE_SET_REG0[0] SRAM_ENABLE_SET OSPI_ECC_SEC_ENABLE_CLR_REG0[0] SRAM_ENABLE_CLR	Event Flag and Event Mask for the ECC Interrupts.
OSPI_ECC_DED_STATUS_REG0[0] SRAM_PEND	OSPI_ECC_DED_ENABLE_SET_REG0[0] SRAM_ENABLE_SET OSPI_ECC_DED_ENABLE_CLR_REG0[0] SRAM_ENABLE_CLR	
OSPI_ECC_AGGR_STATUS_SET[1-0] PARITY	OSPI_ECC_AGGR_ENABLE_SET[0] PARITY	
OSPI_ECC_AGGR_STATUS_SET[3-2] TIMEOUT	OSPI_ECC_AGGR_ENABLE_SET[1] TIMEOUT	
OSPI_ECC_AGGR_STATUS_CLR[1-0] PARITY	OSPI_ECC_AGGR_ENABLE_CLR[0] PARITY	
OSPI_ECC_AGGR_STATUS_CLR[3-2] TIMEOUT	OSPI_ECC_AGGR_ENABLE_CLR[1] TIMEOUT	

12.3.2.4.8 OSPI Data Interface

12.3.2.4.8.1 Data Interface Address Remapping

The incoming data interface address, by default, maps directly to the address sent serially to the FLASH device. If the FLASH device has a 24-bit address, then the 24 LSB's of the data address is forwarded. A remap feature is available to remap all incoming data addresses to ADDRESS + N, where N is the value stored in the OSPI_FLASH_CFG_REMAP_ADDR_REG[31-0] VALUE_FLD bit field. It is enabled via the OSPI_FLASH_CFG_CONFIG_REG[16] ENB_AHB_ADDR_REMAP_FLD bit. This feature could be used when software needs to move boot code to another FLASH region.

12.3.2.4.8.2 Write Protection

In order to protect the FLASH device, a software controlled write protection feature is supported. Any data write detected (by using DAC), pointing to an area of the FLASH that is protected, is not permitted.

A programmable region of the FLASH device, defined as a number of FLASH 'blocks' starting from a particular block number can be protected. Three programmable registers are provided. The first OSPI_FLASH_CFG_LOWER_WR_PROT_REG register defines the FLASH block that is located at the bottom of the region to be protected. The second OSPI_FLASH_CFG_UPPER_WR_PROT_REG register defines the FLASH block that is located at the top of the region to be protected. The third OSPI_FLASH_CFG_WR_PROT_CTRL_REG register is a control register consisting of 2 bits. The OSPI_FLASH_CFG_WR_PROT_CTRL_REG[0] INV_FLD bit allows software to invert the region that is being protected, causing the programmed region to become the only areas of FLASH memory that is not protected from writes. The OSPI_FLASH_CFG_WR_PROT_CTRL_REG[1] ENB_FLD bit is the write protection enable bit. When this bit is set to 0, the FLASH device is unprotected.

For implementation, the data interface must map the incoming address into its associated FLASH block. A block can be between 1 and 65 KB, programmed via the OSPI_FLASH_CFG_DEV_SIZE_CONFIG_REG register.

12.3.2.4.8.3 Access Forwarding

For legal accesses, the data interface will forward all accesses to one of two access controllers - the direct access and the indirect access controllers. Assuming DAC has been enabled via the OSPI_FLASH_CFG_CONFIG_REG[7] ENB_DIR_ACC_CTRL_FLD bit, then by default all accesses will be forwarded to this controller. Before any accesses can be forwarded to INDAC, it must first be configured by software. This process is fully explained in Section 12.3.2.4.10, *OSPI Indirect Controller (INDAC)*. If DAC is

Copyright © 2026 Texas Instruments Incorporated

disabled, any incoming access that cannot be forwarded to INDAC will be completed immediately with an error. If DAC is enabled, the same access will be forwarded and serviced by DAC.

12.3.2.4.9 OSPI Direct Access Controller (DAC)

Direct access refers to the operation where data interface accesses directly trigger a read or write to FLASH memory. It is memory mapped and can be used to both access and directly execute code from external FLASH memory. Any incoming access that is not recognized as being within the programmable indirect trigger region is assumed to be a direct access and will be serviced by the DAC. Note that accesses that use DAC do not use the embedded SRAM. The data transfer stops when read or write burst is carried out. The amount of wait states applied will be dependent on the latency through the controller. Latency is kept to a minimum when the use of XIP read instructions are enabled (see OSPI_FLASH_CFG_CONFIG_REG[18] ENTER_XIP_MODE_IMM_FLD and OSPI_FLASH_CFG_CONFIG_REG[17] ENTER_XIP_MODE_FLD bits).

12.3.2.4.10 OSPI Indirect Access Controller (INDAC)

12.3.2.4.10.1 Indirect Read Controller

The aim of the indirect mode of operation is to read significant numbers of bytes from FLASH memory without requiring a data interface access to trigger it. Instead indirect operations are controlled and triggered by software via specific control/configuration Indirect Read Transfer registers (OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG, OSPI_FLASH_CFG_INDIRECT_READ_XFER_WATERMARK_REG, OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG, and OSPI_FLASH_CFG_INDIRECT_READ_XFER_NUM_BYTES_REG). This block will communicate with an embedded low level SPI protocol state machine module to perform an efficient and optimized FLASH read burst, placing the read data into the local SRAM module ready for fast and low latency delivery to any external controller.

By default, the Indirect Read controller is disabled. Before enabling it, software must configure how much data is required and the start address. The start address and total number of bytes to be fetched is defined in OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG and OSPI_FLASH_CFG_INDIRECT_READ_XFER_NUM_BYTES_REG registers, respectively. Up to two indirect operations can be programmed at any one time. The second operation can be triggered while the first is in progress. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. For more information refer to Section 12.3.2.4.10.3, *Indirect Access Queuing*.

The total number of bytes to read in an indirect operation is not limited by the size of the SRAM. The size of SRAM will only limit the size of requests. In the case of SRAM overrun, the controller will back pressure FLASH reads until space becomes available in the SRAM. Back pressuring the reads on the SPI interface is handled by completing any current read burst, waiting until space in the SRAM becomes available and then issuing a new read burst at the address where the previous terminated burst ended.

An external controller will be able to fetch the data that the controller has read from external FLASH memory by issuing data interface reads to the OSPI module. The address of the incoming read access must be in the range of indirect trigger address programmed via the OSPI_FLASH_CFG_IND_AHB_ADDR_TRIGGER_REG register to indirect trigger address + 2**(indirect trigger address range) - 1. Default value of the range is equal to 16 locations. This allows a 16-beat burst to be applied starting from the indirect trigger address. The smaller bursts are possible to handle effectively as well with this approach. Furthermore it is not strict requirement to push consecutive address sequence. Actual address just has to be in the indirect range to grant SRAM as source. Each valid Indirect Read will cause the internal SRAM to be popped, thereby decoupling the incoming read access address from the FLASH address – that is not direct mapped. Therefore the indirect trigger address does not have any relationship with the FLASH address. It is just to indicate that data should take the SRAM as source instead of the FLASH memory array after triggering of any valid Indirect Read. The FLASH address for Indirect Read is taken from the OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG register. Assuming the requested data is present in the SRAM at the point the data interface access is received by the OSPI module, then the data will be fetched from the SRAM and the response to the read burst will be achieved with

Copyright © 2026 Texas Instruments Incorporated

minimum latency. Once the data has been read from the SRAM, the OSPI module will free up the associated resource in the SRAM.

If a read access is received whose address is not within the range described above then that access will not be completed using the indirect controller. It will instead be serviced by the direct access controller.

If a read access is received whose address is within the range described above but the requested data is not immediately present in the SRAM then wait states will be applied until the data has been read from FLASH and pushed to the SRAM.

If a read burst is received whose access elements traverse the Indirect trigger range, then the accesses within the Indirect trigger range will be processed by the indirect controller and the rest will be taken by the direct access controller. This is likely to be a software configuration error.

The external controller is only permitted to issue 32-bit data interface reads until the last word of an indirect transfer. This helps keep the SRAM control logic less complex. On the final read, the external controller may issue a 16-bit (Halfword) or byte access to complete the transfer. It is also permitted for the external controller to always issue a 32-bit Word read on the last indirect access. The controller will pad the upper bits of the response with zero. The current expectation is that the SRAM will be kept fairly full while the read operation is carried out. The fill level of the SRAM is directly readable by software reading the OSPI_FLASH_CFG_SRAM_FILL_REG register.

An indirect operation may be cancelled at any time by setting 1 to OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[1] CANCEL_FLD bit.

Any bus controller should be allowed to initiate an indirect access. The OSPI module provide software access mechanism to the SRAM fill-level directly via configuration registers and then decide for itself when the data should be fetched from the local SRAM. The fill level watermark register (see OSPI_FLASH_CFG_INDIRECT_READ_XFER_WATERMARK_REG register) is provided. When the SRAM fill level passes this watermark, an interrupt is generated. If the watermark value is > 0, the watermark interrupt is also generated when the final byte of data has been read by the OSPI module and placed in the SRAM, even if the actual SRAM fill level has not risen above the watermark. This last feature is useful to avoid software tracking how much data has been read and resetting the watermark value for the last few bytes of an indirect read transfer.

Two further interrupt sources are provided to help understand the status of an indirect operation. Firstly, an interrupt is generated when an indirect operation has completed. Secondly, an interrupt is generated if an Indirect Read operation was requested but could not be accepted due to the fact 2 indirect operations have already been buffered by the OSPI module.

Setting the OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[0] START_FLD bit starts an indirect read operation. OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[2] RD_STATUS_FLD bit is available to check the status.

12.3.2.4.10.1.1 Indirect Read Transfer Process

The following sequence can be followed:

1. Setup OSPI_FLASH_CFG_CONFIG_REG register.
2. Setup the indirect transfer's FLASH start address in the OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG register.
3. Setup the number of bytes to be transferred in the OSPI_FLASH_CFG_INDIRECT_READ_XFER_NUM_BYTES_REG register.
4. Setup the indirect transfer's trigger address in the OSPI_FLASH_CFG_IND_AHB_ADDR_TRIGGER_REG register.
5. Setup the indirect transfer's trigger address range in the OSPI_FLASH_CFG_INDIRECT_TRIGGER_ADDR_RANGE_REG register.
6. If the watermark interrupt feature is to be used, set the OSPI_FLASH_CFG_INDIRECT_READ_XFER_WATERMARK_REG register which will cause an interrupt to

Copyright © 2026 Texas Instruments Incorporated

be generated when the fill level increases beyond the watermark level. Setting the watermark can be useful indication to software when to read the next part of the indirect read transfer. Note that if the watermark is set to a value other than zero, the watermark interrupt will always trigger once the final byte of indirect transfer has been fetched and placed in the embedded SRAM, even if the watermark value is higher than the actual completed fill level.

7. Trigger Indirect Read access by setting the OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[0] START_FLD bit to 1.
8. If the watermark interrupt feature is to be used, wait for watermark interrupt. Else poll the SRAM fill level via the OSPI_FLASH_CFG_SRAM_FILL_REG register to decide when sufficient data is in the SRAM to trigger data fetches.
9. Read the expected amount of data from SRAM. If there is still more data to fetch in order to complete the indirect read transfer, then loop back to step 8. Otherwise continue to step 10.
10. The completion status of the Indirect Read operation can be polled via the OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[5] IND_OPS_DONE_STATUS_FLD bit.
11. An Indirect Complete interrupt will be generated when the Indirect read operation has completed.

12.3.2.4.10.2 Indirect Write Controller

The aim of the indirect mode of operation is to perform bulk transfer of data from the processor into a FLASH memory in the most efficient manner. The fewest possible write cycles inside the FLASH device will be carried out for the indirect transfer, thus maximizing the life of the device. Indirect write operation can be thought of from a software perspective as the inverse of the indirect read. It is controlled and triggered by software via specific control/configuration Indirect Write Transfer registers (for more information see the following registers: OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG, OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_WATERMARK_REG, OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_START_REG, and OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_NUM_BYTES_REG). This block will await delivery of the write data via the external data interface controller, placing it in the local SRAM before communicating with the existing legacy SPI core to perform an efficient and optimized FLASH write burst.

By default, the indirect write controller is disabled. Before enabling it, the software must configure how much data is required and the start address. The start address and total number of bytes to be written is defined in OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_START_REG and OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_NUM_BYTES_REG registers, respectively. Up to two indirect operations can be programmed at any one time. The second operation can be triggered while the first is in progress. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. The Indirect write queuing is very similar to indirect read queuing. For more information refer to Section 12.3.2.4.10.3, *Indirect Access Queuing*.

The total number of bytes to write in an indirect operation is not limited by the size of the SRAM. The size of SRAM will only limit the amount of data that can be accepted from the external controller. In the case of an SRAM overrun, the controller will back pressure the data interface with wait states. Note the fill level of the SRAM is readable via programmable OSPI_FLASH_CFG_SRAM_FILL_REG register and this can be used to avoid this situation.

An external controller will provide the write data and will transfer this to the OSPI module by issuing data interface writes. The address of the incoming write access must be in the range of Indirect trigger address programmed via the OSPI_FLASH_CFG_IND_AHB_ADDR_TRIGGER_REG register to Indirect trigger address + 2**(Indirect trigger address range) - 1. Default value of the range is equal to 16 locations. This allows a 16-beat burst to be applied starting from the Indirect trigger address. The smaller bursts are possible to handle effectively as well with this approach. Furthermore it is not strict requirement to push consecutive address sequence. Actual address just has to be in the Indirect Range to grant SRAM as source. Each write will cause the internal SRAM to be pushed, thereby decoupling the incoming write access address from the FLASH address – that is not direct mapped. Therefore Indirect trigger address does not have any relationship with FLASH address. It is just to indicate that data should take SRAM as source instead of FLASH Memory array after triggering of any valid Indirect Write. The FLASH address for Indirect Write is taken from the

Copyright © 2026 Texas Instruments Incorporated

OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_START_REG register. Assuming the SRAM is not full at the point the data interface access is received by the OSPI module, then the data will be pushed to the SRAM with minimum latency.

If a write access is received whose address is not within the range described above then that access will not be completed using the indirect controller. It will instead be serviced by the direct access controller.

If a write access is received whose address is within the range described above but the SRAM is full then wait states will be applied until some or all of the data has been pushed from the SRAM to the FLASH.

If a write burst is received whose access elements traverse the Indirect trigger range, then the accesses within the Indirect trigger range will be processed by the indirect controller, and the rest will be taken by the direct access controller. This is likely to be a software configuration error.

The external controller is only permitted to issue 32-bit data interface writes until the last word of an indirect transfer. This helps keep the SRAM control logic less complex. On the final write, the external controller may issue a 32-bit word, 16-bit (halfword) or a byte access to complete the transfer. If the number of bytes to write is less than 4 on the last transfer, the controller is still permitted to issue a 32-bit transfer. In these cases, the extra bytes are discarded by the controller.

When the SRAM holds a number of bytes equal to or greater than the size of a FLASH page (which itself is programmed into the OSPI module, with a default of 256 bytes) or when the SRAM holds all remaining bytes of the currently executing indirect transfer, the OSPI module will initiate a write burst to the flash command generator.

An indirect operation may be cancelled at any time by setting 1 to the OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[1] CANCEL_FLD bit.

Any bus controller should be allowed to initiate an indirect access. The OSPI module provide software access mechanism to the SRAM fill-level directly via the configuration registers and then decide for itself when the data should be written to the local SRAM. The fill level watermark register (see OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_WATERMARK_REG register) is provided. When the SRAM fill level falls below this watermark, an interrupt is generated.

Two further interrupt sources are provided to help understand the status of an indirect operation. Firstly, an interrupt is generated when an indirect operation has completed. Secondly, an interrupt is generated if an indirect write operation was requested but could not be accepted due to the fact 2 indirect operations have already been buffered by the OSPI module.

Setting the OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[0] START_FLD bit starts an indirect write operation. The OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[2] WR_STATUS_FLD bit is available to check the status.

12.3.2.4.10.2.1 Indirect Write Transfer Process

The following sequence can be followed:

1. Setup OSPI_FLASH_CFG_CONFIG_REG register.
2. Setup the indirect transfer's FLASH start address in the OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_START_REG register.
3. Setup the number of bytes to be transferred in the OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_NUM_BYTES_REG register.
4. Setup the indirect transfer's trigger address in the OSPI_FLASH_CFG_IND_AHB_ADDR_TRIGGER_REG register.
5. Setup the indirect transfer's trigger address range in the OSPI_FLASH_CFG_INDIRECT_TRIGGER_ADDR_RANGE_REG register.
6. It is functionally valid for software to simply write all the data to the SRAM in one block transfer. However, if the total number of bytes to write is greater than the size of the partitioned SRAM, then it is quite likely the SRAM will become full causing the OSPI to back-pressure the system data bus for a considerable time. This time is based on the FLASH data-rate and the page-write time of the device. To avoid sending all the write

Copyright © 2026 Texas Instruments Incorporated

data in one block transfer, software can make use of the watermark interrupt to identify a convenient time to send data a page at a time to the SRAM module. Alternatively, software can poll the SRAM fill level register directly to identify how empty the SRAM is at any one time in order to make a judgment as to when the most practical time to send the next part of the transfer.

7. If the watermark interrupt feature is to be used, set the OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_WATERMARK_REG register which will cause an interrupt to be generated when the fill level falls below the watermark. The watermark should be set to a number between zero and a page size. That is if the page size is 256 bytes, then setting the watermark to a value between 10 and 250 is reasonable and will cause the interrupt to trigger when the fill level drops below the programmed number. Setting the watermark can be useful to provide an indication to software when to write the next page of data to the SRAM.
8. Trigger Indirect Write access by setting OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[0] START_FLD bit.
9. If the remaining number of bytes still to be transferred into the SRAM for the current indirect transfer is greater than a FLASH page, then write 1 FLASH page worth of data to the SRAM. Otherwise send the remaining data from the indirect transfer to SRAM.
10. If all the data in the indirect transfer has now been sent to the SRAM, then go to 12 and await indirect complete status. Otherwise if there is more data still to be transferred then either:
    - If the watermark interrupt feature is being used, then wait for watermark interrupt.
    - Alternatively the SRAM fill level can be interrogated to identify a convenient time to send more data.
11. Loop back to 9.
12. Optional: The completion status of the Indirect write operation can be polled via OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[5] IND_OPS_DONE_STATUS_FLD.
13. An Indirect Complete interrupt will be generated when the Indirect write operation has completed.

12.3.2.4.10.3 Indirect Access Queuing

Software is permitted to queue up to two indirect transfers for both the indirect write controller and the indirect read controller. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. Any attempt to queue more than two operations will cause an interrupt to be generated. To take advantage of this feature, software should attempt to keep both indirect programming slots full at all times.

From the software perspective, indirect access queuing is achieved by triggering bit 0 of the indirect transfer control register (OSPI_FLASH_CFG_INDIRECT_READ_XFER_CTRL_REG[0] START_FLD bit or OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_CTRL_REG[0] START_FLD bit) twice in short succession. The indirect number of bytes register (OSPI_FLASH_CFG_INDIRECT_READ_XFER_NUM_BYTES_REG or OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_NUM_BYTES_REG register) and the indirect FLASH start address register (OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG or OSPI_FLASH_CFG_INDIRECT_WRITE_XFER_START_REG register) must be setup with the relevant transfer data before START_FLD bit can be triggered for each transfer. Since these registers will change regularly, the hardware must keep sampled versions of these registers for the duration of the indirect transfer.

The internal register block will only issue an indirect start trigger to the key underlying datapath blocks one at a time. There are 2 independent datapath blocks in the indirect access controller that will receive and independently sample this information. The first is the datapath block on the data bus side of the SRAM. For indirect reads, this is a read interface, for indirect writes, it is a write interface. The second is the datapath block on the FLASH side of the SRAM. For indirect reads, this is a write interface, for indirect writes, it is a read interface. Both blocks will process the indirect transfers at different times. For example, for an indirect read operation, the datapath block on the FLASH side of the SRAM will be able to start processing the second queued transfer as soon as the last byte of the first transfer has been written to the SRAM. Before commencing the second transfer, this block must resample the OSPI_FLASH_CFG_INDIRECT_READ_XFER_NUM_BYTES_REG and OSPI_FLASH_CFG_INDIRECT_READ_XFER_START_REG registers. Similarly, the datapath block on the bus

Copyright © 2026 Texas Instruments Incorporated

side will resample the same registers locally when it has forwarded all the FLASH data associated with the first indirect transfer from the SRAM onto the data bus.

12.3.2.4.10.4 Consecutive Writes and Reads Using Indirect Transfers

It is permitted for software to trigger an indirect read operation while an indirect write operation is in progress. Similarly it is permitted to trigger an indirect write while an indirect read operation is in progress. Indirect write operations will take overall precedence.

12.3.2.4.10.5 Accessing the SRAM

The SRAM depth is separated in two segmets. The lower segment is reserved for indirect read use.
The upper segment is for indirect write use only. The size of each segment is programmable via the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG register. This feature allows to allocate how many bits of the SRAM address bus are allocated to indirect read. By default, this is set so that exactly half of the SRAM is portioned for use by the indirect read controller. To ensure the read data bus is not directly fed by the SRAM read data through combinatorial logic, an extra bank of holding registers is included in the indirect read data path. These registers act as an extra location to be added to the allocated number of SRAM locations for indirect read.

To illustrate how the SRAM (and the extra bank of holding registers) can be allocated between indirect read and write, the following example is provided. The depth of the SRAM in this example is configured to be 8 bits. This is equal to 256 locations.

- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0x00, then 256 locations are allocated to indirect writes and 1 location to indirect reads.
- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0x01, then 255 locations are allocated to indirect writes and 2 locations to indirect reads.
- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0x02, then 254 locations are allocated to indirect writes and 3 locations to indirect reads.
- And so on until.
- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0xFD, then 3 locations are allocated to indirect writes and 254 locations to indirect reads.
- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0xFE, then 2 locations are allocated to indirect writes and 255 locations to indirect reads.
- If the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG[7-0] ADDR_FLD field is set to 0xFF, then 1 location is allocated to indirect writes and 256 locations to indirect reads.

Note

A value of 0xFF or 0x00 in the OSPI_FLASH_CFG_SRAM_PARTITION_CFG_REG register should be avoided by software, as only the bottom 8 bits of the SRAM fill level are accessible through software (up to 255 limit) via the OSPI_FLASH_CFG_SRAM_FILL_REG register. If the fill level reaches 256 on either the indirect read or write side, it will appear when reading the Fill Level to be 0.

There are four SRAM sources that are arbitrated and muxed onto the single SRAM port. Up to three sources can access this port at any one time. The sources are described as follows:

- Indirect Write, Write source. This is located on the data bus side of the SRAM.
- Indirect Write, Read source. This is located on the FLASH side of the SRAM.
- Indirect Read, Write source. This is located on the FLASH side of the SRAM.
- Indirect Read, Read source. This is located on the data bus side of the SRAM.

A fixed priority arbitration scheme is implemented. Table 12-167 shows priority allocated to these sources.

Table 12-167. SRAM Access Priority

SRAM Access Priority		
Indirect Write	Write to SRAM (from System Data Bus)	3rd (exclusive with Data Bus Read Request)
	Read from SRAM (from OSPI Module)	2nd

Copyright © 2026 Texas Instruments Incorporated

Table 12-167. SRAM Access Priority (continued)

Indirect Read	Write to SRAM (from OSPI Module)	1st
	Read from SRAM (from System Data Bus)	3rd (exclusive with Data Bus Write Request)

Note

With the exception of the write port during an Indirect Read operation (on the FLASH side of the SRAM), the logic driving all four sources must not assume single cycle completion. Writes to the SRAM during an indirect read must be allowed to complete immediately to avoid data loss. Therefore this port is given maximum priority.

12.3.2.4.11 OSPI Software-Triggered Instruction Generator (STIG)

The DAC and INDAC are used to transfer data. In order to access the volatile and non-volatile configuration registers, the legacy SPI Status register, other status/protection registers as well as to perform ERASE functions, a separate software controller is required. The software triggered instruction generator (STIG) is controlled using the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register by setting up the command to issue to the FLASH device. This is a generic controller and can be used to perform any instruction that the FLASH device supports from the extended SPI protocol. Configuring of instructions which are not compliant with the specification of the FLASH devices could cause unpredicted behavior of the controller. OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[31-24] CMD_OPCODE_FLD bits should be set different than OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[7-0] RD_OPCODE_NON_XIP_FLD and OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG[7-0] WR_OPCODE_FLD. The OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[0] CMD_EXEC_FLD bit is used to trigger the command. The OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[1] CMD_EXEC_STATUS_FLD bit is used by software to poll the status of the command execution. For reads, when the command has been serviced (OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[1] CMD_EXEC_STATUS_FLD bit toggles from '1' to '0'), up to 8 bytes of read data will be placed in the OSPI_FLASH_CFG_FLASH_RD_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_RD_DATA_UPPER_REG registers. For writes, the write data should be placed in the OSPI_FLASH_CFG_FLASH_WR_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_WR_DATA_UPPER_REG registers.

The completion of the STIG request could be also checked by the corresponding interrupt. The occurrence of the interrupt indicates that the controller is ready for accepting a new STIG request. It is important to notice that completion of the STIG request is not equivalent to completion it on SPI side. For example, if STIG is configured to the command composed of data to transmit only, the data is taken from the corresponding STIG register fields and put into TX FIFO. Since all bytes to write are known, another STIG can be queued before serialization of the current one is completed.

There are some commands which require more data to read than 8 bytes (for example READ ID command). The additional STIG Memory Bank is implemented in order to accommodate these data if needed. The STIG Memory Bank (internal component of the controller) is controlled by the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[2] STIG_MEM_BANK_EN_FLD bit. If enabled, the number of bytes to read in the STIG is extended to 16 as defined in OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[18-16] NB_OF_STIG_READ_BYTES_FLD bit field. It should be noticed that there are very few commands (excluding Read Array ones which are not intended to handle effectively in STIG Mode but in Direct/Indirect Modes) which return more than 8 bytes to the controller. If the maximum number of bytes to Read using STIG in target application is less than 16, the depth of the STIG Memory Bank can be set smaller what will result in saving noticeable part of the area.

If number of bytes to Read in the STIG as defined in OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[18-16] NB_OF_STIG_READ_BYTES_FLD bit field exceeds the Memory Bank Depth, remaining data will overwrite the STIG Memory Bank locations starting from its first address. OSPI_FLASH_CFG_FLASH_RD_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_RD_DATA_UPPER_REG keep the last 8 bytes read from the Flash Device

Copyright © 2026 Texas Instruments Incorporated

by STIG when Memory Bank is enabled. Therefore, for example if the user wants to get just a single byte from the last eight bytes from long continuous read SPI data chain, there is no need to access the STIG Memory Bank since data can be taken from suitable Flash Command Read Data register. In order to access more data, STIG Memory Bank data request should be triggered. It is controlled by the OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG and works analogously for triggering STIG from the functional standpoint.

OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[0] TRIGGER_MEM_BANK_REQ_FLD bit is used to trigger the command, bit OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[1] MEM_BANK_REQ_IN_PROGRESS_FLD is used by software to poll the status of the command execution. When MEM_BANK_REQ_IN_PROGRESS_FLD bit toggles from "1" to "0", the byte of data (OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[15-8] MEM_BANK_READ_DATA_FLD) from corresponding address (OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG[28-20] MEM_BANK_ADDR_FLD bit field) is valid. The address should be set before triggering the STIG Memory Bank access. Each consecutive STIG access overwrites the previous one so that the data in the Bank always fit into byte index fetched by the last STIG access configured to use the Memory Bank (first incoming byte equals first address of the Memory Bank, second one equals the second address and so on).

12.3.2.4.11.1 Servicing a STIG Request

A STIG request will cause the OSPI Flash controller to interrogate the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register to determine what and how many bytes it should send to the FLASH device. The OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[31-24] CMD_OPCODE_FLD field of this register indicate the instruction to be sent and is always pushed first. If there is an address to send, then the address (the size of which is also programmed in the same register) is sent next. The address itself is stored in the OSPI_FLASH_CFG_FLASH_CMD_ADDR_REG register. If Mode bits are enabled by OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[18] ENB_MODE_BIT_FLD bit. OSPI_FLASH_CFG_MODE_BIT_CONFIG_REG[7-0] MODE_FLD bit field are being sent right after address. If OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[18] ENB_MODE_BIT_FLD and OSPI_FLASH_CFG_CONFIG_REG[29] CRC_ENABLE_FLD are both enabled, STIG will replace XIP Mode bits (not applicable for CRC aware SPI interface) for automatically calculated address CRC byte. Therefore, to execute CRC aware STIGs (meaning the commands requiring sending address CRC byte), ENB_MODE_BIT_FLD bit should always be set. If there are any dummy cycles to send (the size of which is also programmed in OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register) then those are sent next. If there is data to write or read (the size of which is also programmed in OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register) then for the case of writes, up to 8 bytes can be sent (as stored in the Flash Command Write Data registers, OSPI_FLASH_CFG_FLASH_WR_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_WR_DATA_UPPER_REG registers) next. In the read case, when the read data has been collected from the FLASH device, the OSPI Flash Controller stores that in the Flash Command Read Data Registers (OSPI_FLASH_CFG_FLASH_RD_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_RD_DATA_UPPER_REG registers). Up to 8 bytes can be get if OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[2] STIG_MEM_BANK_EN_FLD bit is disabled or up to 512 when enabled. When the OSPI Flash controller starts to service a STIG request, it sets the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[1] CMD_EXEC_STATUS_FLD bit to indicate a command execution is in progress. When the OSPI Flash controller is in the auto-polling state, servicing a STIG request is slightly different. Most of devices are largely inaccessible after a program operation until the device has completed that write. Some group of them has a possibility to suspend programming page. It can be controlled by the OSPI_FLASH_CFG_POLLING_FLASH_STATUS_REG[8] DEVICE_STATUS_VALID_FLD bit, which indicate active auto-polling phase. After requesting a STIG, the OSPI Flash Controller immediately issues appropriate OPCODE to Memory. During servicing a STIG (in auto-polling phase) the status bit of command execution remains steady and other parts of transfer such as ADDRESS or DUMMY BITS, and so forth, are disabled (to issued Program Suspend Command is needed OPCODE only). There is a programmable option to add delay between every repetitive poll operation (delay is defined by OSPI_FLASH_CFG_WRITE_COMPLETION_CTRL_REG[31-24] POLL_REP_DELAY_FLD bit field). This feature is implemented to free up SPI bandwidth if needed.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.4.12 OSPI Arbitration Between Direct / Indirect Access Controller and STIG

When multiple controllers are active simultaneously, a simple fixed-priority arbitration scheme is used to arbitrate between each interface and access the external FLASH. The fixed priority is defined as follows, highest priority first.

- The Indirect Access Write
- The Direct Access Write
- The STIG
- The Direct Access Read
- The Indirect Access Read

12.3.2.4.13 OSPI Command Translation

Requests issued by the direct access controller, the indirect access controller or the STIG will be translated into a sequence of byte transfers to send downstream (before serialization to the FLASH device). These sequences depend on the requested transfer but an example of a typical 1-byte non sequential READ is shown below:

INSTRUCTION OPCODE -> ADDRESS -> Mode Byte -> Dummy Bytes -> 1 byte of don't care

For sequential accesses, an extra byte of data per read is pushed to the FLASH device on the back of the above sequence assuming it can be done so with no gap between each transferred byte.

When PHY mode is enabled and consequently no clock divider is configured, latency caused by multi domain synchronization may make an extra byte insufficient to avoid the transfer gap. To ensure the sequential access non-interrupted and keep the maximum performance of the controller, PHY Pipeline Mode is implemented. When enabled, number of don't care bytes is calculated based on the configuration.

The actual sequence sent to the FLASH device depends on the requested transfers, whether the transfer is non-sequential or sequential, whether the device has been configured in XIP mode and the state of the main Device Instruction Type programmable registers (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG and OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG).

For writes, the write enable latch (or WEL) within the FLASH device itself must be high before a write sequence can be issued. The OSPI Flash Controller will automatically issue the write enable latch command before triggering a write command via the direct or indirect access controllers (DAC/INDAC) – that is the user does not need to perform this operation. For increasing flexibility and performance user can turn off this feature by setting the OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG[8] WEL_DIS_FLD bit. The opcode for WREN is typically 0x06 and is common between devices.

When write requests from the direct or indirect access controllers are no longer being received and all outstanding requests have been sent, the FLASH device will automatically start the page program write cycle. Any incoming request at this time will be held in wait states until the cycle has completed. The OSPI Flash Controller will automatically poll the FLASH device legacy SPI status register to identify when the write cycle has completed. This is achieved by sending the RDSR opcode to the FLASH device and waiting until the device itself has indicated the write cycle has completed (until the Write in Progress bit has cleared to zero and the write enable latch bit has also cleared to zero or device is ready bit has set to one). The WREN and the RDSR device instructions are the only ones that are sent by the controller under the hood. For any other specific instruction that the user determines should be sent to the device (for example if the device needs to be unprotected before a write command is issued), these should be handled separately by issuing FLASH commands via the STIG.

There is an option to trigger HOLD or RESET feature on I/Os of the Flash Device. The HOLD one is generally common across the devices and takes an alternative function of DQ3 pin (applicable when device operates neither in Quad SPI mode nor DDR). The transfer can be hold and then resumed by dedicated software trigger field (OSPI_FLASH_CFG_CONFIG_REG[4] HOLD_PIN_FLD). The devices which have the HOLD feature on DQ3 usually need another dedicated pin for hardware reset and ones without HOLD feature usually have alternative reset on DQ3 what makes the additional reset pin being redundant. The controller supports both variants and reset selection register field (OSPI_FLASH_CFG_CONFIG_REG[6] RESET_CFG_FLD) allows the user to configure which hardware reset solution is implemented in the device under usage.

Copyright © 2026 Texas Instruments Incorporated

After configuration is done, it is possible to trigger HOLD or RESET features using I/Os (OSPI_FLASH_CFG_CONFIG_REG[4] HOLD_PIN_FLD or OSPI_FLASH_CFG_CONFIG_REG[5] RESET_PIN_FLD bits). After HOLD activation the controller is introduced into waiting state and any other operations should not be requested before de-asserting of HOLD configuration bit. The HOLD feature is useful when any SPI transaction needs to be prolonged in order to adjust it into specific point in time. Note that any HOLD trigger issued during active SPI transaction may be synchronized into reference clock domain at the time the SPI transfer turns to be finished. In this case, there is nothing to hold so the low level SPI logic will not activate HOLD on DQ3. To check if HOLD request suspended the transfer OSPI_FLASH_CFG_CONFIG_REG[31] IDLE_FLD bit can be polled for. If SPI is not in the IDLE state, the transfer was successfully suspended. It is important for the software to take care of resetting OSPI_FLASH_CFG_CONFIG_REG[4] HOLD_PIN_FLD bit before newly triggered SPI transaction. In case HOLD request is set before the beginning of the transfer it will be HOLD right after it starts what may not always be a goal. The hardware RESET needs to be activated when CS is high (no valid transaction is present on SPI bus). It can be checked by polling of OSPI_FLASH_CFG_CONFIG_REG[31]. If the controller is in the IDLE state and no other transfer requests are queued to perform, the hardware RESET can be triggered. The RESET feature is useful when any write, program or erase operation needs to be cancelled. No transfer request is permitted before driving the reset back to being inactive. Triggering HOLD or RESET on DQ3 at the time the device is configured to work in Quad SPI mode or DDR will overwrite transfer data on DQ3 with '0'. This behavior is considered as a software error so it is advisable for the system to make sure that the flash device was introduced to suitable SPI mode (that is by polling its configuration register) before triggering alternative DQ3 function. There are four independent reset outputs implemented to separate between multiple devices connected to the controller (up to 4 are supported). The decision which reset output is to be activated after triggering OSPI_FLASH_CFG_CONFIG_REG[5] RESET_PIN_FLD bit is made based on OSPI_FLASH_CFG_CONFIG_REG[9] PERIPH_SEL_DEC_FLD and OSPI_FLASH_CFG_CONFIG_REG[13-10] PERIPH_CS_LINES_FLD bits. Reset output OSPI_ECC_VECTOR is to be directly driven into corresponding dedicated RESET pins of the devices with separated RESET pin and alternatively, Reset output OSPI_ECC_VECTOR is to be control OSPI_ECC_VECTOR of the DQ3 RESET devices enabling separating of DQ3 Controller Outputs on SoC integration level.

The controller supports all combinations of CPHA and CPOL for Serial Clock. It allows the controller to support any SPI target devices not limited to Flash Memories. Multiple-SPI flash devices use just a subset of these combinations depending on the Transfer Mode as defined in Table 12-168.

Table 12-168. Flash SPI Modes

(SEL_CLK_POL_FLD, SEL_CLK_PHASE_FLD)	Edge Mode	Support
0x00 (SPI MODE 0)	SDR	Yes
0x01 (SPI MODE 1)	SDR	No
0x10 (SPI MODE 2)	SDR	No
0x11 (SPI MODE 3)	SDR	No
0x00 (SPI MODE 0)	DDR	Yes
0x01 (SPI MODE 1)	DDR	No
0x10 (SPI MODE 2)	DDR	No
0x11 (SPI MODE 3)	DDR	No

12.3.2.4.14 Selecting the Flash Instruction Type

In order to send the correct READ and WRITE opcodes, software should initialize the OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG and the OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG registers. These registers include fields to setup the required instruction opcodes that is intended to be used to access the FLASH (default is basic READ and basic page program) as well as the instruction type, edge mode (DDR or SDR) and whether the instruction uses single, dual, quad or octal pins for address and data transfer. Providing this level of control to the user provides a future proofed generic solution. To ensure the controller can operate from a reset state, the registers will be reset to an opcode compatible with SIO devices what can be modified using BOOT feature.

Copyright © 2026 Texas Instruments Incorporated

Despite being applicable for both READs and WRITEs, the OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[9-8] INSTR_TYPE_FLD field only appears once – it is not included in the OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG register. If software sets this to anything other than '0', then the address transfer type and the data transfer type bits of both OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG and OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG registers become don't care. It is made available to allow software to support the less common FLASH instructions where the opcode, address and data are sent on 2 or 4 lanes (the opcode from most instructions are sent serially to the FLASH device, even for dual/quad instructions).

There are devices capable to handling Read Operations in Dual Data Rate Mode (DDR) (it is also called Dual Transfer Rate Mode (DTR)). That means they can issue and capture the data on both rising and falling edges during working with dedicated command type. This enables the controller to maintain throughput at twice lower frequency of OSPI clock. The Device Read Instruction Register has DDR enable bit which informs Octal-SPI Flash Controller that opcode written into Read Opcode field is capable with DDR command type. The other field defined in OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[19-16] DDR_READ_DELAY_FLD which enables the controller to shift the transmitted data in DDR mode. By default, data are shifted by 1 clock cycle to ensure hold timing greater than 0 during DDR transactions. It may not be sufficient for high reference clock frequency in accordance with the high dividers.

Table 12-169 shows how software should configure the OSPI module for selected specific READ and WRITE instruction supported by the abovementioned device.

Table 12-169. READ and WRITE Instruction Configuration

READ							
OPCODE	OPCODE sent over how many lanes / edge mode?	ADDRESS / DUMMY / MODE sent over how many lanes / edge mode?	DATA bytes sent over how many lanes / edge mode?	Instruction Type (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[9-8] INSTR_TYPE_FLD)	Address transfer type (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[13-12] ADDR_XFER_TYPE_STD_MODE_FLD)	Data transfer type (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[17-16] DATA_XFER_TYPE_EXT_MODE_FLD)	DDR bit enable (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[10] DDR_EN_FLD)
READ	1/SDR	1/SDR	1/SDR	0	0	0	0
FAST_READ	1/SDR	1/SDR	1/SDR	0	0	0	0
DTR_FAST_READ	1/SDR	1/DDR	1/DDR	0	0	0	1
DOFR (Dual O/p Fast Read)	1/SDR	1/SDR	2/SDR	0	0	1	0
DIOFR (Dual I/O Fast Read)	1/SDR	2/SDR	2/SDR	0	1	1	0
DDIOFR (DTR Dual I/ O Fast Read)	1/SDR	2/DDR	2/DDR	0	1	1	1
QOFR (Quad O/p Fast Read)	1/SDR	1/SDR	4/SDR	0	0	2	0
QIOFR (Quad I/O Fast Read)	1/SDR	4/SRD	4/SDR	0	2	2	0
DQIOFR (DTR Quad I/ O Fast Read)	1/SDR	4/DDR	4/DDR	0	2	2	1
OOFR (Octal O/p Fast Read)	1/SDR	1/SDR	8/SDR	0	0	3	0
OIOFR (Octal I/O Fast Read)	1/SDR	8/SDR	8/SDR	0	3	3	0

Copyright © 2026 Texas Instruments Incorporated

Table 12-169. READ and WRITE Instruction Configuration (continued)

DOIOFR (DTR Octal O/p Fast Read)	1/SDR	1/DDR	8/DDR	0	0	3	1
4DOIOFR (4-byte DTR Octal I/O Fast Read)	1/SDR	8/DDR	8/DDR	0	3	3	1
DCFR (Dual Command Fast Read)	2/SDR	2/SDR	2/SDR	1	Don't care	Don't care	0
DDCFR (DTR Dual Command Fast Read)	2/SDR	2/DDR	2/DDR	1	Don't care	Don't care	1
QCFR (Quad Command Fast Read)	4/SDR	4/SRD	4/SDR	2	Don't care	Don't care	0
DQCFR (DTR Quad Command Fast Read)	4/SDR	4/DDR	4/DDR	2	Don't care	Don't care	1
OCFR (Octal Command Fast Read)	8/SDR	8/SDR	8/SDR	3	Don't care	Don't care	0
4DOCFR (4-byte DTR Octal Command Fast Read)	8/SDR	8/DDR	8/DDR	3	Don't care	Don't care	1

WRITE

OPCODE	OPCODE sent over how many lanes?	ADDRESS / DUMMY / MODE sent over how many lanes?	DATA bytes sent over how many lanes?	Instruction Type (OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[9-8] INSTR_TYPE_FLD)	Address transfer type (OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG[13-12] ADDR_XFER_TYPE_STD_MODE_FLD)	Data transfer type (OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG[17-16] DATA_XFER_TYPE_EXT_MODE_FLD)
PP	1	1	1	0	0	0
DIFP (Dual Input Fast Program)	1	1	2	0	0	1
DIEFP (Dual Input Extended Fast Program)	1	2	2	0	1	1
QIFP (Quad Input Fast Program)	1	1	4	0	0	2
QIEFP (Quad Input Extended Fast Program)	1	4	4	0	2	2
OIFP (Octal Input Fast Program)	1	1	8	0	0	3
OIEFP (Octal Input Extended Fast Program)	1	8	8	0	3	3
DCPP (Dual Command Fast Program)	2	2	2	1	Don't care	Don't care

Copyright © 2026 Texas Instruments Incorporated

Table 12-169. READ and WRITE Instruction Configuration (continued)

QCPP (Quad Command Fast Program)	4	4	4	2	Don't care	Don't care
OCPP (Octal Command Fast Program)	8	8	8	3	Don't care	Don't care

Note

This data are applicable for both 3-byte or 4-byte address variants of the commands if did not indicate otherwise.

Note

In DTR protocol all transfer phases (including opcode) take DDR edge mode independently on the command under execution. DTR protocol is to be enabled by OSPI_FLASH_CFG_CONFIG_REG[24] ENABLE_DTR_PROTOCOL_FLD bit. It has higher priority than DDR Mode enable bit from OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[10] DDR_EN_FLD.

12.3.2.4.15 OSPI Data Interface

12.3.2.4.16 OSPI PHY Module

OSPI module fully integrates PHY module dedicated to more flexible and power efficient transfers.

The PHY module communicates with the OSPI Flash controller via the aforementioned PHY Interface and handles data transfer on low-level stage of design hierarchy. However, when the OSPI_RCLK is configured to be equal to the SPI clock instead of alternative approach using clock divider, there is just one OSPI_RCLK cycle (not 4 or more) within single SPI period or half period for DDR Mode (SPI Control Module works on reference clock). Given that OSPI_RCLK is the input clock for RX FIFO and the output one for TX FIFO, the PHY solution incurs more restrictive requirement for value of system clock in order to synchronize data without SPI transfer interruption. For example, when the controller operates in DDR 1× octal Mode, 2 bytes of data (equivalent to one RX FIFO location) is gathered within just single OSPI_RCLK cycle. The controller cannot predict next data access while operating in the Direct Mode (meaning its size or whether it is sequential to the previous one or not). As a result, if the OSPI_HCLK is not significantly greater than OSPI_RCLK, the SPI transfer has to be suspended until the Flash Command Generator forwards new data to TX FIFO.

An optional PHY Pipeline Mode is implemented to avoid the necessity of stable clocking of the system clock for the Direct Mode when the PHY mode is enabled and to keep maximum performance while ensuring correct operation of the OSPI controller with the PHY using low frequencies from all its domains. This mode is a trade-off between large software overhead when operating in the Indirect Mode and the described limitations present in the Direct Mode. For more information about PHY Pipeline Mode, see Section 12.3.2.4.16.1, *PHY Pipeline Mode*.

When DDR 2× Mode is granted based on configuration – SPI transfer is automatically performed using the PHY module even if the OSPI_FLASH_CFG_CONFIG_REG[3] PHY_MODE_ENABLE_FLD is de-asserted. SDR 2× commands are handled with PHY module paths being bypassed. Nevertheless, dividers of 2, 4 or 6 for DDR and divider of 2 for SDR should not be configured based on controller requirements and these configurations are perceived as a software error.

Note

Please refer to the OSPI Controller PHY Tuning Algorithm Application Note for additional details related to the OSPI PHY Module and Tuning Algorithm,

Copyright © 2026 Texas Instruments Incorporated

12.3.2.4.16.1 PHY Pipeline Mode

This mode is used for Direct Read Mode of operation. If any other operations are intended to be executed, it is recommended to disable PHY Pipeline Mode and re-enable for subsequent Direct Reads in PHY mode. Since there is comprehensive software mechanism controlling Read data transfers in Indirect Mode, pipeline of data interface accesses is not effective for this mode. Enable PHY Pipeline feature when at least four 4-byte-sized data words are predicted to be read in sequentially. The Flash Command Generator pipelines and puts them into TX FIFO which causes CS to remain active because low level SPI protocol controller controls TX FIFO fill level. In order to correctly trigger Direct Read in Pipeline Mode TX FIFO must be empty. Therefore first polling of OSPI_FLASH_CFG_CONFIG_REG[31] IDLE_FLD bit needs to be done. The sequential data transfer will be interrupted when the data target select signal of the data interface is asserted to low. This information is also detected by Data Target Module which informs the Flash Command Generator that the next access is invalid and a TX FIFO locations can be flushed transparently for the system.

In PHY Pipeline Mode it is recommended for Data Controller not to introduce wait states in between consecutive occurrences of the data interface signal that indicates transfer has finished. It will ensure regular transfer rate on data side. Introducing wait states gradually slows data transfer rate down and may finally cause SPI transfer interruption because of TX FIFO data starvation. The system, however, may need to introduce some number of wait states after completion of sequential transfer (composed of 4-byte sized data words) for progressing the data. The dedicated buffer is implemented in Data Target Controller which collects all incoming data during wait states injection. In order to keep SPI transfer uninterrupted, number of wait states should be as little as possible. The higher the OSPI_HCLK/ OSPI_RCLK ratio the more wait states can be introduced without SPI transfer interruption. In case the system is able to launch a new transfer before wait states overflow, buffered data transfer to the host will continue. It compensates slowed down transfer by introducing wait states. In case the system is not able to launch a new transfer before wait states overflow, next incoming transfer is considered non-sequential and is executed after all pipelined data is flushed.

This mode can be enabled when following conditions are met:

- OSPI_HCLK > OSPI_RCLK (Comparing the slow data clock with the fast reference one makes Pipeline Mode ineffective – Suspend of SPI Transfer would be possible. Consequently, this condition has to be met to operate in this mode.)
- Only 4-byte sized Data Words are permitted (This ensures more data clock cycles for synchronization of FIFOs between consecutive pulses of the signal indicating transfer has finished.)
- The transfer with introduced wait states or non-sequential transfers can only be triggered in between at least four 4-byte sized Data Bursts sequential accesses (16 Bytes) to be sure that Data Controller can trust buffered incoming data during wait states injection.
- Do not use Pipeline Mode along with Continuous Mode (XIP). Benefit of XIP is limited for bulk data transfers intended to execute in Pipeline Mode.

12.3.2.4.16.2 Read Data Capturing by the PHY Module

Read Data Capturing by the PHY module is useful, as the user is not responsible for the design dedicated DLL being compatible with the Octal-SPI Flash Controller. Another benefit is an option to adjust both SPI clock and sampling clock in a very wide range to fit them into individual requirements of any system. If loopback clock (OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[0] BYPASS_FLD) and PHY mode (OSPI_FLASH_CFG_CONFIG_REG[3] PHY_MODE_ENABLE_FLD) are both enabled, the loopback clock is driven into RX DLL instead of gated reference clock. Because of the architecture of DLL, loopback clock needs to be provided in SPI Mode 0. If DQS (OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG[8] DQS_ENABLE_FLD) and PHY mode (OSPI_FLASH_CFG_CONFIG_REG[3] PHY_MODE_ENABLE_FLD) are both enabled, the DQS is driven into RX DLL instead of gated reference clock.

Copyright © 2026 Texas Instruments Incorporated

12.3.2.5 OSPI Programming Guide

12.3.2.5.1 Configuring the OSPI Controller for Use After Reset

The OSPI controller has been designed to wake up in a state that is suitable for performing basic reads and writes using the direct access controller. The BASIC read (opcode 0x03) and BASIC write (opcode 0x02) instructions are operations supported by all target devices. The controller also wakes up with a baud rate divider setting of divide-by-32. Assuming the reference clock is operating at 400 MHz after reset, then this means the effective SPI clock is just 12.5 MHz. This should be slow enough to meet all timing requirements of all target devices without any further device programming.

If the target device does not use 3 address bytes, the device size configuration register must be modified to the appropriate size.

If software plans to write to the device, and the number of bytes per device page is not equal to 256, then the device size configuration register must also be modified.

While not a requirement, it is prudent for software to enable the write protect feature prior to enabling the OSPI controller. This will block any data writes from taking effect. To do so, the protection registers (OSPI_FLASH_CFG_LOWER_WR_PROT_REG, OSPI_FLASH_CFG_UPPER_WR_PROT_REG and OSPI_FLASH_CFG_WR_PROT_CTRL_REG) should be setup and the number of bytes per device block in the device size configuration register should also be setup.

After Power-on Reset (POR), software can read from and write to the FLASH device (albeit slowly). Enabling/Disabling the controller and DAC is achieved with just one write to corresponding fields of the OSPI_FLASH_CFG_CONFIG_REG register. User shall take note to maintain the default values of the baud rate divisor and the default state of SEL_CLK_POL_FLD/ SEL_CLK_PHASE_FLD bits of this register. A write data value of 0x00780081 is recommended.

12.3.2.5.2 Configuring the OSPI Controller for Optimal Use

Note

When using the OSPI Controller, the opcodes in OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[7-0] RD_OPCODE_NON_XIP_FLD, OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG[7-0] WR_OPCODE_FLD and OSPI_FLASH_CFG_WRITE_COMPLETION_CTRL_REG[7-0] OPCODE_FLD bit fields shall not match the opcode in the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[31-24] CMD_OPCODE_FLD bit field.

For high speed transfers PHY mode can be enabled and for optimal configuration PHY Pipeline mode is recommended. For more information, see *PHY Pipeline Mode*.

To access the flash optimally, software must configure the controller accurately:

1. Wait until any pending STIG or INDAC operation has completed or poll OSPI_FLASH_CFG_CONFIG_REG[31] IDLE_FLD bit.
2. Disable the DAC through OSPI_FLASH_CFG_CONFIG_REG[7] ENB_DIR_ACC_CTLR_FLD bit. It is permitted, but not necessary to also disable the OSPI controller completely via OSPI_FLASH_CFG_CONFIG_REG[0] ENB_SPI_FLD bit.
3. Update the OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG and OSPI_FLASH_CFG_DEV_INSTR_WR_CONFIG_REG registers for the instruction type you wish to use for indirect and direct writes and reads.
4. Update the OSPI_FLASH_CFG_MODE_BIT_CONFIG_REG[7-0] MODE_FLD bit field if mode bits have been enabled in the OSPI_FLASH_CFG_DEV_INSTR_RD_CONFIG_REG[20] MODE_BIT_ENABLE_FLD bit.
5. Update the OSPI_FLASH_CFG_DEV_SIZE_CONFIG_REG if the contents are incorrect. Note parts or all of this register may have been updated after initialization. The number of address bytes is a key configuration setting required for performing reads and writes. The number of bytes per page is required for performing

Copyright © 2026 Texas Instruments Incorporated

any write. The number of bytes per device block is only required if the write protect feature is used. If the default values are correct for the target device, or if some of the values (not including the number address bytes) were incorrect but device writes were not permitted.

6. Update the OSPI_FLASH_CFG_DEV_DELAY_REG. This register allows the user to tweak how the chip select is driven after each FLASH access. This is required as each device may have different timing requirements. As the serial clock frequency is increased, these timing requirements become more important. Note the numbers programmed in this register are based on the period of reference clock. Example: A device needs 50ns minimum time before CS can be re-asserted after it has been de-asserted. By default, the controller will only provide a minimum of 1 SCLK period. When the device is operating at 100 MHz, the SCLK period is only 10ns, so 40ns extra is required. Since the register defines the number of reference clock cycles to add, and reference clock is running at 400 MHz (2.5ns period), then the user should program a value of at least 16 to the OSPI_FLASH_CFG_DEV_DELAY_REG[31-24] D_NSS_FLD. This delay can be extended during auto-polling phase. There is possibility to define the polling repetition delay in the OSPI_FLASH_CFG_WRITE_COMPLETION_CTRL_REG[31-24] POLL_REP_DELAY_FLD bit field.
7. Update the OSPI_FLASH_CFG_REMAP_ADDR_REG register, if required. Affects DAC path only.
8. Setup and enable write protection registers (OSPI_FLASH_CFG_LOWER_WR_PROT_REG, OSPI_FLASH_CFG_UPPER_WR_PROT_REG and OSPI_FLASH_CFG_WR_PROT_CTRL_REG) if they are required and if they have not already been setup from post initialization.
9. Enable required interrupts via the OSPI_FLASH_CFG_IRQ_MASK_REG register.
10. Setup the baud rate divisor in the OSPI_FLASH_CFG_CONFIG_REG[22-19] MSTR_BAUD_DIV_FLD to define the required clock frequency of the target device.
11. Update the OSPI_FLASH_CFG_RD_DATA_CAPTURE_REG register. This register will delay when the read data is captured and can help when the read data path from the device to the controller is long and the device clock frequency is high. An update to this register may not be necessary.
12. Enable the OSPI controller and the DAC via the OSPI_FLASH_CFG_CONFIG_REG.

12.3.2.5.3 Using the Flash Command Control Register (STIG Operation)

The OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register provides software means to access the FLASH device in a flexible and programmable manner. This is known as a STIG operation (Software Triggered Instruction Generator). The instruction opcode, number of address bytes (if any), the address itself, number of dummy cycles (if any), number of write data bytes (if any), the write data itself and the number of read data bytes (if any) can be programmed. Once these have been programmed, software can trigger the command via OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[0] CMD_EXEC_FLD bit and wait for its acceptance by polling OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[1] CMD_EXEC_STATUS_FLD bit. When CMD_EXEC_STATUS_FLD bit turns de-asserted, another STIG can be triggered. This method of accessing the FLASH is the typical mechanism that software would use to access the FLASH device's registers, as well as for performing ERASE operations. It can also be used to access the FLASH array itself, although the maximum of 8 data bytes may be read or written at any one time, defined in the Flash Command Write and Read Data registers (OSPI_FLASH_CFG_FLASH_RD_DATA_LOWER_REG, OSPI_FLASH_CFG_FLASH_RD_DATA_UPPER_REG, OSPI_FLASH_CFG_FLASH_WR_DATA_LOWER_REG and OSPI_FLASH_CFG_FLASH_WR_DATA_UPPER_REG). This number of bytes can be extended for Read Data commands using additional STIG Memory Bank controlled by OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG[2] STIG_MEM_BANK_EN_FLD and OSPI_FLASH_CFG_FLASH_COMMAND_CTRL_MEM_REG.

Commands issued using this interface have a higher priority than all other READ accesses coming from data interface, and will therefore interrupt any READ commands being requested by the indirect or direct controllers.

12.3.2.5.4 Using SPI Legacy Mode

SPI legacy mode allows software to access the internal TX-FIFO and RX-FIFO directly, thus bypassing the direct, indirect and STIG controllers.

Legacy mode allows the user to issue any FLASH instruction to the device, but does place a heavy software overhead in order to manage the fill levels of the FIFO's effectively. This is because the legacy SPI core

Copyright © 2026 Texas Instruments Incorporated

is bi-directional in nature, with data continuously being transferred in either direction while the chip select is enabled. Even if the driver only wishes to read data from the FLASH device, dummy data must be written out to ensure the chip select stays active, and vice versa for write transactions.

Since the TX-FIFO and RX-FIFO are of limited depth, software has a responsibility to maintain the FIFO levels to ensure the TX-FIFO does not become exhausted during the instruction execution and the RX-FIFO doesn't overflow. This can place a lot of overhead on software. Interrupts are provided to indicate when the fill levels pass programmable watermarks, which are themselves programmable registers OSPI_FLASH_CFG_TX_THRESH_REG and OSPI_FLASH_CFG_RX_THRESH_REG.

The limited depth may impose the limitation over execution of some specific SPI commands in legacy mode. Note that the controller interprets all transmitted bytes as valid. For example, if the Flash Device was configured to return valid data after many dummy cycles, the TX FIFO could become full before the controller sends all of dummy data.

12.3.2.5.5 Entering XIP Mode from POR

XIP is a mode that can be entered in a non-volatile way if the device has XIP enabled as a non-volatile configuration setting. Software will not be able to discover the state of XIP from POR via FLASH status register reads as the only operation a FLASH device will recognize when XIP mode is enabled is an XIP read operation.

If it is already known that the device will enter XIP from POR, then the OSPI_FLASH_CFG_MODE_BIT_CONFIG_REG[7-0] MODE_FLD and OSPI_FLASH_CFG_CONFIG_REG[18] ENTER_XIP_MODE_IMM_FLD bits should be set in initial boot.

If it is not already known that the device will enter XIP from POR, and XIP from POR may be supported by the attached FLASH device, then software can attempt to exit XIP mode by issuing an XIP exit command using a STIG command (via the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG register). To do this, software must be aware of the mode bit requirements of that device, as XIP entry and exit changes per device.

12.3.2.5.6 Entering XIP Mode Otherwise

XIP mode is supported in most FLASH devices. Some of them use signature bits that are sent to the device immediately following the address bytes, other use signature bits and also require a FLASH device configuration register write to enable XIP. For the FLASH devices that must be compliant to the OSPI controller, the following steps can be taken by software to enter XIP mode:

1. Disable the DAC and INDAC (OSPI_FLASH_CFG_CONFIG_REG[7] ENB_DIR_ACC_CTLR_FLD) to ensure no new data read accesses will be sent to the FLASH device.
2. (Optional)Configure the OSPI_FLASH_CFG_FLASH_CMD_CTRL_REG to issue a VCR write to FLASH memory, because XIP mode must first be enabled for some devices.
3. Configure the XIP mode bits in the OSPI_FLASH_CFG_MODE_BIT_CONFIG_REG[7-0] MODE_FLD bit field.
4. Enable the local controllers XIP mode by setting OSPI_FLASH_CFG_CONFIG_REG[17] ENTER_XIP_MODE_FLD bit.
5. Re-enable the DAC and, if required, the INDAC.

12.3.2.5.7 Exiting XIP Mode

To exit XIP mode, software should first disable the DAC and INDAC to ensure no new data read accesses will be sent to the FLASH device. It should then set mode bits to other than the established in the corresponding Flash Device specifications. These are dependent on the FLASH device and manufacturer. Software should then reset OSPI_FLASH_CFG_CONFIG_REG[17] ENTER_XIP_MODE_FLD.

Note the FLASH device must see a READ instruction before it can disable its internal XIP mode state, so this means XIP mode will internally stay active until the next READ instruction is serviced. User must take care to ensure that XIP mode is disabled before the end of any READ sequence.

Copyright © 2026 Texas Instruments Incorporated

12.3.3 General-Purpose Memory Controller (GPMC)

This section describes the General-Purpose Memory Controller (GPMC) for the device.

12.3.3.1 GPMC Overview

The General-Purpose Memory Controller is a unified memory controller dedicated for interfacing with external memory devices like:

- Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- Asynchronous, synchronous, and page mode (available only in non-multiplexed mode) burst NOR flash devices
- NAND flash
- Pseudo-SRAM devices

Figure 12-144 shows the GPMC0 module overview.

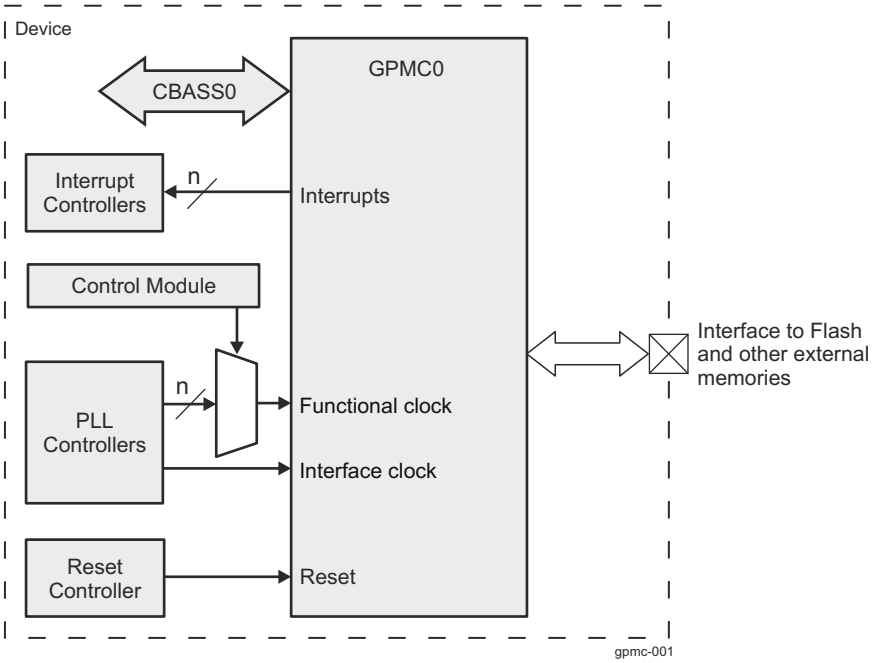


Figure 12-144. GPMC0 Overview

12.3.3.1.1 GPMC Features

The main features of the GPMC are:

- 8-, 16- or 32-bit-wide data path to external memory device
- Supports up to 4 chip select regions of programmable size and programmable base addresses in a total address space of 1GB
- Supports on-the-fly error code detection using the Bose-Chaudhurl-Hocquenghem (BCH) (t = 4, 8, or 16) or Hamming code to improve the reliability of NAND with a minimum effect on software (NAND flash with 512-byte page size or greater)
- Fully pipelined operation for optimal memory bandwidth usage
- The clock to the external memory is provided from GPMC_FCLK divided by 1, 2, 3, or 4
- Supports programmable autoclock gating when no access is detected
- Independent and programmable control signal timing parameters for setup and hold time on a per-chip basis. Parameters are set according to the memory device timing parameters with a timing granularity of one GPMC_FCLK clock cycle.
- Flexible internal access time control (wait state) and flexible handshake mode using external WAIT pin monitoring

Copyright © 2026 Texas Instruments Incorporated

- Support bus keeping
- Support bus turnaround
- Prefetch and write-posting engine associated with DMA controller at system level to achieve full performance from the NAND device with minimum effect on NOR/SRAM concurrent access
- 32-bit interconnect target interface which supports non-wrapping and wrapping burst of up to 16x32 bits.

The GPMC supports the following various access types:

- Asynchronous read/write access
- Asynchronous read page access (4-8-16-32 Word16, 4-8-16 Word32)
- Synchronous read/write access
- Synchronous read/write burst access without wrap capability (4-8-16-32 Word16, 4-8-16 Word32)
- Synchronous read/write burst access with wrap capability (4-8-16-32 Word16, 4-8-16 Word32)
- Address-data-multiplexed (AD) access
- Address-address-data (AAD) multiplexed access
- Little-endian access only

The GPMC can communicate with a wide range of external devices:

- External asynchronous or synchronous 8-bit wide memory or device (non burst device)
- External asynchronous or synchronous 16-bit wide memory or device
- External asynchronous or synchronous 32-bit wide memory or device
- External 16-bit non-multiplexed NOR flash device
- External 16- and 32-bit address and data multiplexed NOR Flash device
- External 8-bit and 16-bit NAND flash device
- External 16-bit and 32-bit pseudo-SRAM (pSRAM) device

Note

Page mode is available only in non-multiplexed mode.

12.3.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.2 GPMC Environment

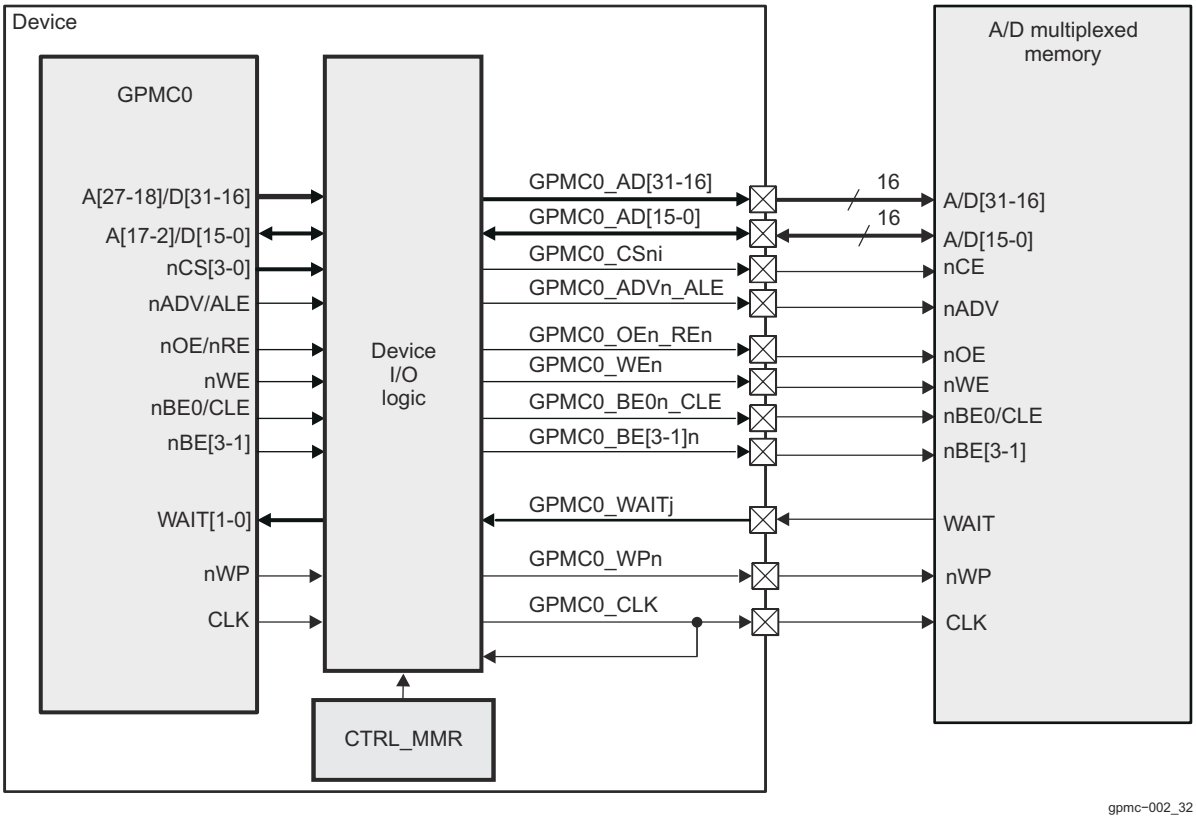
The GPMC0 module is hereinafter referred to as GPMC.

This section describes the GPMC0 external connections (environment).

12.3.3.2.1 GPMC Modes

This section shows four GPMC0 external connection options:

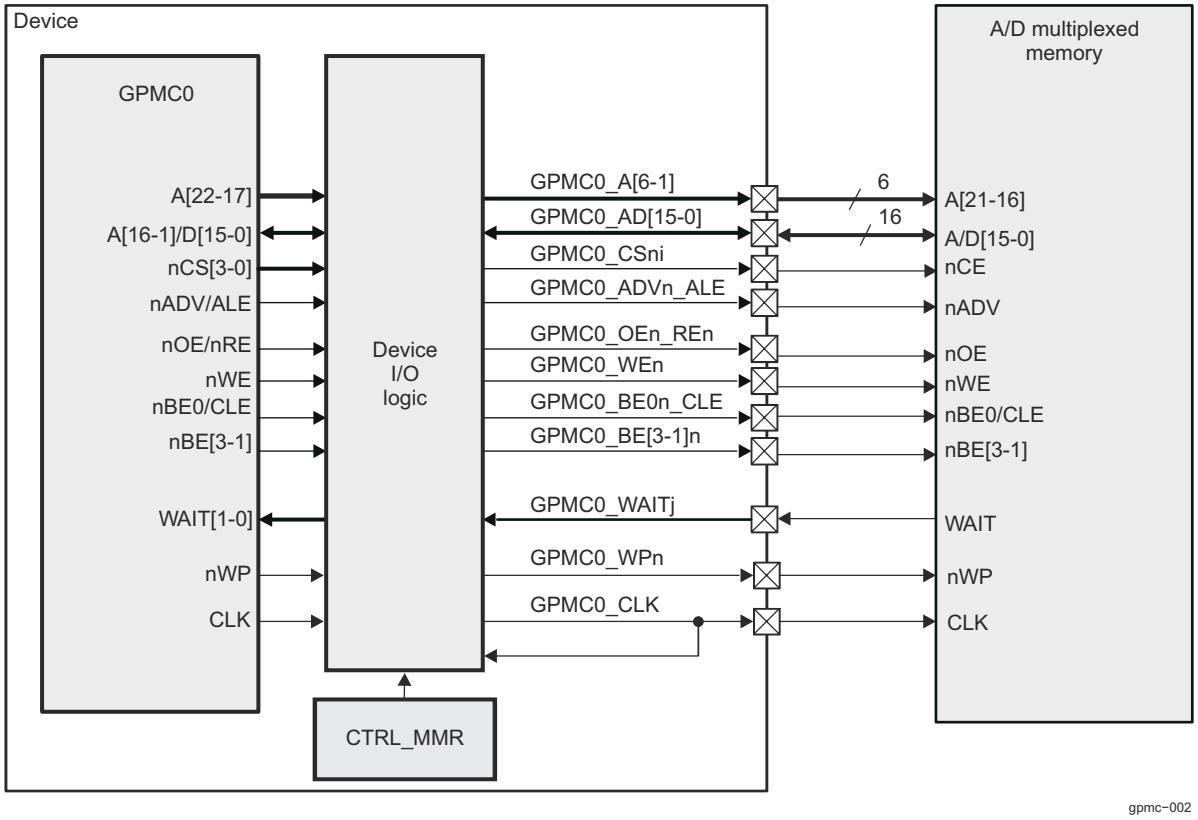
- Figure 12-145 shows a connection between the GPMC0 and a 32-bit synchronous address/data-multiplexed external memory device.
- Figure 12-146 shows a connection between the GPMC0 and a 16-bit synchronous address/data-multiplexed (or AAD-multiplexed but this protocol uses fewer address pins) external memory device.
- Figure 12-147 shows a connection between the GPMC0 and a 16-bit synchronous non-multiplexed external memory device.
- Figure 12-148 shows a connection between the GPMC0 and an 8-bit synchronous non-multiplexed external memory device.
- Figure 12-149 shows a connection between the GPMC0 and an 8-bit NAND device.



i = 0 to 3
j = 0 to 1

Figure 12-145. GPMC to 32-Bit Address/Data-Multiplexed Memory

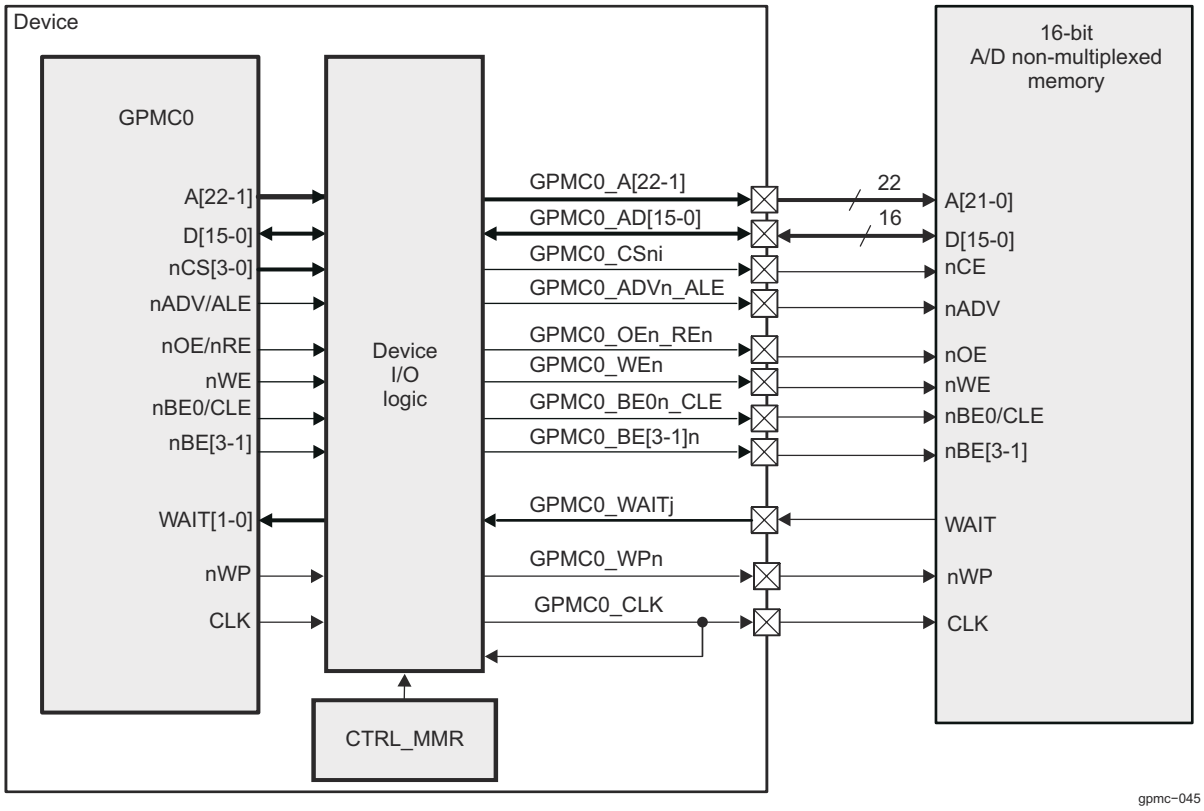
Copyright © 2026 Texas Instruments Incorporated



i = 0 to 3
j = 0 to 1

Figure 12-146. GPMC to 16-Bit Address/Data-Multiplexed Memory

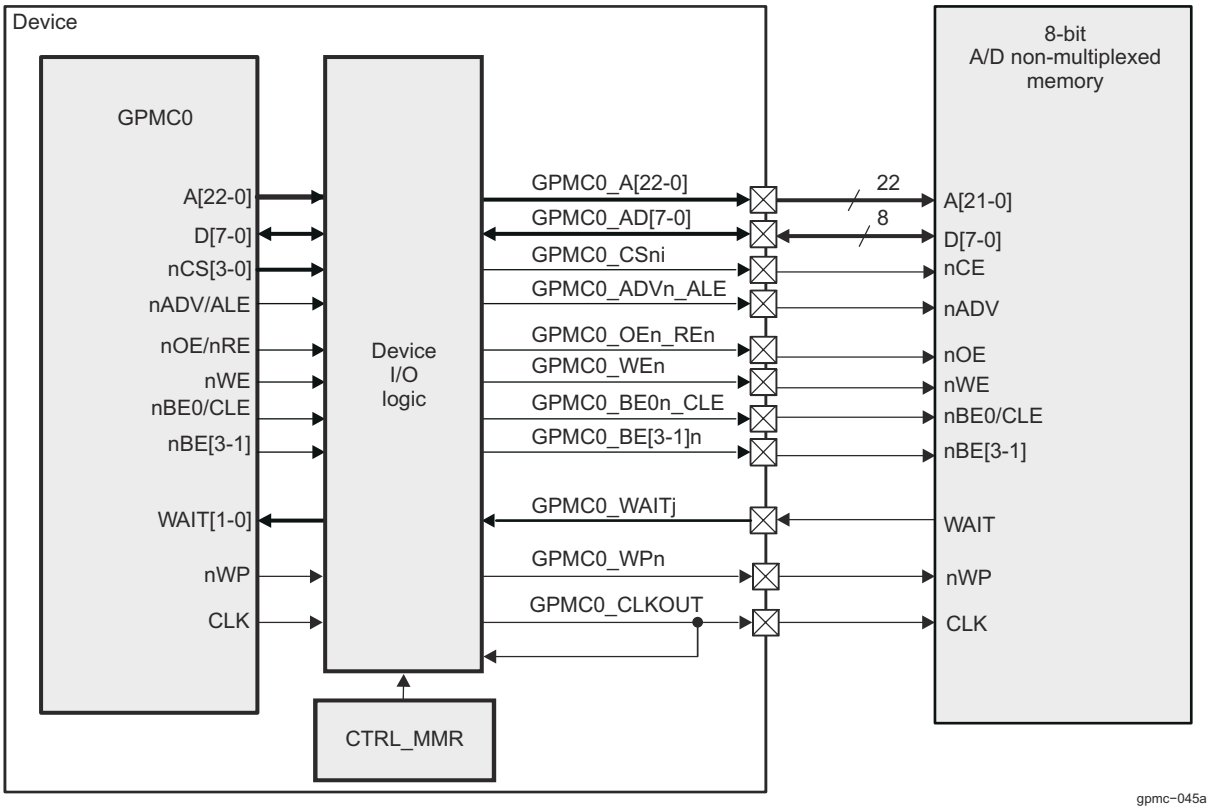
Copyright © 2026 Texas Instruments Incorporated



i = 0 to 3
j = 0 to 1

Figure 12-147. GPMC to 16-Bit Non-Multiplexed Memory

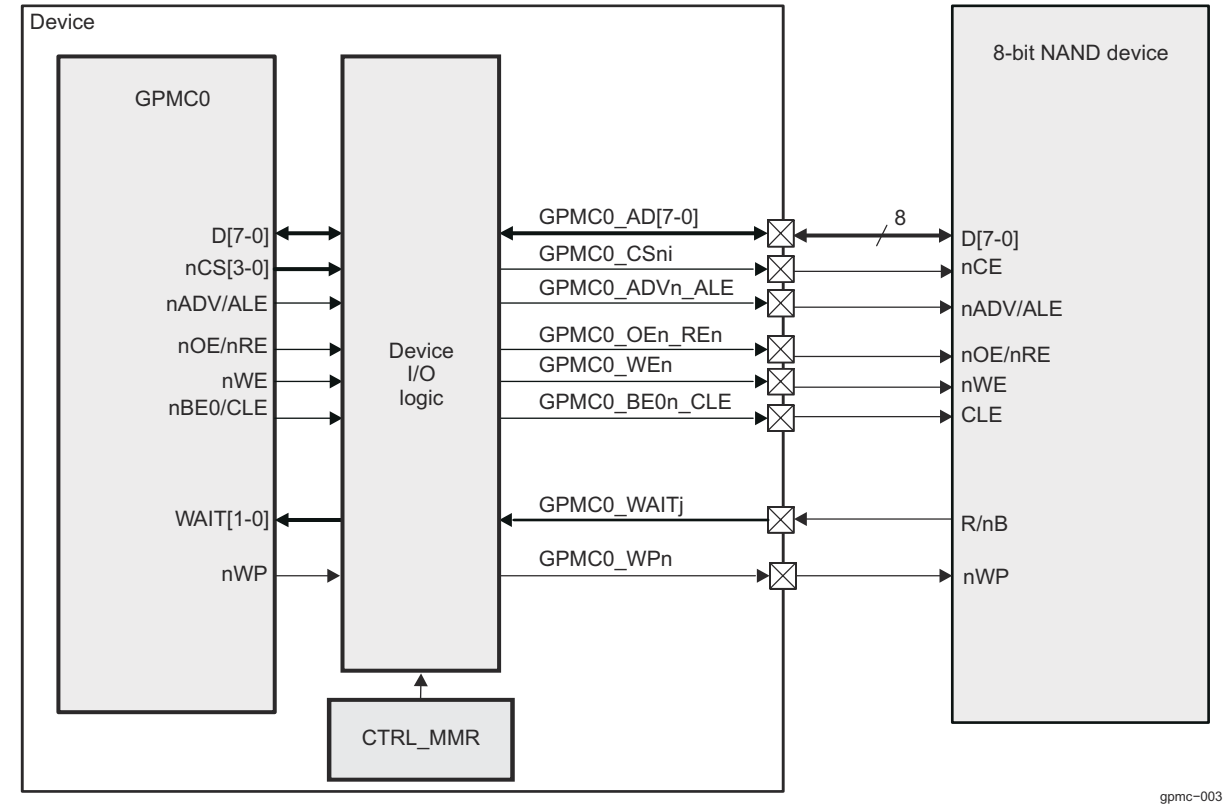
Copyright © 2026 Texas Instruments Incorporated



i = 0 to 3
j = 0 to 1

Figure 12-148. GPMC to 8-Bit Non-Multiplexed Memory

Copyright © 2026 Texas Instruments Incorporated



i = 0 to 3
j = 0 to 1

Figure 12-149. GPMC to 8-Bit NAND Device

12.3.3.2.2 GPMC I/O Signals

Table 12-170 lists the GPMC subsystem input/output (I/O) pins.

Table 12-170. GPMC I/O Signals (Controller Mode)

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
			GPMC0	
A[22-0]	GPMC0_A[22-0]	O	23-bit output address bus	-
A[27-2]/D[31-0]	GPMC0_AD[31-0]	I/O	Multiplexed address/data	-
nCS[3-0]	GPMC0_CSn[3-0]	O	Chip-selects (active low)	-
CLK	GPMC0_CLK	O	Clock generated for the external memory or device. For more information, see *GPMC0 Integration*.	-
CLKLB				
N/A	GPMC0_FCLK_MUX	O	Free running clock. GPMC functional clock (GPMC0_FCLK) propagated on a device pad. For more information on the GPMC0_FCLK_MUX integration, see *GPMC0 Integration*.	-
nADV/ALE	GPMC0_ADVn_ALE	O	Address valid (active low). Also used as address latch enable (active high) for NAND protocol memories.	-
nOE/nRE	GPMC0_OEn_REn	O	Output enable (active low). Also used as read enable (active low) for NAND protocol memories.	-
nWE	GPMC0_WEn	O	Write enable (active low)	-
nBE0/CLE	GPMC0_BE0n_CLE	O	Lower-byte enable (active low). Also used as command latch enable for NAND protocol memories.	-
nBE[3-1]	GPMC0_BE[3-1]n	O	Upper-byte enable (active low)	-

Copyright © 2026 Texas Instruments Incorporated

Table 12-170. GPMC I/O Signals (Controller Mode) (continued)

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
WAIT[1-0]	GPMC0_WAIT[1-0]	I	External wait signal for NOR and NAND protocol memories. Can be mapped on any of the chip-selects.	-
nWP	GPMC0_WPn	O	Write protect (active low)	-
DIR	GPMC0_DIR	O	This signal can be used to control an external buffer direction. Also controls the signal direction of D[15-0]. Low during transmit (for write access: data OUT from GPMC0 to memory). High during receive (for read access: data IN from memory to GPMC0).	-

(1) I = Input; O = Output; I/O - Bidirectional
(2) HiZ = High Impedance

Note

For GPMC output clock signal (CLK) to work properly, the RXACTIVE bit of the appropriate CTRLMMR_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Table 12-171 shows the use of address and data GPMC pins based on the type of external device.

Table 12-171. GPMC Pin Multiplexing Options

GPMC Pin	Multiplexed Address Data 32-Bit Device	Multiplexed Address Data 16-Bit Device	non-multiplexed Address Data 16-Bit Device (incomplete 28-bit address range)	non-multiplexed Address Data 8-Bit Device (incomplete 28-bit address range)	16-Bit NAND Device	8-Bit NAND Device
GPMC0_A[22]	Not used	Not used	A22	A22	Not used	Not used
GPMC0_A[21]	Not used	Not used	A21	A21	Not used	Not used
GPMC0_A[20]	Not used	Not used	A20	A20	Not used	Not used
GPMC0_A[19]	Not used	Not used	A19	A19	Not used	Not used
GPMC0_A[18]	Not used	Not used	A18	A18	Not used	Not used
GPMC0_A[17]	Not used	Not used	A17	A17	Not used	Not used
GPMC0_A[16]	Not used	Not used	A16	A16	Not used	Not used
GPMC0_A[15]	Not used	Not used	A15	A15	Not used	Not used
GPMC0_A[14]	Not used	Not used	A14	A14	Not used	Not used
GPMC0_A[13]	Not used	Not used	A13	A13	Not used	Not used
GPMC0_A[12]	Not used	Not used	A12	A12	Not used	Not used
GPMC0_A[11]	Not used	Not used	A11	A11	Not used	Not used
GPMC0_A[10]	Not used	A26	A10	A10	Not used	Not used
GPMC0_A[9]	Not used	A25	A9	A9	Not used	Not used
GPMC0_A[8]	Not used	A24	A8	A8	Not used	Not used
GPMC0_A[7]	Not used	A23	A7	A7	Not used	Not used
GPMC0_A[6]	Not used	A22	A6	A6	Not used	Not used
GPMC0_A[5]	Not used	A21	A5	A5	Not used	Not used
GPMC0_A[4]	Not used	A20	A4	A4	Not used	Not used
GPMC0_A[3]	Not used	A19	A3	A3	Not used	Not used
GPMC0_A[2]	Not used	A18	A2	A2	Not used	Not used

Copyright © 2026 Texas Instruments Incorporated

Table 12-171. GPMC Pin Multiplexing Options (continued)

GPMC Pin	Multiplexed Address Data 32-Bit Device	Multiplexed Address Data 16-Bit Device	non-multiplexed Address Data 16-Bit Device (incomplete 28-bit address range)	non-multiplexed Address Data 8-Bit Device (incomplete 28-bit address range)	16-Bit NAND Device	8-Bit NAND Device
GPMC0_A[1]	Not used	A17	A1	A1	Not used	Not used
GPMC0_A[0](1)	Not used	A0 - Not used	Not used	A0	Not used	Not used
GPMC0_AD[31]	D31	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[30]	D30	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[29]	D29	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[28]	D28	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[27]	D27	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[26]	D26	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[25]	A27/D25	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[24]	A26/D24	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[23]	A25/D23	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[22]	A24/D22	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[21]	A23/D21	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[20]	A22/D20	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[19]	A21/D19	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[18]	A20/D18	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[17]	A19/D17	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[16]	A18/D16	Not used	Not used	Not used	Not used	Not used
GPMC0_AD[15]	A17/D15	A16/D15	D15	Not used	D15	Not used
GPMC0_AD[14]	A16/D14	A15/D14	D14	Not used	D14	Not used
GPMC0_AD[13]	A15/D13	A14/D13	D13	Not used	D13	Not used
GPMC0_AD[12]	A14/D12	A13/D12	D12	Not used	D12	Not used
GPMC0_AD[11]	A13/D11	A12/D11	D11	Not used	D11	Not used
GPMC0_AD[10]	A12/D10	A11/D10	D10	Not used	D10	Not used
GPMC0_AD[9]	A11/D9	A10/D9	D9	Not used	D9	Not used
GPMC0_AD[8]	A10/D8	A9/D8	D8	Not used	D8	Not used
GPMC0_AD[7]	A9/D7	A8/D7	D7	D7	D7	D7
GPMC0_AD[6]	A8/D6	A7/D6	D6	D6	D6	D6
GPMC0_AD[5]	A7/D5	A6/D5	D5	D5	D5	D5
GPMC0_AD[4]	A6/D4	A5/D4	D4	D4	D4	D4
GPMC0_AD[3]	A5/D3	A4/D3	D3	D3	D3	D3
GPMC0_AD[2]	A4/D2	A3/D2	D2	D2	D2	D2
GPMC0_AD[1]	A3/D1	A2/D1	D1	D1	D1	D1
GPMC0_AD[0]	A2/D0	A1/D0	D0	D0	D0	D0

(1) Used to effectively address 8-bit (only) non-multiplexed memories

With all device types, the GPMC does not drive unnecessary address lines. They stay at their reset value of 0x0.

Address mapping supports address/data-multiplexed 16- or 32-bit-wide devices:

- The NOR flash memory controller still supports non-multiplexed address and data memory devices.
- Multiplexing mode can be selected through the GPMC_CONFIG1_i[9-8] MUXADDDATA bit field (where i = 0 to 3).

Copyright © 2026 Texas Instruments Incorporated

12.3.3.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4 GPMC Functional Description

The GPMC basic programming model offers maximum flexibility to support various access protocols for each of the four configurable chip-selects. Use optimal chip-select settings, based on the characteristics of the external device:

- Different protocols can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- The address and data bus can be multiplexed on the same external bus.
- Read and write access can be independently defined as asynchronous or synchronous.
- System requests (byte, 16-bit word, burst) are performed through single or multiple accesses. External access profiles (single, multiple with optimized burst length, native- or emulated-wrap) are based on external device characteristics (supported protocol, bus width, data buffer size, native-wrap support).
- System burst read or write requests are synchronous-burst (multiple-read or multiple-write). When neither burst nor page mode is supported by external memory or ASIC devices, system burst read or write requests are translated to successive single synchronous or asynchronous accesses (single reads or single writes). 8-bit wide devices are supported only in single synchronous or single asynchronous read or write mode.
- To simulate a programmable internal-wait-state, an external WAIT pin can be monitored to dynamically control external access at the beginning (initial access time) of and during a burst access.

Each control signal is controlled independently for each chip-select. The internal functional clock of the GPMC (GPMC_FCLK) is used as a time reference to specify the following:

- Read- and write-access duration
- Most GPMC external interface control-signal assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time between accesses, when required

12.3.3.4.1 GPMC Block Diagram

Figure 12-150 shows the GPMC functional block diagram. The GPMC consists of six blocks:

- Interconnect port interface
- Address decoder, GPMC configuration, and chip-select configuration register file
- Access engine
- Prefetch and write-posting engine
- Error correction code engine (ECC)
- External device/memory port interface

Copyright © 2026 Texas Instruments Incorporated

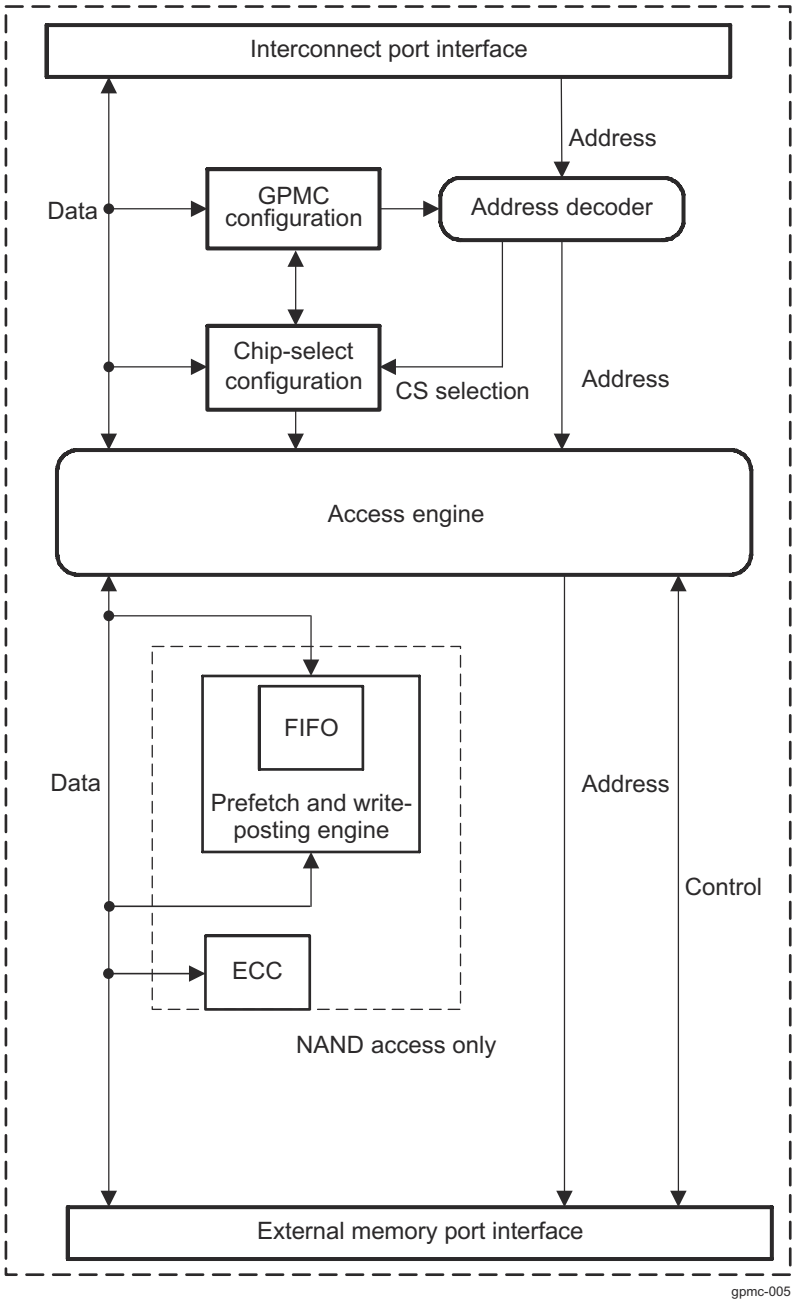


Figure 12-150. GPMC Block Diagram

The GPMC can access various external devices. The flexible programming model allows a wide range of attached device types and access schemes.

Based on the programmed configuration bit fields stored in the GPMC registers, the GPMC can generate the timing of all control signals depending on the attached device and access type.

Given the chip-select decoding and its associated configuration registers, the GPMC selects the appropriate control signal timing for the device type.

12.3.3.4.2 GPMC Clock Configuration

Table 12-172 describes the GPMC clocks.

Copyright © 2026 Texas Instruments Incorporated

Table 12-172. GPMC Clocks

Signal	I/O(1)	Description
GPMC_FCLK	I	Functional clock
GPMC_ICLK	I	Interface clock
CLK (GPMC_CLKOUT pin)	O	External clock provided to synchronous external memory devices and to DCC5 in the device.

(1) I = Input; O = Output; I/O - Bidirectional

The GPMC output clock (CLK) is generated by the GPMC from the internal GPMC_FCLK clock. The source of the GPMC_FCLK is described in *GPMC0 Clocks*. The GPMC output clock is configured using the GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER bit field (where i = 0 to 3), as shown in Table 12-173.

Table 12-173. GPMC Output Clock Configuration

Source Clock	GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER	GPMC Output Clock Provided to External Memory Device
GPMC_FCLK	00	GPMC_FCLK
	01	GPMC_FCLK/2
	10	GPMC_FCLK/3
	11	GPMC_FCLK/4

When using synchronous interface protocols, the GPMC output clock (CLK), toggles only during the read or write access cycle. In some applications, it may be desirable to have a continuous clock running at the GPMC interface clock frequency for clocking attached devices. This option is enabled by an optional clock path from the GPMC functional clock input (GPMC_FCLK) to the GPMC_FCLK_MUX pin. This output clock, to GPMC_FCLK_MUX pin, can be selected through the standard MUXMODE selection of the GPMC_FCLK_MUX pin PADCONFIG control register.

Note that when using such synchronous interface protocols with the continuous clock option, user should ensure that the GPMC outputs are timed to the same frequency (GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER = 0).

12.3.3.4.3 GPMC Power Management

Table 12-174 describes the local power-management features available for the GPMC module.

Table 12-174. GPMC Local Power-Management Features

Feature	Registers	Description
Clock autogating	*GPMC_SYSCONFIG*[0] AUTOIDLE	This bit allows a local power optimization inside the module, by gating the GPMC_ICLK clock upon the internal activity.
Target idle modes	*GPMC_SYSCONFIG*[4-3] SIDLEMODE	Force-idle, no-idle and smart-idle modes are available.

12.3.3.4.4 GPMC Interrupt Requests

The GPMC generates one interrupt request (see *GPMC0 Hardware Requests*).

Table 12-175 lists the event flags, and their mask, that can cause module interrupts.

Table 12-175. GPMC Interrupt Events

Event Flag	Event Mask	Sensitivity	Description
GPMC_IRQSTATUS[9] WAIT1EDGE DETECTIONSTATUS	GPMC_IRQENABLE[9] WAIT1EDGE DETECTIONENABLE	Edge	Wait1 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT1 signal. The rising or falling edge detection of Wait1 is selected through the GPMC_CONFIG[9] WAIT1PINPOLARITY bit.

Copyright © 2026 Texas Instruments Incorporated

Table 12-175. GPMC Interrupt Events (continued)

Event Flag	Event Mask	Sensitivity	Description
GPMC_IRQSTATUS[8] WAIT0EDGE DETECTIONSTATUS	GPMC_IRQENABLE[8] WAIT0EDGE DETECTIONENABLE	Edge	Wait0 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT0 signal. The rising or falling edge detection of Wait0 is selected through the GPMC_CONFIG[8] WAIT0PINPOLARITY bit.
GPMC_IRQSTATUS[1] TERMINAL COUNTSTATUS	GPMC_IRQENABLE[1] TERMINAL COUNTENABLE	Level	Terminal count event: Triggered on prefetch process completion; that is, when the number of currently remaining data to be requested reaches 0.
GPMC_IRQSTATUS[0] FIFOEVENTSTATUS	GPMC_IRQENABLE[0] FIFOEVENTENABLE	Level	FIFO event interrupt: Indicates available FIFO levels for write-posting mode and prefetch mode. GPMC_PREFETCH_CONFIG1[2] DMAMODE must be set to 0.

12.3.3.4.5 GPMC Interconnect Port Interface

Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The GPMC interconnect interface is a pipelined interface including a 16 × 32-bit word write buffer.

Any system host can issue external access requests through the GPMC.

The device system can issue the following requests through this interface:

- One 8-, 16-, or 32-bit interconnect access (read/write)
- Two incrementing 32-bit interconnect accesses (read/write)
- Two wrapped 32-bit interconnect accesses (read/write)
- Four incrementing 32-bit interconnect accesses (read/write)
- Four wrapped 32-bit interconnect accesses (read/write)
- Eight incrementing 32-bit interconnect accesses (read/write)
- Eight wrapped 32-bit interconnect accesses (read/write)

Only linear burst transactions are supported; interleaved burst transactions are not supported. Only power-of-two-length precise bursts 2 × 32, 4 × 32, 8 × 32, and 16 × 32, with the burst base address aligned on the total burst size, are supported (this limitation applies to incrementing bursts only).

This interface also provides one interrupt and one DMA request line for specific event control.

It is recommended to program the *GPMC_CONFIG1_i*[24-23] ATTACHEDDEVICEPAGELENGTH bit field according to the page length of the effective attached device and to enable the *GPMC_CONFIG1_i*[31] WRAPBURST bit if the attached device supports wrapping burst.

It is possible, however, to emulate wrapping burst on a nonwrapping memory by providing relevant addresses within the page or by splitting transactions. Bursts larger than the memory page length are chopped into multiple burst transactions. Because of the alignment requirements, a page boundary is never crossed.

12.3.3.4.6 GPMC Address and Data Bus

The current application supports GPMC connection to NAND devices and to address/data-multiplexed memories or devices. Connection to address/data-non-multiplexed memories or devices is supported with an address range of 256MB.

Depending on the GPMC configuration of each chip-select, address and data bus lines that are not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

- For address/data-multiplexed and AAD-multiplexed NOR devices, the address is multiplexed on the data bus.

Copyright © 2026 Texas Instruments Incorporated

- 8-bit-wide NOR devices do not use GPMC I/O: GPMC_AD[15-8] for data (they are used for address if needed).
- 16-bit-wide NAND devices do not use GPMC I/O: GPMC_A[22-0].
- 8-bit-wide NAND devices do not use GPMC I/O: GPMC_A[22-0] and GPMC I/O: GPMC_AD[15-8].

12.3.3.4.6.1 GPMC I/O Configuration Setting

Note

In this section and the following sections, the i in GPMC_CONFIGx_i stands for the GPMC chip-select i, where i = 0 to 3.

To select a NAND device, program the following register fields:

- GPMC_CONFIG1_i[11-10] DEVICETYPE = 0b10
- GPMC_CONFIG1_i[9-8] MUXADDDATA = 0b00

To select an address/data-multiplexed device, program the following register fields:

- GPMC_CONFIG1_i[11-10] DEVICETYPE = 0b00
- GPMC_CONFIG1_i[9-8] MUXADDDATA = 0b10

To select an address/address/data-multiplexed device, program the following register fields:

- GPMC_CONFIG1_i[11-10] DEVICETYPE = 0b00
- GPMC_CONFIG1_i[9-8] MUXADDDATA = 0b01

To select an address/data-non-multiplexed device, program the following register fields:

- GPMC_CONFIG1_i[11-10] DEVICETYPE = 0b00
- GPMC_CONFIG1_i[9-8] MUXADDDATA = 0b00

12.3.3.4.7 GPMC Address Decoder and Chip-Select Configuration

Addresses are decoded accordingly with the address request of the chip-select and the content of the chip-select base address register file, which includes a set of global GPMC configuration registers and four sets of chip-select configuration registers.

The GPMC configuration register file is memory-mapped and can be read or written with byte, 16-bit word, or 32-bit word accesses. The register file must be configured as a noncacheable, nonbufferable region to prevent any desynchronization between host execution (write request) and the completion of register configuration (write completed with register updated).

After the chip-select is configured, the access engine accesses the external device, drives the external interface control signals, and applies the interface protocol based on user-defined timing parameters and settings.

12.3.3.4.7.1 Chip-Select Base Address and Region Size

Any external memory or ASIC device attached to the GPMC external interface can be accessed by any device system host within the GPMC 128MB address space. For more information, see *Memory Map*.

Note

Even though GPMC supports total address space of 1GB, only 128MB are physically available in this device.

The GPMC 128MB address space can be divided into a maximum of four chip-select regions with programmable base address and programmable chip-select size. The chip-select size is programmable from 16MB to 256MB (must be a power-of-two) and is defined by the mask field. Attached memory smaller than the programmed chip-select region size is accessed through the entire chip-select region (aliasing).

Each chip-select has a 6-bit base address encoding and 4-bit decoding mask, which must be programmed according to the following rules:

Copyright © 2026 Texas Instruments Incorporated

- The programmed chip-select region base address must be aligned on the chip-select region size address boundary and is limited to a power-of-two address value. During access decoding, the value of the register base address is used to compare the address with the address bit line mapping, as shown in Figure 12-151 (with A0 as the device system byte-address line). The base address is programmed through the GPMC_CONFIG7_i[5-0] BASEADDRESS bit field.
- The register mask is used to exclude some address lines from the decoding. A register mask bit field set to 0 suppresses the associated address line from the address comparison (incoming address bit line is don't care). The value of the register mask must be limited to the subsequent value, based on the desired chip-select region size. Any other value has an undefined result. When multiple chip-select regions with overlapping addresses are enabled concurrently, access to these chip-select regions is cancelled and a GPMC access error is posted. The mask field is programmed through the GPMC_CONFIG7_i[11-8] MASKADDRESS bit field.

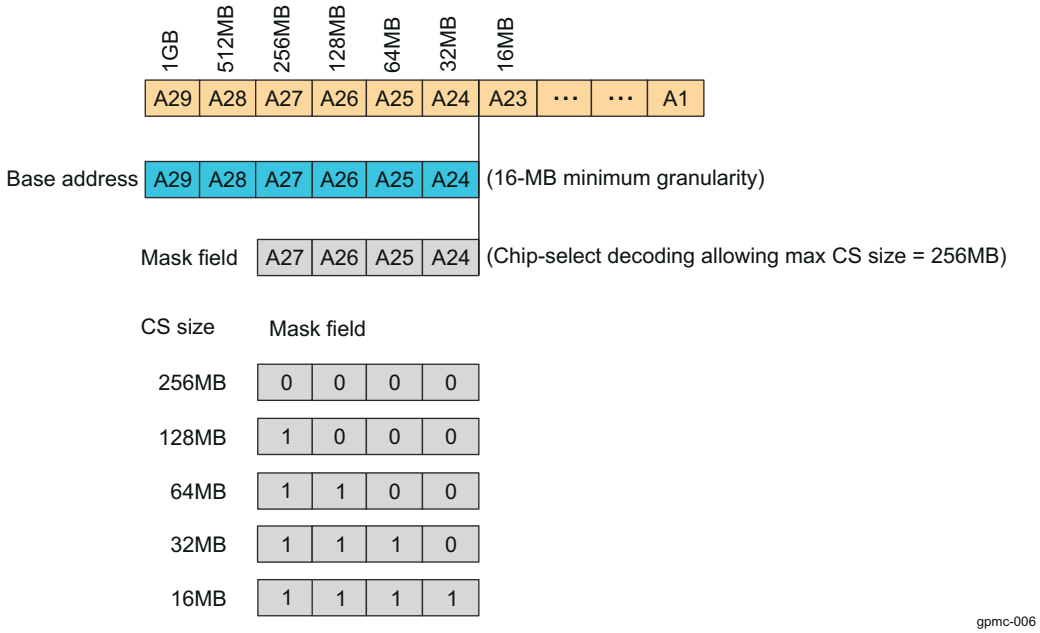


Figure 12-151. Chip-Select Address Mapping and Decoding Mask

For example, to map the 128MB address space (from 2000 0000h to 27FF FFFFh), the GPMC_CONFIG7_i[5-0] BASEADDRESS bit field should be set to 20h.

Chip-select configuration (base and mask address or any protocol and timing settings) must be performed while the associated chip-select is disabled through the GPMC_CONFIG7_i[6] CSVALID bit (where i stands for the GPMC chip-select i, where i = 0 to 3). In addition, a chip-select configuration can be disabled only if there is no ongoing access to that chip-select. This requires monitoring the activity of the prefetch or write-posting engine if the engine is active on the chip-select. Also, the write buffer state must be monitored to wait for any posted write completion to the chip-select.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select region) is not propagated to the external interface and a GPMC access error is posted. In case of overlapping chip-selects, an error is generated and no access occurs on either chip-select.

CS0 is the only chip-select region enabled after a power up or GPMC reset.

Copyright © 2026 Texas Instruments Incorporated

CAUTION

Although the GPMC interface can drive up to four chip-selects, the frequency specified for this interface is for a specific load. If this load is exceeded, the maximum frequency cannot be reached. One solution is to implement a board with buffers to allow the slowest device to maintain the total load on the lines at the value specified in the device-specific Datasheet.

12.3.3.4.7.2 Access Protocol

12.3.3.4.7.2.1 Supported Devices

The access protocol of each chip-select can be independently specified through the *GPMC_CONFIG1_i*[11-10] DEVICETYPE parameter (where i = 0 to 3) for:

- Random-access synchronous or asynchronous memory, such as NOR flash and SRAM
- NAND flash asynchronous devices

Note

For more information about the NAND flash GPMC basic programming model and NAND support, see Section 12.3.3.4.11, *GPMC NAND Device Basic Programming Model*, and Section 12.3.3.4.11.1, *NAND Memory Device in Byte or Word16 Stream Mode*.

12.3.3.4.7.2.2 Access Size Adaptation and Device Width

Each chip-select can be independently configured through the *GPMC_CONFIG1_i*[13-12] DEVICESIZE bit field (where i = 0 to 3) to interface with a 16- or 8-bit-wide device. System requests with data width greater than the external device data bus width are split into successive accesses according to the external device data-bus width and little-endian data organization.

12.3.3.4.7.2.3 Address/Data-Multiplexing Interface

For random synchronous or asynchronous memory interfacing (DEVICETYPE = 0b00), an address- and data-multiplexing protocol can be selected through the *GPMC_CONFIG1_i*[9-8] MUXADDDATA bit field (where i = 0 to 3). The nADV signal must be used as the external device address latch control signal. For the associated chip-select configuration, nADV assertion and deassertion time and nOE assertion time must be set to the appropriate value to meet the address latch setup/hold time requirements of the external device. See *Module Integration*.

Note

This address/data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data-multiplexing protocol. See Section 12.3.3.4.11, *NAND Device Basic Programming Model*.

12.3.3.4.7.3 External Signals

12.3.3.4.7.3.1 WAIT Pin Monitoring Control

GPMC access time can be dynamically controlled using an external GPMC_WAIT pin when the external device access time is not deterministic and cannot be defined and controlled using only the GPMC internal RDACCESSTIME, WRACCESSTIME, and PAGEBURSTACCESSTIME wait-state generator.

The GPMC features two input WAIT pins: GPMC_WAIT1, and GPMC_WAIT0. These pins allow control of external devices with different WAIT pin polarity. They also allow the overlap of WAIT pin assertion from different devices without affecting access to devices for which the WAIT pin is not asserted.

- The GPMC_CONFIG1_i[17-16] WAITPINSELECT bit field (where i = 0 to 3) selects which input GPMC_WAIT pin is used for the device attached to the corresponding chip-select.

Copyright © 2026 Texas Instruments Incorporated

- The polarity of the WAIT pin is defined through the WAITxPINPOLARITY bit of the GPMC_CONFIG register. A WAIT pin configured to be active low means that low level on the WAIT signal indicates that the data is not ready and that the data bus is invalid. When a WAIT pin is inactive, data is valid.

The GPMC access engine can be configured per chip-select to monitor or not the WAIT pin of the external memory device, based on the access type: read or write.

- The GPMC_CONFIG1_i[22] WAITREADMONITORING bit defines whether or not the WAIT pin must be monitored during read accesses.
- The GPMC_CONFIG1_i[21] WAITWRITEMONITORING bit defines whether or not the WAIT pin must be monitored during write accesses.

The GPMC access engine can be configured to monitor the WAIT pin of the external memory device asynchronously or synchronously with the GPMC output clock, depending on the access type: synchronous or asynchronous (the GPMC_CONFIG1_i[29] READTYPE and GPMC_CONFIG1_i[27] WRITETYPE bits).

12.3.3.4.7.3.1.1 Wait Monitoring During Asynchronous Read Access

When WAIT pin monitoring is enabled for read accesses (WAITREADMONITORING), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state.

During asynchronous read accesses with WAIT pin monitoring enabled, the WAIT pin must be at a valid level (asserted or deasserted) for at least two GPMC clock cycles before RDACCESSTIME completes, to ensure correct dynamic access-time control through WAIT pin monitoring. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

In this context, RDACCESSTIME is used as a wait invalid timing window and is set to such a value that the WAIT pin is at a valid state two GPMC clock cycles before RDACCESSTIME completes.

Similarly, during a multiple-access cycle (for example, asynchronous read page mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-deasserted state. Wait monitoring pipelining is also applicable to multiple accesses (access within a page).

- Wait monitored as active freezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, wait monitored as asserted extends the current access time in the page. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, wait monitored as inactive completes the current access time and starts the next access phase in the page. The data bus is considered valid, and data are captured during this clock cycle. In case of a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their related control timing value and according to the CYCLETIME counter status.

When a delay larger than two GPMC clocks must be observed between wait-pin deactivation time and data valid time (including the required GPMC and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data-capture time and the effective unlock of the CYCLETIME counter. This extra delay can be programmed in the *GPMC_CONFIG1_i*[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3).

Note

- The WAITMONITORINGTIME parameter does not delay the WAIT pin active or inactive detection, nor does it modify the two GPMC clocks pipelined detection delay.
- This extra delay is expressed as a number of GPMC output clock cycles, even though the access is defined as asynchronous, and no GPMC output clock is provided to the external device. Still, because *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER is used as a divider for the GPMC clock, it must be programmed to define the correct WAITMONITORINGTIME delay.

Figure 12-152 shows wait behavior during an asynchronous single read access.

Copyright © 2026 Texas Instruments Incorporated

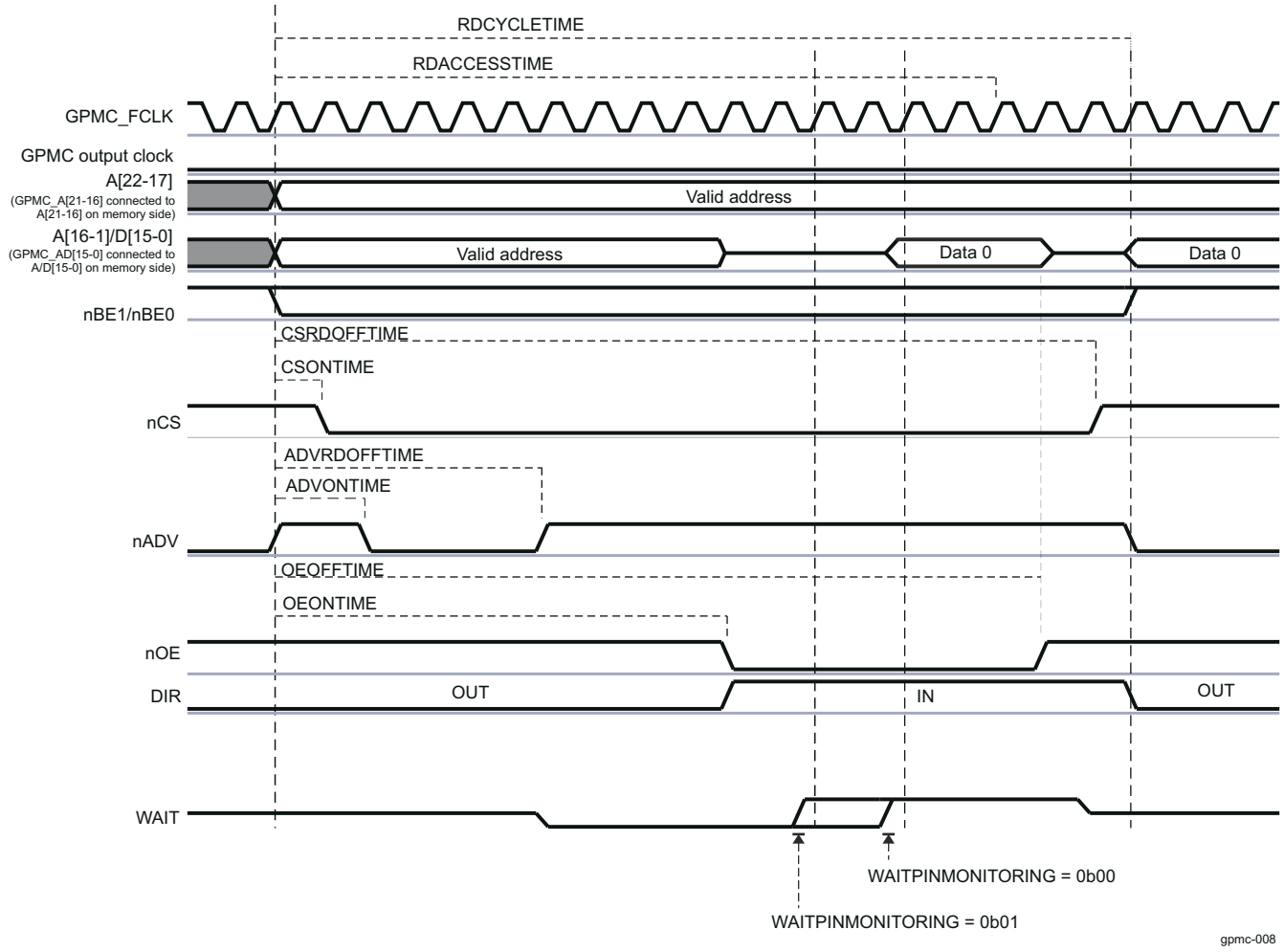


Figure 12-152. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)

Note

The WAIT signal is active low. *GPMC_CONFIG1_i*[19-18] WAITMONITORINGTIME = 0b00, or 0b01.

12.3.3.4.7.3.1.2 Wait Monitoring During Asynchronous Write Access

When WAIT pin monitoring is enabled for write accesses (*GPMC_CONFIG1_i*[21] WAITWRITEMONITORING bit = 0x1), the wait invalid timing window is defined by the WRACCESSTIME field. WRACCESSTIME must be set so that the WAIT pin is at a valid state two GPMC clock cycles before WRACCESSTIME completes. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

- Wait monitored as active freezes the CYCLETIME counter. This informs the GPMC that the data bus is not captured by the external device. The control signals are kept in their current state. The data bus still drives the data.
- Wait monitored as inactive unfreezes the CYCLETIME counter. This informs that the data bus is correctly captured by the external device. All signals, including the data bus, are controlled according to their related control timing value and to the CYCLETIME counter status.

When a delay larger than two GPMC clock cycles must be observed between wait-pin deassertion time and the effective data write into the external device (including the required GPMC data setup time and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data write

Copyright © 2026 Texas Instruments Incorporated

time into the external device and the effective unfreezing of the CYCLETIME counter. This extra delay can be programmed in the *GPMC_CONFIG1_i*[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3).

Note

- The WAITMONITORINGTIME parameter does not delay the WAIT pin assertion or deassertion detection, nor does it modify the two GPMC clock cycles pipelined detection delay.
- This extra delay is expressed as a number of GPMC output clock cycles, even though the access is defined as asynchronous, and even though no clock is provided to the external device. Still, because the *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER bit field is used as a divider for the GPMC clock, it must be programmed to define the correct WAITMONITORINGTIME delay.

12.3.3.4.7.3.1.3 Wait Monitoring During Synchronous Read Access

During synchronous accesses with WAIT pin monitoring enabled, the WAIT pin is captured synchronously with GPMC output clock, using the rising edge of this clock.

The WAIT signal can be programmed to apply to the same clock cycle in which it is captured. Alternatively, it can be sampled one or two GPMC output clock cycles ahead of the clock cycle to which it applies. This pipelining is applicable to the entire burst access and to all data phases in the burst access. This wait pipelining depth is programmed in the GPMC_CONFIG1_i[19-18] WAITMONITORINGTIME bit field (where i = 0 to 3), and is expressed as a number of GPMC output clock cycles.

In synchronous mode, when WAIT pin monitoring is enabled (the GPMC_CONFIG1_i[22] WAITREADMONITORING bit), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state detection.

Depending on the programmed value of WAITMONITORINGTIME, the WAIT pin must be at a valid level, either asserted or deasserted:

- In the same clock cycle the data is valid if WAITMONITORINGTIME = 0 (at RDACCESSTIME completion)
- In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC_FCLK clock cycles before RDACCESSTIME completion if WAITMONITORINGTIME is not equal to 0

Similarly, during a multiple-access cycle (burst mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the WAIT-INACTIVE state. The wait pipelining-depth programming applies to the whole burst access.

- Wait monitored as active freezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in a lock state), wait monitored as active extends the current access time in the burst. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in lock state), wait monitored as inactive completes the current access time and starts the next access phase in the burst. The data bus is considered valid, and data are captured during this clock cycle. In a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their relative control timing value and the CYCLETIME counter status.

Figure 12-153 shows wait behavior during a synchronous read burst access.

Copyright © 2026 Texas Instruments Incorporated

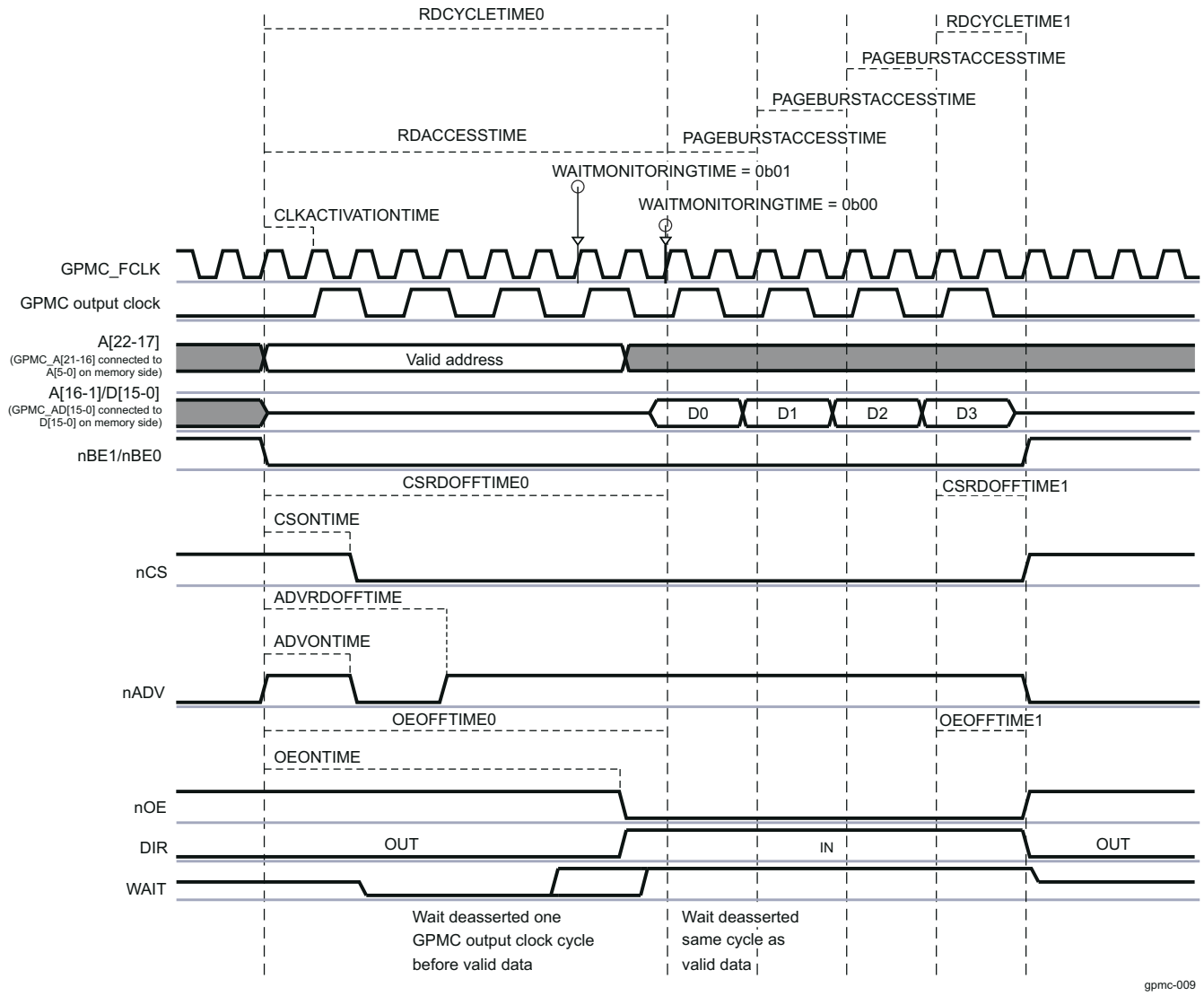


Figure 12-153. Wait Behavior During a Synchronous Read Burst Access

Note

The WAIT signal is active low. WAITMONITORINGTIME = 0b00, 0b01.

12.3.3.4.7.3.1.4 Wait Monitoring During Synchronous Write Access

During synchronous accesses with WAIT pin monitoring enabled (the *GPMC_CONFIG1_i*[21] WAITWRITEMONITORING bit), the WAIT pin is captured synchronously with GPMC output clock, using the rising edge of this clock.

If enabled, external WAIT pin monitoring can be used in combination with WRACCESSTIME to delay the GPMC output clock capture edge of the effective memory device.

WAIT-monitoring pipelining depth is similar to synchronous read access:

- At WRACCESSTIME completion if WAITMONITORINGTIME = 0
- In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC_FCLK cycles before WRACCESSTIME completion if WAITMONITORINGTIME is not equal to 0

Wait-monitoring pipelining definition applies to whole burst accesses:

Copyright © 2026 Texas Instruments Incorporated

- Wait monitored as active freezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, wait monitored as active indicates that the data bus is not being captured by the external device. Control signals are kept in their current state. The data bus is kept in its current state.
- Wait monitored as inactive unfreezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, wait monitored as inactive indicates the effective data capture of the bus by the external device and starts the next access of the burst. In case of a single access or if this was the last access in a multiple access cycle, all signals, including the data bus, are controlled according to their related control timing value and the CYCLETIME counter status.

Note

WAIT monitoring is supported for all configurations except *GPMC_CONFIG1_i*[19-18] WAITMONITORINGTIME = 0x0 (where i = 0 to 3) for write bursts with a clock divider of 1 or 2 (the *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER bit field is equal to 0x0 or 0x1, respectively).

12.2.3.3.4.7.3.1.5 Wait With NAND Device

For information about the use of the WAIT pin for communication with a NAND flash external device, see Section 12.3.3.4.11.2, *NAND Device-Ready Pin*.

12.2.3.3.4.7.3.1.6 Idle Cycle Control Between Successive Accesses

12.2.3.3.4.7.3.1.6.1 Bus Turnaround (BUSTURNAROUND)

To prevent data-bus contention, an access that follows a read access to a slow memory/device must be delayed (in other words, control the nCS/nOE deassertion to data bus in high-impedance delay).

The bus turnaround is a time-out counter starting after nCS or nOE deassertion time, whichever occurs first, and delays the next access start-cycle time. The counter is programmed through the *GPMC_CONFIG6_i*[3-0] BUSTURNAROUND bit field (where i = 0 to 3).

After a read access to a chip-select with a nonzero BUSTURNAROUND, the next access is delayed until the BUSTURNAROUND delay completes, if the next access is one of the following:

- A write access to any chip-select (the same or different chip-select from which the data was read)
- A read access to a different chip-select than the chip-select from which the data was read access
- A read or write access to a chip-select associated with an address/data-multiplexed device

Note

Bus keeping starts after bus turnaround completion so that DIR changes from IN to OUT after bus turnaround. The bus does not have enough time to go into high-impedance even though it can be driven with the same value before bus turnaround timing.

BUSTURNAROUND delay runs in parallel with *GPMC_CONFIG6_i*[3-0] CYCLE2CYCLEDELAY bit field delays. BUSTURNAROUND is a timing parameter for the ending chip-select access, while CYCLE2CYCLEDELAY is a timing parameter for the following chip-select access. The effective minimum delay between successive accesses is driven by these delay timing parameters and by the access type of the following access (see Figure 12-154 through Figure 12-156).

Another way to prevent bus contention is to define an earlier nCS or nOE deassertion time for slow devices or to extend the value of RDCYCLETIME. Doing this prevents bus contention, but it also affects all accesses of this specific chip-select.

Copyright © 2026 Texas Instruments Incorporated

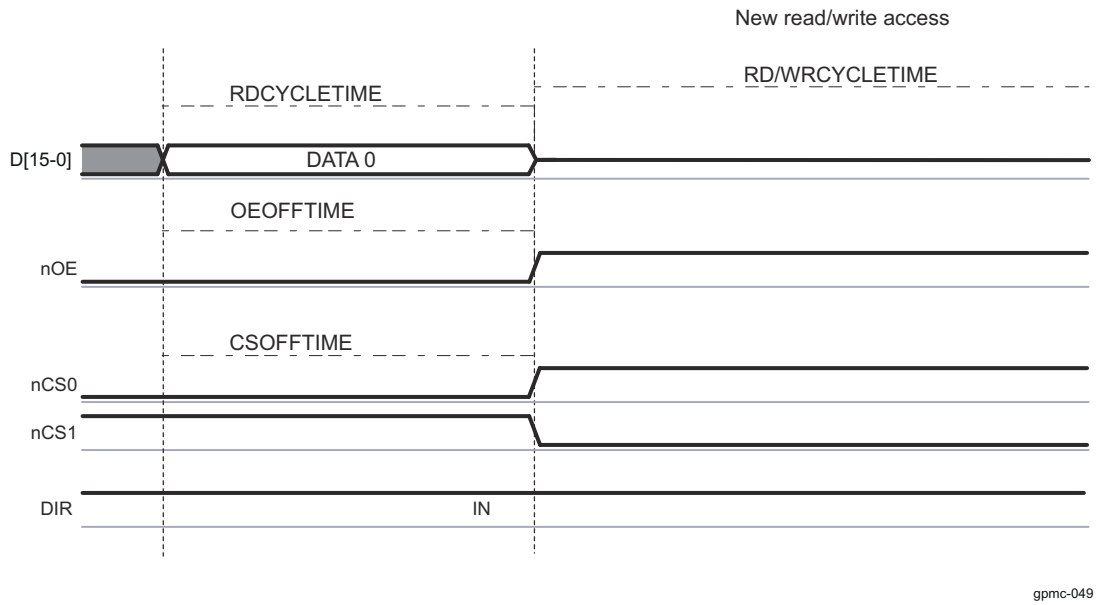


Figure 12-154. Read-to-Read for an Address-Data Multiplexed Device, on Different Chip-Select, Without Bus Turnaround (nCS Attached to a Fast Device)

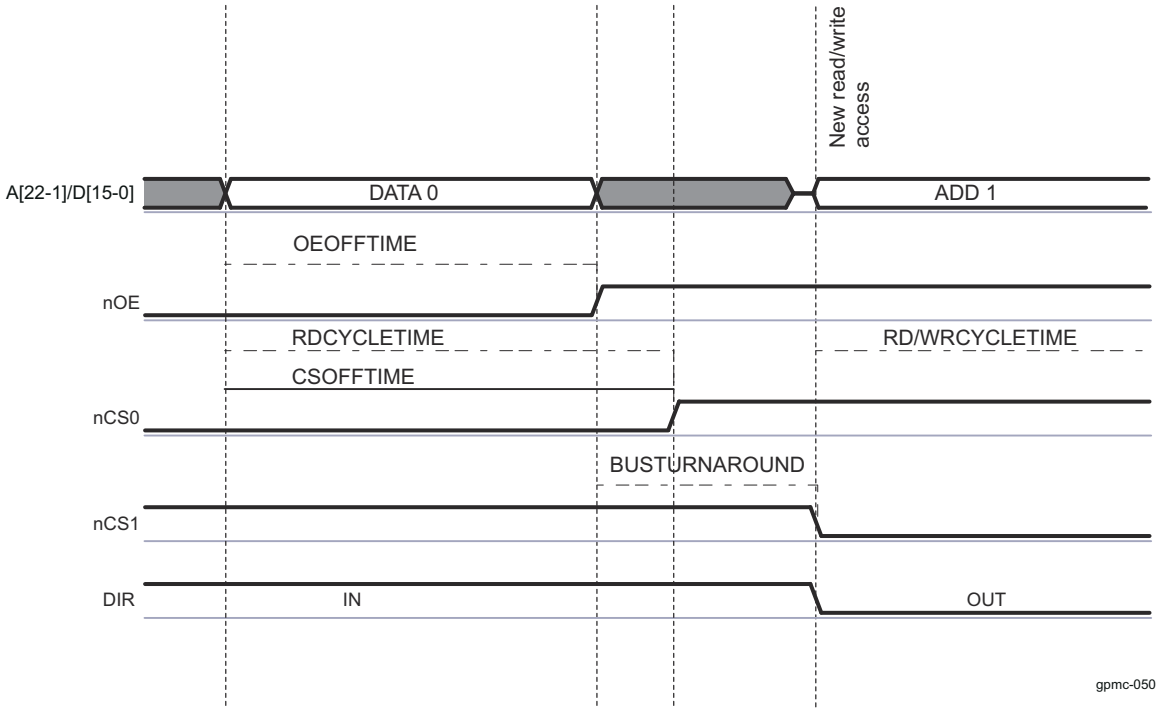


Figure 12-155. Read- to-Read/Write for an Address-Data Multiplexed Device, on Different Chip-Select, With Bus Turnaround

Copyright © 2026 Texas Instruments Incorporated

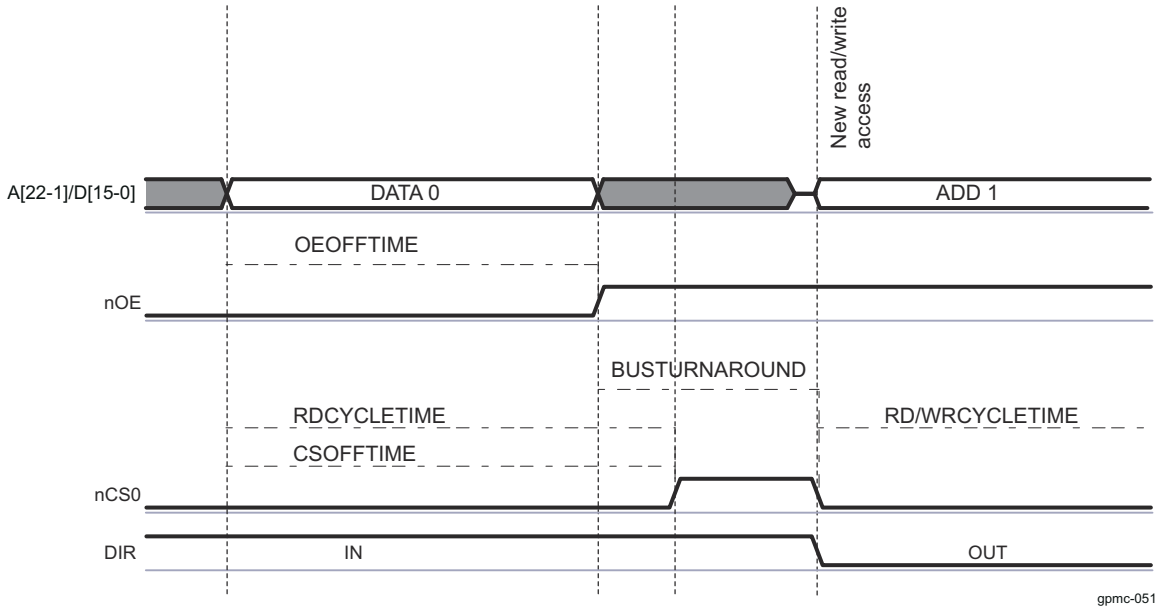


Figure 12-156. Read-to-Read/Write for a Address-Data or AAD-Multiplexed Device, on Same Chip-Select, With Bus Turnaround

12.2.3.3.4.7.3.1.6.2 Idle Cycles Between Accesses to Same Chip-Select (CYCLE2CYCLESAMECSEN, CYCLE2CYCLEDELAY)

Some devices require a minimum chip-select signal inactive time between accesses. The *GPMC_CONFIG6_i*[7] CYCLE2CYCLESAMECSEN bit (where i = 0 to 3) enables insertion of a minimum number of GPMC_FCLK cycles, defined by the *GPMC_CONFIG6_i*[11-8] CYCLE2CYCLEDELAY bit field, between successive accesses of any type (read or write) to the same chip-select.

If CYCLE2CYCLESAMECSEN is enabled, any subsequent access to the same chip-select is delayed until its CYCLE2CYCLEDELAY completes. The CYCLE2CYCLEDELAY counter starts when CSRDOFFTIME/CSWROFFTIME completes.

The same applies to successive accesses occurring during 32-bit word or burst accesses split into successive single accesses when the single-access mode is used (*GPMC_CONFIG1_i*[30] READMULTIPLE = 0 or *GPMC_CONFIG1_i*[28] WRITEMULTIPLE = 0).

All control signals (CS, ADV#/ALE, BE0#/CLE, WE#, and GPMC output clock (CLK)) are kept inactive (ADV#/ALE, BE0#/CLE, and GPMC output clock at low level; and CS, OE#/RE, and WE at high level) during the idle GPMC_FCLK cycles. This prevents back-to-back accesses to the same chip-select without idle cycles between accesses.

12.2.3.3.4.7.3.1.6.3 Idle Cycles Between Accesses to Different Chip-Select (CYCLE2CYCLEDIFFCSEN, CYCLE2CYCLEDELAY)

Because of the pipelined behavior of the system, successive accesses to different chip-selects can occur back-to-back with no idle cycles between accesses. Depending on the control signals (nCS, nADV/ALE, nBE0/CLE, nOE/RE, nWE) assertion and deassertion timing parameters and on the device timing parameters, the assertion times of some control signals may overlap between the successive accesses to a different chip-select. Similarly, some control signals (WE, OE/RE) may not respect required transition times.

To work around overlapping and to observe the required control-signal transitions, a minimum of CYCLE2CYCLEDELAY inactive cycles is inserted between the access being initiated to this chip-select and the previous access ending for a different chip-select. This applies to any type of access (read or write).

If the *GPMC_CONFIG6_i*[6] CYCLE2CYCLEDIFFCSEN bit is enabled, the chip-select access is delayed until CYCLE2CYCLEDELAY cycles have expired since the end of a previous access to a different chip-select.

Copyright © 2026 Texas Instruments Incorporated

CYCLE2CYCLEDELAY count starts at CSRDOFFTIME/CSWROFFTIME completion. All control signals are kept inactive during the idle GPMC_FCLK cycles.

Note

CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDIFFCSEN must be set in the *GPMC_CONFIG6_i* registers to get idle cycles inserted between accesses on this chip-select and after accesses to a different chip-select, respectively.

The CYCLE2CYCLEDELAY delay runs in parallel with the BUSTURNAROUND delay. The BUSTURNAROUND is a timing parameter defined for the ending chip-select access, whereas CYCLE2CYCLEDELAY is a timing parameter defined for the starting chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination, because bus turnaround does not apply to all access types. For more information about bus turnaround, see Section 3.3.4.7.3.1.6.1, *Bus Turnaround (BUSTURNAROUND)*.

Table 12-176 describes the configuration required for idle cycle insertion.

Table 12-176. Idle Cycle Insertion Configuration

1st Access Type	BUSTURNAROUND Timing Parameter	Second Access Type	Chip-Select	Add/Data Multiplexed	CYCLE2CYCLE SAMECSEN Parameter	CYCLE2CYCLE DIFFCSEN Parameter	Idle Cycle Insertion Between the Two Accesses
R/W	= 0	R/W	Any	Any	0	x	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Same	Nonmuxed	x	0	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Different	Nonmuxed	0	0	BUSTURNAROUND cycles are inserted.
R	> 0	R/W	Any	Muxed	0	0	BUSTURNAROUND cycles are inserted.
R	> 0	W	Any	Any	0	0	BUSTURNAROUND cycles are inserted.
W	> 0	R/W	Any	Any	0	0	No idle cycles are inserted if the two accesses are well pipelined.
R/W	= 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted.
R/W	= 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted.
R/W	> 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is max (BUSTURNAROUND, CYCLE2CYCLEDELAY).
R/W	> 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is maximum (BUSTURNAROUND, CYCLE2CYCLEDELAY).

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.7.3.1.7 Slow Device Support (TIMEPARAGRANULARITY Parameter)

All access-timing parameters can be multiplied by 2 by setting the *GPMC_CONFIG1_i*[4] TIMEPARAGRANULARITY bit (where i stands for the GPMC chip-select i, where i = 0 to 3). Increasing all access timing parameters allows support of slow devices.

12.3.3.4.7.3.2 DIR Pin

The DIR pin is used to control I/O direction on the GPMC data bus GPMC_D[15-0]. Depending on pad multiplexing, this signal can be output and used externally to the device, if required. The DIR pin is low during transmit (OUT) and high during receive (IN).

For write accesses, the DIR pin stays OUT from start-cycle time to end-cycle time.

For read accesses, the DIR pin goes from OUT to IN at nOE assertion time and stays IN until:

- BUSTURNAROUND is enabled
  - The DIR pin goes from IN to OUT at end-cycle time plus programmable bus turnaround time.
- BUSTURNAROUND is disabled
  - After an asynchronous read access, the DIR pin goes from IN to OUT at RDACCESSTIME + 1 GPMC_FCLK cycle or when RDCYCLETIME completes, whichever occurs last.
  - After a synchronous read access, the DIR pin goes from IN to OUT at RDACCESSTIME + 2 GPMC_FCLK cycles or when RDCYCLETIME completes, whichever occurs last.

Because of the bus-keeping feature of the GPMC, after a read or write access and with no other accesses pending, the default value of the DIR pin is OUT (see Section 12.3.3.4.8.10, *Bus Keeping Support*). In non-multiplexed devices, the DIR pin stays IN between two successive read accesses to prevent unnecessary toggling.

12.3.3.4.7.3.3 Reset

No reset signal is sent to the external memory device by the GPMC.

GPMC_RST is the reset signal for the GPMC module. It is connected and controlled by LPSC8 in VD_CORE. That reset signal initializes the internal state-machine and the internal configuration registers.

12.3.3.4.7.3.4 Write Protect Signal (nWP)

When connected to the attached memory device, the write protect signal can enable or disable the lockdown function of the attached memory. The nWP output pin value is controlled through the GPMC_CONFIG[4] WRITEPROTECT bit which is common for all chip selects.

12.3.3.4.7.3.5 Byte Enable (nBE1/nBE0)

Byte enable signals (nBE1/nBE0) are:

- Valid (asserted or nonasserted according to the incoming system request) from access start to access completion for asynchronous and synchronous single accesses
- Asserted low from access start to access completion for asynchronous and synchronous multiple read accesses
- Valid (asserted or nonasserted, according to the incoming system request) synchronously to each written data for synchronous multiple write accesses.

12.3.3.4.7.4 Error Handling

When an error occurs in the GPMC, the error information is stored in the GPMC_ERR_TYPE register and the address of the illegal access is stored in the GPMC_ERR_ADDRESS register. The GPMC keeps only the first error abort information until the GPMC_ERR_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the GPMC_ERR_TYPE[0] ERRORVALID bit.

- ERRORNOTSUPPADD occurs when an incoming system request address decoding does not match any valid chip-select region, or if two chip-select regions are defined as overlapped, or if a register file access is tried outside the valid address range of 1KB.

Copyright © 2026 Texas Instruments Incorporated

- ERRORNOTSUPPMCMD occurs when an unsupported command request is decoded at the interconnect interface.
- ERRORTIMEOUT: A time-out mechanism prevents the system from hanging. The start value of the 9-bit time-out counter is defined in the GPMC_TIMEOUT_CONTROL register and enabled with the GPMC_TIMEOUT_CONTROL[0] TIMEOUTENABLE bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory, and then a time-out error occurs. When data are sent from memory, this counter is reset to its start value. With multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access within the burst.

The GPMC does not generate interrupts on these errors. An interrupt generation is handled at interconnect level.

12.3.3.4.8 GPMC Timing Setting

The GPMC offers maximum flexibility to support various access protocols. Most of the timing parameters of the protocol access used by the GPMC to communicate with attached memories or devices are programmable on a chip-select basis. Assertion and deassertion times of control signals are defined to match the attached memory or device timing specifications and to get maximum performance during accesses. For more information about GPMC_CLKOUT and GPMC_FCLK, see Section 12.3.3.4.8.6, *GPMC_CLKOUT*.

Note

In the following sections, the start access time refers to the time at which the access begins.

12.3.3.4.8.1 Read Cycle Time and Write Cycle Time (RDCYCLETIME / WRCYCLETIME)

The GPMC_CONFIG5_i[4-0] RDCYCLETIME and GPMC_CONFIG5_i[12-8] WRCYCLETIME bit fields (where i = 0 to 3) define the address bus and byte-enable valid times for read and write accesses. To ensure a correct duty cycle of GPMC output clock between accesses, RDCYCLETIME and WRCYCLETIME are expressed in GPMC_FCLK cycles and must be multiples of the GPMC output clock cycle. The RDCYCLETIME and WRCYCLETIME bit fields can be set with a granularity of 1 or 2 through the GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY bit.

When RDCYCLETIME or WRCYCLETIME completes, if they are not already deasserted, all control signals (nCS, nADV/ALE, nOE/RE, nWE, and BE0/CLE) are deasserted to their reset values, regardless of their deassertion time parameters.

An exception to this forced deassertion occurs when a pipelined request to the same chip-select or to a different chip-select is pending. In such a case, it is not necessary to deassert a control signal with deassertion time parameters equal to the cycle-time parameter. This exception to forced deassertion prevents any unnecessary glitches. This requirement also applies to BE signals, thus avoiding an unnecessary BE glitch transition when pipelining requests.

Note

All control signals (CS, ADV#/ALE, BE0#/CLE, WE#, and GPMC output clock) are kept inactive (ADV#/ALE, BE0#/CLE, and GPMC output clock at low level; and CS, OE#/RE, and WE at high level) during the idle GPMC_FCLK cycles.

If no inactive cycles are required between successive accesses to the same chip-select or a different chip-select (GPMC_CONFIG6_i[7] CYCLE2CYCLESAMECSEN = 0 or GPMC_CONFIG6_i[6] CYCLE2CYCLEDIFFCSEN = 0, where i = 0 to 3), and if assertion-time parameters associated with the pipelined access are equal to 0, asserted control signals (nCS, nADV/ALE, nBE0/CLE, nWE, and nOE/RE) are kept asserted. This applies to any read/write to read/write access combination.

If inactive cycles are inserted between successive accesses (that is, CYCLE2CYCLESAMECSEN = 1 or CYCLE2CYCLEDIFFCSEN = 1), the control signals are forced to their respective default reset values for the number of GPMC_FCLK cycles defined in CYCLE2CYCLEDELAY.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.8.2 nCS: Chip-Select Signal Control Assertion/Deassertion Time (CSONTIME / CSRDOFFTIME / CSWROFFTIME / CSEXTRADELAY)

The GPMC_CONFIG2_i[3-0] CSONTIME bit field (where i = 0 to 3) defines the nCS signal-assertion time relative to the start access time. It is common for read and write accesses.

The GPMC_CONFIG2_i[12-8] CSRDOFFTIME (read access) and GPMC_CONFIG2_i[20-16] CSWROFFTIME (write access) bit fields define the nCS signal deassertion time relative to start access time.

The CSONTIME, CSRDOFFTIME, and CSWROFFTIME parameters apply to synchronous and asynchronous modes. CSONTIME can be used to control an address and byte-enable setup time before chip-select assertion. CSRDOFFTIME and CSWROFFTIME can be used to control an address and byte-enable hold time after chip-select deassertion.

nCS signal transitions, as controlled through CSONTIME, CSRDOFFTIME, and CSWROFFTIME, can be delayed by a half-GPMC_FCLK period by enabling the GPMC_CONFIG2_i[7] CSEXTRADELAY bit. This half-GPMC_FCLK period provides more granularity on the nCS assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. CSEXTRADELAY is especially useful in configurations where GPMC output clock and GPMC_FCLK have the same frequency, but it can also be used for all GPMC configurations. If enabled, CSEXTRADELAY applies to all parameters that control nCS transitions.

The CSEXTRADELAY bit must be used carefully to avoid control signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than the nCS signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

12.3.3.4.8.3 nADV/ALE: Address Valid/Address Latch Enable Signal Control Assertion/ Deassertion Time (ADVONTIME / ADVRDOFFTIME / ADVWROFFTIME / ADVEXTRADELAY/ADVAADMUXONTIME/ ADVAADMUXRDOFFTIME/ADVAADMUXWROFFTIME)

The *GPMC_CONFIG3_i*[3-0] ADVONTIME field (where i = 0 to 3) defines the nADV/ALE signal-assertion time relative to start access time. It is common to read and write accesses.

The *GPMC_CONFIG3_i*[12-8] ADVRDOFFTIME (read access) and *GPMC_CONFIG3_i*[20-16] ADVWROFFTIME (write access) bit fields define the nADV/ALE signal-deassertion time relative to start access time.

ADVONTIME can be used to control an address and byte-enable valid setup time control before nADV/ALE assertion. ADVRDOFFTIME and ADVWROFFTIME can be used to control an address and byte-enable valid hold time control after nADV/ALE deassertion. ADVRDOFFTIME and ADVWROFFTIME apply to synchronous and asynchronous modes.

The nADV/ALE signal transitions as controlled through ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME can be delayed by a half-GPMC_FCLK period by enabling the *GPMC_CONFIG3_i*[7] ADVEXTRADELAY bit. This half-GPMC_FCLK period provides more granularity on nADV/ALE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. The ADVEXTRADELAY configuration parameter is especially useful in configurations where GPMC output clock and GPMC_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, ADVEXTRADELAY applies to all parameters controlling nADV/ALE transitions.

ADVEXTRADELAY must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than nADV/ALE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

GPMC_CONFIG3_i[6-4] ADVAADMUXONTIME, *GPMC_CONFIG3_i*[26-24] ADVAADMUXRDOFFTIME, and *GPMC_CONFIG3_i*[30-28] ADVAADMUXWROFFTIME parameters have the same functions as ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME, but apply to the first address phase in the AAD-multiplexed protocol. The user must ensure that ADVAADMUXxxOFFTIME is programmed to a value less than or equal to ADVxxOFFTIME. Functionality in AAD-multiplexed mode is undefined if the settings do not comply with this requirement. ADVAADMUXxxOFFTIME can be programmed to the same value as ADVONTIME if no high nADV

Copyright © 2026 Texas Instruments Incorporated

pulse is needed between the two AAD-multiplexed address phases, which is the typical case in synchronous mode. In this configuration, nADV is kept low until it reaches the correct ADVxxOFFTIME.

For more information about the use of ADVONTIME, ADVRDOFFTIME, ADVWROFFTIME, and ADVAADMUXRDOFFTIME and ADVAADMUXWROFFTIME for command latch enable (CLE) and address latch enable (ALE) use for a NAND flash interface, see Section 12.3.3.4.11, *GPMC NAND Access Description*.

12.3.3.4.8.4 nOE/nRE: Output Enable/Read Enable Signal Control Assertion/Deassertion Time (OEONTIME / OEOFFTIME / OEEXTRADELAY / OEAADMUXONTIME / OEAADMUXOFFTIME)

The GPMC_CONFIG4_i[3-0] OEONTIME bit field (where i = 0 to 3) defines the nOE/nRE signal assertion time relative to start access time. It applies only to read accesses.

The GPMC_CONFIG4_i[12-8] OEOFFTIME bit field defines the nOE/nRE signal deassertion time relative to start access time. It applies only to read accesses. nOE/nRE is not asserted during a write cycle.

The OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME parameters apply to synchronous and asynchronous modes. OEONTIME can be used to control an address and byte enable valid setup time control before nOE/nRE assertion. OEOFFTIME can be used to control an address and byte-enable valid hold time control after nOE/nRE assertion.

The OEAADMUXONTIME and OEAADMUXOFFTIME parameters have the same functions as OEONTIME and OEOFFTIME, but apply to the first OE assertion in the AAD-multiplexed protocol for a read phase, or to the only OE assertion for a write phase. The user mut ensure that OEAADMUXOFFTIME is programmed to a value less than OEONTIME. Functionality in AAD-multiplexed mode is undefined if the settings do not comply with this requirement. OEAADMUXOFFTIME must never be equal to OEONTIME because the AAD-multiplexed protocol requires a second address phase with the nOE signal deasserted before nOE can be asserted again to define a read command.

The nOE/RE signal transitions as controlled through OEONTIME, OEOFFTIME, OEAADMUXONTIME, and OEAADMUXOFFTIME can be delayed by a half-GPMC_FCLK period by enabling the GPMC_CONFIG4_i[7] OEEXTRADELAY bit. This half-GPMC_FCLK period provides more granularity on the nOE/RE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. If enabled, OEEXTRADELAY applies to all parameters controlling nOE/nRE transitions.

OEEXTRADELAY must be used carefully, to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program RDCYCLETIME and WRCYCLETIME to be greater than the nOE/RE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

Note

When the GPMC generates a read access to an address-/data-multiplexed device, it drives the address bus until nOE assertion time.

12.3.3.4.8.5 nWE: Write Enable Signal Control Assertion/Deassertion Time (WEONTIME / WEOFFTIME / WEEXTRADELAY)

The GPMC_CONFIG4_i[19-16] WEONTIME bit field (where i = 0 to 3) defines the nWE signal-assertion time relative to start access time. The GPMC_CONFIG4_i[28-24] WEOFFTIME bit field defines the nWE signal-deassertion time relative to start access time. These bit fields apply only to write accesses. nWE is not asserted during a read cycle.

WEONTIME can be used to control an address and byte-enable valid setup time control before nWE assertion. WEOFFTIME can be used to control an address and byte-enable valid hold time control after nWE assertion.

nWE signal transitions as controlled through WEONTIME, and WEOFFTIME can be delayed by a half-GPMC_FCLK period by enabling the GPMC_CONFIG4_i[23] WEEXTRADELAY bit. This half-GPMC_FCLK period provides more granularity on nWE assertion and deassertion time to ensure proper setup and hold time relative to GPMC output clock. If enabled, WEEXTRADELAY applies to all parameters controlling nWE transitions.

Copyright © 2026 Texas Instruments Incorporated

The WEEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the WRCYCLETIME bit field to be greater than the nWE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

12.3.3.4.8.6 GPMC_CLKOUT

The GPMC output clock generated for external synchronous memory or device is GPMC_CLKOUT.

- The GPMC_CLKOUT clock frequency is the GPMC_FCLK functional clock frequency divided by 1, 2, 3, or 4, depending on the GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER bit field (where i = 0 to 3), with a guaranteed 50-percent duty cycle. For information about the duty cycle error, see the device-specific Datasheet.
- The GPMC_CLKOUT clock is activated only when the access in progress is defined as synchronous (read or write access).
- The GPMC_CONFIG1_i[26-25] CLKACTIVATIONTIME bit field (where i = 0 to 3) defines the number of GPMC_FCLK cycles from start access time to GPMC_CLKOUT activation.
- The GPMC_CLKOUT clock is stopped when cycle time completes and is asserted low between accesses.
- The GPMC_CLKOUT clock is kept low when access is defined as asynchronous.

CAUTION

When the cycle time completes, the GPMC_CLKOUT may be high because of the GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER bit field. To ensure correct stoppage of the GPMC_CLKOUT clock within the required 50-percent duty cycle, the user must extend the RDCYCLETIME or WRCYCLETIME value.

Note

To ensure a correct external clock cycle, the following rules must be applied:

- (RDCYCLETIME CLKACTIVATIONTIME) must be a multiple of (GPMCFCLKDIVIDER + 1).
- The PAGEBURSTACCESSTIME value must be a multiple of (GPMCFCLKDIVIDER + 1).

12.3.3.4.8.7 GPMC Output Clock and Control Signals Setup and Hold

Control-signal transition (assertion and deassertion) setup and hold values with respect to the GPMC output clock edge can be controlled in the following ways:

- For the GPMC output clock signal, the GPMC_CONFIG1_i[26-25] CLKACTIVATIONTIME bit field (where i = 0 to 3) allows setup and hold control of control-signal assertion time.
- The use of a divided GPMC output clock allows setup and hold control of the control-signal assertion and deassertion times.
- When the GPMC output clock runs at the GPMC_FCLK frequency so that GPMC output clock edge and control-signal transitions refer to the same GPMC_FCLK edge, the control-signal transitions can be delayed by a half-GPMC_FCLK period to provide minimum setup and hold times. This half-GPMC_FCLK delay is enabled with the CSEXTRADELAY, ADVEXTRADELAY, OEEXTRADELAY, or WEEXTRADELAY parameter. This delay must be used carefully to prevent control-signal overlap between successive accesses to different chip-selects. This implies that the RDCYCLETIME and WRCYCLETIME are greater than the last control-signal deassertion time, including the extra half-GPMC_FCLK cycle.

12.3.3.4.8.8 Access Time (RDACCESSTIME / WRACCESSTIME)

The read/write access time durations can be programmed independently through the GPMC_CONFIG5_i[20-16] RDACCESSTIME and GPMC_CONFIG6_i[28-24] WRACCESSTIME bit fields (where i = 0 to 3). This allows nOE and GPMC data-capture timing parameters to be independent of nWE and memory device data capture timing parameters. The RDACCESSTIME and WRACCESSTIME bit fields can be set with a granularity of 1 or 2 through the GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY bit.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.8.8.1 Access Time on Read Access

In asynchronous read mode, for single and paged accesses, the GPMC_CONFIG5_i[20-16] RDACCESSTIME bit field (where i = 0 to 3) defines the number of GPMC_FCLK cycles from start access time to the GPMC_FCLK rising edge used for the first data capture. RDACCESSTIME must be programmed to the rounded greater value (in GPMC_FCLK cycles) of the read access time of the attached memory device.

In synchronous read mode, for single or burst accesses, RDACCESSTIME defines the number of GPMC_FCLK cycles from the start access time to the GPMC_FCLK rising edge corresponding to the GPMC output clock rising edge used for the first data capture.

GPMC output clock, which is sent to the memory device for synchronization with the GPMC controller, is internally retimed to correctly latch the returned data. The GPMC_CONFIG5_i[4-0] RDCYCLETIME bit field must be greater than RDACCESSTIME to let the GPMC latch the last return data using the internally retimed GPMC output clock.

The external WAIT signal can be used in conjunction with RDACCESSTIME to control the effective GPMC data-capture GPMC_FCLK edge on read access in asynchronous and synchronous modes. For more information about wait monitoring, see Section 12.3.3.4.7.3.1, *WAIT Pin Monitoring Control*.

12.3.3.4.8.8.2 Access Time on Write Access

In asynchronous write mode, the GPMC_CONFIG6_i[28-24] WRACCESSTIME timing parameter is not used to define the effective write access time. Instead, it is used as a wait invalid timing window and must be set to a correct value so that the GPMC_WAIT pin is at a valid state two GPMC output clock cycles before WRACCESSTIME completes. For more information about wait monitoring, see Section 12.3.3.4.7.3.1, *WAIT Pin Monitoring Control*.

In synchronous write mode, for single or burst accesses, WRACCESSTIME defines the number of GPMC_FCLK cycles from the start access time to the GPMC output clock rising edge used by the memory device for the first data capture.

The external WAIT signal can be used in conjunction with WRACCESSTIME to control the effective memory device data-capture GPMC output clock edge for a synchronous write access. For more information about wait monitoring, see Section 12.3.3.4.7.3.1, *WAIT Pin Monitoring Control*.

12.3.3.4.8.9 Page Burst Access Time (PAGEBURSTACCESSTIME)

The *GPMC_CONFIG5_i*[27-24] PAGEBURSTACCESSTIME bit field (where i = 0 to 3) can be set with a granularity of 1 or 2 through the *GPMC_CONFIG1_i*[4] TIMEPARAGRANULARITY bit.

12.3.3.4.8.9.1 Page Burst Access Time on Read Access

In asynchronous page read mode, the delay between successive word captures in a page is controlled through the PAGEBURSTACCESSTIME bit field. The PAGEBURSTACCESSTIME parameter must be programmed to the rounded greater value (in GPMC_FCLK cycles) of the read access time of the attached device.

In synchronous burst read mode, the delay between successive word captures in a burst is controlled through the PAGEBURSTACCESSTIME bit field.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective GPMC data-capture GPMC_FCLK edge on read access. For more information about wait monitoring, see Section 12.3.3.4.7.3.1, *WAIT Pin Monitoring Control*.

12.3.3.4.8.9.2 Page Burst Access Time on Write Access

Asynchronous page write mode is not supported. PAGEBURSTACCESSTIME is irrelevant in this case.

In synchronous burst write mode, PAGEBURSTACCESSTIME controls the delay between successive memory device word captures in a burst.

Copyright © 2026 Texas Instruments Incorporated

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective memory device data capture GPMC output clock edge in synchronous write mode. For more information about wait monitoring, see Section 12.3.3.4.7.3.1, *WAIT Pin Monitoring Control*.

12.3.3.4.8.10 Bus Keeping Support

At the end cycle time of a read access, if no other access is pending, the GPMC drives the bus with the last data read after RDCYCLETIME completes to prevent bus floating and reduce power consumption.

After a write access, if no other access is pending, the GPMC keeps driving the data bus after WRCYCLETIME completes with the same data to prevent bus floating and power consumption.

12.3.3.4.9 GPMC NOR Access Description

For each chip-select configuration, the read access can be specified as asynchronous or synchronous access through the GPMC_CONFIG1_i[29] READTYPE bit (where i = 0 to 3). For each chip-select configuration, the write access can be specified as synchronous or asynchronous access through the GPMC_CONFIG1_i[27] WRITETYPE bit where (i = 0 to 3).

Asynchronous and synchronous read and write access time and related control signals are controlled through timing parameters that refer to GPMC_FCLK. The primary difference of synchronous mode is the availability of a configurable clock interface to control the external device. Synchronous mode also affects data-capture and wait-pin monitoring schemes in read access.

For more information about asynchronous and synchronous access, see the descriptions of GPMC output clock (CLK), RdAccessTime, WrAccessTime, and WAIT pin monitoring.

For more information about timing-parameter settings, see the sample timing diagrams in this chapter.

Note

The address bus and nBE[1-0] are fixed for the duration of a synchronous burst read access, but they are updated for each beat of an asynchronous page-read access.

12.3.3.4.9.1 Asynchronous Access Description

This section describes:

- Asynchronous single-read operation on an address/data multiplexed device
- Asynchronous single write operation on an address/data-multiplexed device
- Asynchronous single read operation on an AAD-multiplexed device
- Asynchronous single write operation on an AAD-multiplexed device
- Asynchronous multiple (page) read operation on a non-multiplexed device

In asynchronous operations GPMC output clock is not provided outside the GPMC and is kept low.

12.3.3.4.9.1.1 Access on Address/Data Multiplexed Devices

12.3.3.4.9.1.1.1 Asynchronous Single-Read Operation on an Address/Data Multiplexed Device

Figure 12-157 shows an asynchronous single read operation on an address/data-multiplexed device.

Copyright © 2026 Texas Instruments Incorporated

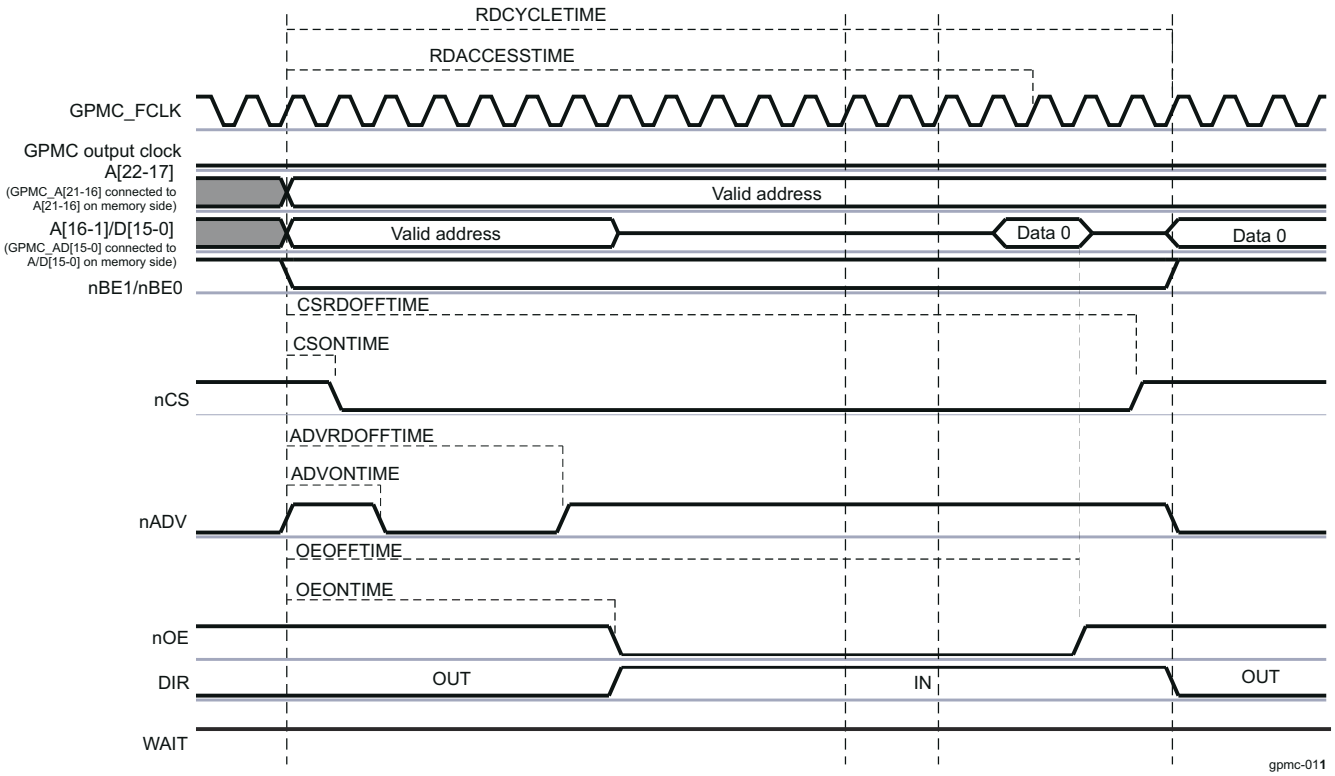


Figure 12-157. Asynchronous Single Read on an Address/Data-Multiplexed Device

For formulas to calculate timing parameters, see Section 12.3.3.5.6.1, *GPMC Timing Parameters Formulas*.

Table 12-216 lists the timing bit fields to set up to configure the GPMC in asynchronous single-read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see Section 12.3.3.4.7.2.3, *Address/Data-Multiplexing Interface*.

Address bits (A[16-1] from a GPMC perspective, A[15-0] from an external device perspective) are placed on the address/data bus, and the remaining address bits GPMC_A[22-16] are placed on the address bus. The address phase ends at nOE assertion, when the DIR signal goes from OUT to IN.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the *GPMC_CONFIG2_i*[3-0] CSONTIME bit field. It controls the address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC_CONFIG2_i[12-8] CSRDOFFTIME bit field. It controls the address hold time from nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the *GPMC_CONFIG3_i*[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the *GPMC_CONFIG3_i*[12-8] ADVRDOFFTIME bit field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the *GPMC_CONFIG4_i*[3-0] OEONTIME bit field.
  - nOE deassertion time is controlled by the *GPMC_CONFIG4_i*[12-8] OEOFFTIME bit field.
- Read data is latched when RDACCESSTIME completes. Access time is defined in the *GPMC_CONFIG5_i*[20-16] RDACCESSTIME bit field.
- Direction signal DIR: DIR goes from OUT to IN at the same time that nOE is asserted.
- The end of the access is defined by the
- GPMC_CONFIG5_i[4-0] RDCYCLETIME parameter.

Copyright © 2026 Texas Instruments Incorporated

In the GPMC, when a 16-bit wide device is attached to the controller, a 32-bit word write access is split into two 16-bit word write accesses. For more information about GPMC access size and type adaptation, see Section 12.3.3.4.9.5, *System Burst Versus External Device Burst Support*.

Between two successive accesses, if an nCS pulse is needed:

- The *GPMC_CONFIG6_i*[11-8] CYCLE2CYCLEDELAY bit field can be programmed with the *GPMC_CONFIG6_i*[7] CYCLE2CYCLESAMECSEN bit enabled.
- The CSWROFFTIME and CSONTIME parameters also allow a chip-select pulse, but this affects all other types of access.

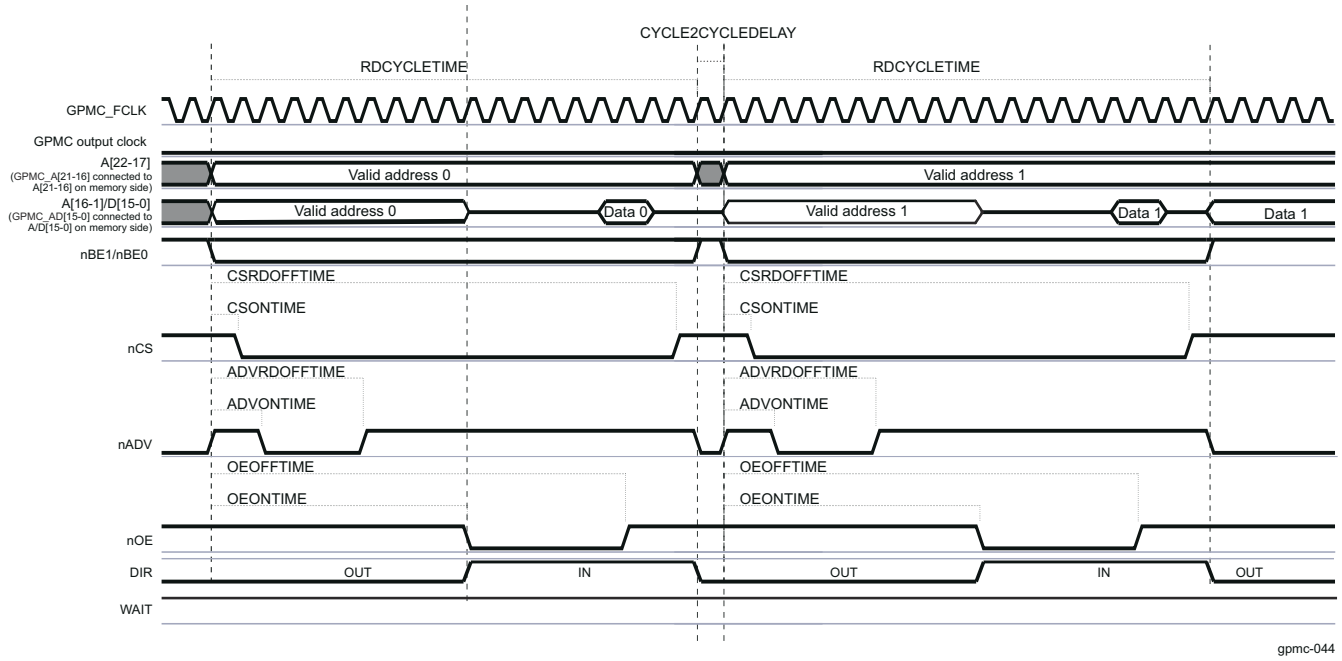


Figure 12-158. Two Asynchronous Single-Read Accesses on an Address/Data-Multiplexed Device (32-Bit Read Split Into 2 x 16-Bit Read)

12.3.3.4.9.1.1.2 Asynchronous Single-Write Operation on an Address/Data-Multiplexed Device

Figure 12-159 shows an asynchronous single-write operation on an address/data-multiplexed device.

Copyright © 2026 Texas Instruments Incorporated

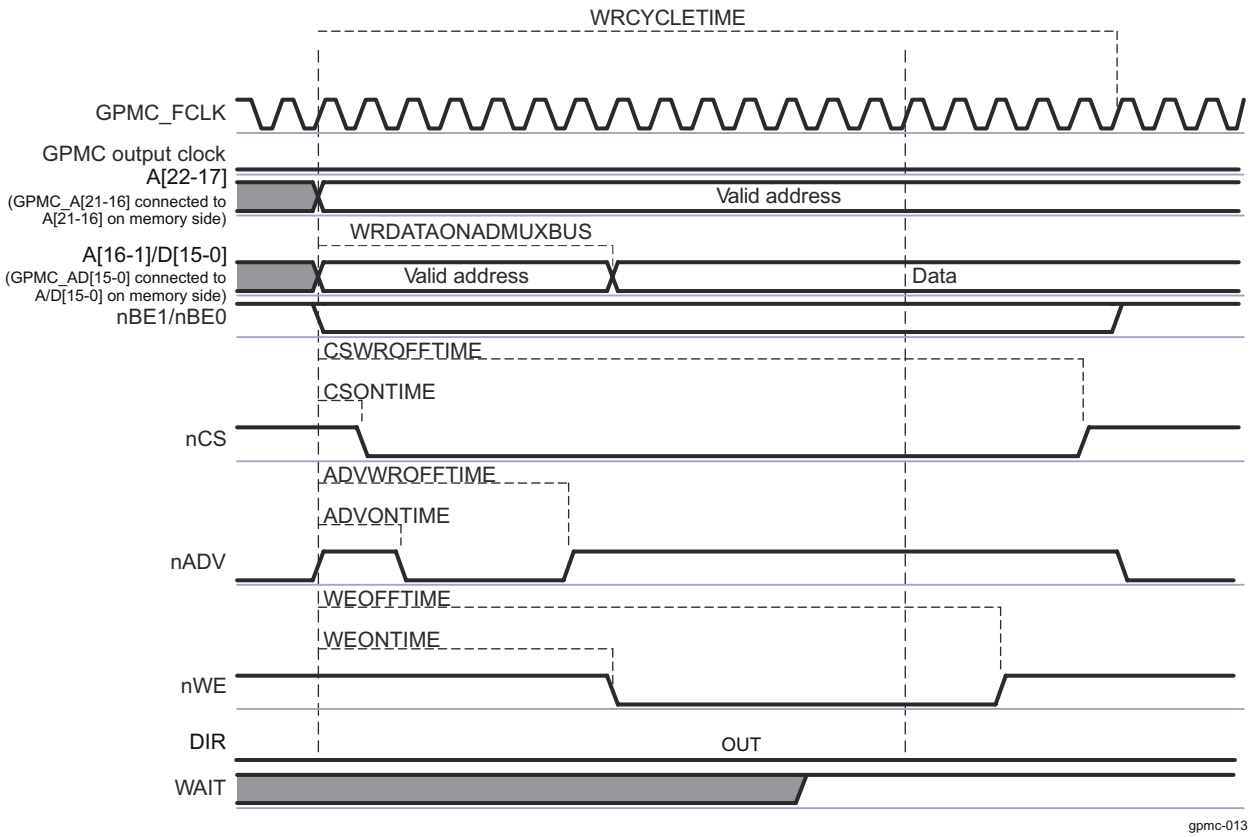


Figure 12-159. Asynchronous Single-Write on an Address/Data-Multiplexed Device

For formulas to calculate timing parameters, see Section 12.3.3.5.6.1, *GPMC Timing Parameters Formulas*.

Table 12-216 lists the timing bit fields to set up to configure the GPMC in asynchronous single-write mode.

When the GPMC generates a write access to an address/data-multiplexed device, it drives the address bus until nWE assertion time. For more information, see Section 12.3.3.4.7.2.3, *Address/Data-Multiplexing Interface*.

The nCS and nADV signals are controlled in the same way as for a asynchronous single-read operation on an address/data-multiplexed device.

- Write enable signal nWE:
  - nWE assertion indicates a write cycle.
  - nWE assertion time is controlled by the GPMC_CONFIG4_i[19-16] WEONTIME bit field.
  - nWE deassertion time is controlled by the GPMC_CONFIG4_i[28-24] WEOFFTIME bit field.
- Direction signal DIR: DIR signal is OUT during the entire access.
- The end of the access is defined by the GPMC_CONFIG5_i[12-8] WRCYCLETIME parameter.

Address bits A[16-1] (GPMC point of view) are placed on the address/data bus at the start of cycle time, and the remaining address bits A[22-17] are placed on the address bus.

Data is driven on the address/data bus at a GPMC_CONFIG6_i[19-16] WRDATAONADMUXBUS time.

Note

Multiple write access in asynchronous mode is not supported. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

After a write operation, if no other access (read or write) is pending, the data bus keeps its previous value. See Section 12.3.3.4.8.10, *Bus Keeping Support*.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.9.1.1.3 Asynchronous Multiple (Page) Write Operation on an Address/Data-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for address/data-multiplexed devices.

If the *GPMC_CONFIG1_i*[28] WRITEMULTIPLE bit is enabled (0x1) with the *GPMC_CONFIG1_i*[27] WRITETYPE bit as asynchronous (0x0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see Section 12.3.3.4.9.3, *Asynchronous and Synchronous Accesses in non-multiplexed Mode*.

12.3.3.4.9.1.2 Access on Address/Address/Data-Multiplexed Devices

12.3.3.4.9.1.2.1 Asynchronous Single Read Operation on an AAD-Multiplexed Device

Figure 12-160 shows an asynchronous single-read operation on an AAD-multiplexed device.

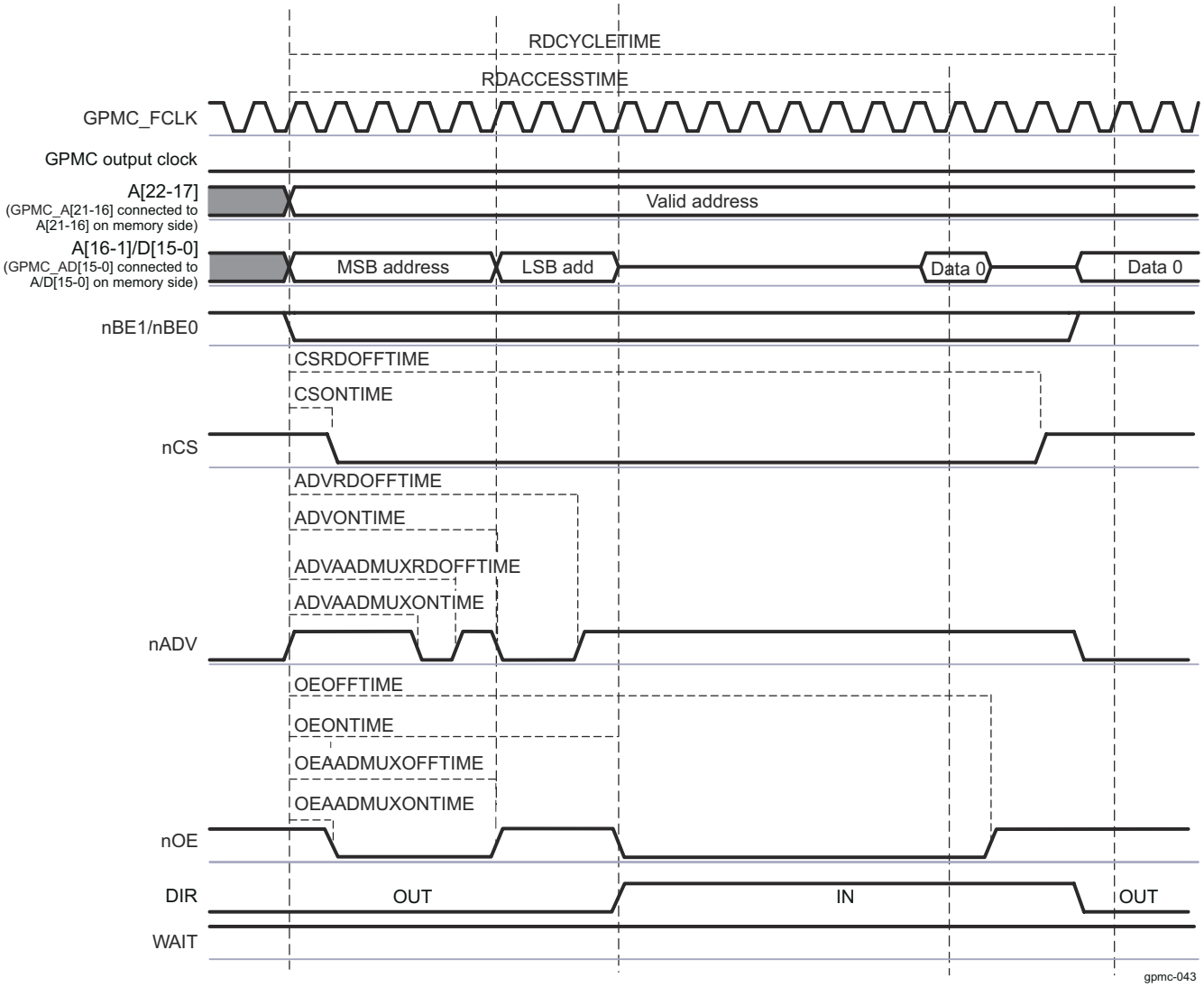


Figure 12-160. Asynchronous Single Read on an AAD-Multiplexed Device

For formulas to calculate timing parameters, see Section 12.3.3.5.6.1, *GPMC Timing Parameters Formulas*.

Table 12-216 lists the timing bit fields to set up to configure the GPMC in asynchronous single write mode.

When the GPMC generates a read access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE

Copyright © 2026 Texas Instruments Incorporated

driven low. The first address phase ends at the first nOE deassertion time. The second phase for LSB address is qualified with nOE driven high. The second address phase ends at the second nOE assertion time, when the DIR signal goes from OUT to IN.

The nCS and DIR signals are controlled in the same way as for an asynchronous single-read operation on an address/data-multiplexed device.

- Address valid signal nADV. nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the *GPMC_CONFIG3_i*[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the *GPMC_CONFIG3_i*[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the *GPMC_CONFIG3_i*[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the *GPMC_CONFIG3_i*[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE. nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the *GPMC_CONFIG4_i*[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the *GPMC_CONFIG3_i*[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the *GPMC_CONFIG4_i*[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the *GPMC_CONFIG4_i*[12-8] OEOFFTIME bit field.

12.3.3.4.9.1.2.2 Asynchronous Single-Write Operation on an AAD-Multiplexed Device

Figure 12-161 shows an asynchronous single-write operation on an AAD-multiplexed device.

Copyright © 2026 Texas Instruments Incorporated

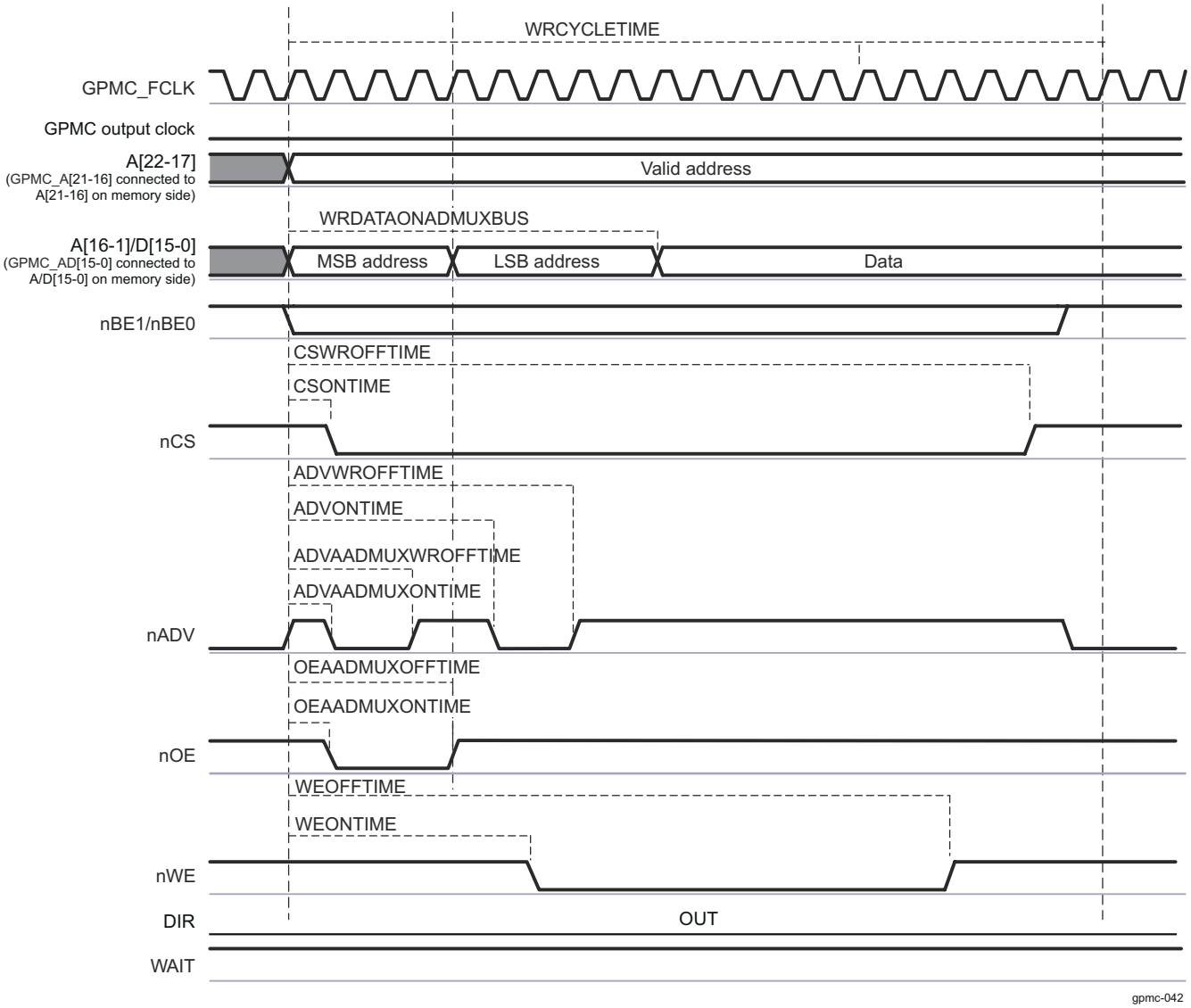


Figure 12-161. Asynchronous Single Write on an AAD-Multiplexed Device

For formulas to calculate timing parameters, see Section 12.3.3.5.6.1, *GPMC Timing Parameters Formulas*.

Table 12-216 lists the timing bit fields to set up to configure the GPMC in asynchronous single-write mode.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

The nCS, nWE, and DIR signals are controlled in the same way as for an asynchronous single-write operation on an address/data-multiplexed device. See Table 12-207, *NAND Memory Type*.

- Address valid signal nADV is asserted and deasserted twice during a write transaction:
  - nADV first assertion time is controlled by the *GPMC_CONFIG3_i*[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the *GPMC_CONFIG3_i*[30-28] ADVAADMUXWROFFTIME bit field.
  - nADV second assertion time is controlled by the *GPMC_CONFIG3_i*[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the *GPMC_CONFIG3_i*[20-16] ADVWROFFTIME bit field.
- Output enable signal nOE is asserted during the address phase of a write transaction:

Copyright © 2026 Texas Instruments Incorporated

- nOE assertion time is controlled by the *GPMC_CONFIG4_i*[6-4] OEAADMUXONTIME bit field.
- nOE deassertion time is controlled by the *GPMC_CONFIG3_i*[15-13] OEAADMUXOFFTIME bit field.

The address bits for the first address phase are driven onto the data bus until nOE deassertion. Data is driven onto the address/data bus at the clock edge defined by the *GPMC_CONFIG6_i*[19-16] WRDATAONADMUXBUS parameter.

12.3.3.4.9.1.2.3 Asynchronous Multiple (Page) Read Operation on an AAD-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for AAD-multiplexed devices.

If the GPMC_CONFIG1_i[28] WRITEMULTIPLE bit is enabled (0x1) with the GPMC_CONFIG1_i[27] WRITETYPE bit as asynchronous (0x0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see Section 12.3.3.4.9.3, *Asynchronous and Synchronous Accessed in non-multiplexed Mode*.

12.3.3.4.9.2 Synchronous Access Description

This section describes read and write synchronous accesses on address/data-multiplexed devices. All information in this section can be applied to any type of memory (non-multiplexed, address and data-multiplexed, or AAD-multiplexed) with the difference limited to the address phase. For accesses on non-multiplexed devices, see Section 12.3.3.4.9.3, *Asynchronous and Synchronous Accessed in non-multiplexed Mode*.

In synchronous operations:

- The GPMC_CLKOUT clock is provided outside the GPMC when accessing the memory device.
- The GPMC_CLKOUT clock is derived from the GPMC_FCLK clock using the *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER bit field. In the following section i stands for the chip-select number, i = 0 to 3.
- The *GPMC_CONFIG1_i*[26-25] CLKACTIVATIONTIME bit field specifies that the GPMC_CLKOUT is provided outside the GPMC for 0 to 2 GPMC_FCLK cycles after start access time until RDCYCLETIME or WRCYCLETIME completes.

12.3.3.4.9.2.1 Synchronous Single Read

Figure 12-162 and Figure 12-163 show a synchronous single-read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.

Copyright © 2026 Texas Instruments Incorporated

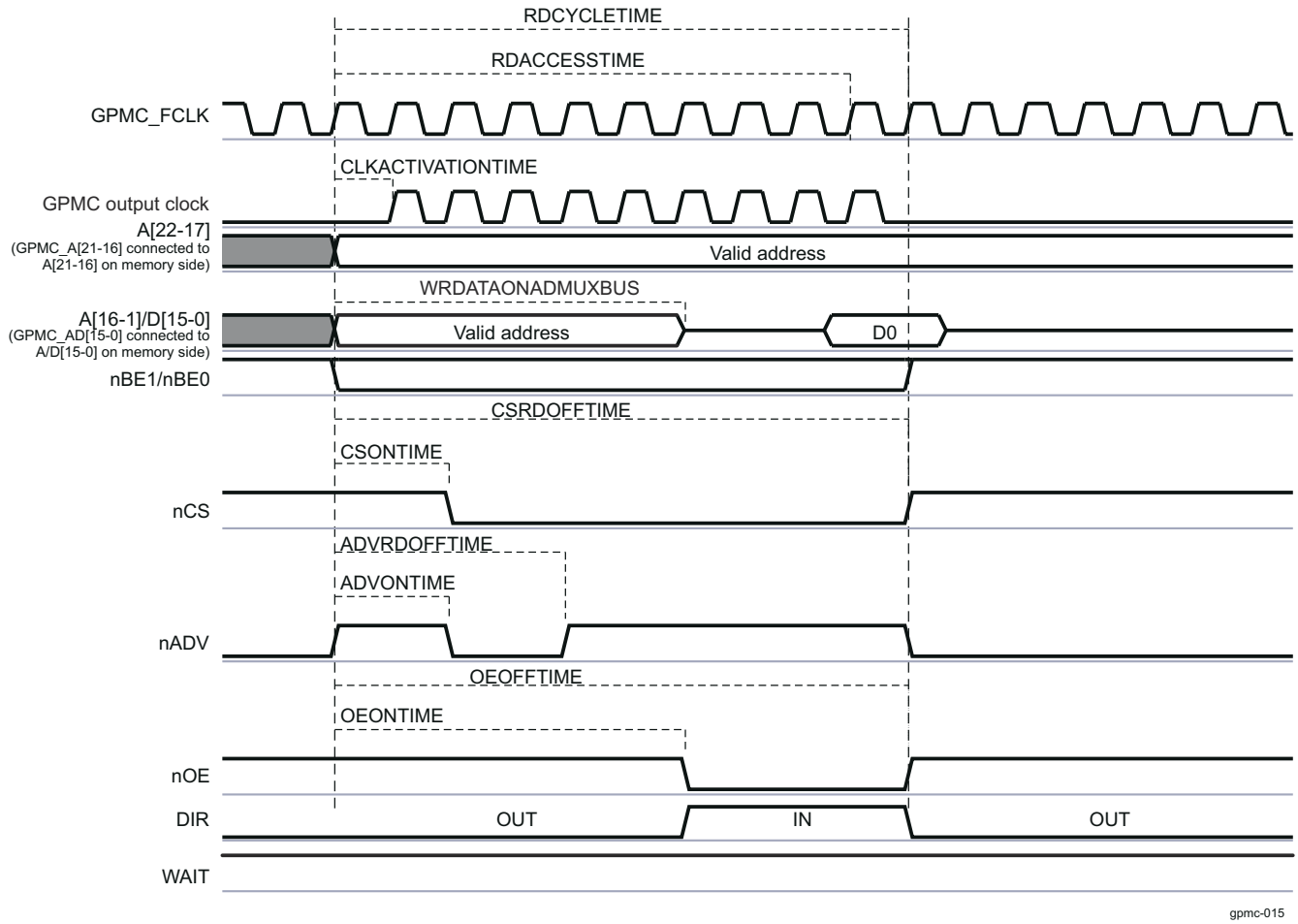


Figure 12-162. Synchronous Single Read (GPMCFCLKDIVIDER = 0)

Copyright © 2026 Texas Instruments Incorporated

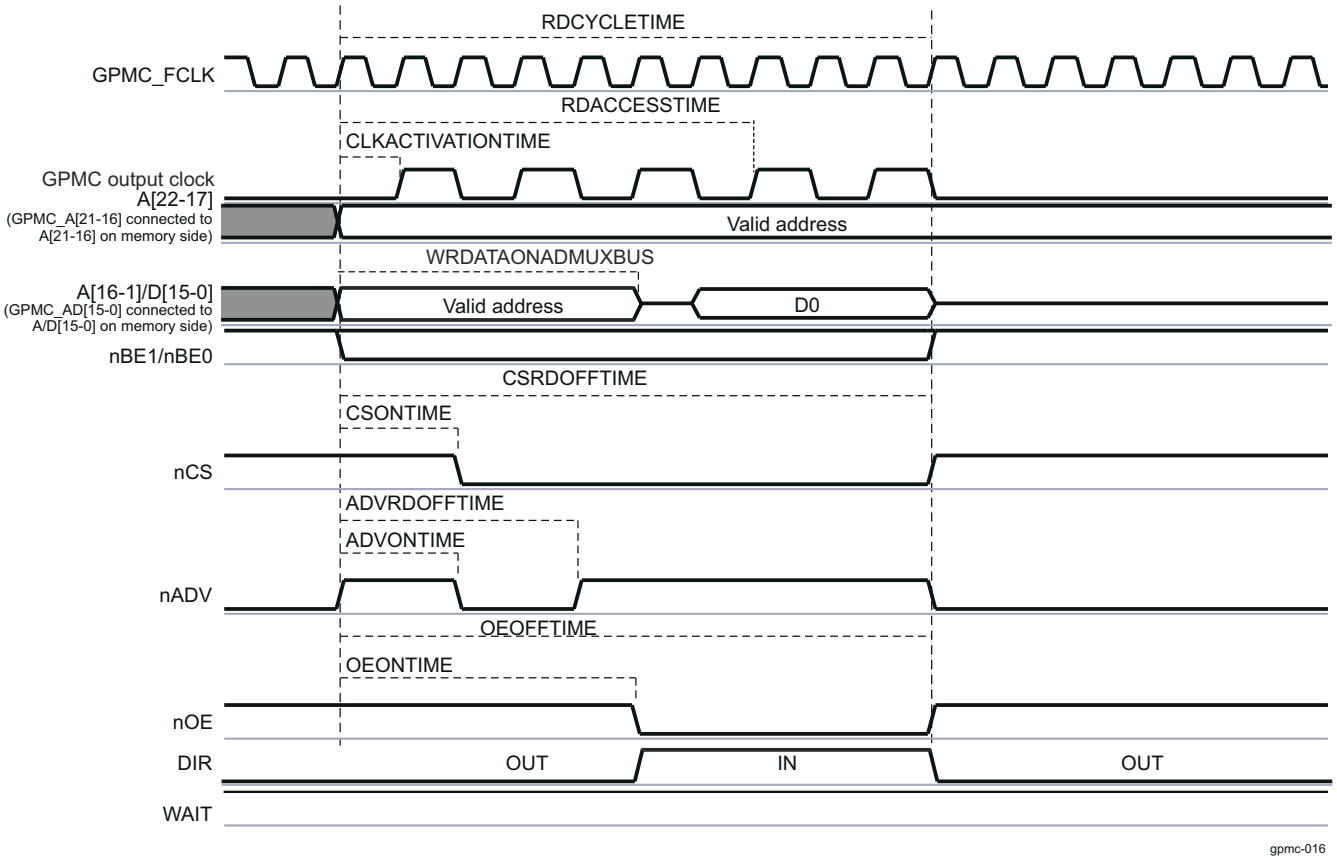


Figure 12-163. Synchronous Single Read (GPMCFCLKDIVIDER = 1)

For formulas to calculate timing parameters, see Section 12.3.3.5.6.1, *GPMC Timing Parameters Formulas*.

Table 12-216 lists the timing bit fields to set up to configure the GPMC in asynchronous single-read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see Section 12.3.3.4.7.2.3, *Address/Data-Multiplexing Interface*.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC_CONFIG2_i[3-0] CSONTIME bit field and ensures address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC_CONFIG2_i[12-8] CSRDOFFTIME bit field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC_CONFIG3_i[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the GPMC_CONFIG3_i[12-8] ADVRDOFFTIME bit field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC_CONFIG4_i[3-0] OEONTIME bit field.
  - nOE deassertion time is controlled by the GPMC_CONFIG4_i[12-8] OEOFFTIME bit field.
- Initial latency for the first read data is controlled by GPMC_CONFIG5_i[20-16] RDACCESSTIME bit field or by monitoring the WAIT signal.
- Total access time (the GPMC_CONFIG5_i[4-0] RDCYCLETIME bit field) corresponds to RDACCESSTIME plus the address hold time from nCS deassertion, plus time from RDACCESSTIME to CSRDOFFTIME.
- Direction signal DIR: DIR goes from OUT to IN at the same time as nOE assertion.

Copyright © 2026 Texas Instruments Incorporated

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

The nCS and DIR signals are controlled in the same way as for a synchronous single-read operation on an address/data-multiplexed device.

- Address valid signal nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the GPMC_CONFIG3_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC_CONFIG3_i[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC_CONFIG3_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC_CONFIG3_i[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the GPMC_CONFIG4_i[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the GPMC_CONFIG3_i[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the GPMC_CONFIG4_i[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the GPMC_CONFIG4_i[12-8] OEOFFTIME bit field.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 12.3.3.4.8.10, *Bus Keeping Support*.

12.3.3.4.9.2.2 Synchronous Multiple (Burst) Read (4-, 8-, 16-Word16 Burst With Wraparound Capability)

Figure 12-164 and Figure 12-165 show a synchronous multiple-read operation with GPMCFCLKDivider equal to 0 and 1, respectively.

Copyright © 2026 Texas Instruments Incorporated

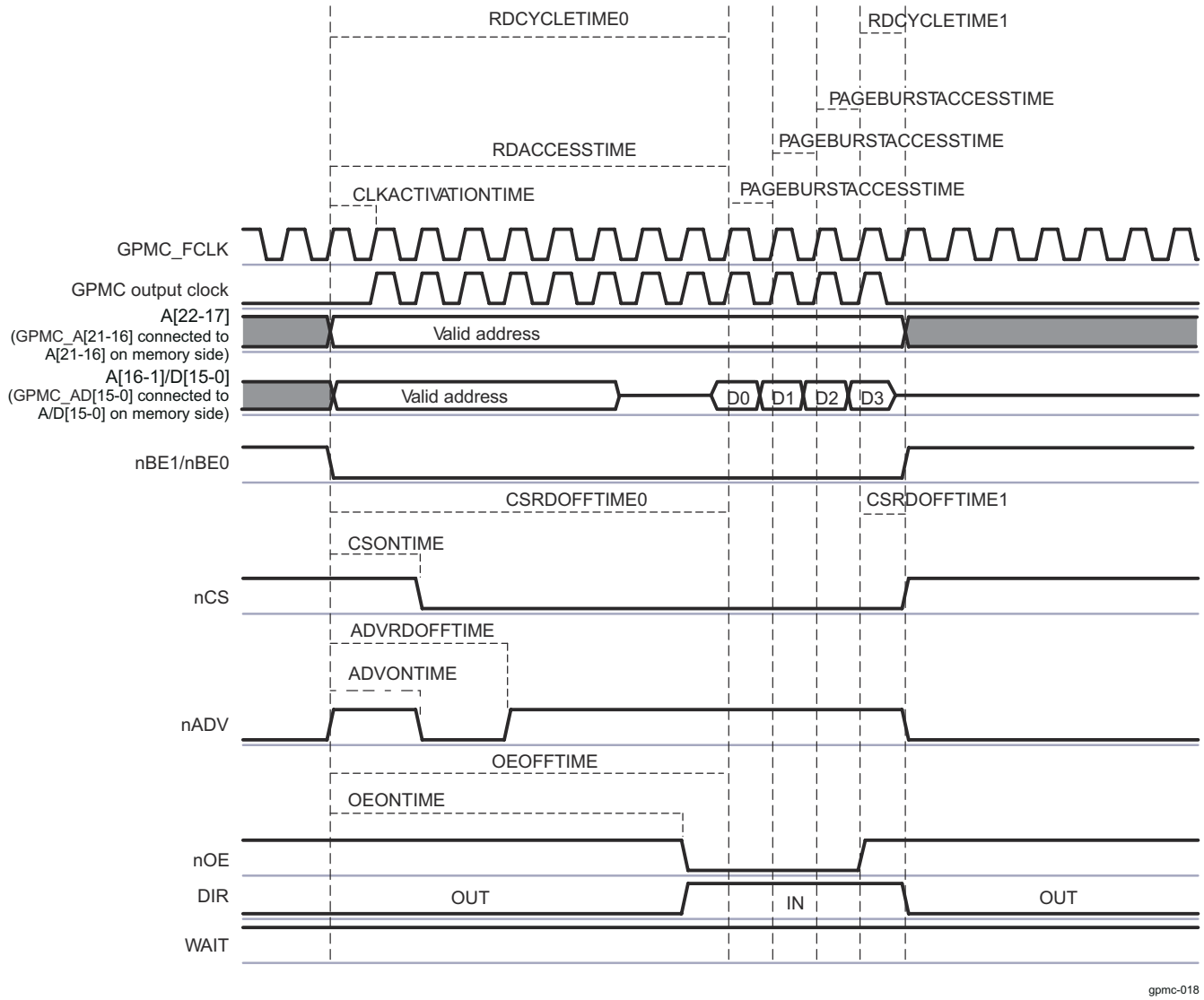


Figure 12-164. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0)

Copyright © 2026 Texas Instruments Incorporated

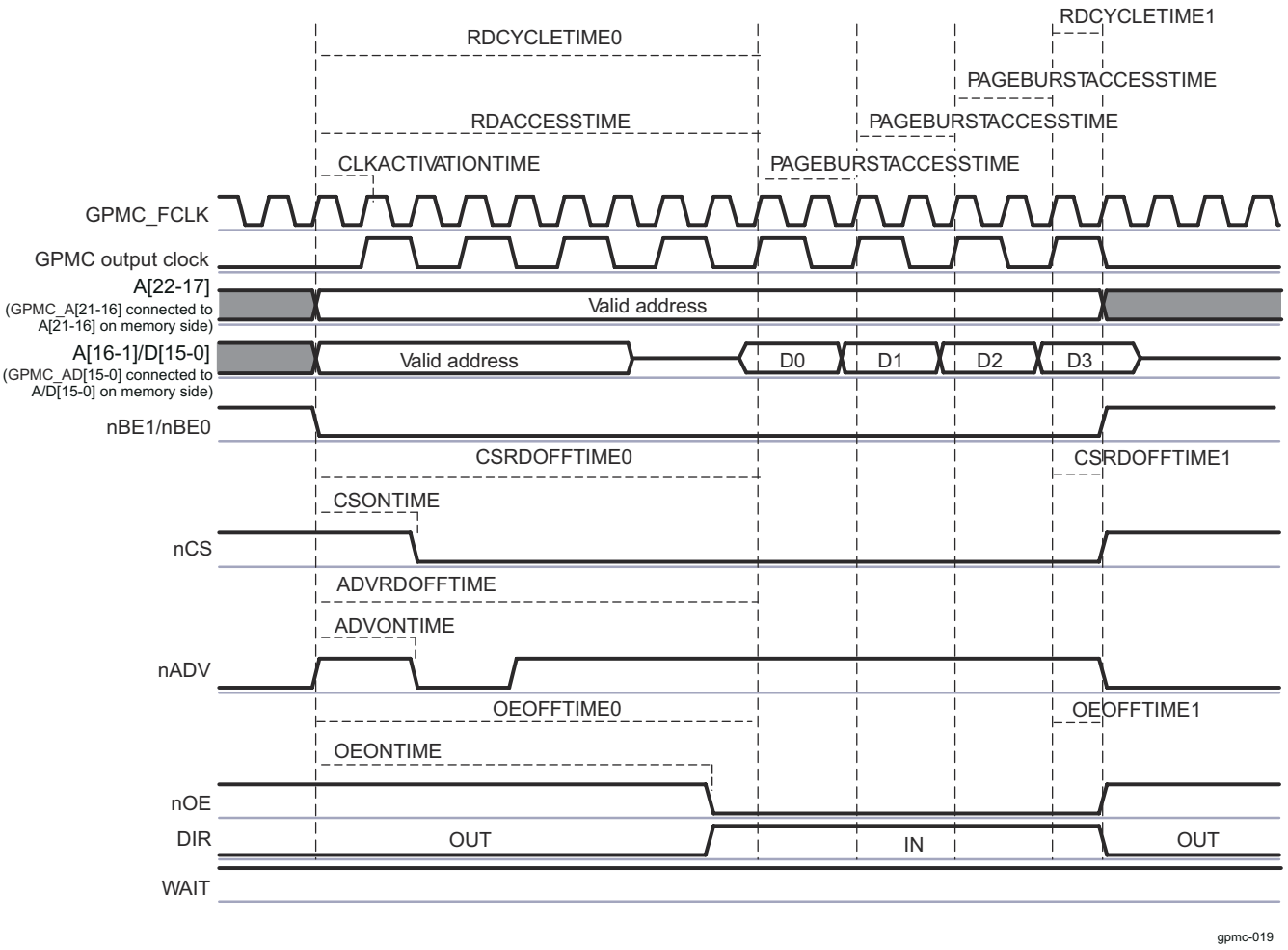


Figure 12-165. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1)

When the GPMC_CONFIG5_i[20-16] RDACCESSTIME bit field completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC_CONFIG5_i[27-24] PAGEBURSTACCESSTIME bit field multiplied by the number of remaining data transactions.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as for a synchronous single-read operation. See Table 12-202, *NOR Memory Type*.

Initial latency for the first read data is controlled by RDACCESSTIME or by monitoring the WAIT signal. Successive read data are provided by the memory device every one or two GPMC_CLKOUT cycles. The PAGEBURSTACCESSTIME parameter must be set accordingly with the GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER bit field and the memory-device internal configuration. Depending on the device page length, the GPMC checks the device page crossing during a new burst request and purposely inserts initial latency (of RDACCESSTIME) when required.

Total access time GPMC_CONFIG5_i[4-0] RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time from nCS deassertion. In Figure 12-165, the programmed value of RDCYCLETIME equals RDCYCLETIME0 + RDCYCLETIME1.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 12.3.3.4.8.10, *Bus Keeping Support*.

Burst wraparound is enabled through the GPMC_CONFIG1_i[31] WRAPBURST bit and allows a 4-, 8-, or 16-Word16 linear burst access to wrap within its burst-length boundary through the GPMC_CONFIG1_i[24-23] ATTACHEDDEVICEPAGELENGTH bit field.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.9.2.3 Synchronous Single Write

Burst write mode is used for synchronous single or burst accesses.

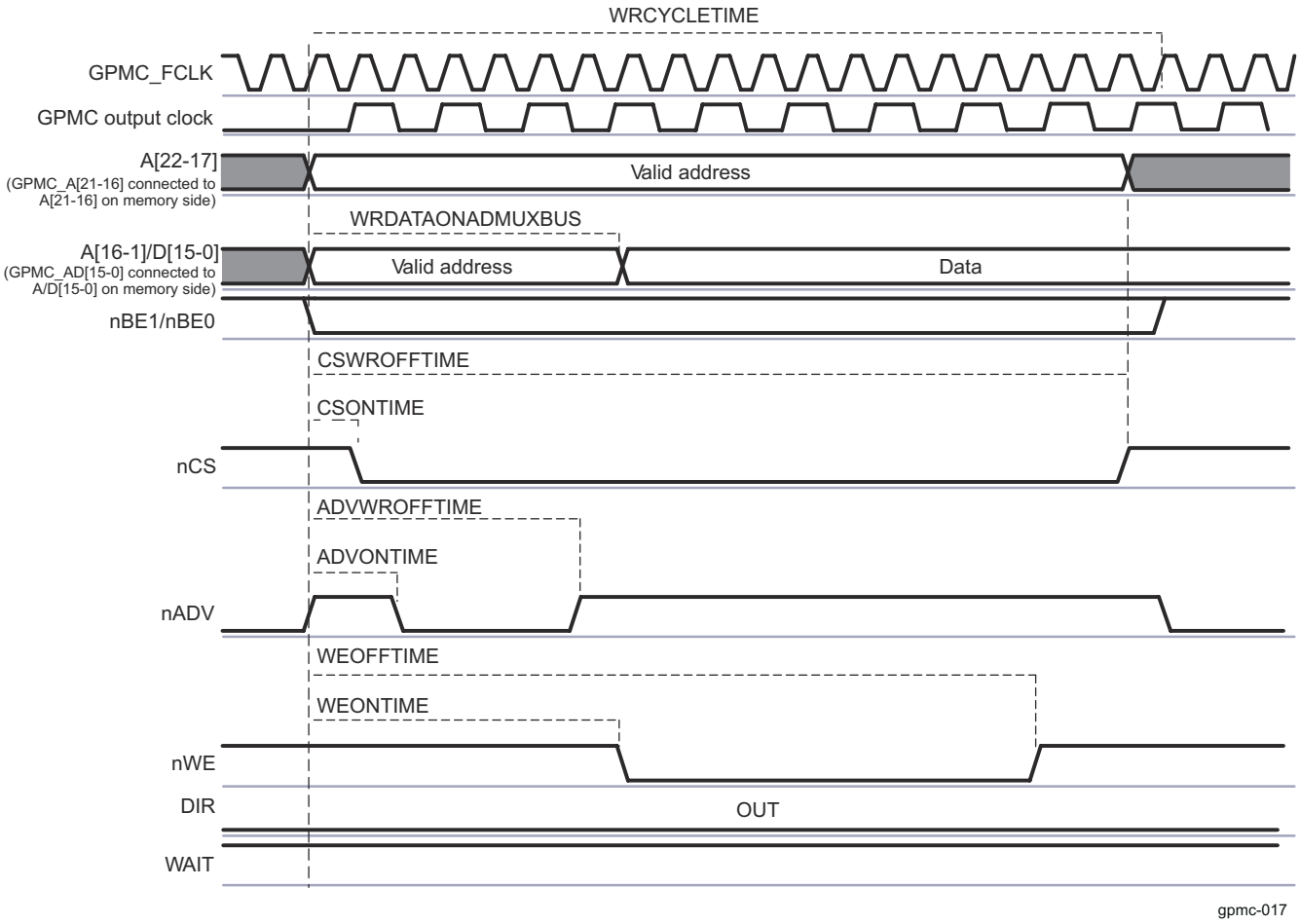


Figure 12-166. Synchronous Single Write on an Address/Data-Multiplexed Device

When the GPMC generates a write access to an address/data-multiplexed device, it drives the data bus (with address bits A[16-1]) until the GPMC_CONFIG6_i[19-16] WRDATAONADMUXBUS bit field time. The first data of the burst is driven on the address/data bus at WRDATAONADMUXBUS time.

12.3.3.4.9.2.4 Synchronous Multiple (Burst) Write

Synchronous burst write mode provides synchronous single or consecutive accesses.

Figure 12-167 shows a synchronous burst write access when the chip-select is configured in address/data-multiplexed mode.

Copyright © 2026 Texas Instruments Incorporated

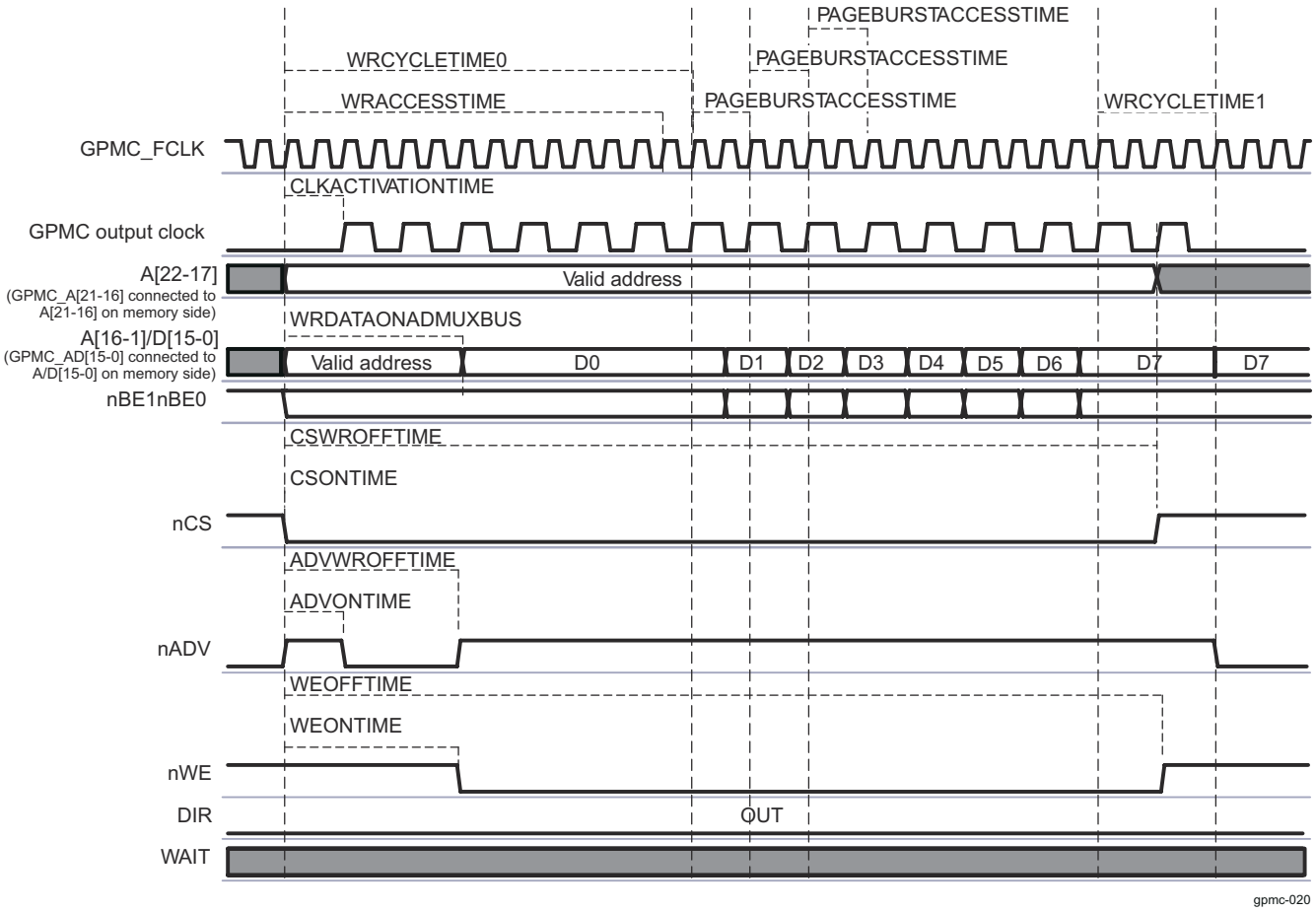


Figure 12-167. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode

Figure 12-168 shows the same synchronous burst write access when the chip-select is configured in address/address/data-multiplexed (AAD-multiplexed) mode.

Copyright © 2026 Texas Instruments Incorporated

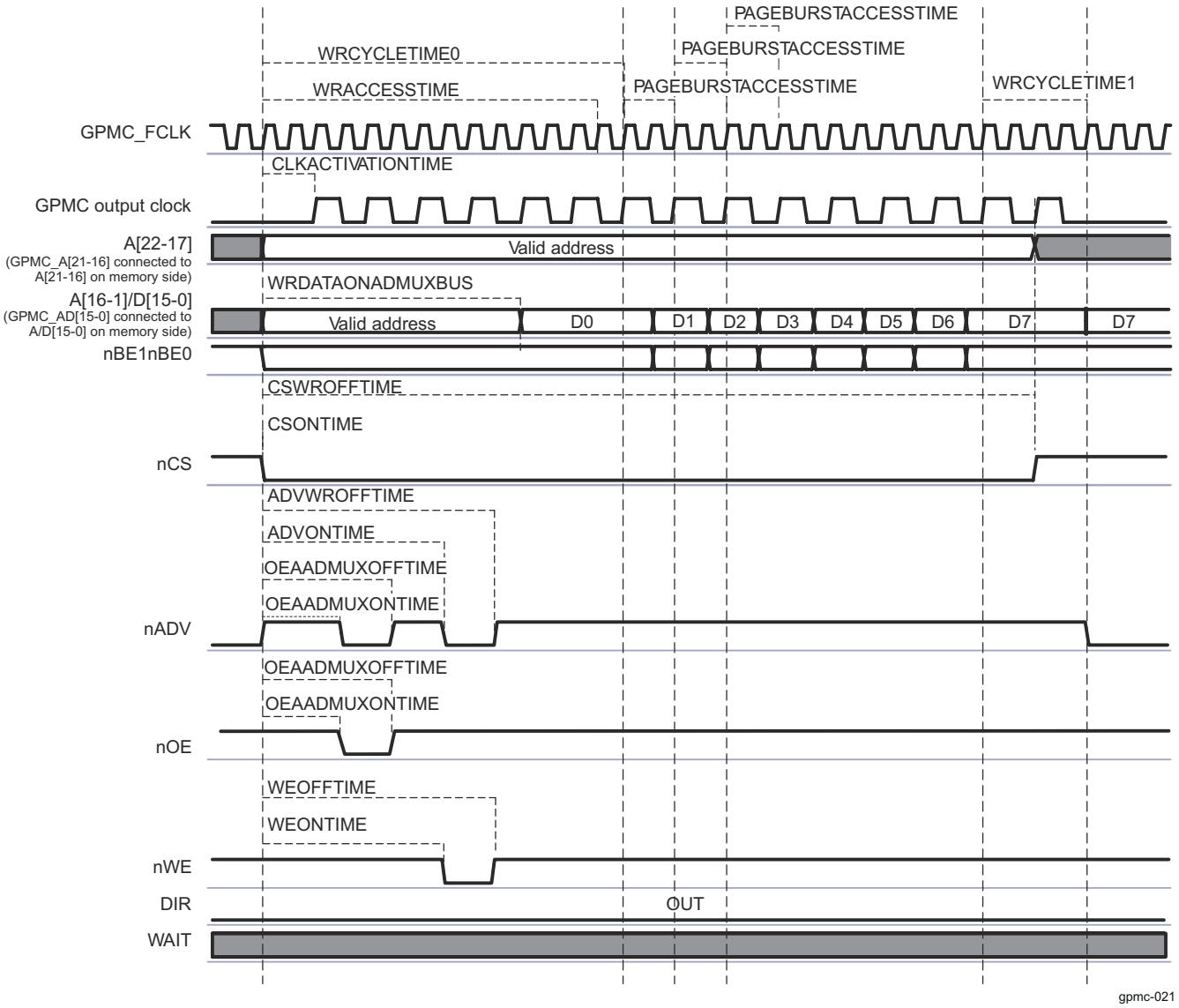


Figure 12-168. Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode

The first data of the burst is driven on the A/D bus at the GPMC_CONFIG6_i[19-16] WRDATAONADMUXBUS bit field.

When WRACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC_CONFIG5_i[27-24] PAGEBURSTACCESSTIME bit field multiplied by the number of remaining data transactions.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For more information, see Section 12.3.3.4.7.2.3, *Address/Data-Multiplexing Interface*.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC_CONFIG2_i[3-0] CSONTIME bit field (where i = 0 to 3) and ensures address setup time to nCS assertion.
  - nCS deassertion time controlled by the GPMC_CONFIG2_i[20-16] CSWROFFTIME bit field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC_CONFIG3_i[3-0] ADVONTIME bit field.
  - nADV deassertion time is controlled by the GPMC_CONFIG3_i[20-16] ADVWROFFTIME bit field.

Copyright © 2026 Texas Instruments Incorporated

- Write enable signal nWE:
  - nWE assertion indicates a read cycle.
  - nWE assertion time is controlled by the GPMC_CONFIG4_i[19-16] WEONTIME bit field.
  - nWE deassertion time is controlled by the GPMC_CONFIG4_i[28-24] WEOFFTIME bit field.

Note

The nWE falling edge must not be used to control the time when the burst first data is driven in the address/data bus, because some new devices require the nWE signal to be low during the address phase.

- Direction signal DIR is OUT during the entire access.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with nOE driven low. The second phase for LSB address is qualified with nOE driven high. The address phase ends at nWE assertion time.

The nCS, and DIR signals are controlled as previously described.

- Address valid signal nADV is asserted and deasserted twice during a read transaction:
  - nADV first assertion time is controlled by the GPMC_CONFIG3_i[6-4] ADVAADMUXONTIME bit field.
  - nADV first deassertion time is controlled by the GPMC_CONFIG3_i[26-24] ADVAADMUXRDOFFTIME bit field.
  - nADV second assertion time is controlled by the GPMC_CONFIG3_i[3-0] ADVONTIME bit field.
  - nADV second deassertion time is controlled by the GPMC_CONFIG3_i[12-8] ADVRDOFFTIME bit field.
- Output Enable signal nOE is asserted and deasserted twice during a read transaction (nOE second assertion indicates a read cycle):
  - nOE first assertion time is controlled by the GPMC_CONFIG4_i[6-4] OEAADMUXONTIME bit field.
  - nOE first deassertion time is controlled by the GPMC_CONFIG4_i[15-13] OEAADMUXOFFTIME bit field.
  - nOE second assertion time is controlled by the GPMC_CONFIG4_i[3-0] OEONTIME bit field.
  - nOE second deassertion time is controlled by the GPMC_CONFIG4_i[12-8] OEOFFTIME bit field.

First write data is driven by the GPMC at GPMC_CONFIG6_i[19-16] WRDATAONADMUXBUS, when in address/data-multiplexed configuration. The next write data of the burst is driven on the bus at WRACCESSTIME + 1 during GPMC_CONFIG5_i[27-24] PAGEBURSTACCESSTIME GPMC_FCLK cycles. The last data of the synchronous burst write is driven until GPMC_CONFIG5_i[12-8] WRCYCLETIME completes.

- WRACCESSTIME is defined in the GPMC_CONFIG6_i[28-24] bit field.
- The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the memory-device internal configuration.

Total access time GPMC_CONFIG5_i[12-8] WRCYCLETIME corresponds to WRACCESSTIME plus the address hold time from nCS deassertion. In Figure 12-167, the programmed value of WRCYCLETIME equals WRCYCLETIME0 + WRCYCLETIME1. WRCYCLETIME0 and WRCYCLETIME1 delays are not actual parameters and are only a graphical representation of the full WRCYCLETIME value.

After a write operation, if no other access (read or write) is pending, the data bus keeps the previous value. See Section 12.3.3.4.8.10, *Bus Keeping Support*.

12.3.3.4.9.3 Asynchronous and Synchronous Accesses in non-multiplexed Mode

Page mode is available only in non-multiplexed mode.

- Asynchronous single-read operation on a non-multiplexed device
- Asynchronous single-write operation on a non-multiplexed device
- Asynchronous multiple- (page mode) read operation on a non-multiplexed device
- Synchronous operations on a non-multiplexed device

12.3.3.4.9.3.1 Asynchronous Single-Read Operation on non-multiplexed Device

Figure 12-169 shows an asynchronous single-read operation on a non-multiplexed device.

Copyright © 2026 Texas Instruments Incorporated

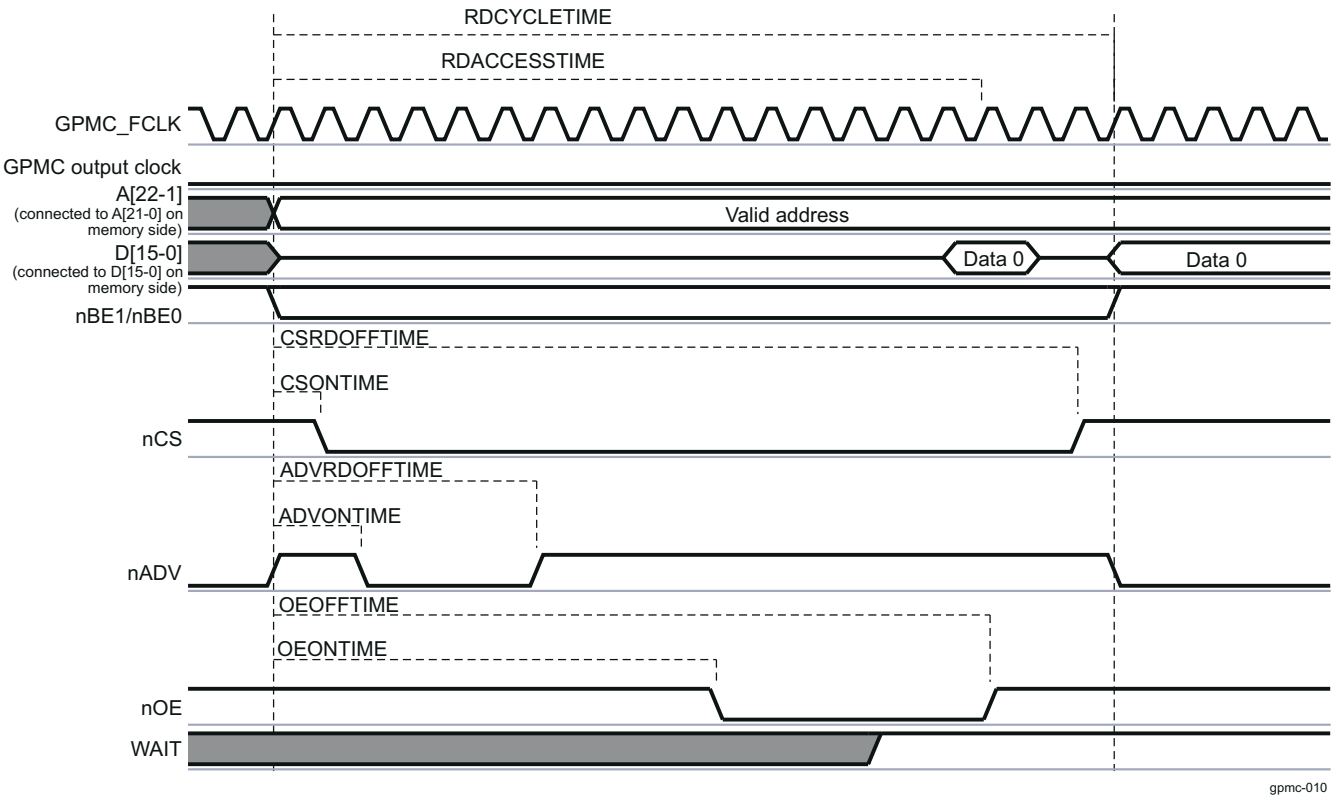


Figure 12-169. Asynchronous Single Read on an Address/Data-non-multiplexed Device

The 22-bit address (For a 16-bit data memory device, hence GPMC A[0] is not necessary to be output) is driven onto the address bus A[22-1] and the 16-bit data is driven onto the data bus D[15-0].

Read data is latched at GPMC_CONFIG1_5[20-16] RDACCESSTIME completion time. The end of the access is defined by the GPMC_CONFIG1_5[4-0] RDCYCLETIME parameter.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see Table 12-207, *NAND Memory Type*).

12.3.3.4.9.3.2 Asynchronous Single-Write Operation on non-multiplexed Device

Figure 12-170 shows an asynchronous single-write operation on a non-multiplexed device.

Copyright © 2026 Texas Instruments Incorporated

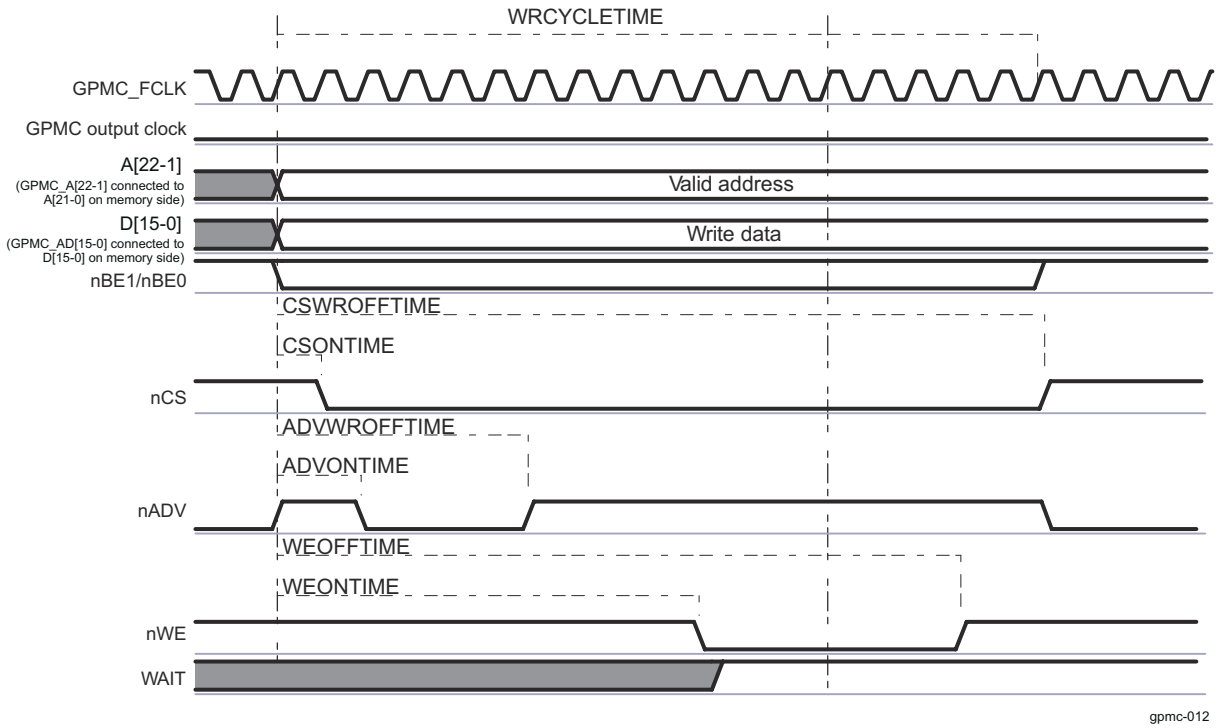


Figure 12-170. Asynchronous Single Write on an Address/Data-non-multiplexed Device

The nCS, nADV, nWE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see Table 12-207).

12.3.3.4.9.3.3 Asynchronous Multiple (Page Mode) Read Operation on non-multiplexed Device

Figure 12-171 shows an asynchronous multiple-read operation on a non-multiplexed device in which two word32 host read accesses to the GPMC are split into one multiple- (page mode of 4 word16) read access to the attached device.

Copyright © 2026 Texas Instruments Incorporated

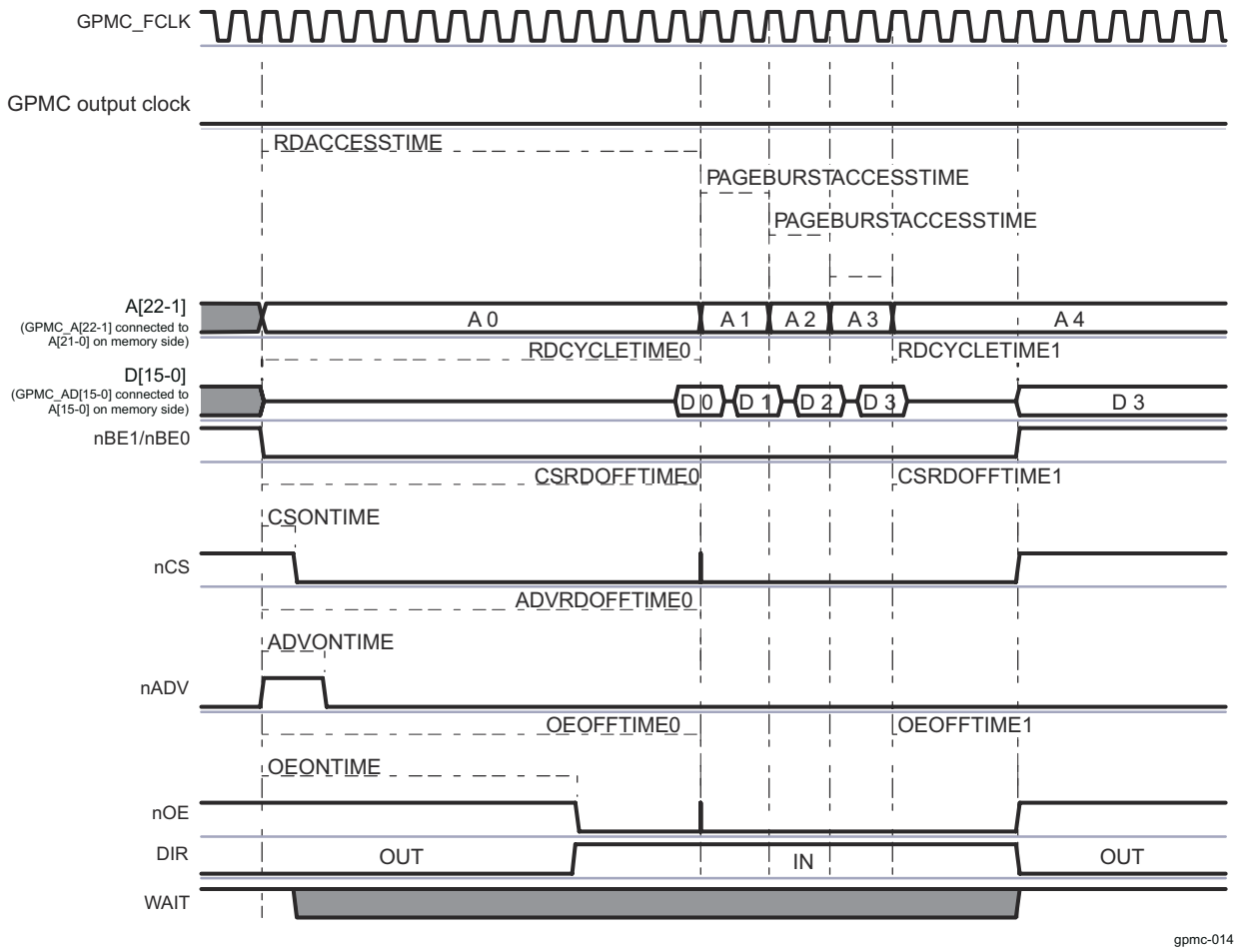


Figure 12-171. Asynchronous Multiple (Page Mode) Read

Note

The WAIT signal is active low.

The nCS, nADV, nOE, and DIR signals are controlled in the same way as address/data-multiplexed accesses (see Table 12-207).

When RDACCESSTIME completes, control signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

Read data is latched at *GPMC_CONFIG5_i*[20-16] RDACCESSTIME completion time (where i = 0 to 3). The end of the access is defined by the *GPMC_CONFIG5_i*[4-0] RDCYCLETIME parameter.

During consecutive accesses, the GPMC increments the address after each data read completes.

Delay between successive read data in the page is controlled by the *GPMC_CONFIG5_i*[27-24] PAGEBURSTACCESSTIME parameter. Depending on the device page length, the GPMC can control device page crossing during a burst request and insert initial RDACCESSTIME latency. Page crossing is possible only with a new burst access, meaning a new initial access phase is initiated.

Total access time RDCYCLETIME corresponds to RDACCESSTIME, plus the address hold time, starting from the nCS deassertion.

- The read cycle time is defined in the *GPMC_CONFIG5_i*[4-0] RDCYCLETIME bit field.

Copyright © 2026 Texas Instruments Incorporated

- In Figure 12-171, the programmed value of RDCYCLETIME equals RDCYCLETIME0 (before paged accesses) + RDCYCLETIME1 (after paged accesses).

12.3.3.4.9.3.4 Synchronous Operations on a non-multiplexed Device

All information for this section is equivalent to similar operations for address/data-multiplexed or AAD-multiplexed accesses. The only difference resides in the address phase. See Section 12.3.3.5.3, *GPMC Configuration in NOR Mode*.

12.3.3.4.9.4 Page and Burst Support

Each chip-select can be configured to process system single or burst requests into successive single accesses or asynchronous page/synchronous burst accesses, with appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through the GPMC. The GPMC_CONFIG1_i[30] READMULTIPLE and GPMC_CONFIG1_i[28] WRITMULTIPLE bits (where i = 0 to 3) are associated with the READTYPE and WRITETYPE parameters.

Note

- Asynchronous write page mode is not supported.
- 8-bit-wide device support is limited to nonburstable devices (READMULTIPLE and WRITEMULTIPLE are ignored).
- Not applicable to NAND device interfacing.

12.3.3.4.9.5 System Burst vs External Device Burst Support

The device system can issue the following requests to the GPMC:

- Byte, 16-bit word, 32-bit word requests (byte-enable-controlled). This is always a single request from the interconnect point of view.
- Incrementing fixed-length bursts of two, four, and eight words
- Wrapped (critical word access first) fixed-length burst of two, four, or eight words

To process a system request with the optimal protocol, the READMULTIPLE (and READTYPE) and WRITEMULTIPLE (and WRITETYPE) parameters must be set according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length bursts. The maximum length that can be issued is defined per chip-select by the *GPMC_CONFIG1_i*[24-23] ATTACHEDDEVICEPAGELENGTH bit field (where i = 0 to 3). When the value of ATTACHEDDEVICEPAGELENGTH is less than the length of the system burst request (including the appropriate access size adaptation according to the device width), the GPMC splits the system burst request into multiple bursts. Within the specified 4-, 8-, or 16-word value, the value of the ATTACHEDDEVICEPAGELENGTH bit field must correspond to the maximum length burst supported by the memory device configured in fixed-length burst mode (as opposed to continuous burst mode).

To get optimal performance from memory devices that natively support 16 Word16-length-wrapping burst capability (critical word access first), the ATTACHEDDEVICEPAGELENGTH parameter must be set to 16 words and the *GPMC_CONFIG1_i*[31] WRAPBURST bit (where i = 0 to 3) must be set to 1. Similarly DEVICEPAGELENGTH is set to 4 and 8 for memories supporting 4 and 8 Word16-length-wrapping burst, respectively.

When the memory device does not offer (or is not configured to offer) native 16 Word16-length-wrapping burst, the WRAPBURST parameter must be cleared, and the GPMC access engine emulates the wrapping burst by issuing the appropriate burst sequences according to the value of ATTACHEDDEVICEPAGELENGTH.

When the memory device does not support native-wrapping burst, there is usually no difference in behavior between a fixed-burst length mode and a continuous-burst mode configuration (except for a potential power increase from a memory-speculative data prefetch in a continuous burst read). However, even though continuous burst mode is compatible with GPMC behavior, because the GPMC access engine issues only

Copyright © 2026 Texas Instruments Incorporated

fixed-length burst and does not benefit from continuous burst mode, it is best to configure the memory device in fixed-length burst mode.

The memory device maximum-length burst (configured in fixed-length burst wrap or nonwrap mode) usually corresponds to the memory device data buffer size. Memory devices with a minimum of 16 half-word buffers are the most appropriate (especially with wrap support), but memory devices with smaller buffer size (4 or 8) are also supported, assuming that the *GPMC_CONFIG1_i*[24-23] ATTACHEDDEVICEPAGELENGTH bit field is set accordingly to 4 or 8 words.

The device system issues only requests with addresses or starting addresses for nonwrapping burst requests; that is, the request size boundary is aligned. In case of an eight-word-wrapping burst, the wrapping address always occurs on the eight-word boundary. As a consequence, all words requested must be available from the memory data buffer when the buffer size is equal to or greater than the value of ATTACHEDDEVICEPAGELENGTH. This usually means that data can be read from or written to the buffer at a constant rate (number of cycles between data) without wait-states between data accesses. If the memory does not behave this way (nonzero wait-state burstable memory), WAIT pin monitoring must be enabled to dynamically control data access completion within the burst.

Note

When the system burst request length is less than the value of ATTACHEDDEVICEPAGELENGTH, the GPMC proceeds with the required accesses.

12.3.3.4.10 GPMC pSRAM Access Specificities

pSRAM devices are SRAM-pin-compatible low-power memories that contain a self-refreshed DRAM memory array. The GPMC_CONFIG1_i[11-10] DEVICETYPE bit field (where i = 0 to 3) must be set to 0b00.

The pSRAM device uses the NOR protocol. It supports the following operations:

- Asynchronous single read
- Asynchronous page read
- Asynchronous single write
- Synchronous single read and write
- Synchronous burst read
- Synchronous burst write (not supported by NOR flash memory)

pSRAM devices must be powered up and initialized in a predefined manner according to the specifications of the attached device.

pSRAM devices can be programmed to use either mode: fixed or variable latency. pSRAM devices can automatically schedule autorefresh operations, which force the GPMC to use its WAIT signal capability when read or write operations occur during an internal self-refresh operation, or they can automatically include the autorefresh operation in the access time. These devices do not require additional WAIT signal capability or a minimum nCS high pulse width between consecutive accesses to ensure that the correct internal refresh operation is scheduled.

12.3.3.4.11 GPMC NAND Access Description

NAND (8-bit and 16-bit) memory devices using a standard NAND asynchronous address/data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings.

As for any other type of memory compatible with the GPMC interface, accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices. This interleaved capability limits the system to *chip enable don't care* NAND devices, because the chip-select allocated to the NAND device must be deasserted if accesses to other chip-selects are requested.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.11.1 NAND Memory Device in Byte or 16-bit Word Stream Mode

NAND devices require correct command and address programming before data array read or write accesses. The GPMC does not include specific hardware to translate a random address system request into a NAND-specific multiphase access. In that sense, GPMC NAND support, as opposed to random memory-map device support, is data stream-oriented (byte or 16-bit word).

The GPMC NAND programming model depends on a software driver for address and command formatting with the correct data address pointer value according to the block and page structure. Because of NAND structure and protocol interface diversity, the GPMC does not support automatic command and address phase programming, and software drivers must access the NAND device ID to ensure that correct command and address formatting are used for the identified device.

NAND device data read and write accesses are achieved through an asynchronous read or write access. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Any chip-select region can be qualified as a NAND region to constrain the nADV/ALE signal as ALE (ALE active high, default state value at low) during address program access, and the nBE0/CLE signal as CLE (CLE active high, default state value at low) during command program access. GPMC address lines are not used (the previous value is not changed) during NAND access.

12.3.3.4.11.1.1 Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode

The GPMC_CONFIG7_i register (where i = 0 to 3) associated with a NAND device region interfaced in byte or word stream mode can be initialized with a minimum size of 16MB, because any address location in the chip-select memory region can be used to access a NAND data array. The NAND flash protocol specifies an address sequence where address bits are passed through the data bus in a series of write accesses with the ALE pin asserted. After this address phase, all operations are streamed and the system requests address is irrelevant.

CAUTION

To allow correct command, address, and data-access controls, the GPMC_CONFIG1_i register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters listed in Table 12-177. Failure to comply with these settings corrupts the NAND interface protocol.

Table 12-177. Chip-Select Configuration for NAND Interfacing

Bit Field	Register	Value	Comments
WRAPBURST	GPMC_CONFIG1_i[31] (1)	0	No wrap
READMULTIPLE	GPMC_CONFIG1_i[30]	0	Single access
READTYPE	GPMC_CONFIG1_i[29]	0	Asynchronous mode
WRITEMULTIPLE	GPMC_CONFIG1_i[28]	0	Single access
WRITETYPE	GPMC_CONFIG1_i[27]	0	Asynchronous mode
CLKACTIVATIONTIME	GPMC_CONFIG1_i[26-25]	0b00	
ATTACHEDDEVICEPAGELENGTH	GPMC_CONFIG1_i[24-23]	Don't care	Single-access mode
WAITREADMONITORING	GPMC_CONFIG1_i[22]	0	Wait not monitored by GPMC access engine
WAITWRITEMONITORING	GPMC_CONFIG1_i[21]	0	Wait not monitored by GPMC access engine
WAITMONITORINGTIME	GPMC_CONFIG1_i[19-18]	Don't care	Wait not monitored by GPMC access engine
WAITPINSELECT	GPMC_CONFIG1_i[17-16]		Select which wait is monitored by edge detectors
DEVICESIZE	GPMC_CONFIG1_i[13-12]	0b00 or 0b01	8- or 16-bit interface

Copyright © 2026 Texas Instruments Incorporated

Table 12-177. Chip-Select Configuration for NAND Interfacing (continued)

Bit Field	Register	Value	Comments
DEVICETYPE	GPMC_CONFIG1_i[11-10]	0b10	NAND device in stream mode
MUXADDDATA	GPMC_CONFIG1_i[9-8]	0b00	non-multiplexed mode
TIMEPARAGRANULARITY	GPMC_CONFIG1_i[4]	0	Timing achieved with best GPMC clock granularity
GPMCFCLKDIVIDER	GPMC_CONFIG1_i[1-0]	Don't care	Asynchronous mode

(1) i = 0 to 3

The GPMC_CONFIG1_i to GPMC_CONFIG4_i registers (where i = 0 to 3) associated with a NAND device region must be initialized with the correct control-signal timing value according to the NAND device timing parameters.

12.3.3.4.11.1.2 NAND Device Command and Address Phase Control

NAND devices require multiple address programming phases. The software driver must issue the correct number of command and address program accesses, according to the device command set and the device address-mapping scheme.

NAND device-command and address-phase programming is achieved through write requests to the GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i register locations (where i = 0 to 3) with the correct command and address values. These locations are mapped in the associated chip-select register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Command and address values are not latched during the access and cannot be read back at the register location.

- Only write accesses must be issued to these locations, but the GPMC does not discard any read access. Accessing a NAND device with nOE and CLE or ALE asserted (read access) can produce undefined results.
- Write accesses to the GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i register locations must be posted for faster operations (where i = 0 to 3). The GPMC_CONFIG[0] NANDFORCEPOSTEDWRITE bit enables write accesses to these locations as posted, even if they are defined as nonposted.

A write buffer is used to store write transaction information before the external device is accessed:

- Up to eight consecutive posted write accesses can be accepted and stored in the write buffer.
- For nonposted write, the pipeline is one deep.
- An GPMC_STATUS[0] EMPTYWRITEBUFFERSTATUS bit stores the empty status of the write buffer.

The GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i registers (where i = 0 to 3) are 32-bit word locations, which means any 32-or 16-bit word access is split into 4- or 2-byte accesses if an 8-bit-wide NAND device is attached. For multiple-command phase or multiple-address phase, the software driver can use 32- or 16-bit word access to these registers, but it must consider the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to the GPMC_NAND_COMMAND_i and GPMC_NAND_ADDRESS_i registers can be used, and any of the four byte locations of the registers is valid.

The same applies to a GPMC_NAND_COMMAND_i and a GPMC_NAND_ADDRESS_i (where i = 0 to 3) 32-bit word write access to a 16-bit-wide NAND device (split into two 16-bit word accesses). In the case of a 16-bit word write access, the MSByte of the 16-bit word value must be set according to the NAND device requirement (usually 0). Either 16-bit word location or any one of the four byte locations of the registers is valid.

12.3.3.4.11.1.3 Command Latch Cycle

Writing data at the GPMC_NAND_COMMAND_i location (where i = 0 to 3) places the data as the NAND command value on the bus, using a regular asynchronous write access.

- nCE is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- CLE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.

Copyright © 2026 Texas Instruments Incorporated

- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE and nRE (nOE) are maintained inactive.

Figure 12-172 shows the NAND command latch cycle.

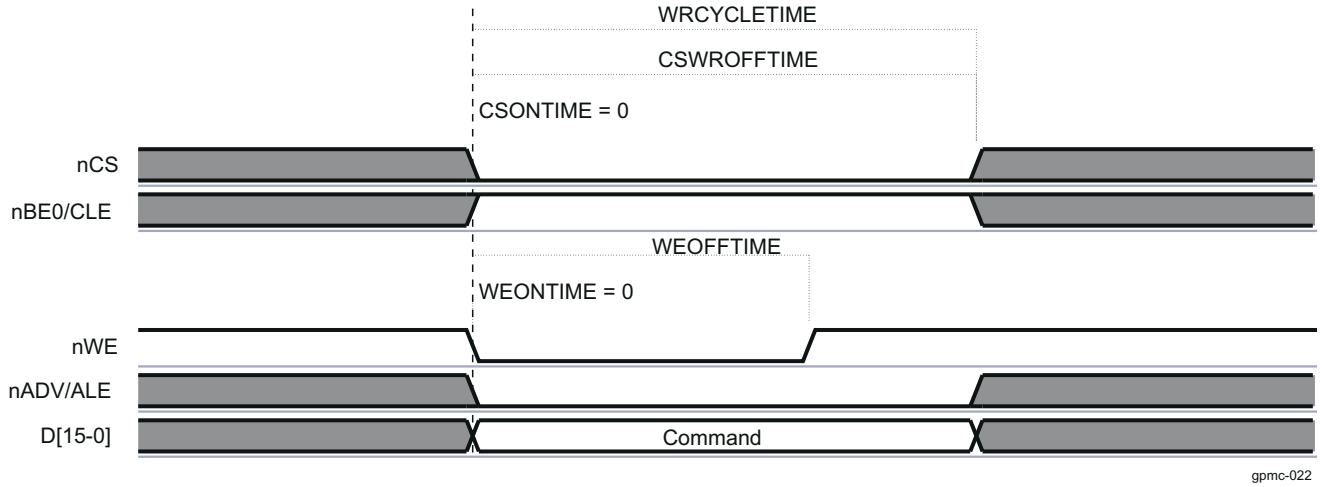


Figure 12-172. NAND Command Latch Cycle

Note

CLE is shared with the nBE0 output signal and has an inverted polarity from BE0. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, nBE0 (also nBE1) must not toggle, because it is shared with CLE.

NAND flash memories do not use byte-enable signals.

12.3.3.4.11.1.4 Address Latch Cycle

Writing data at the *GPMC_NAND_ADDRESS_i* location (where i = 0 to 3) places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- ALE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- CLE and nRE (nOE) are maintained inactive.

Figure 12-173 shows the NAND address latch cycle.

Copyright © 2026 Texas Instruments Incorporated

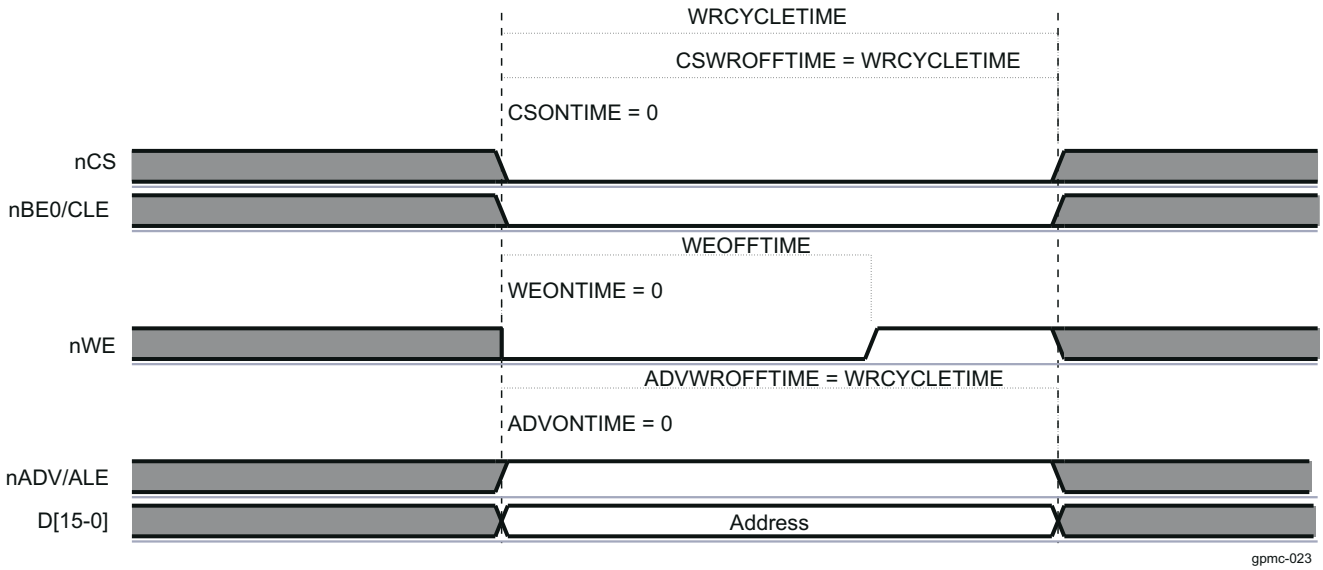


Figure 12-173. NAND Address Latch Cycle

Note

ALE is shared with the nADV output signal and has an inverted polarity from ADV. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, ALE is kept stable.

12.3.3.4.11.1.5 NAND Device Data Read and Write Phase Control in Stream Mode

NAND device data read and write accesses are achieved through a read or write request to the chip-select-associated memory region at any address location in the region or through a read or write request to the GPMC_NAND_DATA_i location (where i = 0 to 3) mapped in the chip-select-associated control register region. GPMC_NAND_DATA_i is not a true register, but an address location to enable nRE or nWE signal control. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Reading data from the GPMC_NAND_DATA_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

- nCS is controlled by the CSONTIME and CSRDOFFTIME timing parameters.
- nRE is controlled by the OEONTIME and OEOFFTIME timing parameters.
- To take advantage of nRE high-to-data invalid minimum timing value, RDACCESSTIME can be set so that data are effectively captured after nRE deassertion. This allows optimization of NAND read access cycle time completion. For optimal timing parameter settings, see the NAND device and the device timing parameters.

ALE, CLE, and nWE are maintained inactive.

Figure 12-174 shows the NAND data read cycle.

Copyright © 2026 Texas Instruments Incorporated

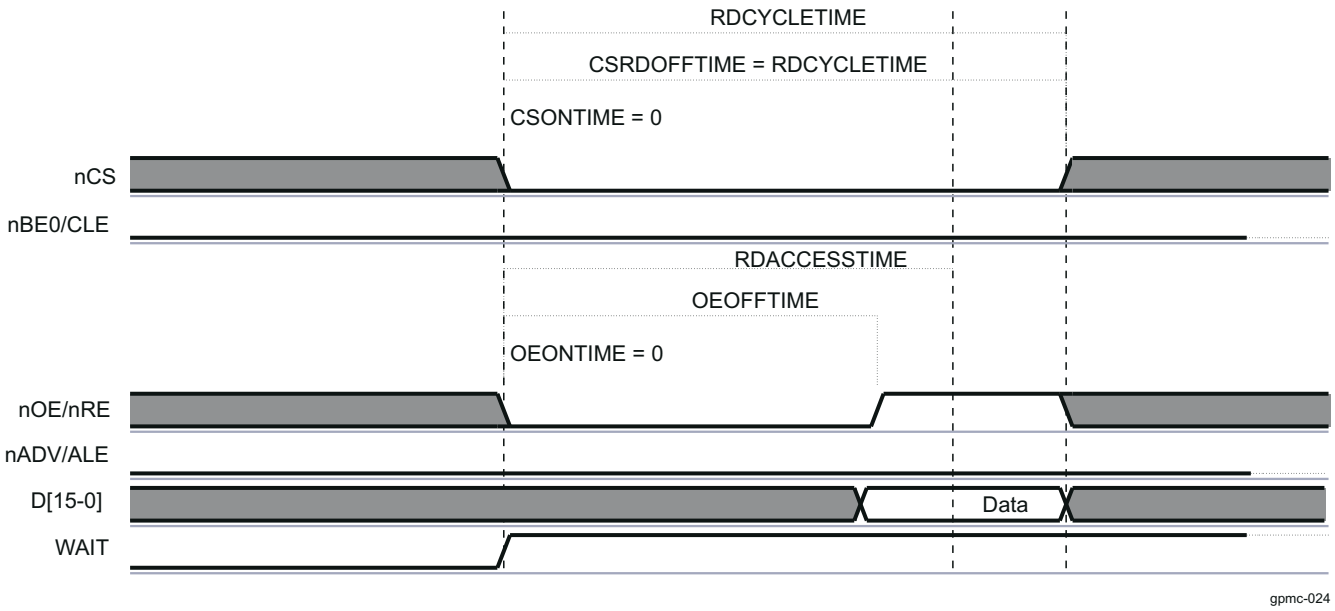


Figure 12-174. NAND Data Read Cycle

Writing data to the GPMC_NAND_DATA_i location or to any location in the associated chip-select memory region activates an asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE, CLE, and nRE (nOE) are maintained inactive.

Figure 12-175 shows the NAND data write cycle.

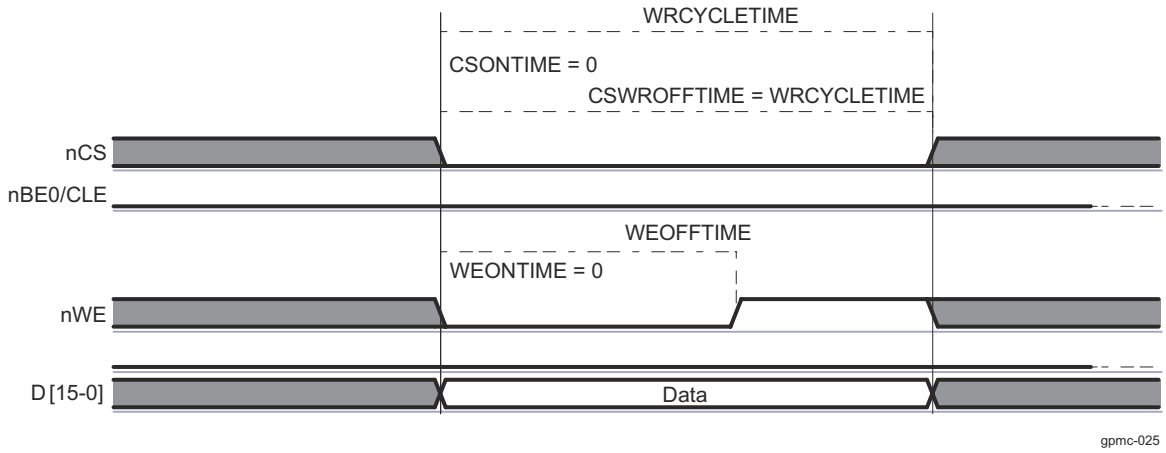


Figure 12-175. NAND Data Write Cycle

12.3.3.4.11.1.6 NAND Device General Chip-Select Timing Control Requirement

For most NAND devices, read data access time is dominated by nCS-to-data-valid timing and has faster nRE-to-data-valid timing. Successive accesses with nCS deassertions between accesses are affected by this timing constraint. Because accesses to a NAND device can be interleaved with other chip-select accesses, there is no certainty that nCS always stays low between two accesses to the same chip-select. Moreover, an nCS deassertion time between the same chip-select NAND accesses is likely to be required as follows: the nCS deassertion requires programming CYCLETIME and RDACCESSTIME according to the nCS-to-data-valid critical timing.

Copyright © 2026 Texas Instruments Incorporated

To get full performance from NAND read and write accesses, the prefetch engine can dynamically reduce the following on back-to-back NAND accesses (to the same memory) and suppress the minimum nCS high pulse width between accesses:

- RDCYCLETIME
- WRCYCLETIME
- RDACCESSTIME
- WRACCESSTIME
- CSRDOFFTIME
- CSWROFFTIME
- ADVRDOFFTIME
- ADVWROFFTIME
- OEOFFTIME
- WEOFFTIME

For more information about optimal prefetch engine access, see Section 12.3.3.4.11.4, *Prefetch and Write-Posting Engine*.

Some NAND devices require minimum write-to-read idle time, especially for device-status read accesses following status-read command programming (write access). If such write-to-read transactions are used, a minimum nCS high pulse width must be set. For this, CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDELAY must be set according to the appropriate timing requirement to prevent any timing violation.

NAND devices usually have an important nRE high-to-data bus in three-state mode. This requires a bus turnaround setting (BUSTURNAROUND = 1) so that the next access to a different chip-select is delayed until the BUSTURNAROUND delay completes. Back-to-back NAND read accesses to the same NAND flash are not affected by the programmed bus turnaround delay.

12.3.3.4.11.1.7 Read and Write Access Size Adaptation

12.3.3.4.11.1.7.1 8-Bit-Wide NAND Device

Host 16- and 32-bit word read and write access requests to a chip-select associated with an 8-bit-wide NAND device are split into successive read and write byte accesses to the NAND memory device. Byte access is ordered according to little-endian organization. A NAND 8-bit-wide device must be interfaced on the D0D7 interface bus lane. GPMC data accesses are justified on this bus lane when the cs is associated with an 8-bit-wide NAND device.

12.3.3.4.11.1.7.2 16-Bit-Wide NAND Device

Host 32-bit word read and write access requests to a chip-select associated with a 16-bit-wide NAND device are split into successive read and write 16-bit word accesses to the NAND memory device. 16-bit word access is ordered according to little-endian organization.

Host byte read and write access requests to a 16-bit-wide NAND device are completed as 16-bit accesses on the device itself, because there is no byte-addressing capability on 16-bit-wide NAND devices. This means that the NAND device address pointer is incremented on a 16-bit word basis and not on a byte basis. For a read access, only the requested byte is given back to the host, but the remaining byte is not stored or saved by the GPMC, and the next byte or 16-bit word read access gets the next 16-bit word NAND location. For a write access, the invalid byte part of the 16-bit word is driven to FF, and the next byte or 16-bit word write access programs the next 16-bit word NAND location.

Generally, byte access to a 16-bit-wide NAND device must be avoided, especially when ECC calculation is enabled. 8- or 16-bit ECC-based computations are corrupted by a byte read to a 16-bit-wide NAND device, because the nonrequested byte is considered invalid on a read access (not captured on the external data bus; FF is fed to the ECC engine) and is set to FF on a write access.

Host requests (read/write) issued in the chip-select memory region are translated in successive single or split accesses (read/write) to the attached device. Therefore, incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the access adaptation of the 32-bit to 8- or 16-bit device.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.11.2 NAND Device-Ready Pin

The NAND memory device provides a ready pin to indicate data availability after a block/page opening and to indicate that data programming is complete. The ready pin can be connected to one of the wait GPMC input pins; data read accesses must not be tried when the ready pin is sampled inactive (device is not ready) even if the associated chip-select WAITREADMONITORING bit field is set. The duration of the NAND device busy state after the block/page opening is so long (up to 50 micro second) that accesses occurring when the ready pin is sampled inactive can stall GPMC access and eventually cause a system time-out.

Note

If a read access to a NAND flash is done using WAIT monitoring mode, the device is blocked during a page opening, and so is the GPMC. If the correct settings are used, other chip-selects can be used while the memory processes the page opening command.

To avoid a time-out caused by a block/page opening delay in NAND flash, disable the WAIT pin monitoring for read and write accesses (that is, set the *GPMC_CONFIG1_i*[21] WAITWRITEMONITORING and *GPMC_CONFIG1_i*[22] WAITREADMONITORING bits to 0, where i = 0 to 3), and use one of the following methods instead:

- Use software to poll the WAITxSTATUS bit (where x = 0 to 1) of the *GPMC_STATUS*.
- Configure an interrupt that is generated on the WAIT signal change (through the *GPMC_IRQENABLE* register bits[11-8]).

Even if the READWAITMONITORING bit is not set, the external memory nR/B pin status is captured in the programmed wait bit in the *GPMC_STATUS* register.

The READWAITMONITORING bit method must be used for other memories than NAND flash, if they require the use of a WAIT signal.

12.3.3.4.11.2.1 Ready Pin Monitored by Software Polling

The ready signal state can be monitored through the GPMC_STATUS[9-8] WAITxSTATUS bit (where x = 0 to 1). Software must monitor the ready pin only when the signal is declared valid. Refer to the NAND device timing parameters to set the correct software temporization to monitor ready only after the invalid window is complete from the last read command written to the NAND device.

12.3.3.4.11.2.2 Ready Pin Monitored by Hardware Interrupt

Each GPMC_WAIT input pin can generate an interrupt when a wait-to-no-wait transition is detected. Depending on whether the GPMC_CONFIG[9-8] WAITxPINPOLARITY bits (where x = 0 to 1) is active low or active high, the wait-to-no-wait transition is a low-to-high external WAIT signal transition or a high-to-low external WAIT signal transition, respectively.

The wait transition pin detector must be cleared before any transition detection. This is done by writing 1 to the WAITxEDGEDETECTIONSTATUS bit (where x = 0 to 1) of the GPMC_IRQSTATUS register according to the GPMC_WAIT pin used for the NAND device-ready signal monitoring. To detect a wait-to-no-wait transition, the transition detector requires a wait active time detection of a minimum of two GPMC_FCLK cycles. Software must incorporate precautions to clear the wait transition pin detector before wait (busy) time completes.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit in the GPMC_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register is set.

The WAITMONITORINGTIME bit field does not affect wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC_IRQSTATUS register according to the GPMC_WAIT pin used for NAND device ready signal monitoring.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.11.3 ECC Calculator

The GPMC includes an error code correction (ECC) calculator circuitry that enables ECC calculation on the fly during data read or data program (that is, write) operations. The page size supported by the ECC calculator in one calculation/context is 512 bytes.

The user can choose from two different algorithms with different error correction capabilities through the *GPMC_ECC_CONFIG*[16] ECCALGORITHM bit:

1. Hamming code for 1-bit error code correction on 8- or 16-bit NAND flash organized with page size greater than 512 bytes
2. Bose-Chaudhurl-Hocquenghem (BCH) code for 4- to 16-bit error correction

The GPMC does not handle the error code correction directly. During writes, the GPMC computes parity bits. During reads, the GPMC provides enough information for the processor to correct errors without reading the data buffer all over again.

The Hamming code ECC is based on a 2-dimensional (2D) (row and column) bit parity accumulation. This parity accumulation is accomplished on the programmed number of bytes or 16-bit words read from the memory device, or is written to the memory device in stream mode.

Because the ECC engine includes only one accumulation context, it can be allocated to only one chip-select at a time through the *GPMC_ECC_CONFIG*[3-1] ECCCS bit field. Even if two chip-selects use different ECC algorithms, one the Hamming code and the other a BCH code, they must define separate ECC contexts because some of the ECC registers are common to all types of algorithms.

12.3.3.4.11.3.1 Hamming Code

All references to ECC in this subsection refer to the 1-bit error correction Hamming code.

The ECC is based on a 2D (row and column) bit parity accumulation known as the Hamming code. The parity accumulation is done for a programmed number of bytes or 16-bit word read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction, and the software NAND driver must read the multiple ECC calculation results, compare them to the expected code value, and take the appropriate corrective actions according to the error handling strategy (ECC storage in spare byte, error correction on read, block invalidation).

The ECC engine includes a single accumulation context. It can be allocated to a single designated chip-select at a time, and parallel computations on different chip-selects are not possible. Because it is allocated to a single chip-select, the ECC computation is not affected by interleaved GPMC accesses to other chip-selects and devices. The ECC accumulation is sequentially processed in the order of data read from or written to the memory on the designated chip-select. The ECC engine does not differentiate read accesses from write accesses and does not differentiate data from command or status information. Software must ensure that only relevant data are passed to the NAND flash memory while the ECC computation engine is active.

The starting NAND page location must be programmed first, followed by an ECC accumulation context reset with an ECC enabling, if required. The NAND device accesses discussed in the following sections must be limited to data read or write until the specified number of ECC calculations is complete.

12.3.3.4.11.3.1.1 ECC Result Register and ECC Computation Accumulation Size

The GPMC includes up to nine ECC result registers (*GPMC_ECCj_RESULT*, where j = 1 to 9) to store ECC computation results when the specified number of bytes or 16-bit words has been computed.

The ECC result registers are used sequentially: one ECC result is stored in one ECC result register on the list, the next ECC result is stored in the next ECC result register on the list, and so forth, until the last ECC computation. The value of the *GPMC_ECCj_RESULT* register is valid only when the programmed number of bytes or 16-bit words has been accumulated, which means that the same number of bytes or 16-bit words has been read from or written to the NAND device in sequence.

Copyright © 2026 Texas Instruments Incorporated

The *GPMC_ECC_CONTROL*[3-0] ECCPOINTER bit field must be set to the correct value to select the ECC result register to be used first in the list for the incoming ECC computation process. The ECCPointer can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The value of the *GPMC_ECCj_RESULT* register (where j = 1 to 9) can be considered valid when ECCPOINTER equals j + 1. When the *GPMC_ECCj_RESULT* register (where j = 9) is updated, ECCPOINTER is frozen at 10, and ECC computing is stopped (ECCENABLE = 0).

The ECC accumulator must be reset before any ECC computation accumulation process. The *GPMC_ECC_CONTROL*[8] ECCCLEAR bit must be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each *GPMC_ECCj_RESULT* register, where j = 1 to 9), the number of bytes or 16-bit words used for ECC computing accumulation can be selected from between two programmable values.

The ECCjRESULTSIZE bits (where j = 1 to 9) in the *GPMC_ECC_SIZE_CONFIG* register select which programmable size value (ECCSIZE0 or ECCSIZE1) must be used for this ECC result (stored in the *GPMC_ECCj_RESULT* register).

The ECCSIZE0 and ECCSIZE1 bit fields allow selection of the number of bytes or 16-bit words used for ECC computation accumulation. Any even values from 2 to 512 are allowed.

Flexibility in the number of ECCs computed and the number of bytes or 16-bit words used in the successive ECC computations enables different NAND page error-correction strategies. Usually based on 256 or 512 bytes and on 128 or 256 16-bit word, the number of ECC results required is a function of the NAND device page size. Specific ECC accumulation size can be used when computing the ECC on the NAND spare byte.

For example, with a 2-KB data page, 8-bit-wide NAND device, eight ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24 spare bytes area where the eight ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC then provides nine *GPMC_ECCj_RESULT* registers (j = 1 to 9) to store the results. In this case, ECCSIZE0 is set to 256, and ECCSIZE1 is set to 24; the ECC[1-8]RESULTSIZE bits are set to 0, and the ECC9RESULTSIZE bit is set to 1.

12.3.3.4.11.3.1.2 ECC Enabling

The *GPMC_ECC_CONFIG*[3-1] ECCCS bit field selects the allocated chip-select. The *GPMC_ECC_CONFIG*[0] ECCENABLE bit enables ECC computation on the next detected read or write access to the selected chip-select.

The following fields must not be changed or cleared while an ECC computation is in progress:

- CCPOINTER
- ECCCLEAR
- ECCSIZE
- ECCjRESULTSIZE (where j = 1 to 9)
- ECC16B
- ECCCS

The ECC accumulator and ECC result register must not be changed or cleared while an ECC computation is in progress.

Table 12-178 describes the ECC enable settings.

Table 12-178. ECC Enable Settings

Bit Field	Register	Value	Comments
ECCCS	*GPMC_ECC_CONFIG*	0–3	Selects the chip-select where ECC is computed
ECC16B	*GPMC_ECC_CONFIG*	0/1	Selects column number for ECC calculation
ECCCLEAR	*GPMC_ECC_CONTROL*	0–7	Clears all ECC result registers
ECCPOINTER	*GPMC_ECC_CONTROL*	0–7	A write to this bit field selects the ECC result register where the first ECC computation is stored. Set to 1 by default.

Copyright © 2026 Texas Instruments Incorporated

Table 12-178. ECC Enable Settings (continued)

Bit Field	Register	Value	Comments
ECCSIZE1	*GPMC_ECC_SIZE_CONFIG*	0x00–0xFF	Defines ECCSIZE1
ECCSIZE0	*GPMC_ECC_SIZE_CONFIG*	0x00–0xFF	Defines ECCSIZE0
ECCjRESULTSIZE (j from 1 to 9)	*GPMC_ECC_SIZE_CONFIG*	0/1	Selects the size of ECCn result register
ECCENABLE	*GPMC_ECC_CONFIG*	1	Enables the ECC computation

12.3.3.4.11.3.1.3 ECC Computation

The ECC algorithm is a multiple parity bit accumulation computed on the odd and even bit streams extracted from the byte or Word16 streams. The parity accumulation is split into row and column accumulations, as shown in Figure 12-176 and Figure 12-177. The intermediate row and column parities are used to compute the upper level row and column parities. Only the final computation of each parity bit is used for ECC comparison and correction.

P1o = bit7 XOR bit5 XOR bit3 XOR bit1 on each byte of the data stream
P1e = bit6 XOR bit4 XOR bit2 XOR bit0 on each byte of the data stream

P2o = bit7 XOR bit6 XOR bit3 XOR bit2 on each byte of the data stream
P2e = bit5 XOR bit4 XOR bit1 XOR bit0 on each byte of the data stream

P4o = bit7 XOR bit6 XOR bit5 XOR bit4 on each byte of the data stream
P4e = bit3 XOR bit2 XOR bit1 XOR bit0 on each byte of the data stream

Each column parity bit is XORed with the previous accumulated value.

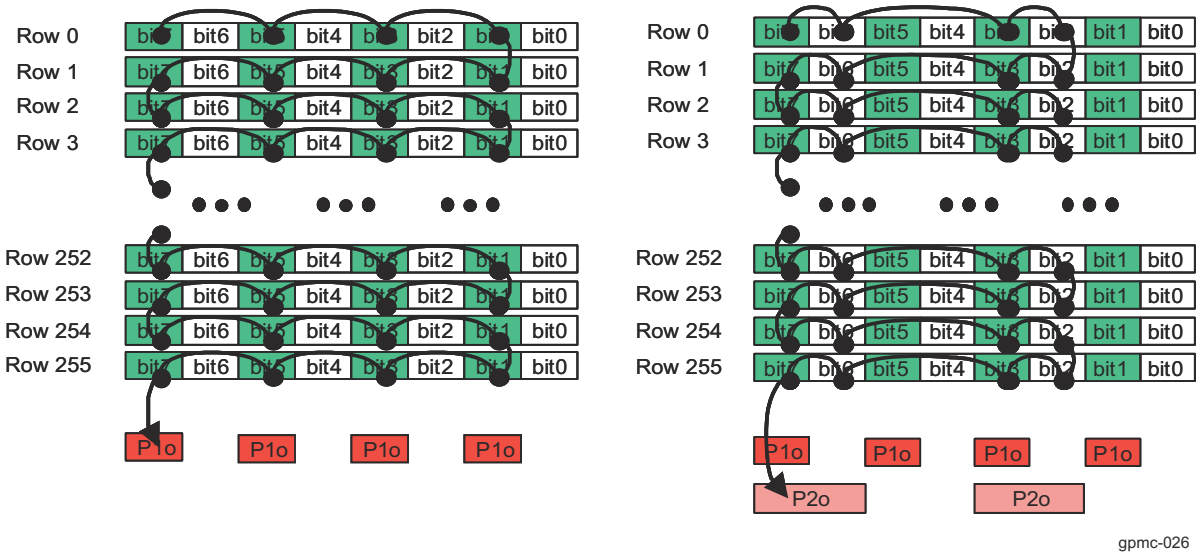


Figure 12-176. Hamming Code Accumulation Algorithm (1/2)

For line parities, the bits of each new data are XORed together, and line parity bits are computed as described below:

P8e = row0 XOR row2 XOR row4 XOR ... XOR row254
P8o = row1 XOR row3 XOR row5 XOR ... XOR row255

P16e = row0 XOR row1 XOR row4 XOR row5 XOR ... XOR row252 XOR row 253
P16o = row2 XOR row3 XOR row6 XOR row7 XOR ... XOR row254 XOR row 255

Copyright © 2026 Texas Instruments Incorporated

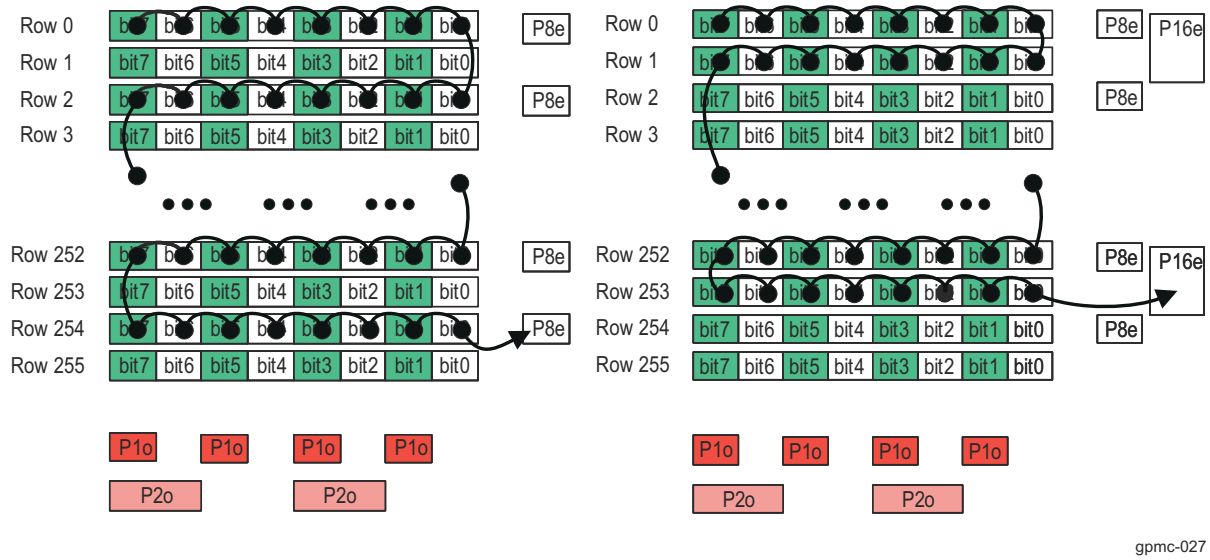


Figure 12-177. Hamming Code Accumulation Algorithm (2/2)

Unused parity bits in the result registers are set to 0.

Figure 12-178 shows ECC computation for a 256-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and sixteen row parity bits (P8o-P16o-P32o--P1024o for odd parities, and P8e-P16e-P32e--P1024e for even parities).

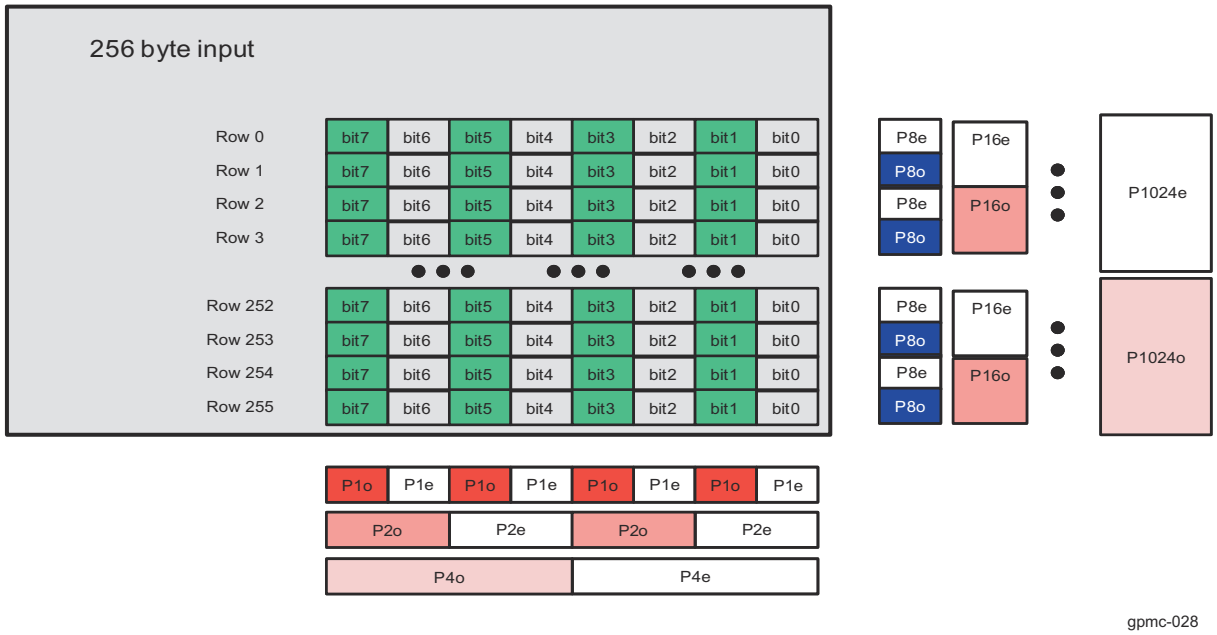


Figure 12-178. ECC Computation for a 256-Byte Data Stream (Read or Write)

Figure 12-179 shows ECC computation for a 512-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and eighteen row parity bits (P8o-P16o-P32o--P1024o- - P2048o for odd parities, and P8e-P16e-P32e--P1024e- P2048e for even parities).

Copyright © 2026 Texas Instruments Incorporated

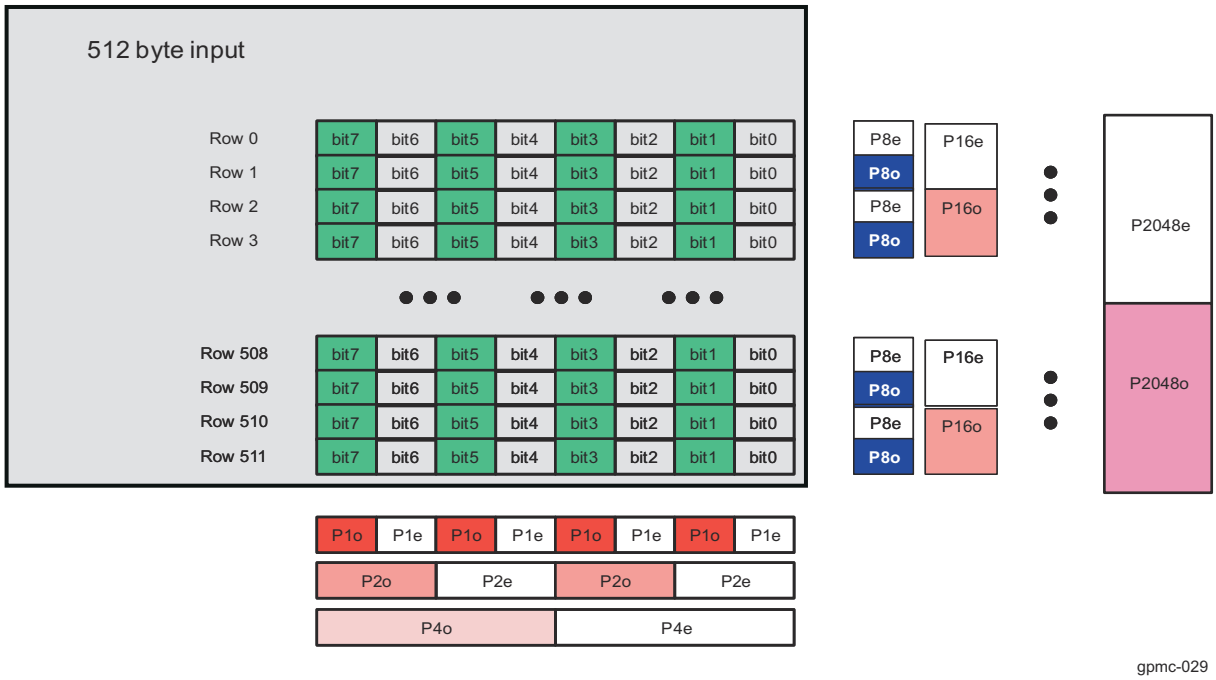


Figure 12-179. ECC Computation for a 512-Byte Data Stream (Read or Write)

For a 2-KB page, four 512 bytes ECC calculations plus 1 for the spare area are required. Results are stored in the *GPMC_ECCj_RESULT* registers (where j = 1 to 9).

12.3.3.4.11.3.1.4 ECC Comparison and Correction

To detect an error, the computed ECC result must be XORed with the parity value stored in the spare area of the accessed page.

- If the result of this logical XOR is all 0s, no error is detected and the read data is correct.
- If every second bit in the parity result is a 1, 1 bit is corrupted and is located at bit address (P2048o, P1024o, P512o, P256o, P128o, P64o, P32o, P16o, P8o, P4o, P2o, P1o). Software must correct the corresponding bit.
- If only 1 bit in the parity result is 1, it is an ECC error and the read data is correct.

12.3.3.4.11.3.1.5 ECC Calculation Based on 8-Bit Word

The 8-bit-based ECC computation is used for 8-bit-wide NAND device interfacing.

The 8-bit-based ECC computation can be used for 16-bit-wide NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit-wide NAND devices. In this case, the 16-bit-wide data read from or written to the NAND device is fragmented into 2 bytes. According to little-endian access, the LSB of the 16-bit-wide data is ordered first in the byte stream used for 8-bit-based ECC computation.

12.3.3.4.11.3.1.6 ECC Calculation Based on 16-Bit Word

ECC computation based on an 16-bit word is used for 16-bit-wide NAND device interfacing. This ECC computation is not supported when interfacing an 8-bit-wide NAND device, and the GPMC_ECC_CONFIG[7] ECC16B bit must be set to 0 when interfacing an 8-bit-wide NAND device.

The parity computation based on 16-bit words affects the row and column parity mapping. The main difference is that the odd and even parity bits P8o and P8e are computed on rows for an 8-bit-based ECC and on columns for a 16-bit based ECC. Figure 12-180 and Figure 12-181 show a 128 Word16 ECC computation scheme and a 256 16-bit word ECC computation scheme.

Copyright © 2026 Texas Instruments Incorporated

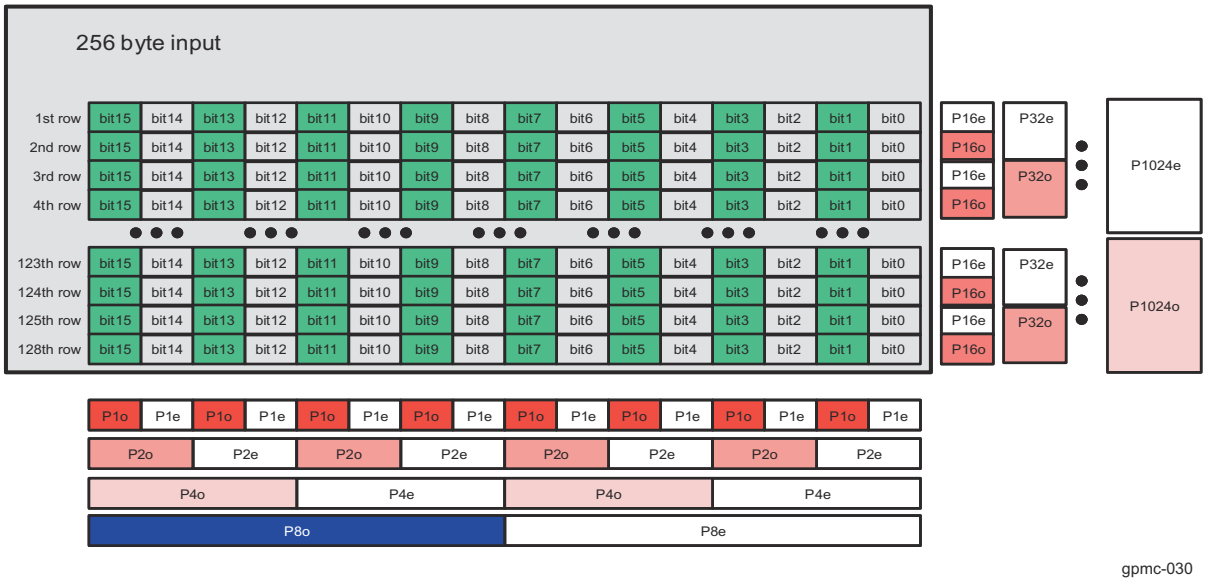


Figure 12-180. 128 Word16 ECC Computation

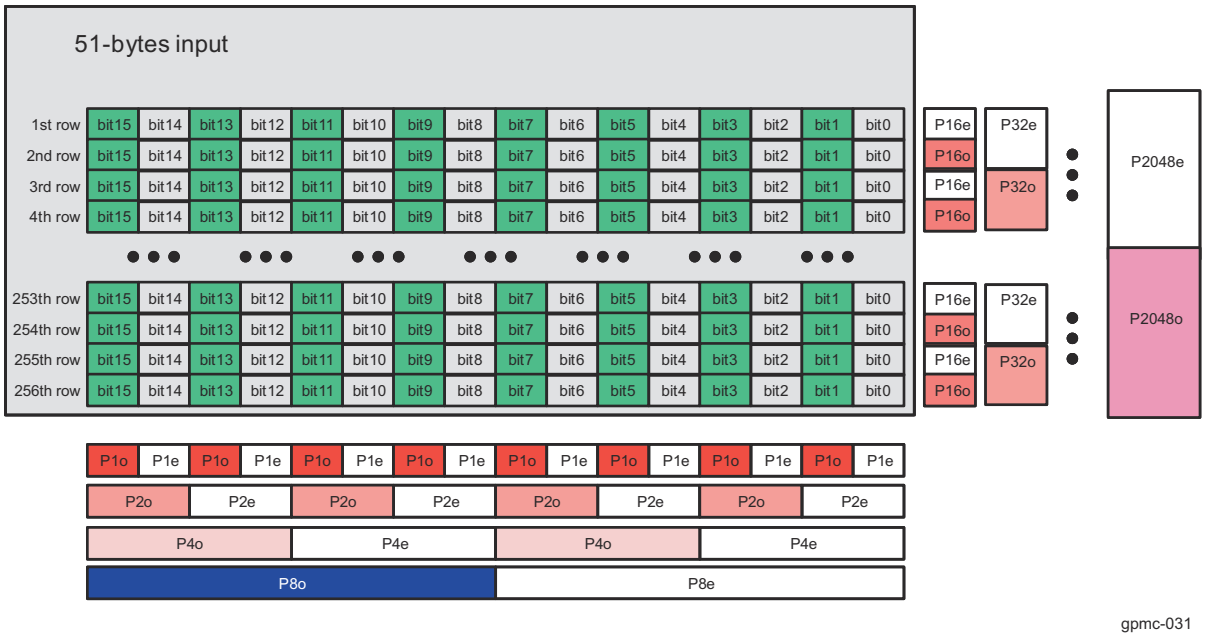


Figure 12-181. 256 Word16 ECC Computation

12.3.3.4.11.3.2 BCH Code

All references to ECC in this subsection refer to the 4- to 16-bit error correction BCH code.

12.3.3.4.11.3.2.1 Requirements

1. Read and write accesses to a NAND flash take place by whole pages, in a predetermined sequence: first the data byte page itself, and then some spare bytes, including the BCH ECC (and other information). The NAND device can cache a full page, including spares, for read and write accesses.
   - Typical page write sequence:
     - Sequential write to NAND cache of main data plus spare data, for a page. ECC is calculated on the fly. Calculated ECC can be inserted on the fly in the spares or replaced by dummy accesses.

Copyright © 2026 Texas Instruments Incorporated

  - When the calculated ECC is replaced by dummy accesses, it must be written to the cache in a second, separate phase. The ECC module is disabled during that time.
  - NAND writes its cache line (page) to the array.
- Typical page read sequence:
  - Sequential read of a page. ECC is calculated on the fly.
  - The status of the ECC module buffers determines the presence of errors.

2. Accesses to several memories can be interleaved by the GPMC, but only one of those memories at a time can be a NAND using the BCH engine; in other words, only one BCH calculation (for example, for a single page) can be ongoing at any time. The sequential nature of NAND accesses ensures that the data is always written or read out in the same order. BCH-relevant accesses are selected by the chip-selects of the GPMC.
3. Each page can hold up to 4KB of data, spare bytes not included. This means up to 8 × 512-byte BCH messages. Because all the data is written or read out first, followed by the BCH ECC, the BCH engine must be able to hold eight 104-bit remainders or syndromes (or smaller, 52-bit ones) at the same time.

   The BCH module can store all remainders internally. After the page start, an internal counter is used to detect the 512-byte sector boundaries. On those boundaries, the current remainder is stored and the divider reset for the next calculation. At the end of the page, the BCH module contains all remainders.
4. NAND access cycles hold 8 or 16 bits of data each (1 or 2 bytes); Each NAND cycle takes at least four cycles of the GPMC internal clock. This means the NAND flash timing parameters must define a RDCYCLETIME and a WRCYCLETIME of at least four clock cycles after optimization when using the BCH calculator.
5. The spare area is assumed to be large enough to hold the BCH ECC; that is, to have a message of at least 13 bytes available per 512-byte sector of data. The zone of unused spare area by the ECC may or may not be protected by the same ECC scheme, by extending the BCH message beyond 512 bytes (the maximum codeword is 1023 bytes long, ECC included, which leaves much space to cover spare bytes).

12.3.3.4.11.3.2.2 Memory Mapping of BCH Codeword

BCH encoding considers a block of data to protect as a polynomial message M(x). In a standard case, 512 bytes of data (that is, 2^{12} bits = 4096 bits) are seen as a polynomial of degree $2^{12} – 1 = 4095$, with parameters ranging from M0 to M4095. For 512 bytes of data, 52 bits are required for 4-bit error correction, 104 bits are required for 8-bit error correction, and 207 bits are required for 16-bit error correction. The ECC is a remainder polynomial R(x) of degree 103 (or 51, depending on the selected mode). The complete codeword C(x) is the concatenation of M(x) and R(x), as described in Table 12-179.

Table 12-179. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits)

	Message M(x)			ECC R(x)		
Bit Number	M4095	...	M0	R103	...	R0

If the message is extended by the addition of spare bytes to be protected by the same ECC, the principle is still valid. For example, a 3-byte extension of the message gives a polynomial message M(x) of degree ((512 + 3) × 8) – 1 = 4119, for a total of 3 + 13 = 16 spare bytes of spare, all protected as part of the same codeword.

The message and the ECC bits are manipulated and mapped in the GPMC byte-oriented system. The ECC bits are stored in the following registers (where i = 0 to 3):

- GPMC_BCH_RESULT_0_i
- GPMC_BCH_RESULT_1_i
- GPMC_BCH_RESULT_2_i
- GPMC_BCH_RESULT_3_i

12.3.3.4.11.3.2.2.1 Memory Mapping of Data Message

The data message mapping must follow the following rules:

- Bit endianness within a byte is little-endian; that is, the bytes LSB is also the lowest-degree polynomial parameter: a byte b7-b0 (with b0 the LSB) represents a segment of polynomial $b7 * x^{(7+i)} + b6 * x^{(6+i)} + ... + b0 * x^{i}$

Copyright © 2026 Texas Instruments Incorporated

- The message is mapped in the NAND starting with the highest-order parameters, that is, in the lowest addresses of a NAND page.
- Byte endianness within the 16-bit words in the NAND is big-endian (that is, the same message mapped in 8- and 16-bit memories has the same content at the same byte address).

Note

The BCH module has no visibility over actual addresses. The most important point is the sequence of data words the BCH sees. However, the NAND page is always scanned incrementally in read and write accesses, which produces the mapping patterns described in the following.

Table 12-180 and Table 12-181 describe the mapping of the same 512-byte vector (typically, a BCH message) in the NAND memory space. The byte address is only an offset module 512 (0x200), because the same page may contain several contiguous 512-byte sectors (BCH blocks). The LSB and MSB are, respectively, the bits M0 and M(2^{12}–1) of the codeword mapping discussed previously. In both cases the data vectors are aligned; that is, their boundaries coincide with the RAM data word boundaries.

Table 12-180. Aligned Message Byte Mapping in 8-Bit NAND

Byte Offset	8-Bit Word
0x000	(MSB) Byte 511 (0x1FF)
0x001	Byte 510 (0x1FE)
...	...
0x1FF	Byte 0 (0x0) (LSB)

Table 12-181. Aligned Message Byte Mapping in 16-Bit NAND

Byte Offset	16-Bit Word MSB	16-Bit Word LSB
0x000	Byte 510 (0x1FE)	(MSB) Byte 511 (0x1FF)
0x002	Byte 508 (0x1FC)	Byte 509 (0x1FD)
...	...	...
0x1FE	Byte 0 (0x0)	(LSB) Byte 1 (0x1)

Table 12-182 through Table 12-187 list the mapping in memory of arbitrarily-sized messages, starting on access (byte or 16-bit word) boundaries for more clarity. Note that message may actually start and stop on arbitrary nibbles. A nibble is a 4-bit entity. The unused nibbles are not discarded, and they can still be used by the BCH module, but as part of the next message section (for example, on the ECC of another sector).

Table 12-182. Aligned Nibble Mapping of Message in 8-Bit NAND

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Least Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
S/2 – 2	Nibble 3	Nibble 2
S/2 – 1	Nibble 1	Nibble 0 (LSB)

Table 12-183. Misaligned Nibble Mapping of Message in 8-Bit NAND

Byte Offset	8-Bit Word	
	4-Bit Most Significant Nibble	4-Bit Least Significant Nibble
1	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
(S + 1) / 2 – 2	Nibble 2	Nibble 1

Copyright © 2026 Texas Instruments Incorporated

Table 12-183. Misaligned Nibble Mapping of Message in 8-Bit NAND (continued)

Byte Offset	8-Bit Word	
(S + 1) / 2 – 1	Nibble 0 (LSB)	

Table 12-184. Aligned Nibble Mapping of Message in 16-Bit NAND

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
S/2 – 4	Nibble 5	Nibble 4	Nibble 7	Nibble 6
S/2 – 2	Nibble 1	Nibble 0 (LSB)	Nibble 3	Nibble 2

Table 12-185. Misaligned Nibble Mapping of Message in 16-Bit NAND (1 Unused Nibble)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S + 1) / 2 – 4	Nibble 4	Nibble 3	Nibble 6	Nibble 5
(S + 1) / 2 – 2	Nibble 0 (LSB)		Nibble 2	Nibble 1

Table 12-186. Misaligned Nibble Mapping of Message in 16-Bit NAND (2 Unused Nibbles)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S + 2) / 2 – 4	Nibble 3	Nibble 2	Nibble 5	Nibble 4
(S + 2) / 2 – 2			Nibble 1	Nibble 0 (LSB)

Table 12-187. Misaligned Nibble Mapping of Message in 16-Bit NAND (3 Unused Nibbles)

Byte Offset	16-Bit Word			
	4-Bit Most Significant Nibble		4-Bit Least Significant Nibble	
0	Nibble S-3	Nibble S-4	(MSB) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S + 3) / 2 – 4	Nibble 2	Nibble 1	Nibble 4	Nibble 3
(S + 3) / 2 – 2			Nibble 0 (LSB)	

Many other cases exist than those given in the previous tables; for example, where the message does not start on a word boundary.

12.3.3.4.11.3.2.2.2 Memory-Mapping of the ECC

The ECC (or remainder) is presented by the BCH module as a single 104-bit (or 52-bit), little-endian vector. Software must fetch those 13 bytes (or 6 bytes) from the module interface and then store them to the spare area (page write) in the NAND or to an intermediate buffer for comparison with the stored ECC (page read). There are no constraints on the ECC mapping inside the spare area: it is software-controlled.

It is advised, however, to maintain a coherence in the respective formats of the message or the ECC remainder once they have been read out of the NAND. The error correction algorithm works from the complete codeword

Copyright © 2026 Texas Instruments Incorporated

(concatenated message and remainder) once an error is detected. The creation of this codeword must be made as straightforward as possible.

There are cases in which the same NAND access contains both data and the ECC protecting that data. This is the case when the data/ECC boundary (which can be on any nibble) does not coincide with an access boundary. The ECC is calculated on the fly following the write. In that case, the write must also contain part of the ECC because it is impossible to insert the ECC on the fly. Instead:

- During the initial page write (BCH encoding), the ECC is replaced by dummy bits. The BCH encoder is by definition turned OFF during the ECC section, so the BCH result is unmodified.
- During a second phase, the ECC is written to the correct location, next to the actual data.
- The completed line buffer is then written to the NAND array.

12.3.3.4.11.3.2.2.3 Wrapping Modes

For a given wrapping mode, the module automatically goes through a specific number of sections as data is being fed into the module. For each section, the BCH core can be enabled (in which case the data is fed to the BCH divider) or not (in which case the BCH simply counts to the end of the section). When enabled, the data is added to the ongoing calculation for a given sector number (for example, number 0).

Wrapping modes are described as follows. To better understand and see the real-life read and write sequences implemented with each mode, see Section 12.3.3.4.11.3.2.3, *Supported NAND Page Mappings and ECC Schemes*.

For each mode:

- A sequence describes the mode in pseudo language, with, for each section, the size and the buffer used for ECC processing (if ON). The programmable lengths are size, size0, and size1.
- A checksum condition is given. If the checksum condition is not respected for a given mode, the module behavior is unpredictable. S is the number of sectors in the page; size0 and size1 are the section sizes programmed for the mode, in nibbles.

Wrapping modes 8 through 11 insert a 1-nibble padding where the BCH processing is OFF. This is intended for t = 4 ECC, where ECC is 6 bytes long and the ECC area is expected to include (at least) 1 unused nibble to remain byte-aligned.

12.3.3.4.11.3.2.2.3.1 Manual Mode (0x0)

This mode is intended for short sequences, added manually to a given buffer through the software data port input. A complete page may be built out of several such sequences.

To process an arbitrary sequence of 4-bit nibbles, accesses to the software data port, containing the appropriate data, must be made. If the sequence end does not coincide with an access boundary (for example, to process 5 nibbles = 20 bits in 16-bit access mode) and those nibbles must be skipped, a number of unused nibbles must be programmed in GPMC_ECC_SIZE_CONFIG[31-22] ECCSIZE1 In the same example, 5 nibbles to process + 3 to discard = 8 nibbles = 2 × 16-bit accesses. Software must set:

- GPMC_ECC_SIZE_CONFIG[21-12] ECCSIZE0 = 0x5
- GPMC_ECC_SIZE_CONFIG[31-22] ECCSIZE1 = 0x3

Note

In the following figures, size and size0 are the same parameter.

Copyright © 2026 Texas Instruments Incorporated

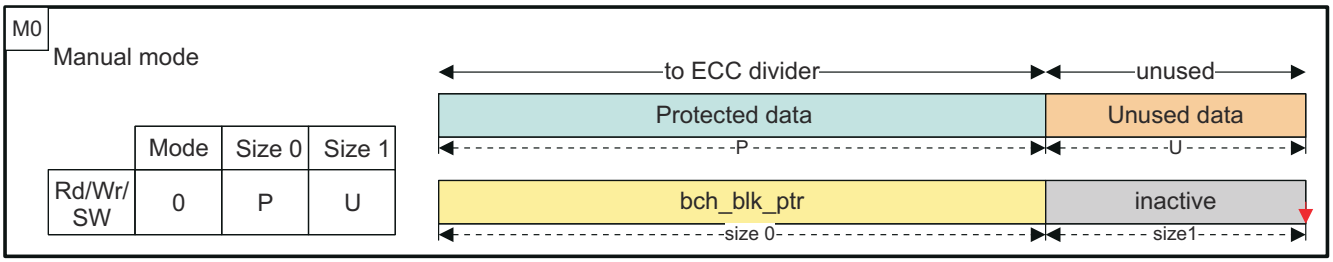


Figure 12-182. Manual Mode Sequence and Mapping

Section processing sequence:

- One time with buffer
  - size0 nibbles of data, processing ON
  - size1 nibbles of unused data, processing OFF

Checksum: size0 + size1 nibbles must fit in a whole number of accesses.

In the following sections, S is the number of sectors in the page.

12.3.3.4.11.3.2.2.3.2 Mode 0x1

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + size1)

12.3.3.4.11.3.2.2.3.3 Mode 0xA (10)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - 1 nibble pad spare, processing OFF
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + 1 + size1)

12.3.3.4.11.3.2.2.3.4 Mode 0x2

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + size1)

12.3.3.4.11.3.2.2.3.5 Mode 0x3

Page processing sequence:

Copyright © 2026 Texas Instruments Incorporated

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – size1)

12.3.3.4.11.3.2.2.3.6 Mode 0x7

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = size0 + (S – size1)

12.3.3.4.11.3.2.2.3.7 Mode 0x8

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – (1 + size1))

12.3.3.4.11.3.2.2.3.8 Mode 0x4

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S – size1)

12.3.3.4.11.3.2.2.3.9 Mode 0x9

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Copyright © 2026 Texas Instruments Incorporated

Checksum: Spare area size (nibbles) = size0 + (S – (1 + size1))

12.3.3.4.11.3.2.2.3.10 Mode 0x5

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + size1)

12.3.3.4.11.3.2.2.3.11 Mode 0xB (11)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S – (size0 + 1 + size1)

12.3.3.4.11.3.2.2.3.12 Mode 0x6

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S – (size0 + size1)

12.3.3.4.11.3.2.3 Supported NAND Page Mappings and ECC Schemes

The following rules apply to the entire mapping description:

- Main data area (sectors) size is hardcoded to 512 bytes.
- Spare area size is programmable.
- All page sections (of main area data bytes, protected spare bytes, unprotected spare bytes, and ECC) are defined as explained in Section 3.3.4.11.3.2.2.1, *Memory Mapping of Data Message*.

Each of the following sections gives a NAND page mapping example (per-sector spare mappings, pooled spare mapping, per-sector spare mapping, with ECC separated at the end of the page).

In the mapping diagrams, sections that belong to the same BCH codeword have the same color (blue or green); unprotected sections are not covered (orange) by the BCH scheme.

Below each mapping diagram, a write (encoding) and read (decoding: syndrome generation) sequence is given, with the number of the active buffers at each point in time (yellow). In the inactive zones (grey), no computing is taking place but the data counter is still active.

Copyright © 2026 Texas Instruments Incorporated

In Figure 12-183 through Figure 12-185, the tables on the left summarize the mode, size0, and size1 parameters to program for, respectively, write and read processing of a page, with the given mapping, where:

- P is the size of spare byte section Protected by the ECC (in nibbles)
- U is the size of spare byte section Unprotected by the ECC (in nibbles)
- E is the size of the ECC (in nibbles)
- S is the number of Sectors per page (two in the current diagrams)

Each time the processing of a BCH block is complete (ECC calculation for write/encoding, syndrome generation for read/decoding, indicated by red arrows), the update pointer is pulsed. The processing for block 0 can be the first or the last to complete, depending on the NAND page mapping and operation (read or write). All examples show a page size of 1 KB + spares; that is, S = 2 sectors of 512 bytes. The same principles can be extended to larger pages by adding more sectors.

The actual BCH codeword size is used during the error location work to restrict the search range: by definition, errors can happen only in the codeword that was actually written to the NAND, not in the mathematical codeword of $n = 2^{13} - 1 = 8191$ bits; that codeword (higher-order bits) is all-zero and implicit during computations.

The actual BCH codeword size depends on the mode, the programmed sizes, and the sector number (all sizes in nibbles):

- Spares mapped and protected per sector (below: see M1-M2-M3-M9-M10):
  - All sectors: (512) + P + E
- Spares pooled and protected by sector 0 (below: see M5-M6):
  - Sector 0 codeword: (512) + P + E
  - Other sectors: (512) + E
- Unprotected spares (below: see M4-M7-M8-M11-M12):
  - All codewords (512) + E

12.3.3.4.11.3.2.3.1 Per-Sector Spare Mappings

In these schemes, each 512-byte sector of the main area has its own dedicated section of the spare area. The spare area of each sector is composed of:

- ECC, which must be located after the data it protects
- Other data, which may or may not be protected by the ECC for its sector.

Copyright © 2026 Texas Instruments Incorporated

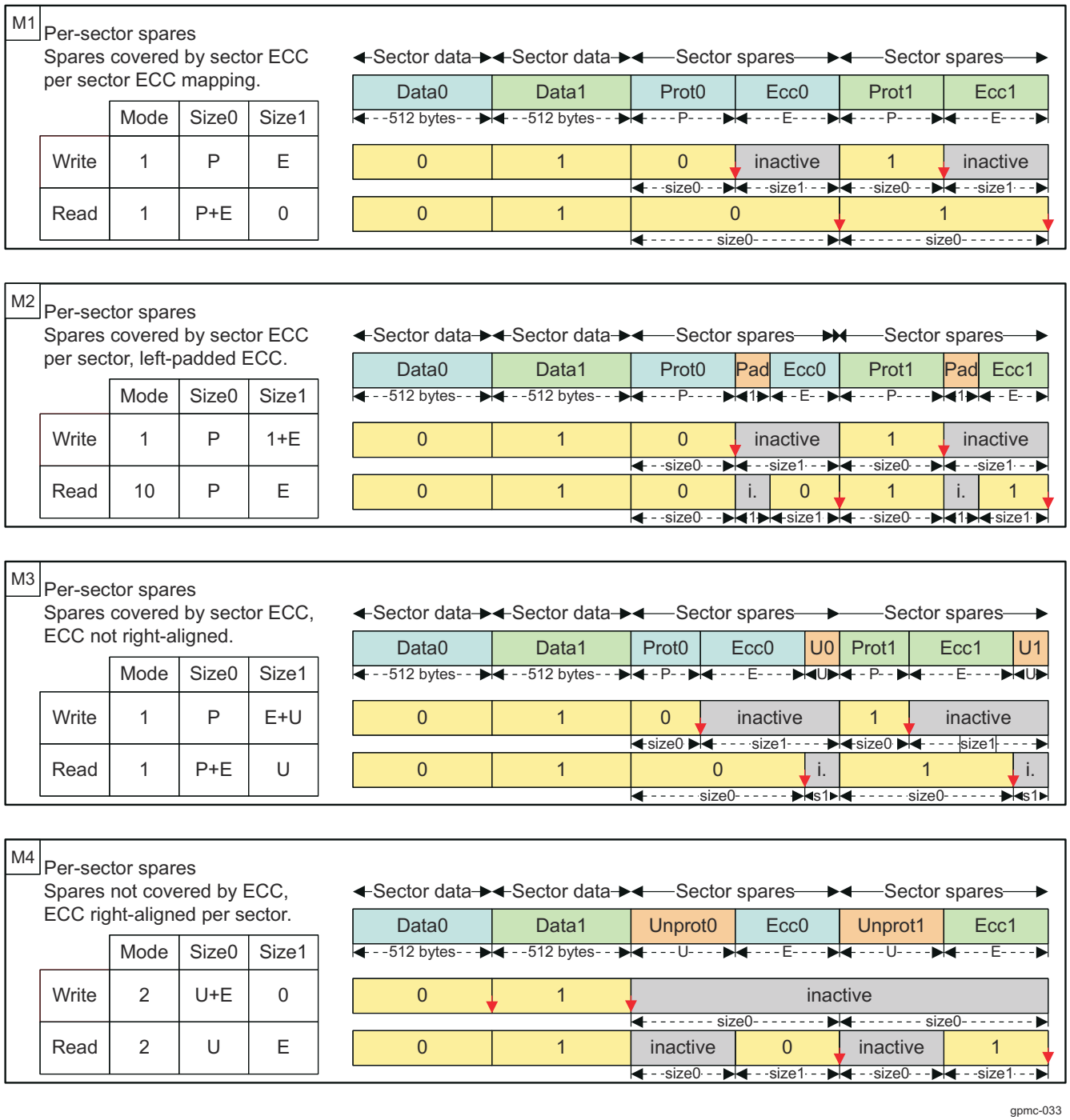


Figure 12-183. NAND Page Mapping and ECC: Per-Sector Schemes

12.3.3.4.11.3.2.3.2 Pooled Spare Mapping

In the following schemes, the spare area is pooled for the page.

- The ECC of each sector is aligned at the end of the spare area.
- The non-ECC spare data may or may not be covered by the ECC of sector 0.

Copyright © 2026 Texas Instruments Incorporated

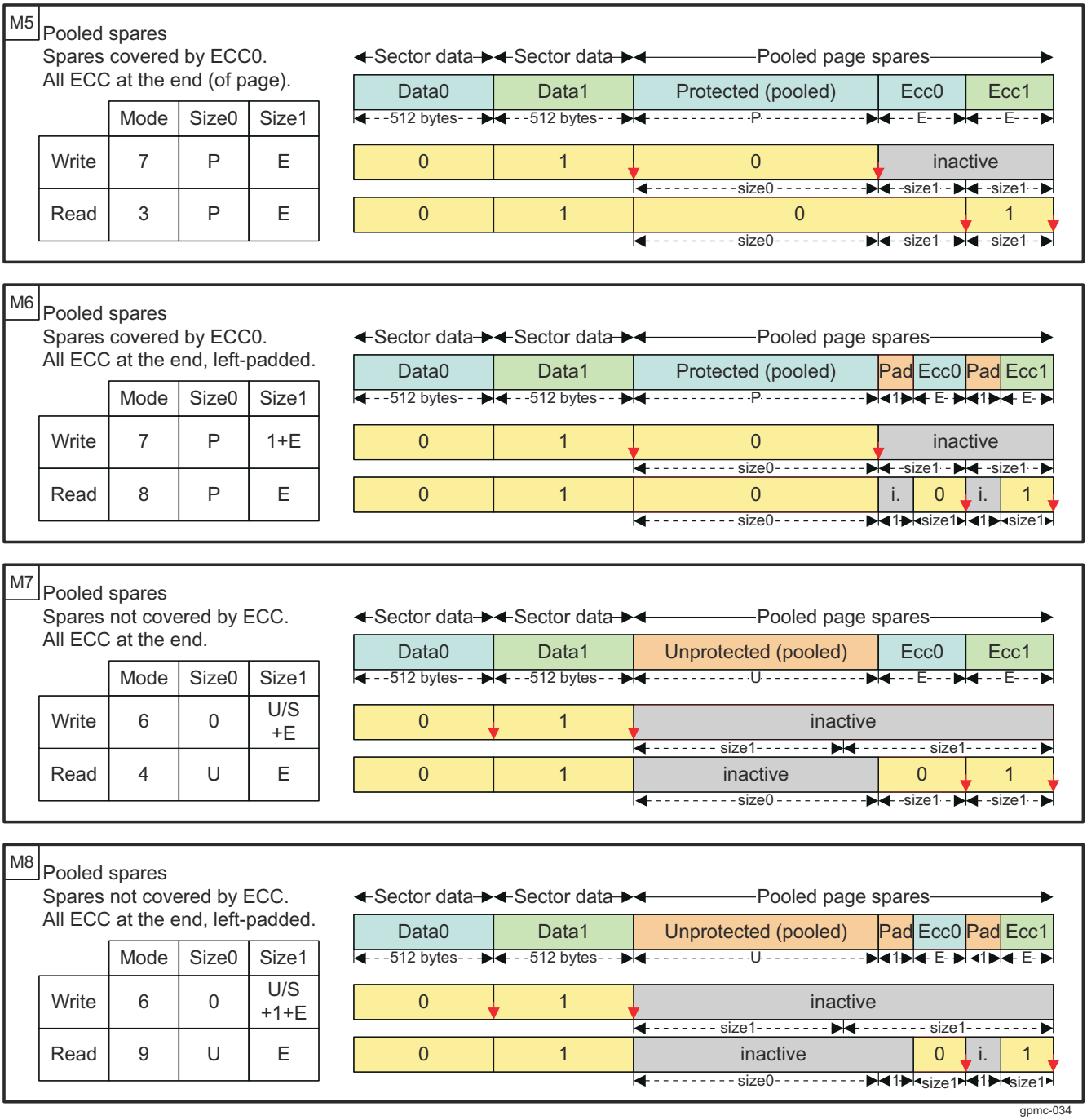


Figure 12-184. NAND Page Mapping and ECC: Pooled Spare Schemes

12.3.3.4.11.3.2.3.3 Per-Sector Spare Mapping, with ECC Separated at the End of the Page

In these schemes, each 512-byte sector of the main area is associated with two sections of the spare area.

- ECC section, all aligned at the end of the page
- Other data section, aligned before the ECCs, each of which may or may not be protected by the ECC for its sector.

Copyright © 2026 Texas Instruments Incorporated

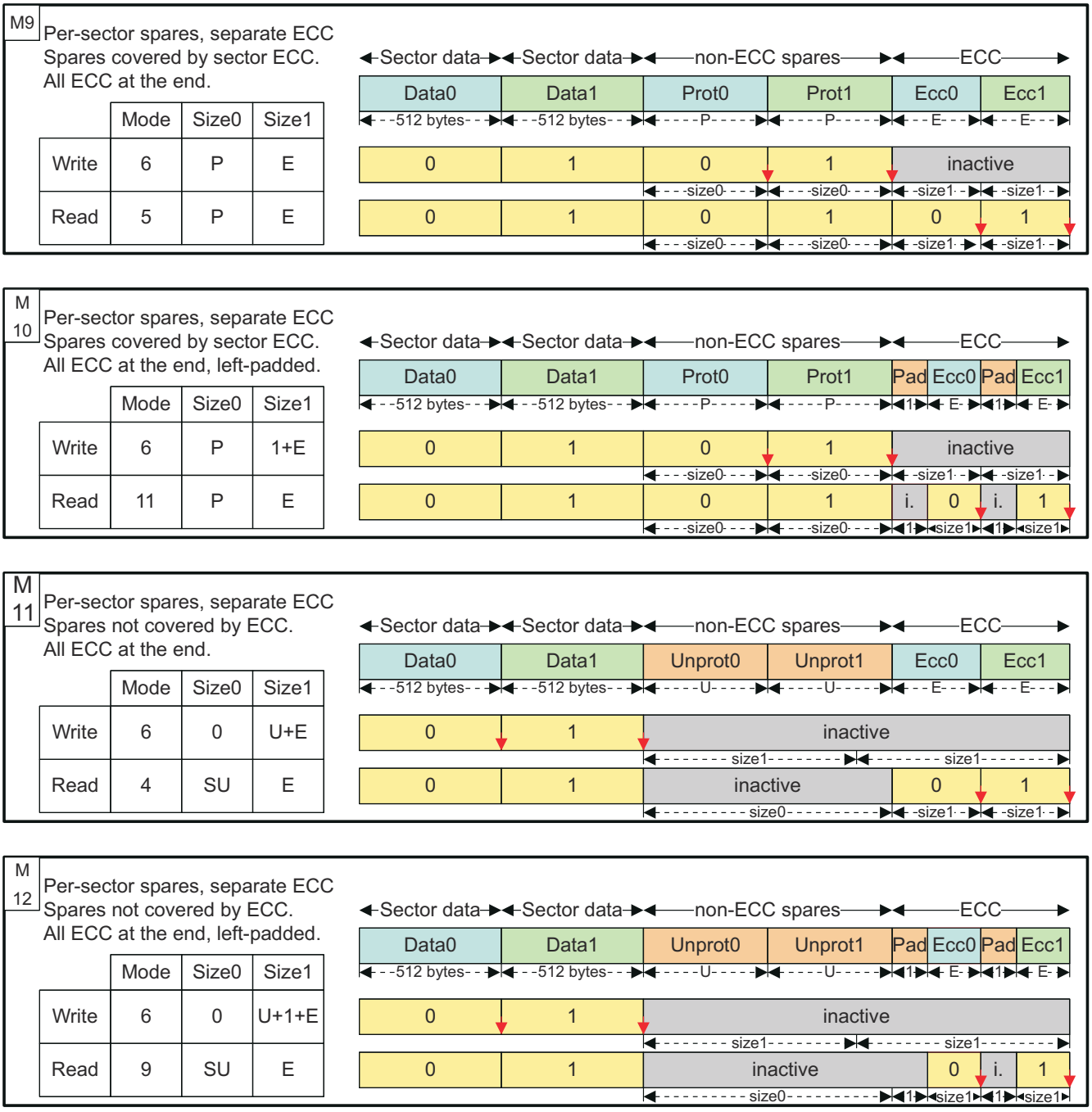


Figure 12-185. NAND Page Mapping and ECC: Per-Sector Schemes, With Separate ECC

12.3.3.4.11.4 Prefetch and Write-Posting Engine

Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

NAND device data access cycles are usually much slower than the CPU system frequency; such NAND read or write accesses issued by the processor affect the overall system performance, especially considering long

Copyright © 2026 Texas Instruments Incorporated

read or write sequences required for NAND page loading or programming. To minimize this effect on system performance, the GPMC includes a prefetch and write-posting engine, which can be used to read from or write to any chip-select location in a buffered manner.

The prefetch and write-posting engine is a simplified embedded-access requester that presents requests to the access engine on a user-defined chip-select target. The access engine interleaves these requests with any request coming from the interconnect interface; as a default, the prefetch and write-posting engine has the lowest priority.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access); thus, it is primarily dedicated to NAND support. The engine does not include an address generator; the request is limited to chip-select target identification. It includes a 64-byte FIFO associated with a DMA request synchronization line, for optimal DMA-based use.

The prefetch and write-posting engine uses an embedded 64-byte (32 16-bit word) FIFO to prefetch data from the NAND device in read mode (prefetch mode) or to store host data to be programmed into the NAND device in write mode (write-posting mode). The FIFO draining and filling (read and write) can be controlled by a device host processor through interrupt synchronization (an interrupt is triggered whenever a programmable threshold is reached) or by a device DMA module through DMA request synchronization, with a programmable request byte size in prefetch or posting mode.

The prefetch and write-posting engine includes a single memory pool. Therefore, only one mode, read or write, can be used at any given time. In other words, the prefetch and write-posting engine is a single-context engine that can be allocated to only one chip-select at a time for a read prefetch or a write-posting process.

The engine does not support atomic command and address phase programming and is limited to linear memory read or write access. As a consequence, it is limited to NAND data-stream access. The engine depends on the NAND software driver to control block and page opening with the correct data address pointer initialization, before the engine can read from or write to the NAND memory device.

Once started, engine data read and write sequencing is based solely on FIFO location availability and until the total programmed number of bytes is read or written.

Any host-concurrent accesses to a different chip-select are correctly interleaved with ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select do not suffer a large latency.

A round-robin arbitration scheme can be enabled to ensure minimum bandwidth to the prefetch and write-posting engine in the case of back-to-back direct memory requests to a different chip-select. If the *GPMC_PREFETCH_CONFIG1*[23] PFPWENROUNDROBIN bit is enabled, the arbitration grants the prefetch and write-posting engine access to the GPMC bus for a number of requests programmed in the *GPMC_PREFETCH_CONFIG1*[19-16] PFPWWEIGHTEDPRIO bit field.

The prefetch/write-posting engine read or write request is routed to the access engine with the chip-select destination ID. After the required arbitration phase, the access engine processes the request as a single access with the data access size equal to the device size specified in the corresponding chip-select configuration.

Note

The destination chip-select configuration must be set to the NAND protocol-compatible configuration for which address lines are not used (the address bus is not changed from its current value). Selecting a different chip-select configuration can produce undefined behavior.

12.3.3.4.11.4.1 General Facts About the Engine Configuration

The engine can be configured only if the GPMC_PREFETCH_CONTROL[0] STARTENGINE bit is deasserted.

The engine must be correctly configured in prefetch or write-posting mode and must be linked to a NAND chip-select before it can be started. The chip-select is linked using the GPMC_PREFETCH_CONFIG1[26-24] ENGINECSSELECTOR bit field.

Copyright © 2026 Texas Instruments Incorporated

In prefetch and write-posting modes, the engine uses byte or 16-bit word access requests, respectively, for an 8- or 16-bit-wide NAND device attached to the linked chip-select. The FIFOTHRESHOLD and TRANSFERCOUNT bit fields must be programmed accordingly as a number of bytes.

When the GPMC_PREFETCH_CONFIG1[7] ENABLEENGINE bit is set, the FIFO entry on the interconnect port side is accessible at any address in the associated chip-select memory region. When the ENABLEENGINE bit is set, any host access to this chip-select is rerouted to the FIFO input. Directly accessing the NAND device linked to this chip-select from the host is still possible through the following registers (where i = 0 to 3):

- GPMC_NAND_COMMAND_i
- GPMC_NAND_ADDRESS_i
- GPMC_NAND_DATA_i

The FIFO entry on the interconnect port can be accessed with byte, 16-bit word, or 32-bit word access size, according to little-endian format, even though the FIFO input is 32 bits wide.

The FIFO control is made easier through the use of interrupts or DMA requests associated with the FIFOTHRESHOLD bit field. The GPMC_PREFETCH_STATUS[30-24] FIFOPOINTER bit field monitors the number of available bytes to be read in prefetch mode or the number of free empty slots that can be written in write-posting mode. The GPMC_PREFETCH_STATUS[13-0] COUNTVALUE bit field monitors the number of remaining bytes to be read or written by the engine according to the value of the TRANSFERCOUNT bit field. The FIFOPOINTER and COUNTVALUE bit fields are always expressed as a number of bytes even if a 16-bit-wide NAND device is attached to the linked chip-select.

In prefetch mode, when the FIFOPOINTER equals 0 (that is, the FIFO is empty), a host read access receives the byte last read from the FIFO as its response. In case of 32-bit word or 16-bit word read accesses, the last byte read from the FIFO is copied the required number of times to fit the requested word size. In write-posting mode, when the FIFOPOINTER equals 0 (that is, the FIFO is full), a host write overwrites the last FIFO byte location. There is no underflow or overflow error reporting in the GPMC.

12.3.3.4.11.4.2 Prefetch Mode

The prefetch mode is selected when the GPMC_PREFETCH_CONFIG1[0] ACCESSMODE bit is cleared.

The NAND software driver must issue the block and page opening (READ) command with the correct data address pointer initialization before the engine can be started to read from the NAND memory device. The engine is started by asserting the GPMC_PREFETCH_CONTROL[0] STARTENGINE bit. The STARTENGINE bit automatically clears when the prefetch process completes.

If required, the ECC calculator engine must be initialized (that is, reset, configured, and enabled) before the prefetch engine is started so that the ECC is computed correctly on all data read by the prefetch engine.

When the GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE bit is cleared, the prefetch engine starts requesting data as soon as the STARTENGINE bit is set. If using this configuration, the host must monitor the NAND device-ready pin so that it sets the STARTENGINE bit only when the NAND device is in a ready state (that is, data is valid for prefetching).

When the SYNCHROMODE bit is set, the prefetch engine starts requesting data when an active-to-inactive WAIT signal transition is detected. The transition detector must be cleared before any transition detection (see Section 12.3.3.4.11.2.2, *Ready Pin Monitored by Hardware Interrupt*). The GPMC_PREFETCH_CONFIG1[5-4] WAITPINSELECTOR bit field selects which GPMC_WAIT pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit is set after the NAND address phase (page opening command), the engine is effectively started only after the actual NAND address phase completion. To prevent GPMC stall during this NAND address phase, set the STARTENGINE bit before NAND address phase completion when in synchronized mode. The prefetch engine starts when an active-to-inactive WAIT signal transition is detected. The STARTENGINE bit is automatically cleared on prefetch process completion.

Copyright © 2026 Texas Instruments Incorporated

The prefetch engine issues a read request to fill the FIFO with the amount of data specified by the GPMC_PREFETCH_CONFIG2[13-0] TRANSFERCOUNT bit field.

Table 12-188 describes the prefetch mode configuration.

Table 12-188. Prefetch Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	0	Prefetch engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1[26-24]	0 to 3	Selects the chip-select associated with a NAND device where the prefetch engine is active.
ACCESSMODE	GPMC_PREFETCH_CONFIG1[0]	0	Selects prefetch mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1[14-8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2[13-0]		Selects the number of bytes to be read or written by the engine to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1[3]	0/1	Selects when the engine starts the access to the chip-select
WAITPINSELECT	GPMC_PREFETCH_CONFIG1[17-16]	0 to 1	Selects WAIT pin edge detector (if GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE = 0x1)
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1[27]	0/1	See Section 12.3.3.4.11.4.6, *Optimizing NAND Access Using the Prefetch and Write-Posting Engine*.
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG1[30-28]		Number of clock cycle removed to timing parameters
ENABLEENGINE	GPMC_PREFETCH_CONFIG1[7]	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	1	Starts the prefetch engine

12.3.3.4.11.4.3 FIFO Control in Prefetch Mode

Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The FIFO can be drained directly by a device host processor or a DMA module channel.

In draining mode, the FIFO status can be monitored through the GPMC_PREFETCH_STATUS[30-24] FIFOPOINTER bit field or through the GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bit. The FIFOPOINTER indicates the current number of available data to be read; FIFOTHRESHOLDSTATUS set to 1 indicates that at least FIFOTHRESHOLD bytes are available from the FIFO.

An interrupt can be triggered by the GPMC if the GPMC_IRQENABLE[0] FIFOEVENTENABLE bit is set. The FIFO interrupt event is logged, and the GPMC_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, all the available bytes must be read, or at least enough bytes to get below the programmed FIFO threshold, and the FIFOEVENTSTATUS bit must be cleared to enable further interrupt events. The FIFOEVENTSTATUS bit must always be reset before asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt generation must be enabled after enabling the STARTENGINE bit.

Prefetch completion can be monitored through the GPMC_PREFETCH_STATUS[13-0] COUNTVALUE bit field. COUNTVALUE indicates the number of currently remaining data to be requested according to the TRANSFERCOUNT value. An interrupt can be triggered by the GPMC when the prefetch process is complete (that is, COUNTVALUE equals 0) if the GPMC_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. At prefetch completion, the TERMINALCOUNT interrupt event is also logged, and the GPMC_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the TERMINALCOUNTSTATUS bit must

Copyright © 2026 Texas Instruments Incorporated

be cleared. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

Note

The COUNTVALUE value is valid only when the prefetch engine is active (started), and an interrupt is only triggered when COUNTVALUE reaches 0, that is, when the prefetch engine automatically goes from an active to an inactive state.

The number of bytes to be prefetched (programmed in TRANSFERCOUNT) must be a multiple of the programmed FIFOTHRESHOLD to trigger the correct number of interrupts allowing a deterministic and transparent FIFO control. If this guideline is respected, the number of ISR accesses is always required and the FIFO is always empty after the last interrupt is triggered. In other cases, the TERMINALCOUNT interrupt must be used to read the remaining bytes in the FIFO (the number of remaining bytes being lower than the FIFOTHRESHOLD value).

In DMA draining mode, the GPMC_PREFETCH_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes are ready to be read from the FIFO. The DMA channel that owns this DMA request must be programmed so that the number of bytes programmed in FIFOTHRESHOLD is read from the FIFO during the DMA request process. The DMA request is kept active until this number of bytes has effectively been read from the FIFO, and no other DMA request can be issued until the ongoing active request is complete.

In prefetch mode, the TERMINALCOUNT event is also a source of DMA requests if the number of bytes to be prefetched is not a multiple of FIFOTHRESHOLD, the remaining bytes in the FIFO can be read by the DMA channel using the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled through the DMA channel programming model.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive-to-active prefetch (the STARTENGINE bit is set to 1). The associated DMA channel must always be enabled after setting the STARTENGINE bit so that the out-of-date active DMA request does not trigger spurious DMA transfers.

12.3.3.4.11.4.4 Write-Posting Mode

The write-posting mode is selected when the *GPMC_PREFETCH_CONFIG1*[0] ACCESSMODE bit is set.

The NAND software driver must issue the correct address pointer initialization command (page program) before the engine can start writing data into the NAND memory device. The engine starts when the *GPMC_PREFETCH_CONTROL*[0] STARTENGINE bit is set to 1. The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the NAND software driver must issue the second cycle program command and monitor the status for programming process completion (adding ECC handling, if required).

If used, the ECC calculator engine must be started (configured, reset, and enabled) before the posting engine is started so that the ECC parities are calculated properly on all data written by the prefetch engine to the associated chip-select.

In write-posting mode, the *GPMC_PREFETCH_CONFIG1*[3] SYNCHROMODE bit must be cleared so that posting starts as soon as the STARTENGINE bit is set and the FIFO is not empty.

If the STARTENGINE bit is set after the NAND address phase (page program command), the STARTENGINE setting is effective only after the actual NAND command completion. To prevent GPMC stall during this NAND command phase, set the STARTENGINE bit field before the NAND address completion and ensure that the associated DMA channel is enabled after the NAND address phase.

The posting engine issues a write request when valid data are available from the FIFO and until the programmed *GPMC_PREFETCH_CONFIG2*[13-0] TRANSFERCOUNT accesses are complete.

The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the NAND software driver must issue the second cycle program command and monitor

Copyright © 2026 Texas Instruments Incorporated

the status for programming process completion. The closing program command phase must be issued only when the full NAND page has been written into the NAND flash write buffer, including the spare area data and the ECC parities, if used.

Table 12-189 describes the write-posting configuration.

Table 12-189. Write-Posting Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	*GPMC_PREFETCH_CONTROL*[0]	0	Write-posting engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	*GPMC_PREFETCH_CONFIG1*[26-24]	0 to 3	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESSMODE	*GPMC_PREFETCH_CONFIG1*[0]	1	Selects write-posting mode
FIFOTHRESHOLD	*GPMC_PREFETCH_CONFIG1*[14-8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	*GPMC_PREFETCH_CONFIG2*[13-0]		Selects the number of bytes to be read or written by the engine from/to the selected chip-select
SYNCHROMODE	*GPMC_PREFETCH_CONFIG1*[3]	0	Engine starts the access to chip-select as soon as STARTENGINE is set.
ENABLEOPTIMIZEDACCESS	*GPMC_PREFETCH_CONFIG1*[27]	0/1	See Section 12.3.3.4.11.4.6, *Optimizing NAND Access Using the Prefetch and Write-Posting Engine*.
CYCLEOPTIMIZATION	*GPMC_PREFETCH_CONFIG1*[30-28]		
ENABLEENGINE	*GPMC_PREFETCH_CONFIG1*[7]	1	Engine enabled
STARTENGINE	*GPMC_PREFETCH_CONTROL*[0]	1	Starts the prefetch engine

12.3.3.4.11.4.5 FIFO Control in Write-Posting Mode

Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

The FIFO can be filled directly by a device host processor or a DMA module channel.

In filling mode, the FIFO status can be monitored through the FIFOPOINTER or through the GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bit. FIFOPOINTER indicates the current number of available free byte places in the FIFO, and the FIFOTHRESHOLDSTATUS bit, when set, indicates that at least FIFOTHRESHOLD free byte places are available in the FIFO.

An interrupt can be issued by the GPMC if the GPMC_IRQENABLE[0] FIFOEVENTENABLE bit is set. When the interrupt is fired, the GPMC_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, enough bytes must be written to fill the FIFO or to get below the programmed threshold, and the FIFOEVENTSTATUS bit must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit must always be cleared before asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt must be enabled after enabling the STARTENGINE bit.

The posting completion can be monitored through the GPMC_PREFETCH_STATUS[13-0] COUNTVALUE bit field. COUNTVALUE indicates the current number of remaining data to be written based on the value of the TRANSFERCOUNT bit field. An interrupt is issued by the GPMC when the write-posting process completes (that is, COUNTVALUE equal to 0) if the GPMC_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. When the interrupt is fired, the GPMC_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the TERMINALCOUNTSTATUS bit must be cleared. The TERMINALCOUNTSTATUS bit must always be cleared before asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

Copyright © 2026 Texas Instruments Incorporated

Note

The value of the COUNTVALUE bit field is valid only if the write-posting engine is active and started, and an interrupt is issued only when COUNTVALUE reaches 0; that is, when the posting engine automatically goes from active to inactive.

In DMA filling mode, the DMAMode bit field in the GPMC_PREFETCH_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes-free places are available in the FIFO. The DMA channel that owns this DMA request must be programmed so that a number of bytes equal to the value programmed in the FIFOTHRESHOLD bit field are written into the FIFO during the DMA access. The DMA request remains active until the associated number of bytes has effectively been written into the FIFO, and no other DMA request can be issued until the ongoing active request completes.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive-to-active prefetch (STARTENGINE set to 1). The associated DMA channel must always be enabled after setting the STARTENGINE bit so that an out-of-date active DMA request does not trigger spurious DMA transfers.

In write-posting mode, DMA or CPU fills the FIFO with no consideration for the associated byte enables. Any byte stored in the FIFO is written into the memory device.

12.3.3.4.11.4.6 Optimizing NAND Access Using the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized for back-to-back accesses if the associated nCS signal is not deasserted between accesses. The GPMC access engine can track prefetch engine accesses to optimize the access timing parameter programmed for the allocated chip-select, if no accesses to other chip-selects (that is, interleaved accesses) occur. Similarly, the access engine also eliminates CYCLE2CYCLEDELAY even if CYCLE2CYCLESAMECSEN is set. This capability is limited to the prefetch and write-posting engine accesses, and accesses to a NAND memory device (through the defined chip-select memory region or through the GPMC_NAND_DATA_i location, where i = 0 to 3) are never optimized.

The GPMC_PREFETCH_CONFIG1[27] ENABLEOPTIMIZEDACCESS bit must be set to enable optimized accesses. To optimize access time, the GPMC_PREFETCH_CONFIG1[30-28] CYCLEOPTIMIZATION bit field defines the number of GPMC_FCLK cycles to be suppressed from the following timing parameters:

- RDCYCLETIME
- WRCYCLETIME
- RDACCESSTIME
- WRACCESSTIME
- CSOFFTIME
- ADVOFFTIME
- OEOFFTIME
- WEOFFTIME

Figure 12-186 shows that in the case of back-to-back accesses to the NAND flash through the prefetch engine, CYCLE2CYCLESAMECSEN is forced to 0 when using optimized accesses. The first access uses the regular timing settings for this chip-select. All accesses after this one use settings reduced by x clock cycles, x being defined by the GPMC_PREFETCH_CONFIG1[30-28] CYCLEOPTIMIZATION bit field.

Copyright © 2026 Texas Instruments Incorporated

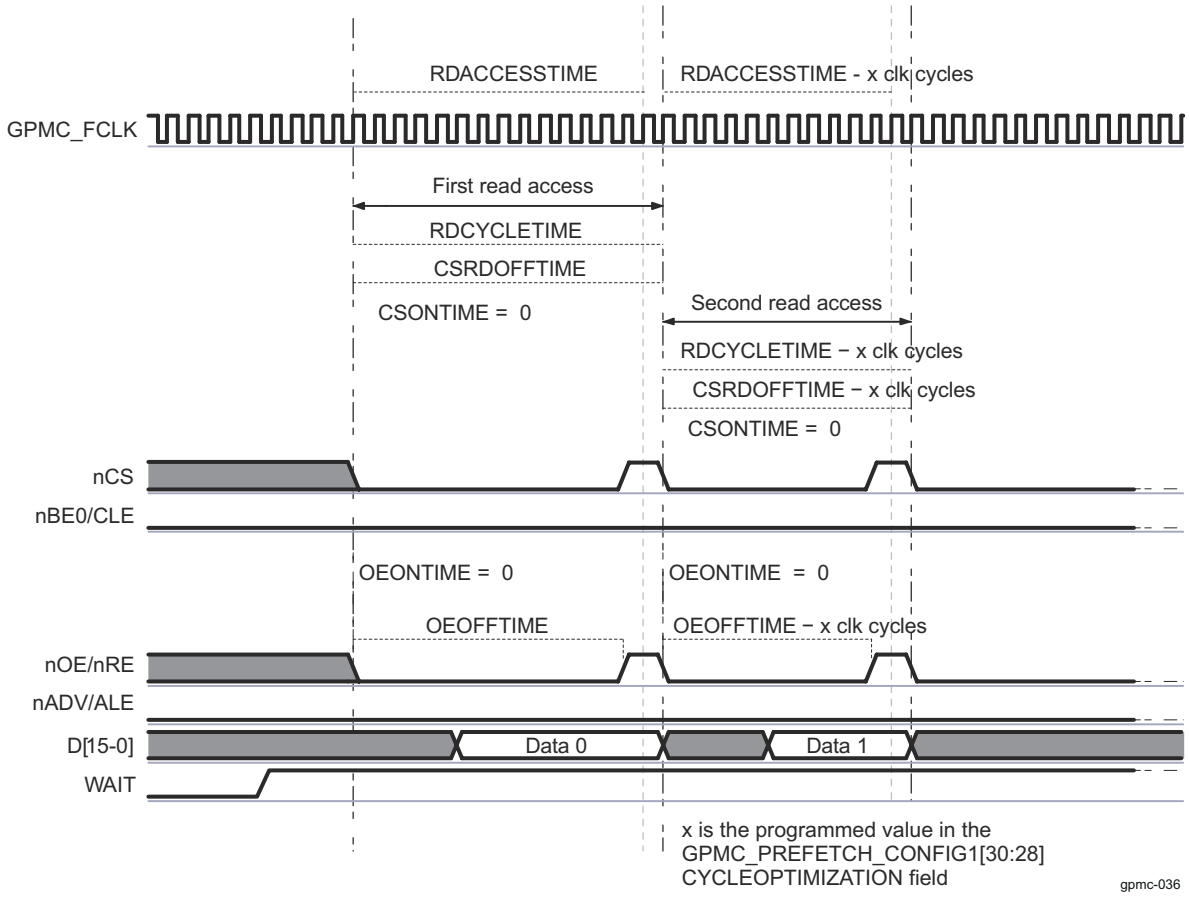


Figure 12-186. NAND Read Cycle Optimization Timing Description

12.3.3.4.11.4.7 Interleaved Accesses Between Prefetch and Write-Posting Engine and Other Chip-Selects

Any on-going read or write access from the prefetch and write-posting engine is completed before an access to any other chip-select can be initiated. As a default, the arbiter uses a fixed-priority algorithm, and the prefetch and write-posting engine has the lowest priority. The maximum latency added to access starting time in this case equals the RDCYCLETIME or WRCYCLETIME (optimized or not) plus the requested BUSTURNAROUND delay for bus turnaround completion programmed for the chip-select to which the NAND device is connected.

Alternatively, a round-robin arbitration can be used to prioritize accesses to the external bus. This arbitration scheme is enabled by setting the *GPMC_PREFETCH_CONFIG1*[23] PFPWENROUNDROBIN bit. When a request to another chip-select is received while the prefetch and write-posting engine is active, priority is given to the new request. The request processed thereafter is the prefetch and write-posting engine request, even if another interconnect request is passed in the mean time. The engine keeps control of the bus for an additional number of requests programmed in the *GPMC_PREFETCH_CONFIG1*[19-16] PFPWWEIGHTEDPRIO bit field. Control is then passed to the direct interconnect request.

As an example, the round-robin arbitration scheme is selected with PFPWWEIGHTEDPRIO set to 0x2. Considering that the prefetch and write-posting engine and the interconnect interface are always requesting access to the external interface, the GPMC grants priority to the direct interconnect access for one request. The GPMC then grants priority to the engine for three requests, and finally back to the direct interconnect access, until the arbiter is reset when one of the two initiators stops initiating requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.12 GPMC Use Cases and Tips

12.3.3.4.12.1 How to Set GPMC Timing Parameters for Typical Accesses

12.3.3.4.12.1.1 External Memory Attached to the GPMC Module

As discussed in the introduction to this chapter, the GPMC module supports the following external memory types:

- Asynchronous or synchronous, 8- or 16-bit-wide memory or device
- 16-bit address/data-multiplexed or non-multiplexed NOR flash device
- 8- or 16-bit NAND flash device

The following examples describe how to calculate GPMC timing parameters by showing a typical parameter setup for the access to be performed.

The example is based on a 512-Mb multiplexed NOR flash memory with the following characteristics:

- Type: NOR flash (address/data-multiplexed mode)
- Size: 512M bits
- Data Bus: 16 bits wide
- Speed: 104-MHz clock frequency
- Read access time: 80 ns

12.3.3.4.12.1.2 Typical GPMC Setup

Table 12-190 lists some of the I/Os of the GPMC module.

Table 12-190. Typical GPMC Setup Signals

Module Pin	I/O(1)	Description
GPMC0_FCLK	Internal	Functional clock. Acts as the time reference.
GPMC0_ICLK	Internal	Interface clock. Acts as the time reference.
GPMC0_CLKOUT	O	External clock provided to the external device for synchronous operations
GPMC0_A[21-16]	O	Address
GPMC0_AD[15-0]	I/O	Data-multiplexed with addresses A[16-1] on memory side
GPMC0_CSn[3-0]	O	Chip-selects
GPMC0_ADVn_ALE	O	Address valid enable
GPMC0_OEn_REn	O	Output enable (read access only)
GPMC0_WEn	O	Write enable (write access only)
GPMC0_WAIT[1-0]	I	Ready signal from memory device. Indicates when valid burst data is ready to be read

(1) I = Input; O = Output

Figure 12-187 shows the typical connection between the GPMC module and an attached NOR Flash memory.

Copyright © 2026 Texas Instruments Incorporated

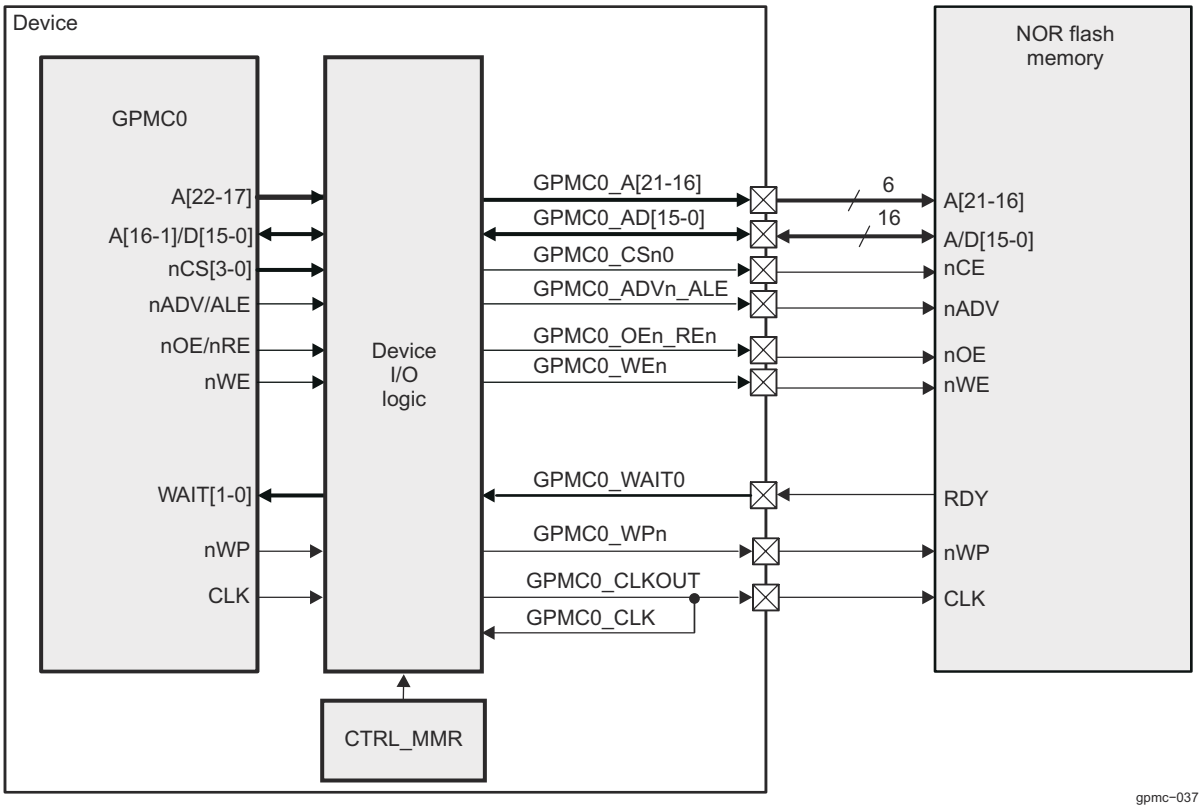


Figure 12-187. GPMC Connection to an External NOR Flash Memory

The following sections demonstrate how to calculate GPMC parameters for three access types:

- Synchronous burst read
- Asynchronous read
- Asynchronous single write

12.3.3.4.12.1.2.1 GPMC Configuration for Synchronous Burst Read Access

Note

The examples in Section 12.3.3.4.12.1.2.1 through Section 12.3.3.4.12.1.2.3 are based on a clock rate of 104 MHz. See the device-specific Datasheet for the maximum frequency appropriate for this device and the memory datasheet for the maximum frequency for the particular memory device.

The clock runs at 104 MHz (f = 104 MHz; T = 9.615 ns).

Table 12-191 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 12-192 shows how to calculate timings for the GPMC using the memory parameters.

Figure 12-188 shows the synchronous burst read access.

Table 12-191. Useful Timing Parameters on the Memory Side

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCES	nCS setup time to clock	0
tACS	Address setup time to clock	3
tIACC	Synchronous access time	80
tBACC	Burst access time valid clock to output delay	5.2

Copyright © 2026 Texas Instruments Incorporated

Table 12-191. Useful Timing Parameters on the Memory Side (continued)

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCEZ	Chip-select to High-Z	7
tOEZ	Output enable to High-Z	7
tAVC	nADV setup time	6
tAVD	nAVD pulse	6
tACH	Address hold time from clock	3

The following terms, which describe the timing interface between the controller and its attached device, are used to calculate the timing parameters on the GPMC side:

- Read access time (GPMC side): Time required to activate the clock + read access time requested on the memory side + data setup time required for optimal capture of a burst of data
- Data setup time (GPMC side): Ensures a good capture of a burst of data (as opposed to taking a burst of data out). One word of data is processed in one clock cycle (T = 9.615 ns). The read access time between two bursts of data is tBACC = 5.2 ns. Therefore, data setup time is a clock period – tBACC = 4.415 ns of data setup.
- Access completion (GPMC side): (Different from page burst access time) Time required between the last burst access and access completion: nCS/nOE hold time (nCS and nOE must be released at the end of an access. These signals are held to allow the access to complete).
- Read cycle time (GPMC side): Read access time + access completion
- Write cycle time for burst access: Not supported for NOR flash memory

Table 12-192. Calculating GPMC Timing Parameters

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
GPMC FCLK Divider	–	–	–	GPMCFCLKDIVIDER = 0x0
ClkActivation Time	min (tCES, tACS)	3	1	CLKACTIVATIONTIME = 0x1
RdAccessTime	roundmax (ClkActivationTime + tIACC + DataSetupTime)	94.03: (9.615 + 80 + 4.415)	10: roundmax (94.03 / 9.615)	RDACCESSTIME = 0xA
PageBurst RdAccessTime	roundmax (tBACC)	roundmax (5.2)	1	PAGEBURSTACCESSTIME = 0x1
RdCycleTime	RdAccess time + max (tCEZ, tOEZ)	101.03: (94.03 + 7)	11	RDCYCLETIME = 0xB
CsOnTime	tCES	0	0	CSONTIME = 0x0
CsReadOffTime	RdCycleTime	-	11	CSRDOFFTIME = 0xB
AdvOnTime	tAVC (1)	0	0	ADVONTIME = 0x0
AdvRdOffTime	tAVD + tAVC (2)	12	2	ADVRDOFFTIME = 0x2
OeOnTime (3)	(ClkActivationTime + tACH) < OeOnTime (ClkActivationTime + tIACC)	–	3, for instance	OEONTIME = 0x3
OeOffTime	RdCycleTime	–	11	OEOFFTIME = 0xB

(1) The external clock provided to the NOR flash is not yet available.
(2) AdvRdOffTime – AdvOnTime = tAVD; thus, AdvRdOffTime = tAVD + AdvOnTime = tAVD + tAVC.
(3) OeOnTime must ensure that addresses are available. It must not exceed the availability of the first burst of data.

Copyright © 2026 Texas Instruments Incorporated

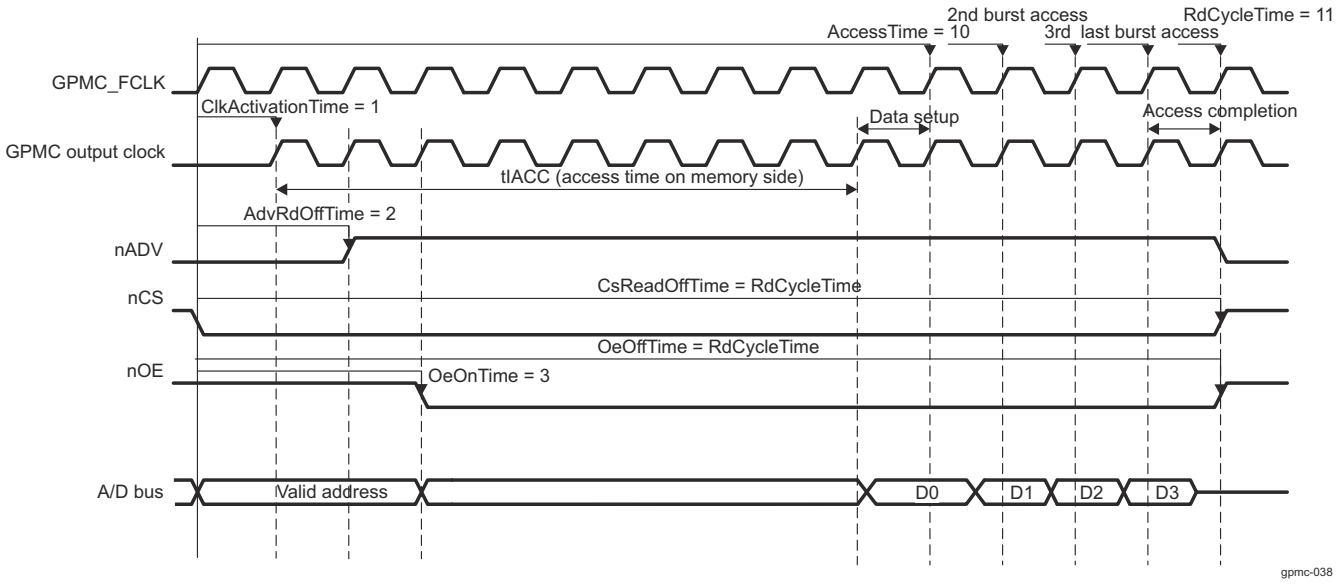


Figure 12-188. Synchronous Burst Read Access (Timing Parameters in Clock Cycles)

12.3.3.4.12.1.2.2 GPMC Configuration for Asynchronous Read Access

The clock runs at 104 MHz (f = 104 MHz; T = 9.615 ns).

Table 12-193 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 12-194 shows how to calculate timings for the GPMC using the memory parameters.

Figure 12-189 shows the asynchronous read access.

Table 12-193. AC Characteristics for Asynchronous Read Access

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCE	Read Access time from nCS low	80
tAAVDS	Address setup time to rising edge of nADV	3
tAVDP	nADV low time	6
tCAS	nCS setup time to nADV	0
tOE	Output enable to output valid	6
tOEZ	Output enable to High-Z	7

Use the following formula to calculate the RdCycleTime parameter for this typical access:

RdCycleTime = RdAccessTime + AccessCompletion = RdAccessTime + 1 clock cycle + tOEZ:

1. On the memory side, the external memory makes the data available to the output bus. This is the memory-side read access time defined in Table 12-193: the number of clock cycles between the address capture (nADV rising edge) and the data valid on the output bus.

   The GPMC requires some hold time to allow the data to be captured correctly and the access to be finished.
2. To read the data correctly, the GPMC must be configured to meet the data setup time requirement of the memory; the GPMC module captures the data on the next rising edge. This is access time on the GPMC side.
3. There must also be a data hold time for correctly reading the data (checking that there is no nOE/nCS deassertion while reading the data). This data hold time is one clock cycle (that is, RdAccessTime + 1).
4. To complete the access, nOE/nCS signals are driven to High-Z. RdAccessTime + 1 + tOEZ is the read cycle time.

Copyright © 2026 Texas Instruments Incorporated

5. Addresses can now be relatched and a new read cycle begun.

Table 12-194. GPMC Timing Parameters for Asynchronous Read Access

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
ClkActivationTime		n/a (asynchronous mode)		
RdAccessTime	round max (tCE)	80	10	RDACCESSTIME = 0xA
PageBurstAccessTime	N/A (single access)			
RdCycleTime	RdAccessTime + 1cycle + tOEZ	96.615	12	RDCYCLETIME = 0xC
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsReadOffTime	RdAccessTime + 1 cycle	89.615	11	CSRDOFFTIME = 0xB
AdvOnTime	tAAVDS	3	1	ADVONTIME = 0x1
AdvRdOffTime	tAAVDS + tAVDP	9	1	ADVRDOFFTIME = 0x1
OeOnTime	OeOnTime >= AdvRdOffTime (multiplexed mode)	-	3, for instance	OEONTIME = 0x3
OeOffTime	RdAccessTime + 1cycle	89.615	11	OEOFFTIME = 0xB

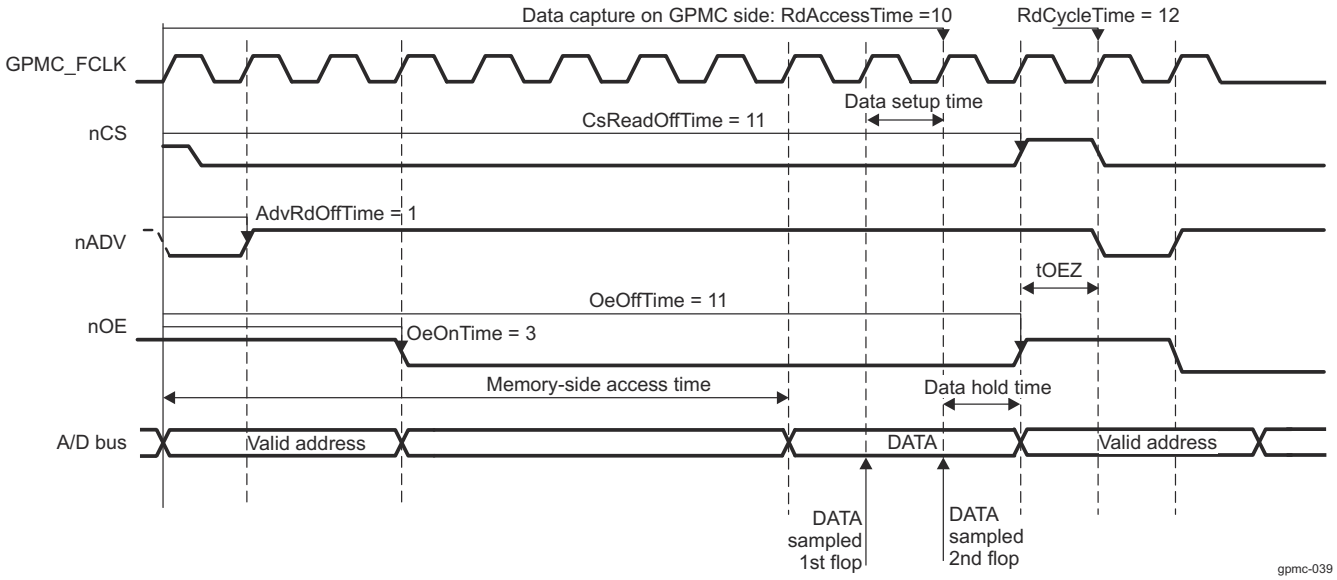


Figure 12-189. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)

12.3.3.4.12.1.2.3 GPMC Configuration for Asynchronous Single Write Access

The clock runs at 104 MHz: (f = 104 MHz; T = 9.615 ns).

Table 12-196 shows how to calculate timings for the GPMC using the memory parameters.

Table 12-195 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Figure 12-190 shows the synchronous burst write access.

Table 12-195. AC Characteristics for Asynchronous Single Write (Memory Side)

AC Characteristics on the Memory Side	Description	Duration (ns)
tWC	Write cycle time	60
tAVDP	nADV low time	6
tWP	Write pulse width	25

Copyright © 2026 Texas Instruments Incorporated

Table 12-195. AC Characteristics for Asynchronous Single Write (Memory Side) (continued)

AC Characteristics on the Memory Side	Description	Duration (ns)
tWPH	Write pulse width high	20
tCS	nCS setup time to nWE	3
tCAS	nCS setup time to nADV	0
tAVSC	nADV setup time	3

For asynchronous single write access, write cycle time is WrCycleTime = WeOffTime + AccessCompletion = WeOffTime + 1. For the AccesCompletion, the GPMC requires one cycle of data hold time (nCS deassertion). For more information, see the device-specific Datasheet.

Table 12-196. GPMC Timing Parameters for Asynchronous Single Write

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Registers Configuration
ClkActivationTime	N/A (asynchronous mode)			
WdAccessTime	Applicable only to WAITMONITORING (the value is the same as for read access)			
PageBurstAccessTime	N/A (single access)			
WrCycleTime	WeOffTime + AccessCompletion	57.615	6	WRCYCLETIME = 0x6
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsWrOffTime	WeOffTime + 1	57.615	6	CSWROFFTIME = 0x6
AdvOnTime	tAVSC	3	1	ADVONTIME = 0x1
AdvWrOffTime	tAVSC + tAVDP	9	1	ADVWROFFTIME = 0x1
WeOnTime	tCS	3	1	WEONTIME = 0x1
WeOffTime	tCS + tWP + tWPH	48	5	WEOFFTIME = 0x5

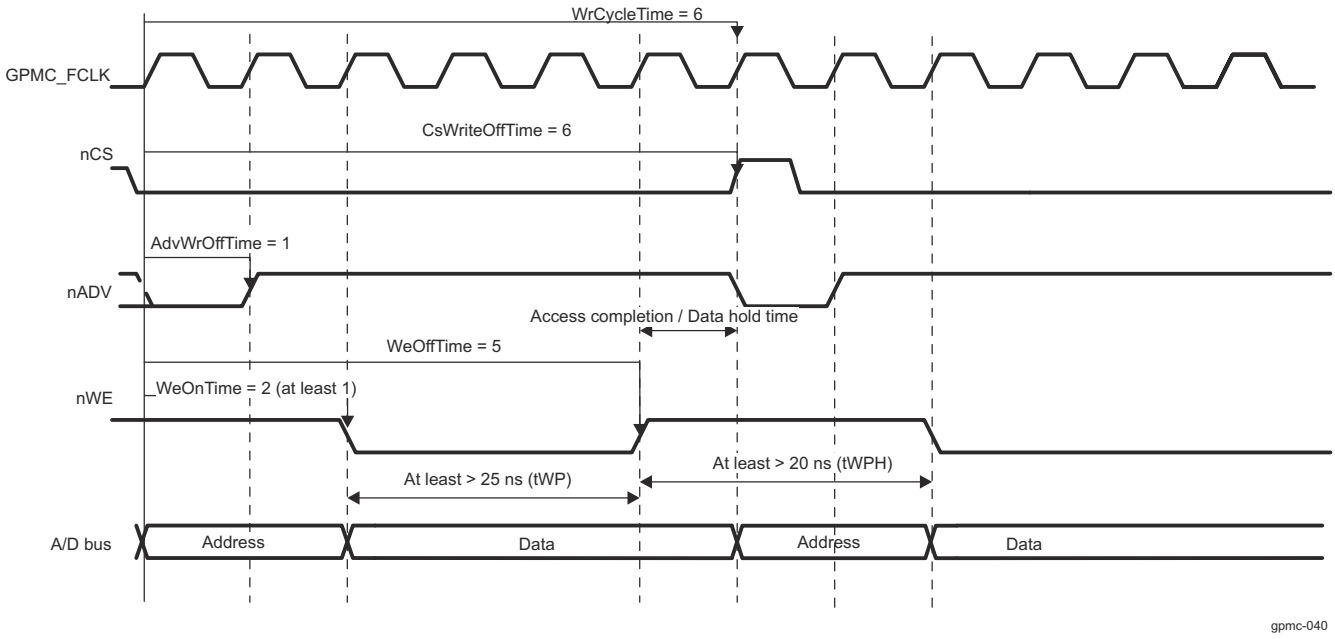


Figure 12-190. Asynchronous Single Write Access (Timing Parameters in Clock Cycles)

12.3.3.4.12.2 How to Choose a Suitable Memory to Use With the GPMC

This section is intended to help the user select a suitable memory device to interface with the GPMC controller.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.12.2.1 Supported Memories or Devices

NAND flash and NOR flash architectures are the two flash technologies. The GPMC supports various types of external memory or devices, basically any one that supports NAND or NOR protocols:

- 8- and 16-bit-wide asynchronous or synchronous memory or device (only 8-bit: nonburst device)
- 16-bit address and data-multiplexed NOR flash devices (pSRAM, and so on)
- 8- and 16-bit NAND flash devices

12.3.3.4.12.2.1.1 Memory Pin Multiplexing

This section describes the interfacing differences of the GPMC supported memories.

Table 12-197. Supported Memory Interfaces

Function	16-Bit Address/ Data-Multiplexed pSRAM or NOR Flash (1)	16-Bit NAND	8-Bit NAND
GPMC_A[22]			
GPMC_A[21]			
GPMC_A[20]			
GPMC_A[19]			
GPMC_A[18]			
GPMC_A[17]			
GPMC_A[16]			
GPMC_A[15]			
GPMC_A[14]			
GPMC_A[13]			
GPMC_A[12]			
GPMC_A[11]			
GPMC_A[10]	A26		
GPMC_A[9]	A25		
GPMC_A[8]	A24		
GPMC_A[7]	A23		
GPMC_A[6]	A22		
GPMC_A[5]	A21		
GPMC_A[4]	A20		
GPMC_A[3]	A19		
GPMC_A[2]	A18		
GPMC_A[1]	A17		
GPMC_A[0]	A16		
GPMC_AD[15]	D15 or A16	IO15	
GPMC_AD[14]	D14 or A15	IO14	
GPMC_AD[13]	D13 or A14	IO13	
GPMC_AD[12]	D12 or A13	IO12	
GPMC_AD[11]	D11 or A12	IO11	
GPMC_AD[10]	D10 or A11	IO10	
GPMC_AD[9]	D9 or A10	IO9	
GPMC_AD[8]	D8 or A9	IO8	
GPMC_AD[7]	D7 or A8	IO7	
GPMC_AD[6]	D6 or A7	IO6	
GPMC_AD[5]	D5 or A6	IO5	
GPMC_AD[4]	D4 or A5	IO4	

Copyright © 2026 Texas Instruments Incorporated

Table 12-197. Supported Memory Interfaces (continued)

Function	16-Bit Address/ Data-Multiplexed pSRAM or NOR Flash (1)	16-Bit NAND	8-Bit NAND
GPMC_AD[3]	D3 or A4	IO3	
GPMC_AD[2]	D2 or A3	IO2	
GPMC_AD[1]	D1 or A2	IO1	
GPMC_AD[0]	D0 or A1	IO0	
GPMC_CLKOUT	CLK		
GPMC_CSn0	nCS0 (chip-select)	nCE0 (chip-enable)	
GPMC_CSn1	nCS1	nCE1	
GPMC_CSn2	nCS2	nCE2	
GPMC_CSn3	nCS3	nCE3	
GPMC_ADVn_ALE	nADV (address valid)	ALE (address latch enable)	
GPMC_OEn_REn	nOE (output enable)	nRE (read enable)	
GPMC_WEn	nWE (Write enable)	nWE (write enable)	
GPMC_BE0n_CLE	nBE0 (byte enable)	CLE (command latch enable)	
GPMC_BE1n	nBE1		
GPMC_WAIT0	WAIT0	R/nB0 (ready/busy)	
GPMC_WAIT1	WAIT1	R/nB1	
GPMC_WPn	nWP (Write Protect)	nWP (Write Protect)	

(1) Addresses seen from the device side. When interfacing to the external device, A1 is connected to the memory A0, A2 to the memory A1, and so on.

12.3.3.4.12.2.1.2 NAND Interface Protocol

NAND flash architecture, introduced in 1989, is a flash technology. NAND is a page-oriented memory device; that is, read and write accesses are done by pages. NAND achieves great density by sharing common areas of the storage transistor, which creates strings of serially connected transistors (in NOR devices, each transistor stands alone). Because of its high density NAND is best suited to devices that require high capacity data storage, such as pictures, music, and data files. NAND nonvolatility makes of it a good storage solution for many applications where mobility, low power, and speed are key factors. Low pin count and simple interface are other advantages of NAND.

Table 12-198 summarizes the NAND interface signals level applied to external device or memories.

Table 12-198. NAND Interface Bus Operations Summary

Bus Operation	CLE	ALE	nCE	nWE (1)	nRE (1)	nWP
Read (cmd input)	H	L	L	RE	H	x
Read (add input)	L	H	L	RE	H	x
Write (cmd input)	H	L	L	RE	H	H
Write (add input)	L	H	L	RE	H	H
Data input	L	L	L	RE	H	H
Data output	L	L	L	H	FE	x
Busy (during read)	x	x	H (2)	H (2)	H (2)	x
Busy (during program)	x	x	x	x	x	H
Busy (during erase)	x	x	x	x	x	H
Write protect	x	x	x	x	x	L
Standby	x	x	H	x	x	H/L

(1) RE stands for rising edge; FE stands for falling edge
(2) Can be nCE high, or WE and nRE high.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.4.12.2.1.3 NOR Interface Protocol

NOR flash architecture, introduced in 1988, is a flash technology. Unlike NAND, which is a sequential access device, NOR is directly addressable; that is, it is designed to be a random access device. NOR is best suited to devices used to store and run code or firmware, usually in small capacities. While NOR has fast read capabilities, it also has slow write and erase functions when compared to the NAND architecture.

Table 12-199 summarizes the level of the NOR interface signals applied to external devices or memories.

Table 12-199. NOR Interface Bus Operations Summary

Bus Operation	CLK	nADV	nCS	nOE	nWE	WAIT	DQ[15-0]
Read (asynchronous)	x	L	L	L	H	Asserted	Output
Read (synchronous)	Running	L	L	L	H	Driven	Output
Read (burst suspend)	Halted	x	L	H	H	Active	Output
Write	x	L	L	H	L	Asserted	Input
Output disable	x	x	L	H	H	Asserted	High-Z
Standby	x	x	H	x	x	High-Z	High-Z

12.3.3.4.12.2.1.4 Other Technologies

Other supported device types interact with the GPMC through the NOR interface protocol.

FPGA (Field-Programmable Gate Array) is a low-power integrated circuit based on an array of programmable logic blocks.

pSRAM (pseudo-static random access memory) is a low-power memory device. pSRAM is based on the DRAM cell with internal refresh and address control features, and interfaces as a synchronous NOR flash. It has synchronous write capability.

Copyright © 2026 Texas Instruments Incorporated

12.3.3.5 GPMC Basic Programming Model

12.3.3.5.1 GPMC High-Level Programming Model Overview

The goal of the basic high-level programming model is to introduce a top-down approach to users that need to configure the GPMC module.

Figure 12-191 and Table 12-200 through Table 12-201 show a programming model top-level diagram for the GPMC, and a description of each step. Each block of the diagram is described in one of the following sections through a set of registers to configure.

Copyright © 2026 Texas Instruments Incorporated

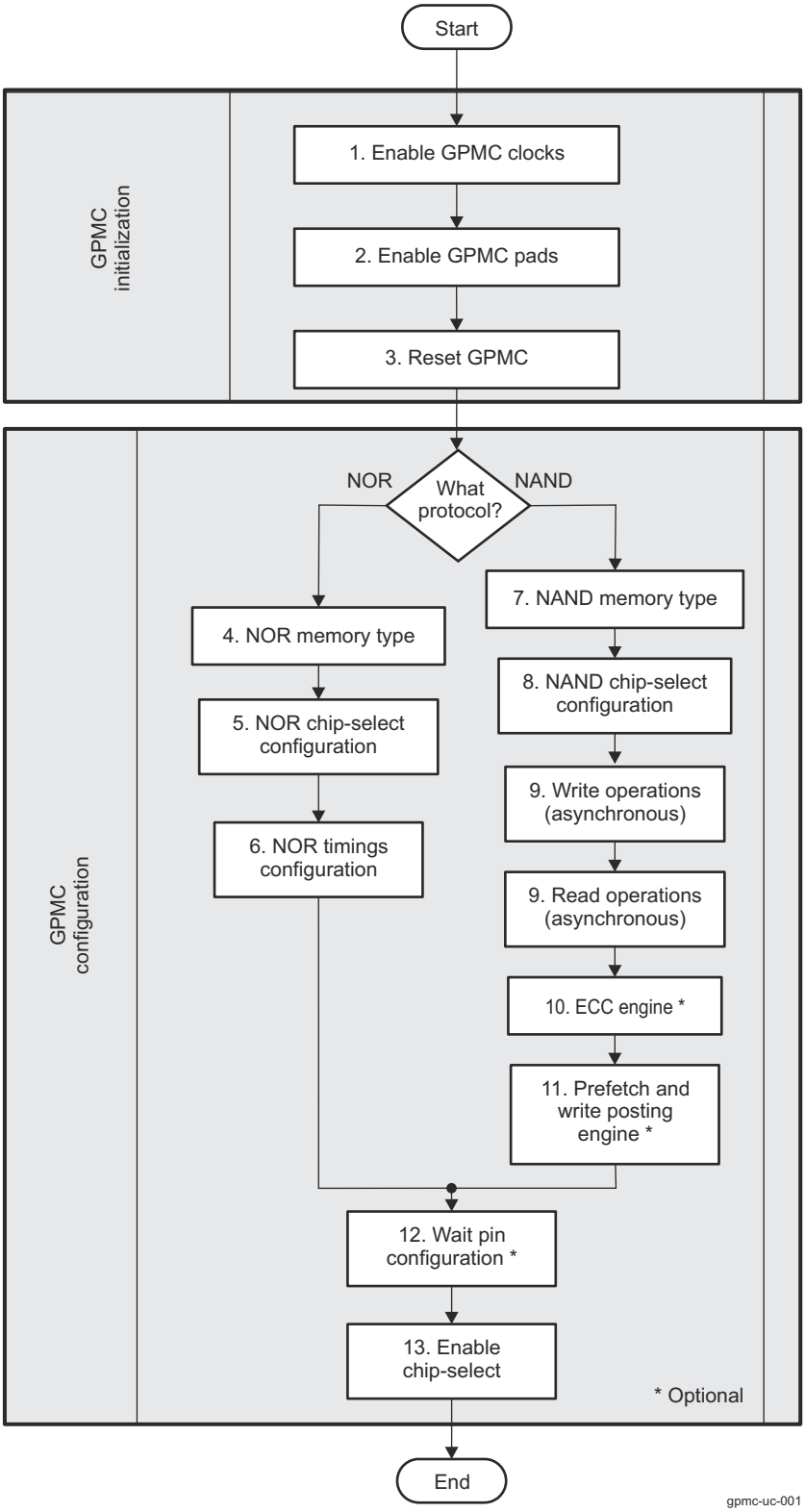


Figure 12-191. Programming Model Top-Level Diagram

Table 12-200. GPMC Configuration in NOR Mode

Step	Description
NOR Memory Type	See Table 12-202.

Copyright © 2026 Texas Instruments Incorporated

Table 12-200. GPMC Configuration in NOR Mode (continued)

Step	Description
NOR Chip-Select Configuration	See Table 12-203.
NOR Timings Configuration	See Table 12-204.
WAIT Pin Configuration	See Table 12-212.
Enable Chip-Select	See Table 12-213.

Table 12-201. GPMC Configuration in NAND Mode

Step	Description
NAND Memory Type	See Table 12-207.
NAND Chip-Select Configuration	See Table 12-208.
Write Operations (Asynchronous)	See Table 12-209.
Read Operations (Asynchronous)	See Table 12-209.
ECC Engine	See Table 12-210.
Prefetch and Write-Posting Engine	See Table 12-211.
WAIT Pin Configuration	See Table 12-212.
Enable Chip-Select	See Table 12-213.

12.3.3.5.2 GPMC Initialization

GPMC can be reset via software bit in LPSC. For more information, see *Reset*.

12.3.3.5.3 GPMC Configuration in NOR Mode

This section gives a generic configuration for parameters related to the NOR memory connected to the GPMC.

Table 12-202. NOR Memory Type

Subprocess Name	Register / Bit Field	Value
Set the NOR protocol.	GPMC_CONFIG1_i[11-10] DEVICETYPE	0x0
Set a device size.	GPMC_CONFIG1_i[13-12] DEVICESIZE	x
Select an address and data multiplexing protocol.	GPMC_CONFIG1_i[9] MUXADDDATA	x
Set the attached device page length.	GPMC_CONFIG1_i[24-23] ATTACHEDDEVICEPAGELENGTH	x
Set the wrapping burst capabilities.	GPMC_CONFIG1_i[31] WRAPBURST	x
Select a timing signals latencies factor.	GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY	x
Select an output clock frequency (1).	GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER	x
Choose an output clock activation time (1).	GPMC_CONFIG1_i[26-25] CLKACTIVATIONTIME	x
Set a single or multiple access for read operations (1).	GPMC_CONFIG1_i[30] READMULTIPLE	x
Set a synchronous or asynchronous mode for read operations.	GPMC_CONFIG1_i[29] READTYPE	x
Set a single or multiple access for write operations.	GPMC_CONFIG1_i[28] WRITEMULTIPLE	x
Set a synchronous or asynchronous mode for write operations.	GPMC_CONFIG1_i[27] WRITETYPE	x

(1) Applies only to synchronous configurations (or non-multiplexed asynchronous for multiple access one)

Table 12-203. NOR Chip-Select Configuration

Subprocess Name	Register/Bit Field	Value
Select the chip-select base address.	GPMC_CONFIG7_i[5-0] BASEADDRESS	x
Select the chip-select mask address.	GPMC_CONFIG7_i[11-8] MASKADDRESS	x

Copyright © 2026 Texas Instruments Incorporated

Table 12-204. NOR Timings Configuration

Subprocess Name	Register/Bit Field	Value
Configure adequate timing parameters in various memory modes.	See *GPMC Timing Parameters*	

Table 12-205. WAIT Pin Configuration

Subprocess Name	Register/Bit Field	Value
Enable or disable WAIT pin monitoring for read operations.	GPMC_CONFIG1_i[22] WAITREADMONITORING	x
Enable or disable WAIT pin monitoring for write operations.	GPMC_CONFIG1_i[21] WAITWRITEMONITORING	x
Select a WAIT pin monitoring time.	GPMC_CONFIG1_i[19-18] WAITMONITORINGTIME	x
Choose the input WAIT pin for the chip-select.	GPMC_CONFIG1_i[17-16] WAITPINSELECT	x

Table 12-206. Enable Chip-Select

Subprocess Name	Register/Bit Field	Value
When all parameters are configured, enable the chip-select.	GPMC_CONFIG7_i[6] CSVALID	x

12.3.3.5.4 GPMC Configuration in NAND Mode

This section gives a generic configuration for parameters related to the NAND memory connected to the GPMC.

Note

Some of the GPMC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

Table 12-207. NAND Memory Type

Subprocess Name	Register/Bit Field	Value
Set the NAND protocol.	*GPMC_CONFIG1_i*[11-10] DEVICETYPE	0x2
Set a device size.	*GPMC_CONFIG1_i*[13-12] DEVICESIZE	x
Set the address and data multiplexing protocol to non-multiplexed attached device.	*GPMC_CONFIG1_i*[9] MUXADDDATA	0x0
Select a timing signals latencies factor.	*GPMC_CONFIG1_i*[4] TIMEPARAGRANULARITY	x
Set a synchronous or asynchronous mode and a single or multiple access for read and write operations.	See Section 12.3.3.5.5, *Set Memory Access*.	x

Table 12-208. NAND Chip-Select Configuration

Subprocess Name	Register/Bit Field	Value
Select the chip-select base address.	*GPMC_CONFIG7_i*[5-0] BASEADDRESS	x
Select the chip-select minimum granularity (16MB).	*GPMC_CONFIG7_i*[11-8] MASKADDRESS	x

Table 12-209. Asynchronous Read and Write Operations

Subprocess Name	Register/Bit Field	Value
Configure adequate timing parameters in asynchronous modes	See Section 12.3.3.5.6, *GPMC Timing Parameters*.	

Table 12-210. ECC Engine

Subprocess Name	Register/Bit Field	Value
Select the ECC result register where the first ECC computation is stored (applies only to Hamming).	*GPMC_ECC_CONTROL*[3-0] ECCPOINTER	x (2)
Clear all ECC result registers.	*GPMC_ECC_CONTROL*[8] ECCCLEAR	Write 1 to clear.
Define ECCSIZE0 and ECCSIZE1.	*GPMC_ECC_SIZE_CONFIG*[21-12] ECCSIZE0 and [31-22] ECCSIZE1	x (1)
Select the size of each of the 9 result registers (size specified by ECCSIZE0 or ECCSIZE1).	*GPMC_ECC_SIZE_CONFIG*[j-1] ECCjRESULTSIZE where j = 1 to 9	x

Copyright © 2026 Texas Instruments Incorporated

Table 12-210. ECC Engine (continued)

Subprocess Name	Register/Bit Field	Value
Select the chip-select where ECC is computed.	*GPMC_ECC_CONTROL*[3-1] ECCCS	x
Select the Hamming code or BCH code ECC algorithm in use.	*GPMC_ECC_CONTROL*[16] ECCALGORITHM	x
Select word size for ECC calculation.	*GPMC_ECC_CONTROL*[7] ECC16B	x
If the BCH code is used, Set an error correction capability and Select a number of sectors to process.	*GPMC_ECC_CONTROL*[13-12] ECCBCHTSEL and *GPMC_ECC_CONTROL*[6-4] ECCTOPSECTOR	x
Enable the ECC computation.	*GPMC_ECC_CONTROL*[0] ECCENABLE	0x1

(1) Depends on the size of each sector in the NAND page
(2) This parameter depends on the numbers of sectors in a page.

Table 12-211. Prefetch and Write-Posting Engine

Subprocess Name	Register/Bit Field	Value
Disable the engine before configuration.	*GPMC_PREFETCH_CONTROL*[0] STARTENGINE	0x0
Select the chip-select associated with a NAND device where the prefetch engine is active.	*GPMC_PREFETCH_CONFIG1*[26-24] ENGINECSSELECTOR	x
Select access direction through prefetch engine, read or write.	*GPMC_PREFETCH_CONFIG1*[0] ACCESSMODE	x
Select the threshold used to issue an interrupt request.	*GPMC_PREFETCH_CONFIG1*[14-8] FIFOTHRESHOLD	x
Select interrupt synchronization mode.	*GPMC_PREFETCH_CONFIG1*[2] DMAMODE	x
Select if the engine immediately starts accessing the memory upon STARTENGINE assertion or if hardware synchronization based on a WAIT signal is used.	*GPMC_PREFETCH_CONFIG1*[3] SYNCHROMODE	x
Select which WAIT pin edge detector should start the engine in synchronized mode.	*GPMC_PREFETCH_CONFIG1*[5-4] WAITPINSELECTOR	x
Enter a number of clock cycles removed to timing parameters (for all back-to-back accesses to the NAND flash except the first one).	*GPMC_PREFETCH_CONFIG1*[30-28] CYCLEOPTIMIZATION	x
Enable the prefetch postwrite engine.	*GPMC_PREFETCH_CONFIG1*[7] ENABLEENGINE	0x1
Select the number of bytes to be read or written by the engine to the selected chip-select.	*GPMC_PREFETCH_CONFIG2*[13-0] TRANSFERCOUNT	x
Start the prefetch engine.	*GPMC_PREFETCH_CONTROL*[0] STARTENGINE	0x1

Table 12-212. WAIT Pin Configuration

Subprocess Name	Register/Bit Field	Value
Selects when the engine starts the access to chip-select.	*GPMC_PREFETCH_CONFIG1*[3] SYNCHROMODE	x
Select which WAIT pin edge detector should start the engine in synchronized mode.	*GPMC_PREFETCH_CONFIG1*[5-4] WAITPINSELECTOR	x

Table 12-213. Enable Chip-Select

Subprocess Name	Register/Bit Field	Value
When all parameters are configured, enable the chip-select.	*GPMC_CONFIG7_i*[6] CSVALID	x

12.3.3.5.5 Set Memory Access

Note

Some of the GMPC features described in this section may not be supported on this family of devices. For more information, see *GPMC Not Supported Features*.

This section describes the bit field to configure to set the GPMC in various memory modes. Table 12-214 and Table 12-215 provide check lists for mode parameters and access type parameters, respectively.

Copyright © 2026 Texas Instruments Incorporated

Table 12-214. Mode Parameters Check List

Register	Bit	Name	Asynchronous				Synchronous			
			Single Read Access	Single Write Access	Multiple Read (Page) Access	Multiple Write (Page) Access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access
GPMC_CONFIG1_i	30	READMULTIPLE	0x0	Don't care	0x1 (1)	Not Supported	0x0	Don't care	0x1	Don't care
GPMC_CONFIG1_i	29	READTYPE	0x0	Don't care	0x0 (1)	Not Supported	0x1	Don't care	0x1	Don't care
GPMC_CONFIG1_i	28	WRITEMULTIPLE	Don't care	0x0	- (1)	Not Supported	Don't care	0x0	Don't care	0x1
GPMC_CONFIG1_i	27	WRITETYPE	Don't care	0x0	- (1)	Not Supported	Don't care	0x1	Don't care	0x1

(1) Multiple read is not supported in address/data-multiplexed and AAD-multiplexed modes. Multiple read is supported in non-multiplexed mode.

Table 12-215. Access Type Parameters Check List

Register	Bit	Name	Access Type		
			non-multiplexed	Address/ Data-Multiplexed	AAD-Multiplexed
GPMC_CONFIG1_i	9-8	MUXADDDATA	0x0	0x2	0x1

12.3.3.5.6 GPMC Timing Parameters

Figure 12-192 shows a programming model diagram for the NOR interfacing timing parameters.

Copyright © 2026 Texas Instruments Incorporated

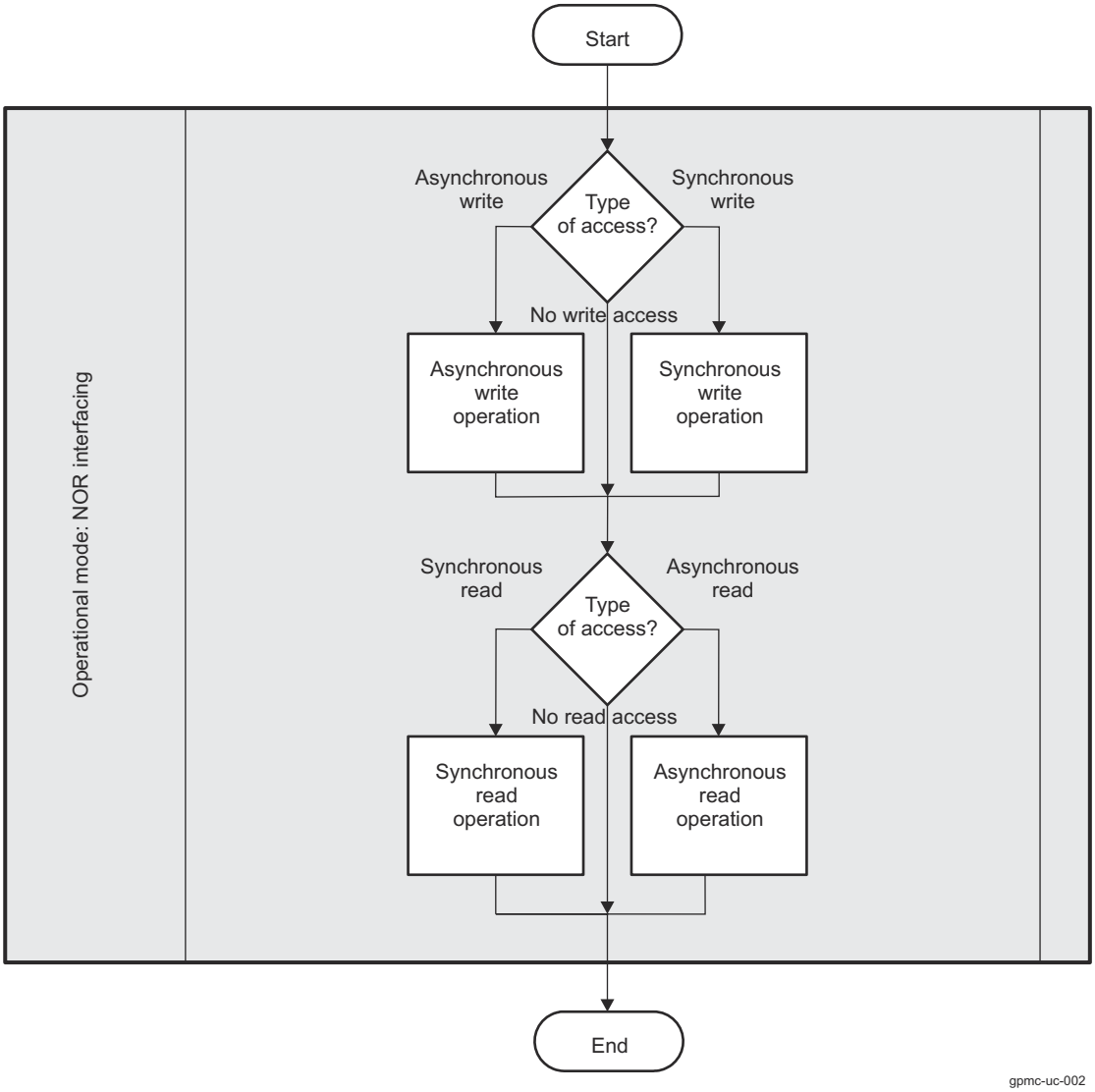


Figure 12-192. NOR Interfacing Timing Parameters Diagram

Table 12-216 lists the bit fields to configure adequate timing parameters in various memory modes.

Table 12-216. Timing Parameters

			Asynchronous			Synchronous				Access Type		
Register	Bit	Name	Single Read Access	Single Write Access	Multiple Read (Page) access	Single Read Access	Single Write Access	Multiple Read (Burst) Access	Multiple Write (Burst) Access	Non-multiplexed	Address/ Data-Multiplexed	AAD Multiplexed
GPMC_CONFIG1_i	9	MUXADDDATA	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	29	READTYPE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	30	READMULTIPLE	y		y	y		y		y	y	y
GPMC_CONFIG1_i	27	WRITETYPE		y			y		y	y	y	y
GPMC_CONFIG1_i	28	WRITEMULTIPLE		y			y		y	y	y	y
GPMC_CONFIG1_i	31	WRAPBURST						y	y	y	y	y

Copyright © 2026 Texas Instruments Incorporated

Table 12-216. Timing Parameters (continued)

			Asynchronous			Synchronous			Access Type			
GPMC_CONFIG1_i	26-25	CLKACTIVATIONTIME				y	y	y	y	y	y	y
GPMC_CONFIG1_i	19-18	WAITMONITORINGTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG1_i	4	TIMEPARAGRANULARITY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	20-16	CSWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG2_i	12-8	CSRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG2_i	7	CSEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG2_i	3-0	CSONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	30-28	ADVAADMUXWROFFTIME		y			y		y			y
GPMC_CONFIG3_i	30-29	ADVAADMUXRDOFFTIME	y		y	y		y				y
GPMC_CONFIG3_i	6-4	ADVAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG3_i	20-16	ADVWROFFTIME		y			y		y	y	y	y
GPMC_CONFIG3_i	12-8	ADVRDOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG3_i	7	ADVEXTRADELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG3_i	3-0	ADVONTIME	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG4_i	15-13	OEAADMUXOFFTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	6-4	OEAADMUXONTIME	y	y	y	y	y	y	y			y
GPMC_CONFIG4_i	28-24	WEOFFTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	23	WEEXTRADELAY		y			y		y	y	y	y
GPMC_CONFIG4_i	19-16	WEONTIME		y			y		y	y	y	y
GPMC_CONFIG4_i	12-8	OEOFFTIME	y		y	y		y		y	y	y
GPMC_CONFIG4_i	7	OEEXTRADELAY	y		y	y		y		y	y	y
GPMC_CONFIG4_i	3-0	OEONTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	27-24	PAGEBURSTACCESSTIME			y			y	y	y	y	y
GPMC_CONFIG5_i	20-16	RDACCESSTIME	y		y	y		y		y	y	y
GPMC_CONFIG5_i	12-8	WRCYCLETIME		y			y		y	y	y	y
GPMC_CONFIG5_i	4-0	RDCYCLETIME	y		y	y		y		y	y	y
GPMC_CONFIG6_i	28-24	WRACCESSTIME		y			y		y	y	y	y
GPMC_CONFIG6_i	19-16	WRDATAONADMUXBUS		y			y		y		y	y
GPMC_CONFIG6_i	11-8	CYCLE2CYCLEDELAY	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	7	CYCLE2CYCLESAMECSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	6	CYCLE2CYCLEDIFFCSEN	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG6_i	3-0	BUSTURNAROUND	y	y	y	y	y	y	y	y	y	y
GPMC_CONFIG7_i	6	CSVALID	y	y	y	y	y	y	y	y	y	y

Copyright © 2026 Texas Instruments Incorporated

12.3.3.5.6.1 GPMC Timing Parameters Formulas

This section is intended to help the user calculate the GPMC timing bit field values. Formulas are not listed exhaustively.

The section describes:

- NAND flash interface timing parameters formulas
- Synchronous NOR flash timing parameters formulas
- Asynchronous NOR flash timing parameters formulas

For complete information, such as OPP and board effects on timings, see the device-specific Datasheet.

12.3.3.5.6.1.1 NAND Flash Interface Timing Parameters Formulas

This section lists formulas to calculate NAND timing parameters. This is the case when GPMC_CONFIG1_i[11-10] DEVICETYPE = 0x2. Table 12-217 describes the NAND timing parameters.

Table 12-217. NAND Formulas Description

Configuration Parameter	Unit	Description
A	ns	Pulse duration – GPMC_WEn valid time
B	ns	Delay time – GPMC_CS valid to GPMC_WEn valid
C	ns	Delay time – GPMC_BE0n_CLE/GPMC_ADVn_ALE high to GPMC_WEn valid
D	ns	Delay time – GPMC_AD[15-0] valid to GPMC_WEn valid
E	ns	Delay time – GPMC_WEn invalid to GPMC_AD[15-0] invalid
F	ns	Delay time – GPMC_WEn invalid to GPMC_BE0n_CLE/GPMC_ADVn_ALE invalid
G	ns	Delay time – GPMC_WEn invalid to GPMC_CS invalid
H	ns	Cycle time – Write cycle time
I	ns	Delay time – GPMC_CS valid to GPMC_OEn_REn valid
J	ns	Setup time – GPMC_AD[15-0] valid to GPMC_OEn_REn invalid
K	ns	Pulse duration – GPMC_OEn_REn valid time
L	ns	Cycle time – Read cycle time
M	ns	Delay time – GPMC_OEn_REn invalid to GPMC_CS invalid

The configuration parameters are calculated through the following formulas. For more information, see the device-specific Datasheet.

A = (WEOffTime – WEOnTime) * (TimeParaGranularity + 1) * GPMC_FCLK period

B = ((WEOnTime – CSOnTime) * (TimeParaGranularity + 1) + 0.5 * (WEExtraDelay – CSExtraDelay)) * GPMC_FCLK period

C = ((WEOnTime – ADVOnTime) * (TimeParaGranularity + 1) + 0.5 * (WEExtraDelay - ADVExtraDelay)) * GPMC_FCLK period

D = (WEOnTime * (TimeParaGranularity + 1) + 0.5 * WEExtraDelay) * GPMC_FCLK period

E = (WrCycleTime – WEOffTime * (TimeParaGranularity + 1) – 0.5 * WEExtraDelay) * GPMC_FCLK period

F = (ADVWrOffTime – WEOffTime * (TimeParaGranularity + 1) + 0.5 * (ADVExtraDelay – WEExtraDelay) * GPMC_FCLK period

G = (CSWrOffTime – WEOffTime * (TimeParaGranularity + 1) + 0.5 * (CSExtraDelay – WEExtraDelay) * GPMC_FCLK period

H = WrCycleTime * (1 + TimeParaGranularity) * GPMC_FCLK period

I = ((OEOnTime – CSOnTime) * (TimeParaGranularity + 1) + 0.5 * (OEExtraDelay – CSExtraDelay)) * GPMC_FCLK period

J = ((RdAccessTime – OEOffTime) * (TimeParaGranularity + 1) – 0.5 * OEExtraDelay)) * GPMC_FCLK period

K = (OEOffTime – OEOnTime) * (1 + TimeParaGranularity) * GPMC_FCLK period

L = RdCycleTime * (1 + TimeParaGranularity) * GPMC_FCLK period

M = (CSRdOffTime – OEOffTime * (TimeParaGranularity + 1) + 0.5 * (CSExtraDelay - OEExtraDelay) * GPMC_FCLK period

Figure 12-193 shows a simplified example of command latch cycle timing where formulas are associated with signal waves.

Copyright © 2026 Texas Instruments Incorporated

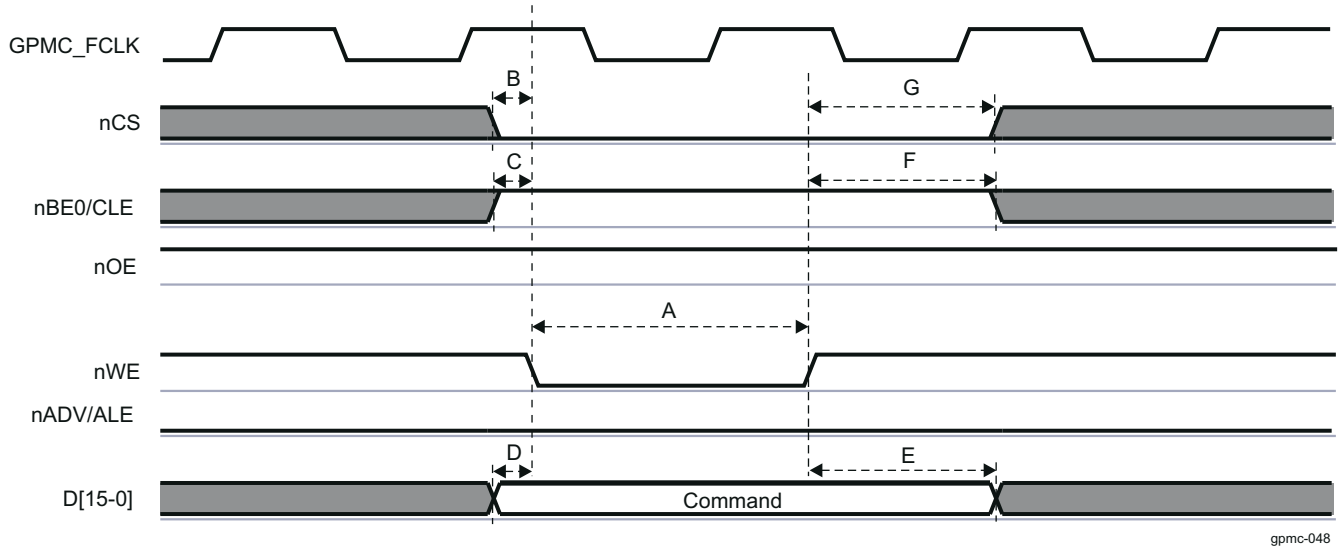


Figure 12-193. NAND Command Latch Cycle Timing Simplified Example

12.3.3.5.6.1.2 Synchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to calculate synchronous NOR timing parameters. This is the case when *GPMC_CONFIG1_i*[11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to synchronous mode. Table 12-218 describes the synchronous NOR formulas.

Table 12-218. Synchronous NOR Formulas Description

Configuration Parameter	Unit	Description
A	ns	Pulse duration – nCS low
B	ns	Delay time – address bus valid to CLK first edge
		Delay time – nBE0 / nBE1 valid to CLK first edge
C	ns	Pulse duration – nBE0 / nBE1 low
D	ns	Delay time – CLK rising edge to nBE0 / nBE1 invalid
		Delay time – CLK rising edge to nADV/ALE invalid
E	ns	Delay time – CLK rising edge to nCS invalid
		Delay time – CLK rising edge to nOE/nRE invalid
F	ns	Delay time – CLK rising edge to nCS transition
G	ns	Delay time – CLK rising edge to nADV/ALE transition
H	ns	Delay time – CLK rising edge to nOE/nRE transition
I	ns	Delay time – CLK rising edge to nWE transition
J	ns	Delay time – CLK rising edge to A[16-1/]D[15-0] data bus transition
		Delay time – CLK rising edge to nBE0 / nBE1 transition
K	ns	Pulse duration – nADV/ALE low
L	ns	Delay time – WAIT invalid to first data latching CLK edge

The configuration parameters are calculated through the following formulas. For more information, see the device-specific Datasheet.

1. For single read accesses:
   A = (CSRDOFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   C = RDCYCLETIME * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   D = (RDCYCLETIME – RDACCESSTIME) * GPMC_FCLK period
   E = (CSRDOFFTIME – RDACCESSTIME) * GPMC_FCLK period

Copyright © 2026 Texas Instruments Incorporated

2. For burst read accesses (where n is the page burst access number):
   A = (CSRDOFFTIME – CSONTIME + (n – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   C = (RDCYCLETIME + (n – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   D = (RDCYCLETIME – (RDACCESSTIME + (n – 1) * PAGEBURSTACCESSTIME)) * GPMC_FCLK period
   E = (CSRDOFFTIME – (RDACCESSTIME + (n – 1) * PAGEBURSTACCESSTIME)) * GPMC_FCLK period
3. For burst write accesses (where n is the page burst access number):
   A = (CSWROFFTIME – CSONTIME + (n – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   C = (WRCYCLETIME + (n – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
   D = (WRCYCLETIME – (RDACCESSTIME + (n – 1) * PAGEBURSTACCESSTIME)) * GPMC_FCLK period
   E = (CSWROFFTIME – (RDACCESSTIME + (n – 1) * PAGEBURSTACCESSTIME)) * GPMC_FCLK period
4. For all accesses:
   For nCS falling edge (chip-select activated):
   - Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period
   - Case where GPMCFCLKDIVIDER = 0x1:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and CSONTIME are odd) or (CLKACTIVATIONTIME and CSONTIME are even)
     F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period otherwise.
   - Case where GPMCFCLKDIVIDER = 0x2:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CSONTIME – CLKACTIVATIONTIME) is a multiple of 3
     F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSONTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
     F = (2 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSONTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

   For nCS rising edge (chip-select deactivated) in reading mode:
   - Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period
   - Case where GPMCFCLKDIVIDER = 0x1:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and CSRDOFFTIME are odd) or (CLKACTIVATIONTIME and CSRDOFFTIME are even)
     F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period otherwise.
   - Case where GPMCFCLKDIVIDER = 0x2:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CSRDOFFTIME – CLKACTIVATIONTIME) is a multiple of 3
     F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSRDOFFTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
     F = (2 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSRDOFFTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

   For nCS rising edge (chip-select deactivated) in writing mode:
   - Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period
   - Case where GPMCFCLKDIVIDER = 0x1:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and CSWROFFTIME are odd) or (CLKACTIVATIONTIME and CSWROFFTIME are even)
     F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period otherwise.
   - Case where GPMCFCLKDIVIDER = 0x2:
     F = 0.5 * CSEXTRADELAY * GPMC_FCLK period, when (CSWROFFTIME – CLKACTIVATIONTIME) is a multiple of 3

Copyright © 2026 Texas Instruments Incorporated

F = (1 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSWROFFTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
F = (2 + 0.5 * CSEXTRADELAY) * GPMC_FCLK period, when (CSWROFFTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nADV falling edge (nADV activated):

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVONTIME are odd) or (CLKACTIVATIONTIME and ADVONTIME are even)
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (ADVONTIME – CLKACTIVATIONTIME) is a multiple of 3
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVONTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  G = (2 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVONTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nADV rising edge (nADV deactivated) in reading mode:

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVRDOFFTIME are odd) or (CLKACTIVATIONTIME and ADVRDOFFTIME are even)
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3
  G = (2 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For nADV rising edge (nADV deactivated) in writing mode:

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVWROFFTIME are odd) or (CLKACTIVATIONTIME and ADVWROFFTIME are even)
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  G = 0.5 * ADVEXTRADELAY * GPMC_FCLK period, when (ADVWROFFTIME – CLKACTIVATIONTIME) is a multiple of 3
  G = (1 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVWROFFTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  G = (2 + 0.5 * ADVEXTRADELAY) * GPMC_FCLK period, when (ADVWROFFTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nOE falling edge (nOE activated):

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and OEONTIME are odd) or (CLKACTIVATIONTIME and OEONTIME are even)

Copyright © 2026 Texas Instruments Incorporated

H = (1 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period otherwise.

- Case where GPMCFCLKDIVIDER = 0x2:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period, when (OEONTIME – CLKACTIVATIONTIME) is a multiple of 3
  H = (1 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period, when (OEONTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  H = (2 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period, when (OEONTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nOE rising edge (nOE deactivated):

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and OEOFFTIME are odd) or (CLKACTIVATIONTIME and OEOFFTIME are even)
  H = (1 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  H = 0.5 * OEEXTRADELAY * GPMC_FCLK period, when (OEOFFTIME – CLKACTIVATIONTIME) is a multiple of 3
  H = (1 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period, when (OEOFFTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  H = (2 + 0.5 * OEEXTRADELAY) * GPMC_FCLK period, when (OEOFFTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nWE falling edge (nWE activated):

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and WEONTIME are odd) or (CLKACTIVATIONTIME and WEONTIME are even)
  I = (1 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period, when (WEONTIME – CLKACTIVATIONTIME) is a multiple of 3
  I = (1 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period, when (WEONTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  I = (2 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period, when (WEONTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nWE rising edge (nWE deactivated):

- Case where *GPMC_CONFIG1_i*[1-0] GPMCFCLKDIVIDER = 0x0:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period
- Case where GPMCFCLKDIVIDER = 0x1:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period, when (CLKACTIVATIONTIME and WEOFFTIME are odd) or (CLKACTIVATIONTIME and WEOFFTIME are even)
  I = (1 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period otherwise.
- Case where GPMCFCLKDIVIDER = 0x2:
  I = 0.5 * WEEXTRADELAY * GPMC_FCLK period, when (WEOFFTIME – CLKACTIVATIONTIME) is a multiple of 3
  I = (1 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period, when (WEOFFTIME – CLKACTIVATIONTIME – 1) is a multiple of 3
  I = (2 + 0.5 * WEEXTRADELAY) * GPMC_FCLK period, when (WEOFFTIME – CLKACTIVATIONTIME – 2) is a multiple of 3

For nADV low pulse duration:

- Read operation:

Copyright © 2026 Texas Instruments Incorporated

K = (ADVRDOFFTIME – ADVONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period

- Write operation:
  K = (ADVWROFFTIME – ADVONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period

For WAIT invalid to first data latching GPMC output clock edge:

- L = WAITMONITORINGTIME * (GPMCFCLKDIVIDER + 1) * GPMC_FCLK period + GPMC output clock period

Figure 12-194 shows a simplified example of a synchronous NOR single read where formulas are associated with signal waves.

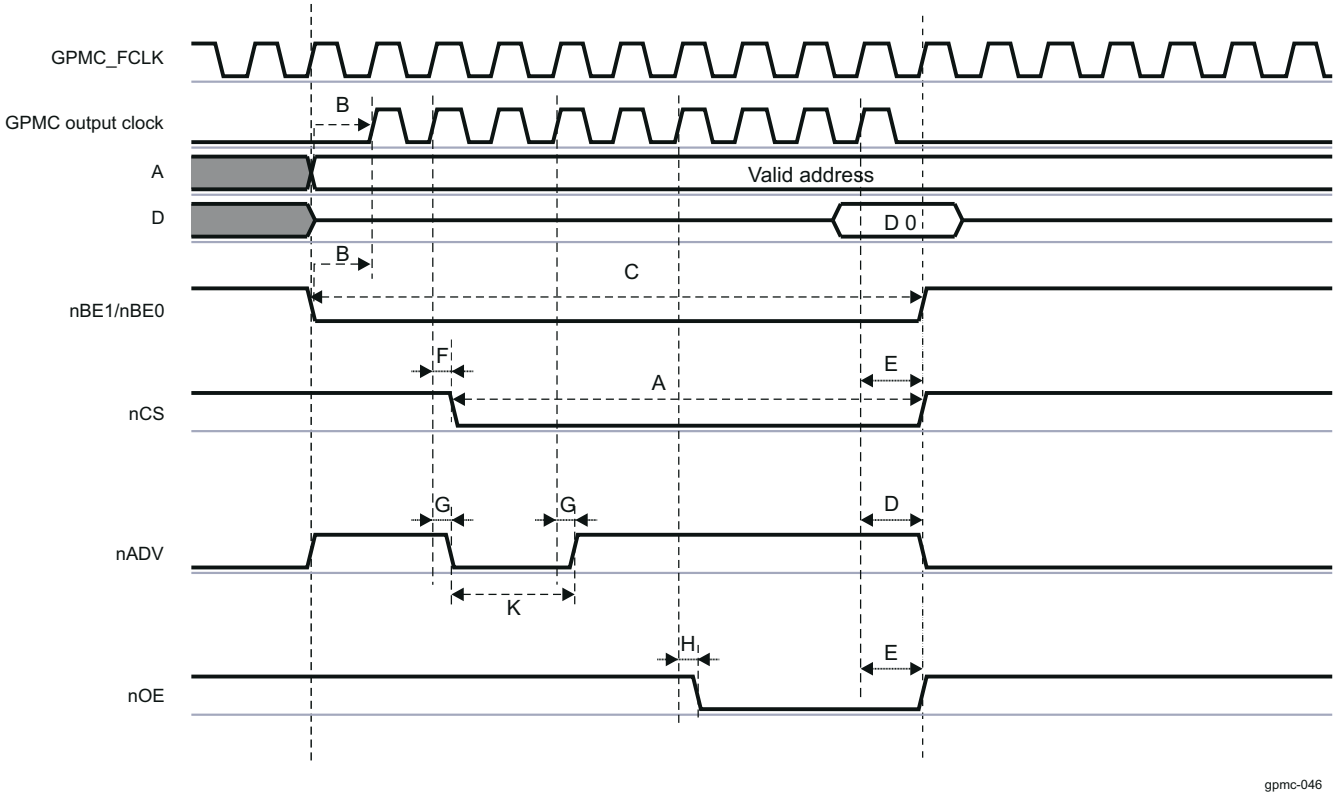


Figure 12-194. Synchronous NOR Single Read Simplified Example

12.3.3.5.6.1.3 Asynchronous NOR Flash Timing Parameters Formulas

This section lists all the formulas to calculate asynchronous NOR timing parameters. This is the case when *GPMC_CONFIG1_i*[11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to asynchronous mode. Table 12-219 describes the asynchronous NOR formulas.

Table 12-219. Asynchronous NOR Formulas Description

Configuration Parameter	Unit	Description
A	ns	Pulse duration – nCS low
B	ns	Delay time – nCS valid to nADV/ALE invalid
C	ns	Delay time – nCS valid to nOE/nRE invalid (single read)
D	ns	Pulse duration – address bus valid - 2nd, 3rd and 4th accesses
E	ns	Delay time – nCS valid to nWE valid
F	ns	Delay time – nCS valid to nWE invalid
G	ns	Address invalid duration between two successive R/W accesses
H	ns	Setup time – read data valid before nOE/nRE high

Copyright © 2026 Texas Instruments Incorporated

Table 12-219. Asynchronous NOR Formulas Description (continued)

Configuration Parameter	Unit	Description
I	ns	Delay time – nCS valid to nOE/nRE invalid (burst read)
J	ns	Delay time – address bus valid to nCS valid
		Delay time – data bus valid to nCS valid
		Delay time – nBE0 / nBE1 valid to nCS valid
K	ns	Delay time – nCS valid to nADV/ALE valid
L	ns	Delay time – nCS valid to nOE/nRE valid
M	ns	Delay time – nCS valid to first data latching edge
N	ns	Pulse duration – nBE0 / nBE1 valid time
O	ns	Delay time – nCS valid to nADV/ALE valid

The configuration parameters are calculated through the following formulas. These formulas are not exhaustive. For more information, see the device-specific Datasheet.

- nCS low pulse:
  For single read: A = (CSRDOFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
  For burst read: A = (CSRDOFFTIME – CSONTIME + (N – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period, where N = page burst access number
  For single write: A = (CSWROFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
  For burst write: A = (CSWROFFTIME – CSONTIME + (N – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period, where N = page burst access number
- nCS valid to nADV/ALE invalid delay:
  For reading: B = ((ADVRDOFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
  For writing: B = ((ADVWROFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- C = ((OEOFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (OEEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- D = PAGEBURSTACCESSTIME * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period
- E = ((WEONTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (WEEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- F = ((WEOFFTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (WEEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- G = CYCLE2CYCLEDELAY * GPMC_FCLK period
- H = ((OEOFFTIME – RDACCESSTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * OEEXTRADELAY) * GPMC_FCLK period
- I = ((OEOFFTIME + (N – 1) * PAGEBURSTACCESSTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (OEEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period, where N = page burst access number
- J = (CSONTIME * (TIMEPARAGRANULARITY + 1) + 0.5 * CSEXTRADELAY) * GPMC_FCLK period
- K = ((ADVONTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- L = ((OEONTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (OEEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period
- M = ((RDACCESSTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) – 0.5 * CSEXTRADELAY) * GPMC_FCLK period
- nBE0 /nBE1 pulse:
  For single read: N = RDCYCLETIME * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period

Copyright © 2026 Texas Instruments Incorporated

For burst read: N = (RDCYCLETIME + (N – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period, where N = page burst access number
For burst write: N = (WRCYCLETIME + (N – 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK period, where N = page burst access number

- O = ((WRCYCLETIME + (N – 1) * PAGEBURSTACCESSTIME – CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY – CSEXTRADELAY)) * GPMC_FCLK period

Figure 12-195 shows a simplified example of an asynchronous NOR single write where formulas are associated with signal waves.

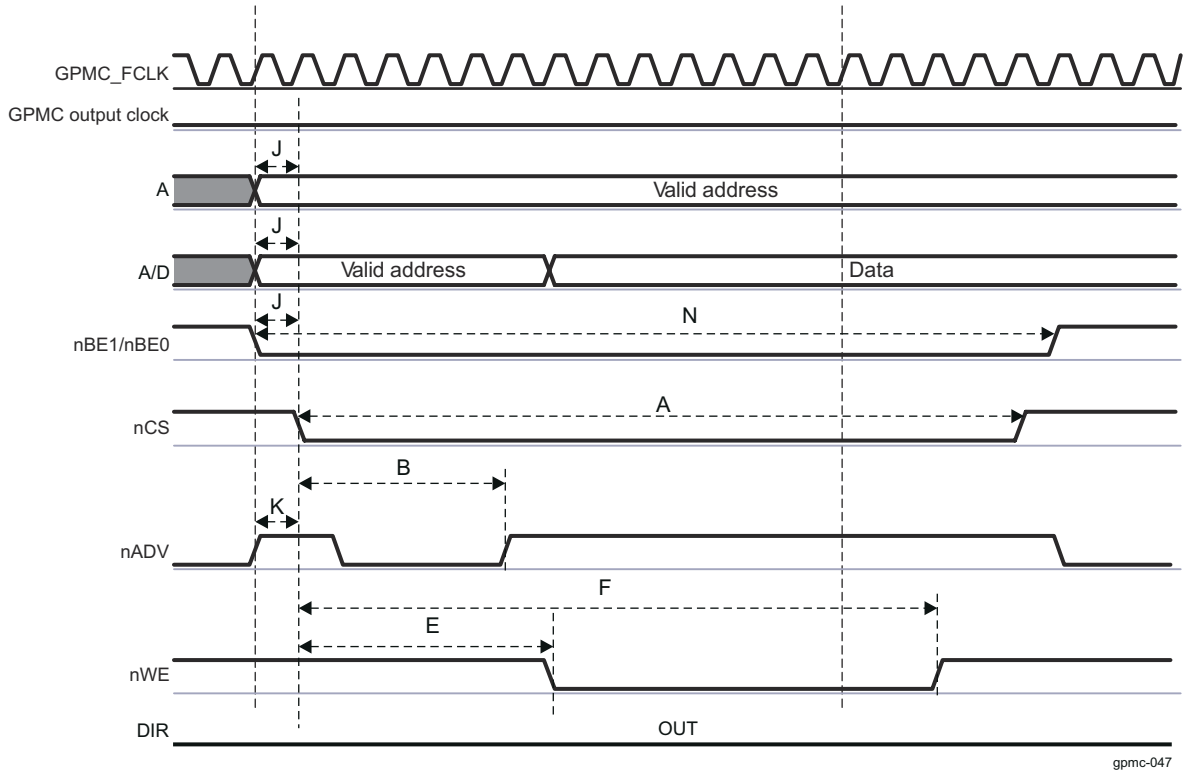


Figure 12-195. Asynchronous NOR Single Write Simplified Example

Note

Write multiple access is not supported in asynchronous mode. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

Copyright © 2026 Texas Instruments Incorporated

12.3.4 Error Location Module (ELM)

This section describes the Error Location Module (ELM) for the device.

12.3.4.1 ELM Overview

The ELM extracts error addresses from generated syndrome polynomials.

The ELM is used with the GPMC. Syndrome polynomials generated on-the-fly when reading a NAND flash page and stored in GPMC registers are passed to the ELM. A host processor can then correct the data block by flipping the bits to which the ELM error-location outputs point.

When reading from NAND flash memories, some level of error-correction is required. In the case of NAND modules with no internal correction capability, sometimes referred to as *bare NANDs*, the correction process is delegated to the memory controller. ELM can be also used to support parallel NOR flash or NAND flash.

The General-Purpose Memory Controller (GPMC) probes data read from an external NAND flash and uses this to compute checksum-like information, called syndrome polynomials, on a per-block basis. Each syndrome polynomial gives a status of the read operations for a full block, including 512 bytes of data, parity bits, and an optional spare-area data field, with a maximum block size of 1023 bytes. Computation is based on a Bose-Chaudhuri-Hocquenghem (BCH) algorithm. The ELM extracts error addresses from these syndrome polynomials.

Based on the syndrome polynomial value, the ELM can detect errors, compute the number of errors, and give the location of each error bit. The actual data is not required to complete the error-correction algorithm. Errors can be reported anywhere in the NAND flash block, including in the parity bits.

The maximum acceptable number of errors that can be corrected depends on a programmable configuration parameter. 4-, 8-, and 16-bit error-correction levels are supported. The ELM depends on a static and fixed definition of the generator polynomial for each error-correction level that corresponds to the generator polynomials defined in the GPMC (there are three fixed polynomial for the three correction error levels). A larger number of errors than the programmed error-correction level may be detected, but the ELM cannot correct them all. The offending block is then tagged as *uncorrectable* in the associated computation exit status register. If the computation is successful, that is, if the number of errors detected does not exceed the maximum value authorized for the chosen correction capability, the exit status register contains the information on the number of detected errors.

When the error-location process completes, an interrupt is triggered to inform the software that its status can be checked. The number of detected errors and their locations in the NAND block can be retrieved from the module through register accesses.

.

Figure 12-196 shows the ELM0 module overview.

Copyright © 2026 Texas Instruments Incorporated

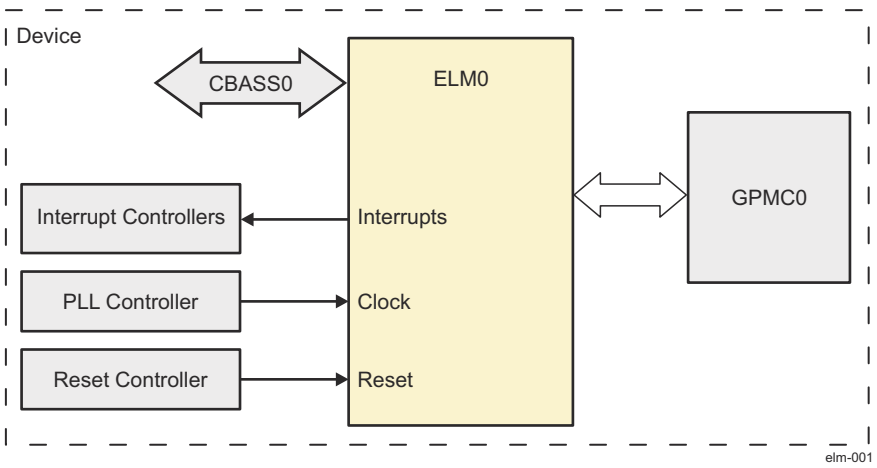


Figure 12-196. ELM0 Overview

12.3.4.1.1 ELM Features

The ELM has the following features:

- 4, 8, and 16 bits per 512-byte block error-location, based on BCH algorithms
- Eight simultaneous processing contexts
- Page-based and continuous modes
- Interrupt generation on error-location process completion:
  - When the full page has been processed in page mode
  - For each syndrome polynomial in continuous mode.

12.3.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.3.4.2 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.4.3 ELM Functional Description

The ELM0 module is hereinafter referred to as ELM.

The ELM is designed around the error-location engine, which handles the computation based on the input syndrome polynomials.

The ELM maps the error-location engine to a standard interconnect interface by using a set of registers to control inputs and outputs.

12.3.4.3.1 ELM Software Reset

To perform a software reset, set the ELM_SYSCONFIG[1] SOFTRESET bit to 1. The ELM_SYSSTS[0] RESETDONE bit indicates that the software reset is complete when its value is 1. When the software reset completes, the ELM_SYSCONFIG[1] SOFTRESET bit is automatically reset.

12.3.4.3.2 ELM Power Management

Note

Some of the ELM features described in this section may not be supported on this family of devices. See the *Module Integration* section for more information about Unsupported Features.

Table 12-220 describes the power-management features available to the ELM.

Table 12-220. Local Power-Management Features

Feature	Registers	Description
Clock autogating	ELM_SYSCONFIG[0] AUTOGATING	This bit allows a local power optimization inside the module by gating the ELM_FICLK clock upon the interface activity.
Idle modes	ELM_SYSCONFIG[4-3] SIDLEMODE	Force-idle, no-idle, and smart-idle modes are available.
Clock activity	ELM_SYSCONFIG[8] CLOCKACTIVITY	The clock can be switched off or maintained.

12.3.4.3.3 ELM Interrupt Requests

Table 12-221 lists the event flags, and their masks, that can cause module interrupts asserting the signal.

Table 12-221. ELM Events

Event Flag	Event Mask	Description
ELM_IRQSTS[8] PAGE_VALID	ELM_IRQEN[8] PAGE_MASK	Page interrupt
ELM_IRQSTS[7] LOC_VALID_7	ELM_IRQEN[7] LOCATION_MASK_7	Error-location interrupt for syndrome polynomial 7
ELM_IRQSTS[6] LOC_VALID_6	ELM_IRQEN[6] LOCATION_MASK_6	Error-location interrupt for syndrome polynomial 6
ELM_IRQSTS[5] LOC_VALID_5	ELM_IRQEN[5] LOCATION_MASK_5	Error-location interrupt for syndrome polynomial 5
ELM_IRQSTS[4] LOC_VALID_4	ELM_IRQEN[4] LOCATION_MASK_4	Error-location interrupt for syndrome polynomial 4
ELM_IRQSTS[3] LOC_VALID_3	ELM_IRQEN[3] LOCATION_MASK_3	Error-location interrupt for syndrome polynomial 3
ELM_IRQSTS[2] LOC_VALID_2	ELM_IRQEN[2] LOCATION_MASK_2	Error-location interrupt for syndrome polynomial 2
ELM_IRQSTS[1] LOC_VALID_1	ELM_IRQEN[1] LOCATION_MASK_1	Error-location interrupt for syndrome polynomial 1
ELM_IRQSTS[0] LOC_VALID_0	ELM_IRQEN[0] LOCATION_MASK_0	Error-location interrupt for syndrome polynomial 0

Copyright © 2026 Texas Instruments Incorporated

12.3.4.3.4 ELM Processing Initialization

ELM_LOCATION_CONFIG global setting parameters must be set before using the error-location engine. The ELM_LOCATION_CONFIG[1-0] ECC_BCH_LEVEL bit field defines the error-correction level used (4-, 8-, or 16-bit error correction). The ELM_LOCATION_CONFIG[26-16] ECC_SIZE bit field defines the maximum buffer length beyond which the engine processing no longer looks for errors.

Software can choose to use the ELM in continuous mode or page mode. If all ELM_PAGE_CTRL[i] SECTOR_i bits (i is the syndrome polynomial number, where i = 0 to 7) are reset, continuous mode is used. In any other case, page mode is implicitly selected.

- Continuous mode: Each syndrome polynomial is processed independently. Results for a syndrome can be retrieved and acknowledged at any time, regardless of the status of the other seven processing contexts.
- Page mode: Syndrome polynomials are grouped into atomic entities: only one page can be processed at any given time, even if all eight contexts are not used for this page. Unused contexts are lost and cannot be affected to any other processing. The full page must be acknowledged and cleared before moving to the next page.

For completion interrupts to be generated correctly, all ELM_IRQEN[i] LOCATION_MASK_i bits (where i = 0 to 7) must be forced to 0 when in page mode, and set to 1 in continuous mode. Additionally, the ELM_IRQEN[8] PAGE_MASK bit must be set to 1 when in page mode.

Software initiates error-location processing by writing a syndrome polynomial into one of the eight possible register sets. Each of these register sets includes seven registers: ELM_SYNDROME_FRAGMENT_0_i to ELM_SYNDROME_FRAGMENT_6_i. The first six registers can be written in any order, but ELM_SYNDROME_FRAGMENT_6_i must be written last because it includes the validity bit, which instructs the ELM that this syndrome polynomial must be processed (the ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID bit).

As soon as one validity bit is asserted (ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID = 0x1, where i = 0 to 7), error-location processing can start for the corresponding syndrome polynomial. The associated ELM_LOCATION_STS_i and ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i registers (where i = 0 to 7) are not reset. Software must not consider them until the corresponding ELM_IRQSTS[i] LOC_VALID_i bit is set.

12.3.4.3.5 ELM Processing Sequence

While the error-location engine is busy processing one syndrome polynomial, further syndrome polynomials can be written. They are processed when the current processing completes.

The engine completes early when:

- No error is detected; that is, when the ELM_LOCATION_STS_i[8] ECC_CORRECTABLE bit is set to 1 and the ELM_LOCATION_STS_i[4-0] ECC_NB_ERRORS bit field is set to 0x0.
- Too many errors are detected; that is, when the ELM_LOCATION_STS_i[8] ECC_CORRECTABLE bit is set to 0 while the ELM_LOCATION_STS_i[4-0] ECC_NB_ERRORS bit field is set with the value output by the error-location engine. The reported number of errors is not ensured if ECC_CORRECTABLE is 0.

If the engine completes early, the associated error-location registers ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i (where i = 0 to 7) are not updated.

In all other cases, the engine goes through the entire error-location process. Each time an error location is found, it is logged in the associated ECC_ERROR_LOCATION bit field. The first error detected is logged in the ELM_ERROR_LOCATION_0_i[12-0] ECC_ERROR_LOCATION bit field; the second is logged in the ELM_ERROR_LOCATION_1_i[12-0] ECC_ERROR_LOCATION bit field, and so on.

Table 12-222 describes the ELM_LOCATION_STS_i value decoding.

Table 12-222. ELM_LOCATION_STS_i Value Decoding

ECC_CORRECTABLE Value	ECC_NB_ERRORS Value	Status	Number of Errors Detected	Action Required

Copyright © 2026 Texas Instruments Incorporated

Table 12-222. ELM_LOCATION_STS_i Value Decoding (continued)

1	0	OK	0	None
1	≠ 0	OK	ECC_NB_ERRORS	Correct the data buffer read based on the ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i results.
0	Any	Failed	Unknown	Software-dependent

12.3.4.3.6 ELM Processing Completion

When the processing for a given syndrome polynomial completes, its ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID bit is reset. It must not be set again until the exit status registers, ELM_LOCATION_STS_i (where i = 0 to 7) for this processing are checked. Failure to comply with this rule leads to potential loss of the first polynomial process data output.

The error-location engine signals the process completion to the ELM. When this event is detected, the corresponding ELM_IRQSTS[i] LOC_VALID_i bit (where i = 0 to 7) is set. The processing exit status is available from the associated ELM_LOCATION_STS_i register, and error locations are stored in order in the ECC_ERROR_LOCATION bit fields. Software must read only valid error-location registers based on the number of errors detected and located.

Immediately after the error-location engine completes, a new syndrome polynomial can be processed, if any is available, as reported by the ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID bit, depending on the configured error-correction level. If several syndrome polynomials are available, a round-robin arbitration is used to select one for processing.

In continuous mode (that is, all bits in ELM_PAGE_CTRL are reset), an interrupt is triggered whenever a ELM_IRQSTS[i] LOC_VALID_i bit is asserted. Software must read the ELM_IRQSTS register to determine which polynomial is processed and retrieve the exit status and error locations (ELM_LOCATION_STS_i and ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i). When done, software must clear the corresponding ELM_IRQSTS[i] LOC_VALID_i bit by setting it to 1. Other status bits must be set to 0 so that other interrupts are not unintentionally cleared. When using this mode, the ELM_IRQSTS[8] PAGE_VALID interrupt is never triggered.

In page mode, the module does not trigger interrupts for the processing completion of each polynomial because the ELM_IRQEN[i] LOCATION_MASK_i bits are cleared. A page is defined using the ELM_PAGE_CTRL register. Each SECTOR_i bit set means the corresponding polynomial i is part of the page processing. A page is fully processed when all tagged polynomials have been processed, as logged in the ELM_IRQSTS[i] LOC_VALID_i bits. The module triggers an ELM_IRQSTS[8] PAGE_VALID interrupt whenever it detects that the full page has been processed. To make sure the next page can be correctly processed, all status bits in the ELM_IRQSTS register must be cleared by using a single atomic-write access.

Note

Do not modify page setting parameters in the ELM_PAGE_CTRL register unless the engine is idle, no polynomial input is valid, and all interrupts have been cleared.

Because no polynomial-level interrupt is triggered in page mode, polynomials cleared in the ELM_PAGE_CTRL[i] SECTOR_i bits (where i = 0 to 7) are processed as usual, but are essentially ignored. Software must manually poll the ELM_IRQSTS bits to check for their status.

Copyright © 2026 Texas Instruments Incorporated

12.3.4.4 ELM Basic Programming Model

12.3.4.4.1 ELM Low-Level Programming Model

12.3.4.4.1.1 Processing Initialization

Table 12-223. ELM Processing Initialization

Step	Register/Bit Field/Programming Model	Value
Resets the module.	ELM_SYSCONFIG[1] SOFTRESET	0x1
Wait until reset is done.	ELM_SYSSTS[0] RESETDONE	0x1
Configure the target interface power management.	ELM_SYSCONFIG[4-3] SIDLEMODE	Set value
Defines the error-correction level used.	ELM_LOCATION_CONFIG[1-0] ECC_BCH_LEVEL	Set value
Defines the maximum buffer length.	ELM_LOCATION_CONFIG[26-16] ECC_SIZE	Set value
Sets the ELM in continuous mode or page mode.	ELM_PAGE_CTRL	Set value
IF continuous mode is used:	All ELM_PAGE_CTRL[i] SECTOR_i (where i = 0 to 7)	0x0
Enable interrupt for syndrome polynomial i.	ELM_IRQEN[i] LOCATION_MASK_i	0x1
ELSE (page mode is used):	One syndrome polynomial i is set ELM_PAGE_CTRL[i] SECTOR_i (where i = 0 to 7)	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	All ELM_IRQEN[i] LOCATION_MASK_i = 0x0 and ELM_IRQEN[8] PAGE_MASK = 0x1	Set value
ENDIF		Set value
Set the input syndrome polynomial i.	ELM_SYNDROME_FRAGMENT_0_i	Set value
	ELM_SYNDROME_FRAGMENT_1_i	Set value
	ELM_SYNDROME_FRAGMENT_5_i	Set value
	ELM_SYNDROME_FRAGMENT_6_i	Set value
Initiates the computation process.	ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID	0x1

12.3.4.4.1.2 Read Results

The engine goes through the entire error-location process and results can be read. Table 12-224 and Table 12-225 describe the processing completion for continuous and page modes, respectively.

Table 12-224. ELM Processing Completion for Continuous Mode

Step	Register/Bit Field/Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the interrupt is generated, or poll the status register.		
Read for which i the error-location process is complete.	*ELM_IRQSTS*[i] LOC_VALID_i	0x1
IF the process fails (too many errors):	*ELM_LOCATION_STS_i*[8] ECC_CORRECTABLE	0x0
It is software dependant.		
ELSE (process successful, the engine completes):	*ELM_LOCATION_STS_i*[8] ECC_CORRECTABLE	0x1
Read the number of errors.	*ELM_LOCATION_STS_i*[4-0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers. Software must correct errors in the data buffer.	*ELM_ERROR_LOCATION_0_i*[12-0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12-0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12-0] ECC_ERROR_LOCATION	
ENDIF		
Clear the corresponding i interrupt.	*ELM_IRQSTS*[i] LOC_VALID_i	0x1

A new syndrome polynomial can be processed after the end of processing (

Copyright © 2026 Texas Instruments Incorporated

ELM_SYNDROME_FRAGMENT_6_i[16] SYNDROME_VALID = 0x0) and after the exit status register check (*ELM_LOCATION_STS_i*

Table 12-225. ELM Processing Completion for Page Mode

Step	Register/Bit Field/Programming Model	Value
Wait until process is complete for syndrome polynomial i: Wait until the interrupt is generated, or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	*ELM_IRQSTS*[8] PAGE_VALID	0x1
Repeat the following actions the necessary number of times. That is, once for each valid defined block in the page.		
Read the process exit status.	*ELM_LOCATION_STS_i*[8] ECC_CORRECTABLE	
IF the process fails (too many errors):	*ELM_LOCATION_STS_i*[8] ECC_CORRECTABLE	0x0
It is software dependent.		
ELSE (process successful, the engine completes):	*ELM_LOCATION_STS_i*[8] ECC_CORRECTABLE	0x1
Read the number of errors.	*ELM_LOCATION_STS_i*[4-0] ECC_NB_ERRORS	
Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers.	*ELM_ERROR_LOCATION_0_i*[12-0] ECC_ERROR_LOCATION	
	ELM_ERROR_LOCATION_1_i[12-0] ECC_ERROR_LOCATION	
	...	
	ELM_ERROR_LOCATION_15_i[12-0] ECC_ERROR_LOCATION	
ENDIF		
End Repeat		
Clear the *ELM_IRQSTS* register.	*ELM_IRQSTS*	0x1FF

12.3.4.4.1.3

Next page can be correctly processed after a page is fully processed, when all tagged polynomials have been processed (*ELM_IRQSTS*[i] LOC_VALID_i = 0x1 for all syndrome polynomials i used in the page).

12.3.4.4.2 Use Case: ELM Used in Continuous Mode

In this example, the ELM is programmed for an 8-bit error-correction capability in continuous mode (see Table 12-226). After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, a nonzero polynomial syndrome is reported from the GPMC (polynomial syndrome 0 is used in the ELM):

- P = 0x0A16ABE115E44F767BFB0D0980.

Table 12-226. Use Case: Continuous Mode

Step	Register/Bit Field/Programming Model	Value
Reset the module.	*ELM_SYSCONFIG*[1] SOFTRESET	0x1
Wait until reset is done.	*ELM_SYSSTS*[0] RESETDONE	0x1
Configure the target interface power management: Smart idle is used.	*ELM_SYSCONFIG*[4-3] SIDLEMODE	0x2
Define the error-correction level used: 8 bits.	*ELM_LOCATION_CONFIG*[1-0] ECC_BCH_LEVEL	0x1
Define the maximum buffer length: 528 bytes (2 × 528 = 1056).	*ELM_LOCATION_CONFIG*[26-16] ECC_SIZE	0x420
Set the ELM in continuous mode.	*ELM_PAGE_CTRL*	0
Enable interrupt for syndrome polynomial 0.	*ELM_IRQEN*[0] LOCATION_MASK_0	0x1
Set the input syndrome polynomial 0.	*ELM_SYNDROME_FRAGMENT_0_i* (where i = 0)	0xFB0D0980
	ELM_SYNDROME_FRAGMENT_1_i (where i = 0)	0xE44F767B
	ELM_SYNDROME_FRAGMENT_2_i (where i = 0)	0x16ABE115
	ELM_SYNDROME_FRAGMENT_3_i (where i = 0)	0x0000000A

Copyright © 2026 Texas Instruments Incorporated

Table 12-226. Use Case: Continuous Mode (continued)

Step	Register/Bit Field/Programming Model	Value
Initiate the computation process.	*ELM_SYNDROME_FRAGMENT_6_i*[16] SYNDROME_VALID (where i = 0)	0x1
Wait until process is complete for syndrome polynomial 0: is generated or poll the status register.		
Read that error-location process is complete for syndrome polynomial 0.	*ELM_IRQSTS*[0] LOC_VALID_0	0x1
Read the process exit status: All errors were successfully located.	*ELM_LOCATION_STS*[8] ECC_CORRECTABLE (where i = 0)	0x1
Read the number of errors: Four errors detected.	*ELM_LOCATION_STS*[4-0] ECC_NB_ERRORS (where i = 0)	0x4
Read the error-location bit addresses for syndrome polynomial 0 of the first four registers: Errors are located in the data buffer at decimal addresses 431, 1062, 1909, 3452.	*ELM_ERROR_LOCATION_0_i* (where i = 0)	0x1AF
	ELM_ERROR_LOCATION_1_i (where i = 0)	0x426
	ELM_ERROR_LOCATION_2_i (where i = 0)	0x775
	ELM_ERROR_LOCATION_3_i (where i = 0)	0xD7C
Clear the corresponding interrupt for polynomial 0.	*ELM_IRQSTS*[0] LOC_VALID_0	0x1

The NAND flash data in the sector are seen as a polynomial of degree 4223 (number of bits in a 528 byte buffer minus 1), with each data bit being a coefficient in the polynomial. When reading from a NAND flash using the GPMC module, computation of the polynomial syndrome assumes that the first NAND word read at address 0x0 contains the highest-order coefficient in the message. Furthermore, in the 16-bit NAND word, bits are ordered from bit 7 to bit 0, and then from bit 15 to bit 8. Based on this convention, an address table of the data buffer can be built. NAND memory addresses in Table 12-227 are given in decimal format.

Table 12-227. 16-Bit NAND Sector Buffer Address Map

NAND Memory Address	Message Bit Addresses in Memory Word															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	4215	4214	4213	4212	4211	4210	4209	4208	4223	4222	4221	4220	4219	4218	4217	4216
1	4175	4174	4173	4172	4171	4170	4169	4168	4183	4182	4181	4180	4179	4178	4177	4176
...																
47	3463	3462	3461	3460	3459	3458	3457	3456	3471	3470	3469	3468	3467	3466	3465	3464
48	3447	3446	3445	3444	3443	3442	3441	3440	3455	3454	3453	3452	3451	3450	3449	3448
49	3431	3430	3429	3428	3427	3426	3425	3424	3439	3438	3437	3436	3435	3434	3433	3432
50	3415	3414	3413	3412	3411	3410	3409	3408	3423	3422	3421	3420	3419	3418	3417	3416
...																
255	135	134	133	132	131	130	129	128	143	142	141	140	139	138	137	136
256	119	118	117	116	115	114	113	112	127	126	125	124	123	122	121	120
257	103	102	101	100	99	98	97	96	111	110	109	108	107	106	105	104
258	87	86	85	84	83	82	81	80	95	94	93	92	91	90	89	88
259	71	70	69	68	67	66	65	64	79	78	77	76	75	74	73	72
260	55	54	53	52	51	50	49	48	63	62	61	60	59	58	57	56
261	39	38	37	36	35	34	33	32	47	46	45	44	43	42	41	40
262	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24
263	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

The table can now be used to determine which bits in the buffer were incorrect and must be flipped. In this example, the first bit to be flipped is bit 4 from the 49th byte read from memory. It is up to the processor to

Copyright © 2026 Texas Instruments Incorporated

correctly map this word to the copied buffer and flip this bit. The same process must be repeated for all detected errors.

12.3.4.4.3 Use Case: ELM Used in Page Mode

In this example, the ELM module is programmed for a 16-bit error-correction capability in page mode (see Table 12-228). After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, four non-zero polynomial syndromes are reported from the GPMC (polynomial syndrome 0, 1, 2, and 3 are used in the ELM):

- P0 = 0xE8B0 12ADDB5A318E05BE B0693DB28330B5CC A329AA05E0B718EF
- P1 = 0xBAD0 49A0D932C22E6669 0948DF08BE093336 79C6BA10E5F935EB
- P2 = 0x69D9 B86ABCD5EC3697FA A6498FEE54556EA0 1579EF7D60BA3189
- P3 = 0x0

Table 12-228. Use Case: Page Mode

Step	Register/Bit Field/Programming Model	Value
Reset the module	*ELM_SYSCONFIG*[1] SOFTRESET	0x1
Wait until reset is done.	*ELM_SYSCONFIG*[0] RESETDONE	0x1
Configure the target interface power management: Smart idle is used.	*ELM_SYSCONFIG*[4-3] SIDLEMODE	0x2
Define the error-correction level used: 16 bits	*ELM_LOCATION_CONFIG*[1-0] ECC_BCH_LEVEL	0x2
Define the maximum buffer length: 528 bytes	*ELM_LOCATION_CONFIG*[26-16] ECC_SIZE	0x420
Set the ELM in page mode (four blocks in a page)	*ELM_PAGE_CTRL*[0] SECTOR_0	0x1
	ELM_PAGE_CTRL[1] SECTOR_1	0x1
	ELM_PAGE_CTRL[2] SECTOR_2	0x1
	ELM_PAGE_CTRL[3] SECTOR_3	0x1
Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.	*ELM_IRQEN*	0x100
Set the input syndrome polynomial 0.	*ELM_SYNDROME_FRAGMENT_0_i* (where i = 0)	0xE0B718EF
	ELM_SYNDROME_FRAGMENT_1_i (where i = 0)	0xA329AA05
	ELM_SYNDROME_FRAGMENT_2_i (where i = 0)	0x8330B5CC
	ELM_SYNDROME_FRAGMENT_3_i (where i = 0)	0xB0693DB2
	ELM_SYNDROME_FRAGMENT_4_i (where i = 0)	0x318E05BE
	ELM_SYNDROME_FRAGMENT_5_i (where i = 0)	0x12ADDB5A
	ELM_SYNDROME_FRAGMENT_6_i (where i = 0)	0xE8B0
Set the input syndrome polynomial 1.	*ELM_SYNDROME_FRAGMENT_0_i* (where i = 1)	0xE5F935EB
	ELM_SYNDROME_FRAGMENT_1_i (where i = 1)	0x79C6BA10
	ELM_SYNDROME_FRAGMENT_2_i ((where i = 1)	0xBE093336
	ELM_SYNDROME_FRAGMENT_3_i (where i = 1)	0x0948DF08
	ELM_SYNDROME_FRAGMENT_4_i (where i = 1)	0xC22E6669
	ELM_SYNDROME_FRAGMENT_5_i (where i = 1)	0x49A0D932
	ELM_SYNDROME_FRAGMENT_6_i (where i = 1)	0xBAD0
Set the input syndrome polynomial 2.	*ELM_SYNDROME_FRAGMENT_0_i* (where i = 2)	0x60BA3189
	ELM_SYNDROME_FRAGMENT_1_i (where i = 2)	0x1579EF7D
	ELM_SYNDROME_FRAGMENT_2_i (where i = 2)	0x54556EA0
	ELM_SYNDROME_FRAGMENT_3_i (where i = 2)	0xA6498FEE
	ELM_SYNDROME_FRAGMENT_4_i (where i = 2)	0xEC3697FA
	ELM_SYNDROME_FRAGMENT_5_i (where i = 2)	0xB86ABCD5
	ELM_SYNDROME_FRAGMENT_6_i (where i = 2)	0x69D9

Copyright © 2026 Texas Instruments Incorporated

Table 12-228. Use Case: Page Mode (continued)

Step	Register/Bit Field/Programming Model	Value
Set the input syndrome polynomial 3.	*ELM_SYNDROME_FRAGMENT_0_i* (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_1_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_2_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_3_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_4_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_5_i (where i = 3)	0x0
	ELM_SYNDROME_FRAGMENT_6_i (where i = 3)	0x0
Initiate the computation process for syndrome polynomial 0	*ELM_SYNDROME_FRAGMENT_6_i*[16] SYNDROME_VALID (where i = 0)	0x1
Initiate the computation process for syndrome polynomial 1	*ELM_SYNDROME_FRAGMENT_6_i*[16] SYNDROME_VALID (where i = 1)	0x1
Initiate the computation process for syndrome polynomial 2	*ELM_SYNDROME_FRAGMENT_6_i*[16] SYNDROME_VALID (where i = 2)	0x1
Initiate the computation process for syndrome polynomial 3	*ELM_SYNDROME_FRAGMENT_6_i*[16] SYNDROME_VALID (where i = 3)	0x1
Wait until process is complete for syndrome polynomial 0, 1, 2, and 3: Wait until the interrupt is generated or poll the status register.		
Wait for page completed interrupt: All error locations are valid.	*ELM_IRQSTS*[8] PAGE_VALID	0x1
Read the process exit status for syndrome polynomial 0: All errors were successfully located.	*ELM_LOCATION_STS*[8] ECC_CORRECTABLE (where i = 0)	0x1
Read the process exit status for syndrome polynomial 1: All errors were successfully located.	*ELM_LOCATION_STS*[8] ECC_CORRECTABLE (where i = 1)	0x1
Read the process exit status for syndrome polynomial 2: All errors were successfully located.	*ELM_LOCATION_STS*[8] ECC_CORRECTABLE (where i = 2)	0x1
Read the process exit status for syndrome polynomial 3: All errors were successfully located.	*ELM_LOCATION_STS*[8] ECC_CORRECTABLE (where i = 3)	0x1
Read the number of errors for syndrome polynomial 0: four errors detected.	*ELM_LOCATION_STS*[4-0] ECC_NB_ERRORS (where i = 0)	0x4
Read the number of errors for syndrome polynomial 1: two errors detected.	*ELM_LOCATION_STS*[4-0] ECC_NB_ERRORS (where i = 1)	0x2
Read the number of errors for syndrome polynomial 2: one error detected.	*ELM_LOCATION_STS*[4-0] ECC_NB_ERRORS (where i = 2)	0x1
Read the number of errors for syndrome polynomial 3: no errors detected.	*ELM_LOCATION_STS*[4-0] ECC_NB_ERRORS (where i = 3)	0x0
Read the error-location bit addresses for syndrome polynomial 0 of the first four registers:	*ELM_ERROR_LOCATION_0_i* (where i = 0)	0x1FE
	ELM_ERROR_LOCATION_1_i (where i = 0)	0x617
	ELM_ERROR_LOCATION_2_i (where i = 0)	0x650
	ELM_ERROR_LOCATION_3_i (where i = 0)	0xA83
Read the error-location bit addresses for syndrome polynomial 1 of the first two registers:	*ELM_ERROR_LOCATION_0_i* (where i = 1)	0x4
	ELM_ERROR_LOCATION_1_i (where i = 1)	0x1036
Read the errors location bit addresses for syndrome polynomial 2 of the first registers:	*ELM_ERROR_LOCATION_0_i* (where i = 1)	0x3E8
Clear the *ELM_IRQSTS* register.	*ELM_IRQSTS*	0x1FF

Copyright © 2026 Texas Instruments Incorporated

12.3.5 Multi-Media Card Secure Digital (MMCSD) Interface

This section describes the MMCSD modules in the device.

12.3.5.1 MMCSD Overview

This device contains one or more MMCSD module instances. Each MMCSD instance includes one MMCSD Host Controller.

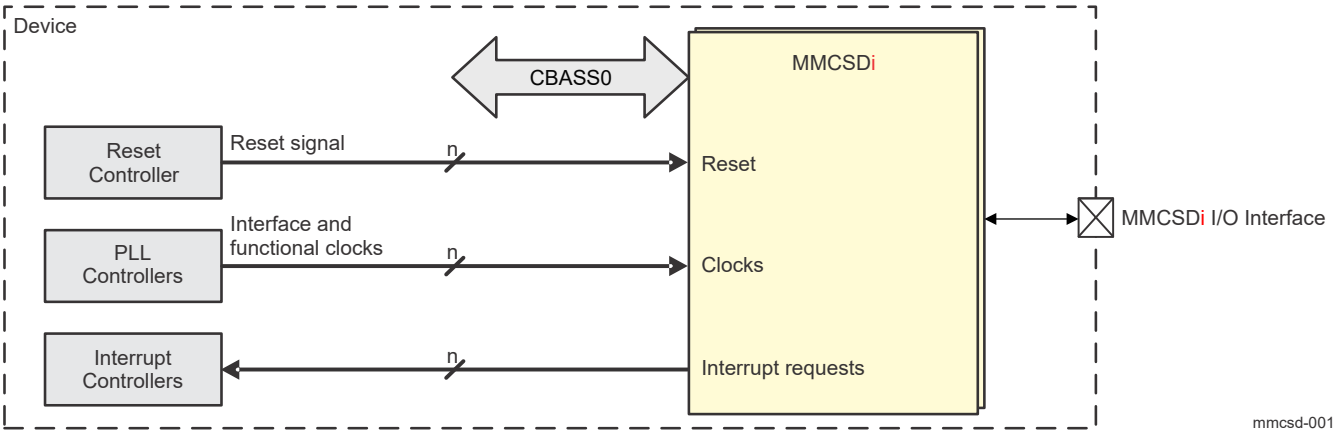
An instance supports either an 8-bit or 4-bit wide data bus.

A host controller can support eMMC and/or SD/SDIO. See the device specific datasheet for interfaces supported by the controller.

The MMCSD Host Controller provides an interface to eMMC 5.1 (embedded Multi-Media Card), SD 3.0 (Secure Digital), and SDIO 3.0 (Secure Digital IO) devices. The MMCSD Host Controller deals with MMC/SD/SDIO protocol at transmission level, data packing, adding cyclic redundancy checks (CRCs), start/end bit insertion, and checking for syntactical correctness.

The MMCSD Host Controller provides accessibility to external MMC/SD/SDIO devices using a Programmed IO method or DMA data transfer method. In programmed IO method, the device CPU transfers data using the Buffer Data Port register (MMCSDi_DATA_PORT). In DMA data transfer method, the MMCSD Host Controller can read or write memory without device CPU intervention.

Figure 12-197 shows the MMCSDi module overview (where i represents a valid instance of MMCSD in a domain. Consult the device specific datasheet for available domains and instances).



A. i represents a valid instance of MMCSD in a domain.

Figure 12-197. MMCSDi Module Overview

Copyright © 2026 Texas Instruments Incorporated

12.3.5.1.1 MMCSD Features

Table 12-229. MMCSD Features

Feature		MMCSD0 (8-bit)	MMCSD1/2 (4-bit)
Muxing of other LVCMOS interfaces onto the MMCSD interface at the SoC level (See the device specific data sheet for supported interfaces, if applicable)			✓
Controller			
eMMC5.1 Host Specification (JESD84-B51)		✓	✓
Integrated DMA controller supporting SD Advanced DMA - ADMA2 and ADMA3 (for more information about ADMA support, see Section 12.3.5.4.4, *Advanced DMA*)		✓	✓
System Bus Interface			
64-bit data width (host interface)		✓	✓
48-bit address		✓	✓
Clock asynchronous to eMMC clock (MMCi_CLK)		✓	
Clock asynchronous to eMMC/SD clock (MMCi_CLK)			✓
Little endian only		✓	✓
Configuration Bus Interface			
32-bit data width (target interface)		✓	✓
Linear incrementing addressing mode		✓	✓
32-bit aligned accesses only		✓	✓
Little endian only		✓	✓
Multi-Media Card Support			
eMMC Electrical Standard 5.1 (JESD84-B51)		✓	✓
Backward compatible with earlier eMMC standards		✓	✓
Legacy MMC SDR	8-bit bus width, 0-25 MHz, 25 MBps	1.8v	
	4-bit bus width, 0-25 MHz, 12.5 MBps		3.3/1.8v
	1-bit bus width, 0-25 MHz, 3.125 MBps		
High Speed SDR	8-bit bus width, 0-50 MHz, 50 MBps	1.8v	
	4-bit bus width, 0-50 MHz, 25 MBps		3.3/1.8v
	1-bit bus width, 0-50 MHz, 6.25 MBps		
High Speed DDR	8-bit bus width, 0-50 MHz, 100 MBps	1.8v	
	4-bit bus width, 0-50 MHz, 50 MBps		3.3/1.8v
HS200 SDR	0-200 MHz, 8-bit bus width, 200 MBps	1.8v	
	0-200 MHz, 4-bit bus width, 100 MBps		1.8v
HS400 DDR	0-200 MHz, 8-bit bus width, 400 MBps	1.8v	
Secure Digital Card Support			
SD Host Controller Standard Specification 4.10 and SD Physical Layer Specification v3.01		✓	✓
SDIO Specification v3.00		✓	✓
Backward compatible with earlier SD card specifications			✓
Default Speed mode	3.3 V signaling, frequency up to 25 MHz, up to 12.5 MBps		✓
High Speed mode	3.3 V signaling, frequency up to 50 MHz, up to 25 MBps		✓
SDR12	UHS-I 1.8 V signaling, frequency up to 25 MHz, up to 12.5 MBps	✓	✓
SDR25	UHS-I 1.8 V signaling, frequency up to 50 MHz, up to 25 MBps	✓	✓
SDR50	UHS-I 1.8 V signaling, frequency up to 100 MHz, up to 50 MBps	✓	✓
DDR50	UHS-I 1.8 V signaling, frequency up to 50 MHz, up to 50 MBps	✓	✓
SDR104	UHS-I 1.8 V signaling, frequency up to 200 MHz, up to 100 MBps	✓	✓

Copyright © 2026 Texas Instruments Incorporated

12.3.5.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

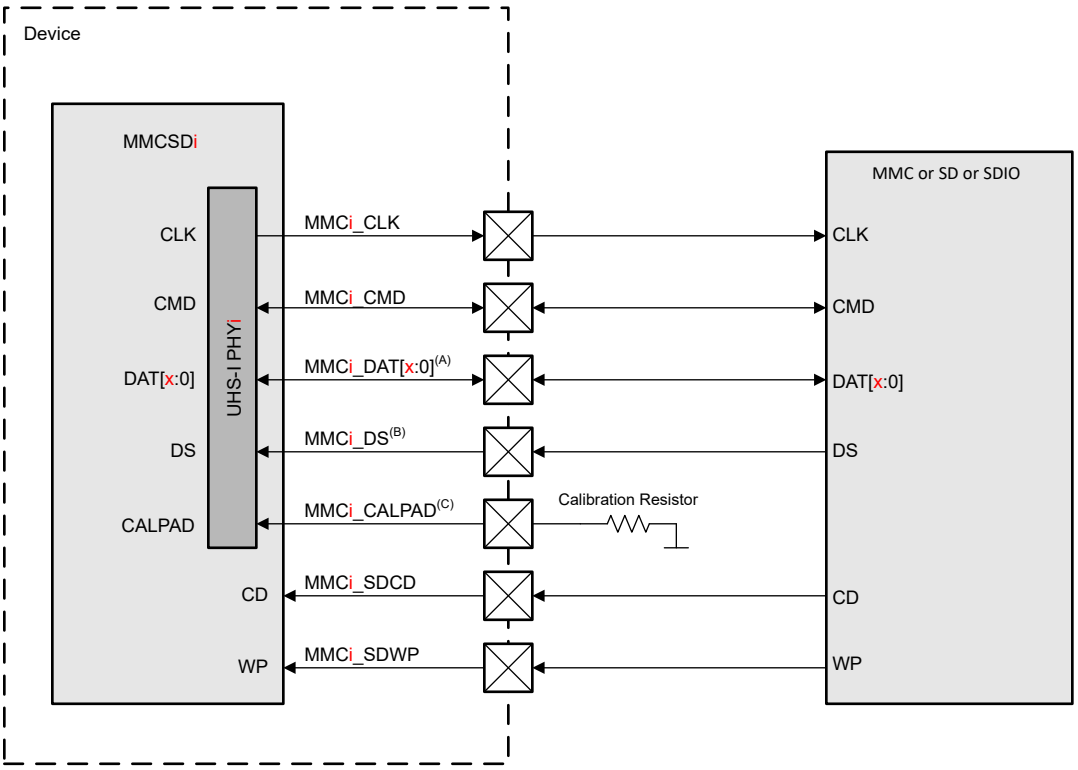
12.3.5.2 MMCSD Environment

The MMCSD instance is hereinafter referred to as MMCSDi module.

This section describes the MMCSDi external connections (environment).

The MMCSD (see the device specific datasheet for supported instances) can be used for connection to 1, 4, or 8-bit devices (dedicated for connection to eMMC devices). The MMCSD (see the device specific datasheet for supported instances) can be used for connection to 1 or 4-bit devices (dedicated for connection to SD or SDIO devices). For each MMCSD an integrated UHS-I PHY provides interface to external device.

Figure 12-198 shows the MMCSDi (where i represents a valid instance of MMCSD in a domain. See the device specific datasheet for available domains and MMCSD instances) connected to MMC, SD, or SDIO device.



A. x=7 for MMCSD (8-bit wide data bus); x=3 for MMCSD (4-bit wide data bus)
B. Used for HS400 mode (See the device specific datasheet for valid instances)
C. Used for PHY Calibration Resistor (See the device specific datasheet for valid instances)

Figure 12-198. MMCSDi Connected to MMC, SD, or SDIO Device

Copyright © 2026 Texas Instruments Incorporated

Table 12-230 describes the MMCSDi I/O signals.

Table 12-230. MMCSDi I/O Signals

Module Pin[4]	Device Level Signal	I/O[1]	Description
	MMCSDi[3]		
CLK	MMCi[3]_CLK	O	External Clock
CMD	MMCi[3]_CMD	I/O	Command Line
DAT[x:0][5]	MMCi[3]_DAT[x:0]	I/O	Data Signals
DS	MMCi[3]_DS	I	Data Strobe
CALPAD	MMCi[3]_CALPAD[2]	A	PHY Calibration Resistor
WP	MMCi[3]_SDWP	I	SD Card Write Protect
CD	MMCi[3]_SDCD	I	SD Card Detect

(1) I = Input; O = Output; HiZ = High Impedance; A = Analog
(2) An external 10 kΩ ±1% resistor must be connected between this pin and VSS. No external voltage should be applied to this pin.
(3) i represents an MMC instance. See the device datasheet for available domains and MMC instances.
(4) Some pins may not be pinned out or relevant to this device. See the device datasheet for specifics.
(5) x=7 for MMCSD (8-bit wide data bus); x=3 for MMCSD (4-bit wide data bus)

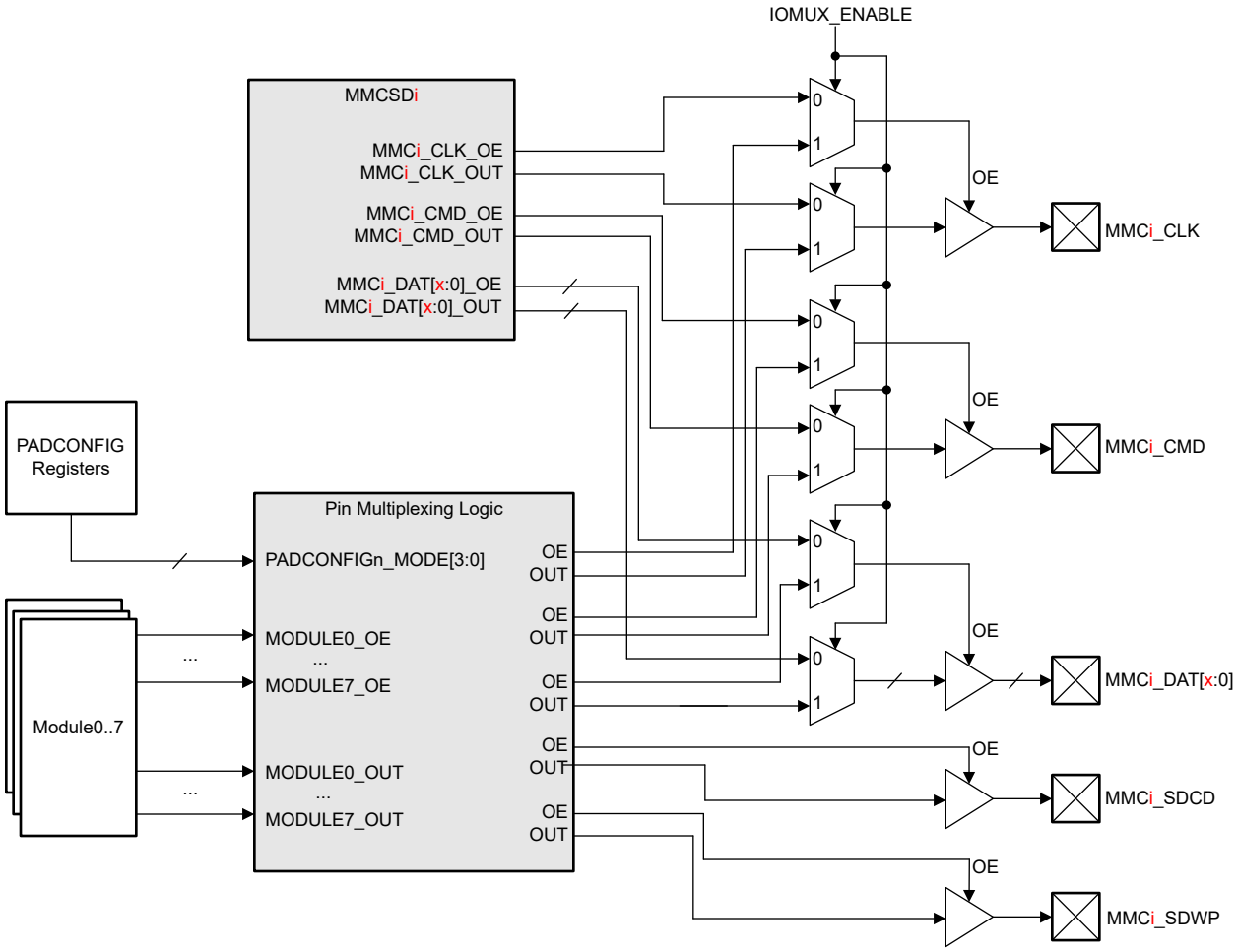
Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), refer to the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.2.1 MMCSD IO Mulitplexer

Figure 12-199 describes the IOMUX_ENABLE multilplexer, which is controlled by the MMC_SSCFG_PHY_CTRL_1_REG[31] IOMUX_ENABLE bit.



A. i represents a valid instance of 4-bit MMCSD only. See the device data sheet for specifics.
B. x=3 for MMCSD (4-bit wide data bus)
C. Only MMCi_CLK, MMCi_CMD and MMCi_DAT[n:0] are routed through the IOMUX_ENABLE multiplexer. MMCi_SDCD and MMCi_SDWP are not routed through the IOMUX_ENABLE multiplexer.
D. The IOMUX_ENABLE multiplexer does not affect the signal functions selected from the respective PADCONFIG register for the MMCi_SDCD or MMCi_SDWP pins.

Figure 12-199. MMCSD IOMUX_ENABLE Multiplexer

Copyright © 2026 Texas Instruments Incorporated

12.3.5.2.2 Protocol and Data Format

The bus protocol between the MMCSD Host Controller and the card is message-based. Each message is represented by one of the following parts:

- Command: A command starts an operation. The command is transferred serially from the MMCSD Host Controller to the card on the CMD line.
- Response: A response is an answer to a command. The response is sent from the card to the MMCSD Host Controller. It is transferred serially on the CMD line.
- Data: Data are transferred from the MMCSD Host Controller to the card or from the card to the MMCSD Host Controller using the data lines.
- Busy: The DAT[0] signal is maintained low by the card as far as it is programming the data received.
- CRC status: The CRC result is sent by the card through the DAT[0] line when executing a write transfer. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status on DAT[0]. In the case of successful transmission, over all active data lines, the card sends a positive CRC status on DAT[0] and starts the data programming procedure.

12.3.5.2.2.1 Protocol

There are two types of data transfer:

- Sequential operation
- Block-oriented operation

There are specific commands for each type of operation (sequential or block-oriented).

For information about commands and programming sequences supported by the MMC, SD, and SDIO devices, see the *Multi-Media Card System Specification*, *SD Memory Card Specifications*, and *SDIO Card Specification (Part E1)*.

Figure 12-200 and Figure 12-201 show how sequential operations are defined. Sequential operation is only for 1-bit transfer and initiates a continuous data stream. The transfer terminates when a stop command follows on the CMD line.

Note

Stream commands are supported only by MMCs.

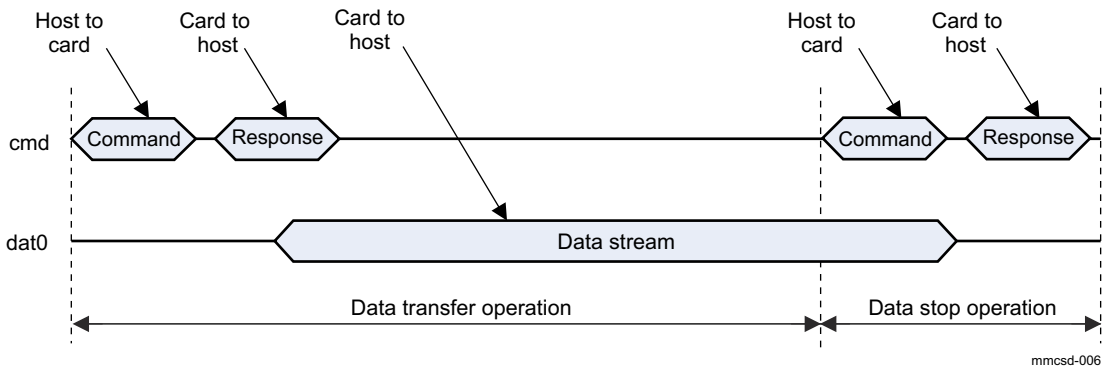


Figure 12-200. Sequential Read Operation (MMCs Only)

Copyright © 2026 Texas Instruments Incorporated

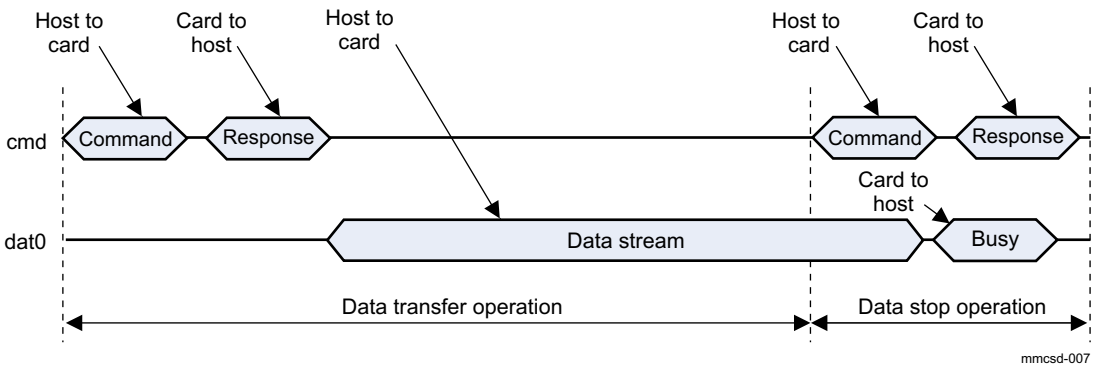


Figure 12-201. Sequential Write Operation (MMCs Only)

Figure 12-202 and Figure 12-203 show how multiple block-oriented operations are defined. A multiple block-oriented operation sends a data block plus CRC bits. The transfer terminates when a stop command follows on the CMD line. These operations are available for all kinds of devices (MMC, SD, SDIO).

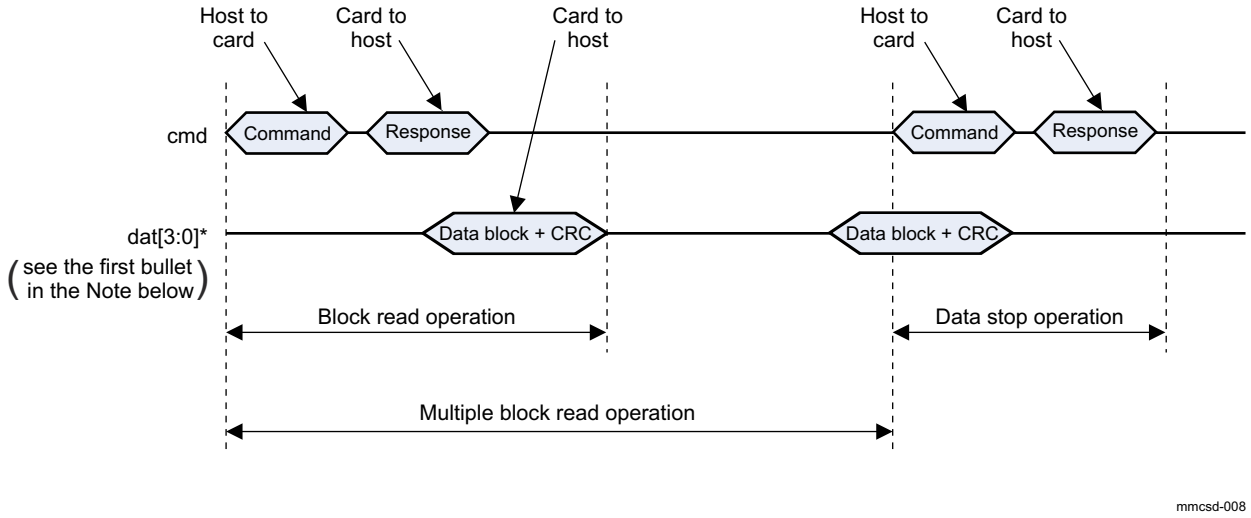


Figure 12-202. Multiple Block Read Operation

Copyright © 2026 Texas Instruments Incorporated

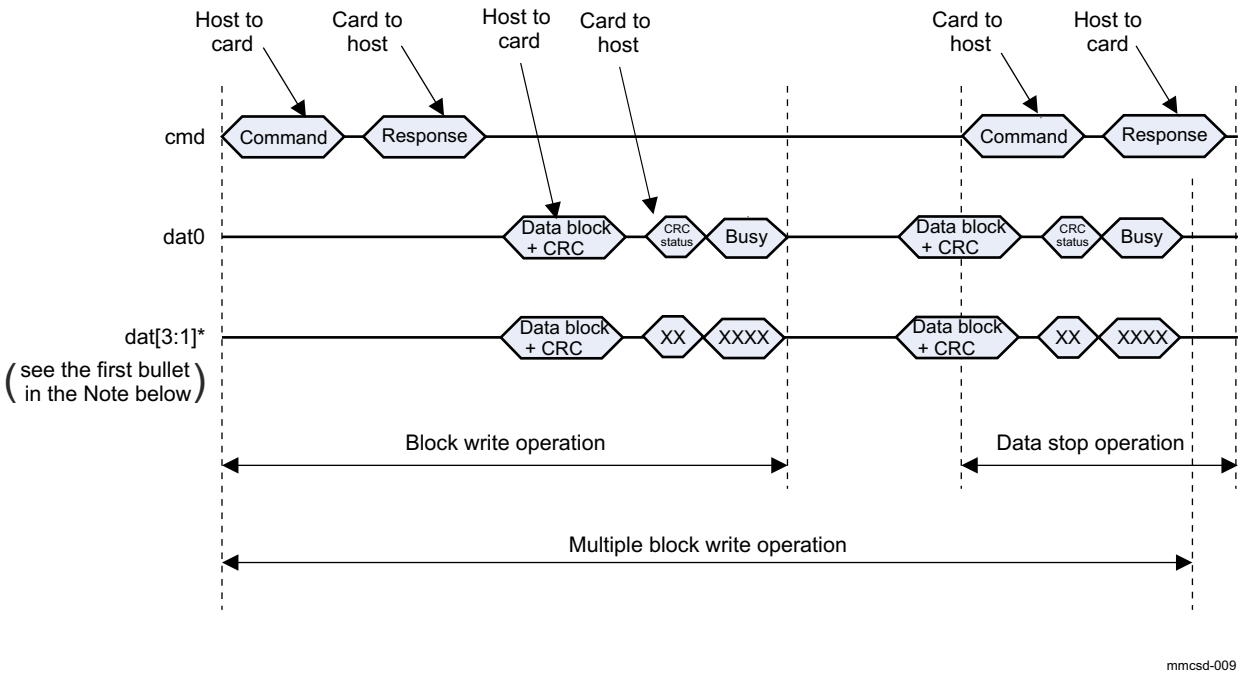


Figure 12-203. Multiple Block Write Operation With Card Busy Signal

Note

- The card busy signal is not always generated by the card. Refer to Figure 12-202 and Figure 12-203, that show a particular case.
- Software must perform a software reset (see MMCSD0_SOFTWARE_RESET / MMCSD1_SOFTWARE_RESET register) after a data time-out to ensure that MMCi_CLK is stopped.
- For multiblock transfer, and especially for MMC devices, a transfer can be aborted without using a stop command. If a CMD23 is used before data transfer to define the number of blocks that will be transferred, then the transfer stops automatically after the last block (if the MMCs supports this feature).

12.3.5.2.2.2 Data Format

12.3.5.2.2.2.1 Coding Scheme for Command Token

Command tokens always start with 0 and end with 1. The second bit is a transmitter bit: 1 for a host command. The content is the command index (coded by 6 bits) and an argument (for example, an address), coded by 32 bits. The content is protected by 7-bit CRC checksum (see Figure 12-204).

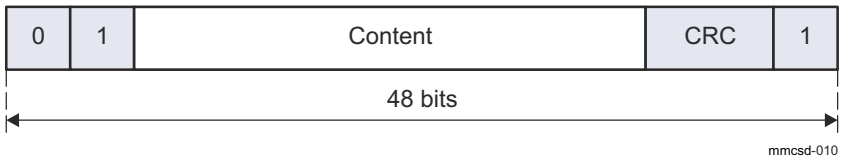


Figure 12-204. Command Token Format

12.3.5.2.2.2.2 Coding Scheme for Response Token

Response tokens always start with 0 and end with 1. The second bit is a transmitter bit: 0 for a card response. The content is different for each type of response (R1, R2, R3, R4, and R5, R6, R7 [for SD]) and the content is protected by 7-bit CRC checksum (see Figure 12-205 and Figure 12-206). Depending on the type of commands sent to the card, the MMCSD0_COMMAND / MMCSD1_COMMAND register must be configured differently to

Copyright © 2026 Texas Instruments Incorporated

avoid false CRC or index errors to be flagged on command response (see Table 12-231). For more information about response types, see the *Multi-Media Card System Specification*, *SD Memory Card Specifications*, and *SDIO Card Specification (Part E1*).

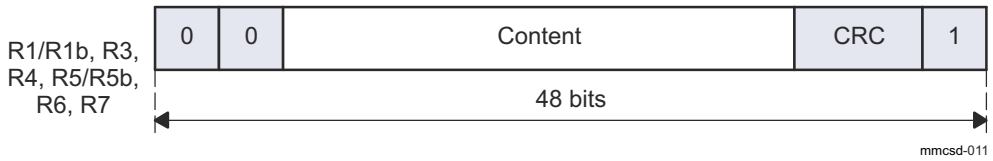


Figure 12-205. Response Token Format (R1/R1b, R3, R4, R5/R5b, R6, R7)

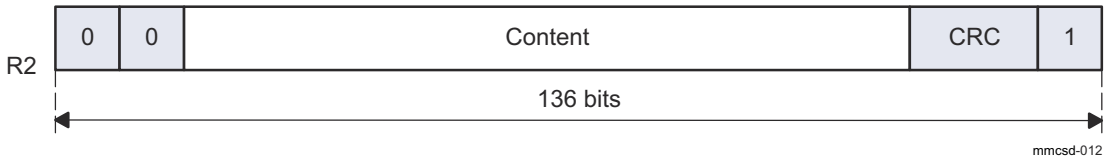


Figure 12-206. Response Token Format (R2)

Table 12-231. Relationship Between Configuration and Name of Response Type

Response Type MMCSD0_COMMAND[1-0] RESP_TYPE_SEL / MMCSD1_COMMAND[1-0] RESP_TYPE_SEL	Index Check Enable MMCSD0_COMMAND[4] CMD_INDEX_CHK_ENA / MMCSD1_COMMAND[4] CMD_INDEX_CHK_ENA	CRC Check Enable MMCSD0_COMMAND[3] CMD_CRC_CHK_ENA / MMCSD1_COMMAND[3] CMD_CRC_CHK_ENA	Name of Response Type
00	0	0	No response
01	0	1	R2
10	0	0	R3 (R4 for SD cards)
10	1	1	R1, R6, R5, (R7 for SD cards)
11	1	1	R1b, R5b

12.3.5.2.2.3 Coding Scheme for Data Token

Data tokens always start with 0 and end with 1 (see Figure 12-207 through Figure 12-209).

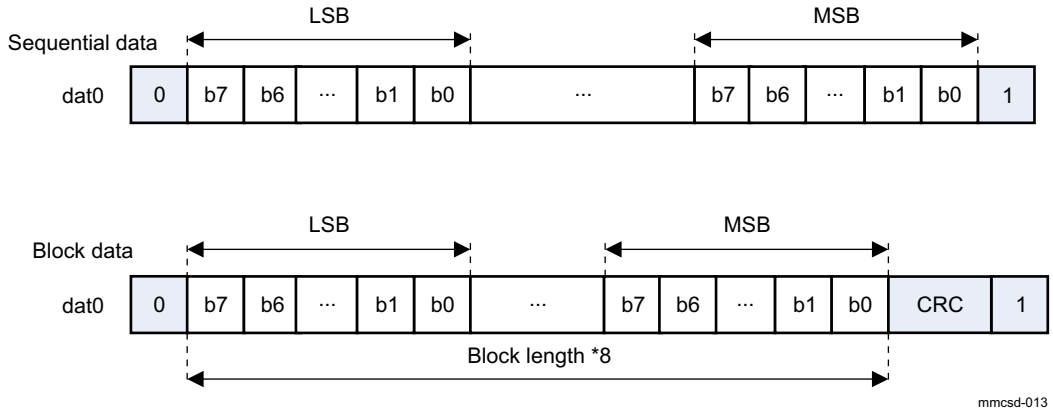


Figure 12-207. Data Token Format for 1-Bit Transfers

Copyright © 2026 Texas Instruments Incorporated

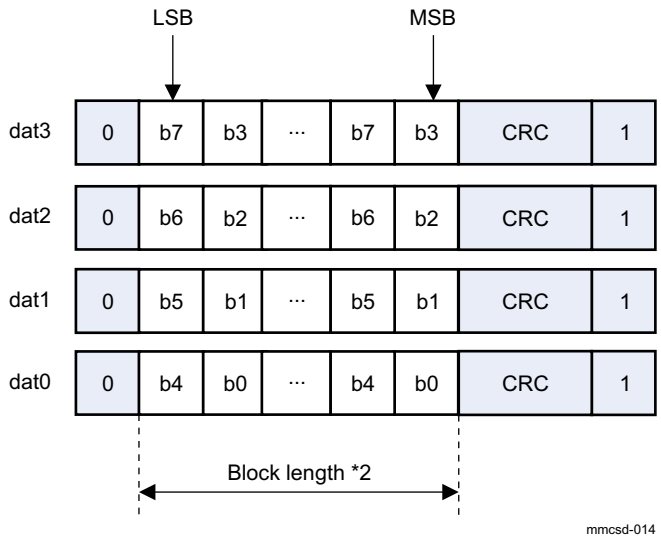


Figure 12-208. Data Token Format for 4-Bit Transfers

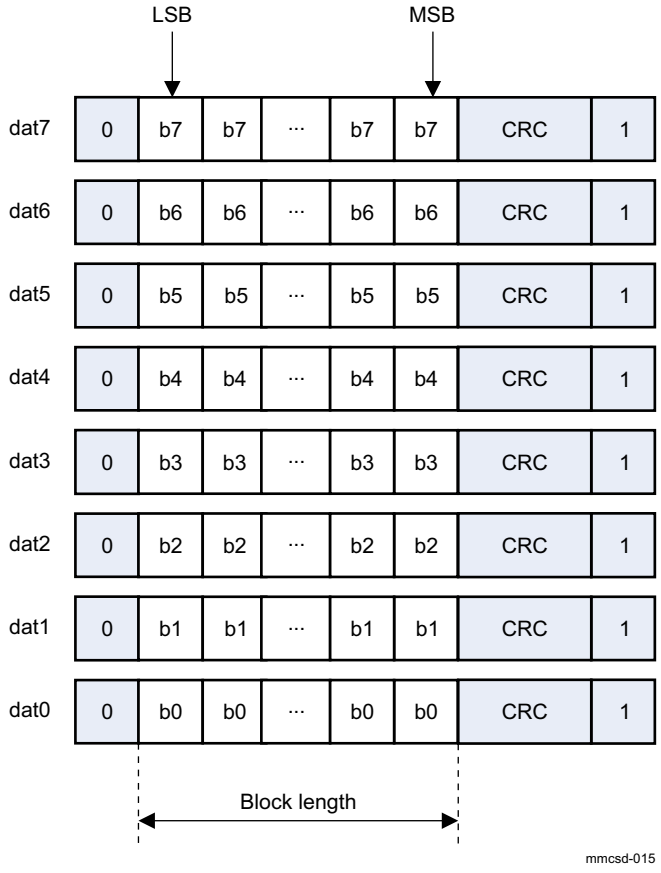


Figure 12-209. Data Token Format for 8-Bit Transfers

12.3.5.3 Integration

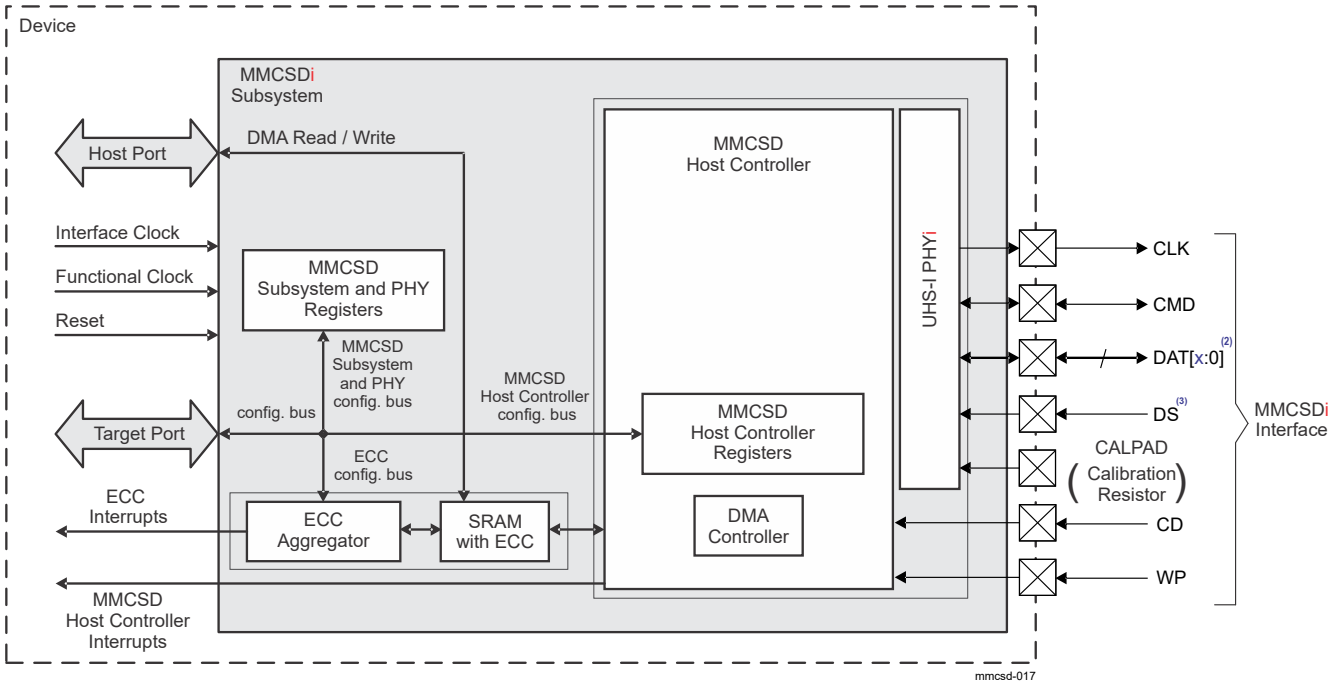
See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.4 MMCSD Functional Description

12.3.5.4.1 Block Diagram

Figure 12-210 shows the MMCSDi module block diagram (where i represents a valid instance of MMCSD in a domain).



A. (1) i represents a valid instance of MMCSD in a domain. Consult the device specific datasheet for available domains and MMCSD instances.

B. (2) x = the maximum number of available DAT signals for an instance of MMCSD in a domain.

C. (3) Used for HS200 mode (Consult the datasheet for supported devices).

Figure 12-210. MMCSDi Module Block Diagram

Basic Blocks:

- MMCSD Host Controller: The MMCSD Host Controller is situated in the MMCSD Subsystem and provides accessibility to external MMC/SD/SDIO devices using a Programmed IO method or DMA data transfer method.
- UHS-I PHY: The integrated UHS-I PHY provides an interface between the MMCSD Host Controller and external MMC/SD/SDIO devices.
- MMCSDi Interface: The MMCSDi Interface includes all used interface pins (for more information, see Table 12-230).
- Host Port: Provides a 64-bit wide read/write interface between the device and the MMCSD Subsystem internal SRAM.
- Target Port: Provides a 32-bit wide interface between the device and the MMCSD Subsystem and MMCSD Host Controller parts.
- MMCSD Host Controller Registers: This block includes set of all MMCSD Host Controller registers.
- MMCSD Subsystem and PHY Registers: This block implements memory-mapped registers at the MMCSD Subsystem level and memory-mapped registers to control and program the MMCSD PHY.
- ECC Aggregator: The ECC Aggregator block facilitates aggregating and reporting internal SRAM ECC errors (for more information, see *ECC Support*).
- Internal SRAM with ECC: The internal SRAM block is used for data storage during read/write transactions.
- DMA Controller: Manages the data transfer between the device memory and the MMCSD Subsystem internal SRAM.

Copyright © 2026 Texas Instruments Incorporated

- System Clocks: There are asynchronous relationship between the interface (system) clock and functional clock (for more information, see *MMCSD Clocks*).
- System Reset: The reset to the MMCSD Subsystem provides reset to the all MMCSD Subsystem parts (for more information, see *MMCSD Clocks*).
- Interrupts: The MMCSD Subsystem sources one MMCSD Host Controller interrupt and four ECC Aggregator interrupts (for more information, see *MMCSD Hardware Requests* and *Interrupt Requests*).

For more information about all MMCSD registers, see *MMCSD Registers*.

The MMCSD Host Controller can use a Programmed IO method (PIO) or DMA data transfer method to access external MMC/SD/SDIO devices.

The target port is used for device CPU access to the MMCSD Host Controller register set. The MMCSD Host Controller register set provides the communication between the device CPU and the MMCSD Host Controller. In PIO method the device CPU transfers data using the MMCSD0_DATA_PORT / MMCSD1_DATA_PORT register. The data flow from the device memory to the external MMC/SD/SDIO device (and vice versa) is through the target port, the MMCSD0_DATA_PORT / MMCSD1_DATA_PORT register in the MMCSD Host Controller, and the PHY.

The host port is used for connection to the DMA controller (for more information about ADMA support, see *Advanced DMA* and MMCSD0_CAPABILITIES / MMCSD1_CAPABILITIES register). In DMA data transfer method the DMA controller uses the host port to transfer data between the device memory and the MMCSD Subsystem internal SRAM. The other side of the internal SRAM is connected to the MMCSD Host Controller. The data flow from the device memory to the external MMC/SD/SDIO device (and vice versa) is through the host port, internal SRAM, MMCSD Host Controller, and the PHY. The DMA controller manages the read/write operations from/to the internal SRAM without device CPU intervention.

12.3.5.4.2 Interrupt Requests

Each MMCSD instance sources one active high level MMCSD Host Controller interrupt and four active high level ECC Aggregator interrupts (see *Module Integration*). The MMCSD Host Controller interrupt is generated based on the bit values in the MMCSD0_NORMAL_INTR_STS / MMCSD1_NORMAL_INTR_STS and MMCSD0_NORMAL_INTR_STS_ENA / MMCSD1_NORMAL_INTR_STS_ENA registers. The ECC Aggregator interrupts are generated based on the ECC errors - single and double bit errors (for more information about ECC Aggregator interrupts, see *ECC Support*).

12.3.5.4.3 ECC Support

The Error Correcting Code (ECC) is a mechanism for providing increased system reliability via reduction of memory software errors by allowing single bit errors to be detected and corrected (SEC) and double bit errors to be detected (DED).

There are two SRAM memories (transmit SRAM and receive SRAM) within each MMCSD instance. These two memories are ECC protected.

The SEC logic detects and corrects a single bit error (single bit error per ECC word or per ECC data segment). The DED logic only detects (does not correct) double errors (double bit errors per ECC word or per ECC data segment).

12.3.5.4.3.1 ECC Aggregator

There are two SRAMs (each with size 4 KB - 512 words × 64-bit) in each MMCSD Subsystem. One SRAM dedicated for transmit and one SRAM dedicated for receive operations.

Figure 12-211 shows the ECC Aggregator block diagram.

Copyright © 2026 Texas Instruments Incorporated

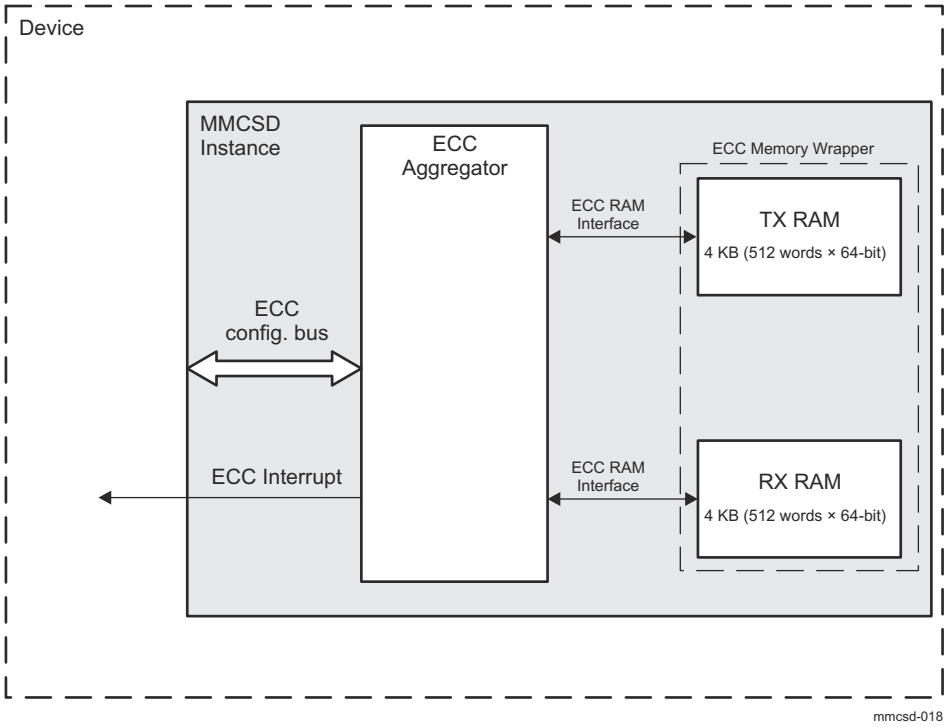


Figure 12-211. ECC Aggregator Block Diagram

Note

For more information about ECC Aggregator Registers, refer to *MMCSD Registers*.

12.3.5.4.4 Advanced DMA

The MMCSD modules support SD Advanced DMA – ADMA2 and ADMA3.

Note

For more information, refer to the SD Association specification titled *"SD Host Controller Simplified Specification, Part A2, Version 4.10"*.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5 MMCSD Programming Guide

12.3.5.5.1 Sequences

This section defines basic sequence flow chart divided into several sub sequences.

Note

UHSII is not supported. For more information, see *Not Supported Features*.

"Wait for interrupts" is used in the flow chart. This means the Host Driver waits until specified interrupts are asserted. If already asserted, then fall through that step in the flow chart. Timeout checking shall be always required to detect no interrupt generated but this is not described in the flow chart.

This specification uses the double box like Figure 12-212, (the step (1) in Figure 12-216), it means that the other flows, which already are shown, shall be performed. Therefore, the interrupt may be included in the other flows.



mmcsd-019

Figure 12-212. Double Box Notation

12.3.5.5.1.1 SD Card Detection

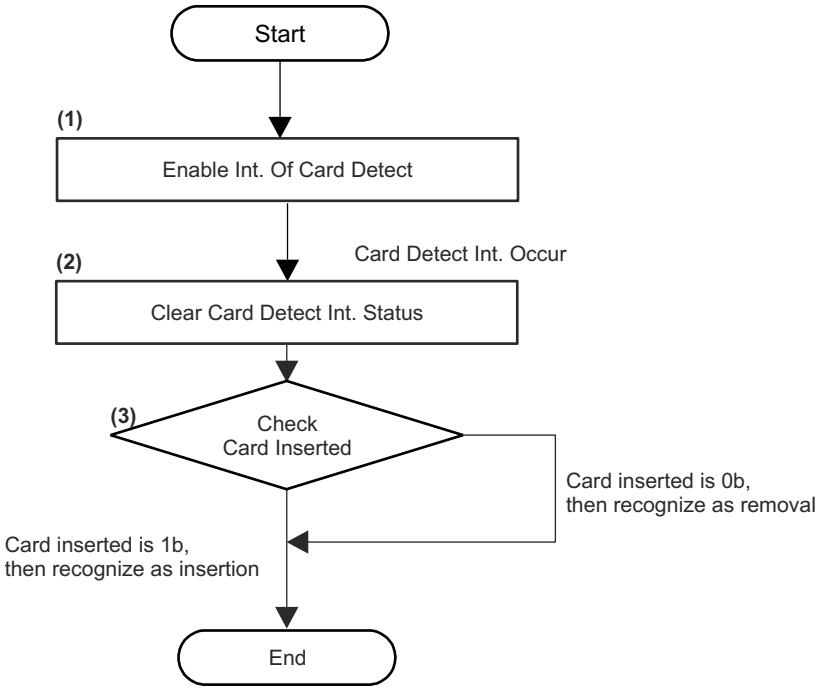


Figure 12-213. SD Card Detect Sequence

The flow chart for detecting a SD card is shown in Figure 12-213.

Each step is executed as follows:

To enable interrupt for card detection, write 1 to the following bits:

- MMCSD0_NORMAL_INTR_STS_ENA[6] CARD_INSERTION
- MMCSD0_NORMAL_INTR_SIG_ENA[6] CARD_INSERTION

Copyright © 2026 Texas Instruments Incorporated

- MMCSD0_NORMAL_INTR_STS_ENA[7] CARD_REMOVAL
- MMCSD0_NORMAL_INTR_SIG_ENA[7] CARD_REMOVAL

(1) When the Host Driver detects the card insertion or removal, clear its interrupt statuses. If Card Insertion interrupt is generated, write 1 to MMCSD0_NORMAL_INTR_STS_ENA[6] CARD_INSERTION bit. If Card Removal interrupt is generated, write 1 to MMCSD0_NORMAL_INTR_STS_ENA[7] CARD_REMOVAL bit.

(2) Check MMCSD0_PRESENTSTATE[16] CARD_INSERTED bit. In the case where MMCSD0_PRESENTSTATE[16] CARD_INSERTED bit is 1, the Host Driver can supply the power and the clock to the SD card. In the case where MMCSD0_PRESENTSTATE[16] CARD_INSERTED bit is 0, the other executing processes of the Host Driver shall be immediately closed.

If miniSD adaptor is used for standard SD slot and miniSD card is inserted or extracted from the adaptor, card detect interrupt may not be generated. When host does not receive response to any commands, miniSD card is extracted or in idle state, the host should try to re-initialize the card. In this case, all card information shall be re-loaded.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.2 SD Clock Control

12.3.5.5.1.2.1 Internal Clock Setup Sequence

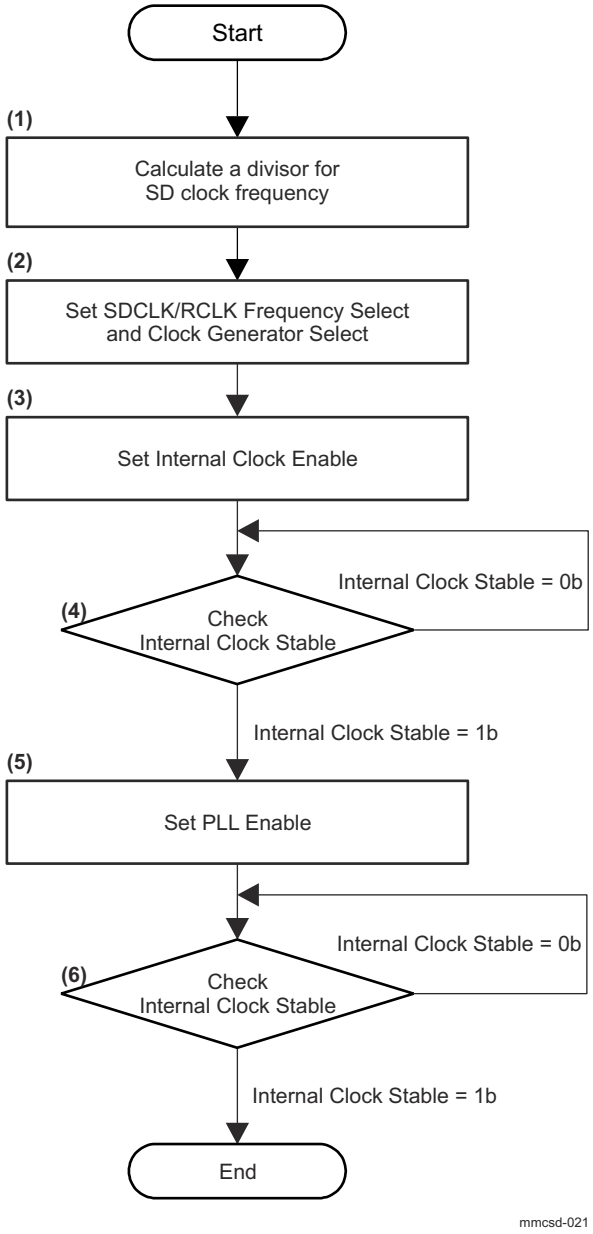


Figure 12-214. Internal Clock Setup Sequence

The sequence for supplying SD Clock to a SD card is described in Figure 12-214. From Version 4.10, MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit is added. This sequence is also applicable to prior versions which do not support MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit.

(1) Calculate a divisor to determine SD Clock frequency for legacy IF or RCLK frequency for UHS-II IF by reading MMCSD0_CAPABILITIES[15-8] BASE_CLK_FREQ and MMCSD0_CAPABILITIES[55-48] CLOCK_MULTIPLIER bit fields. If non-zero value is set to MMCSD0_CAPABILITIES[55-48] CLOCK_MULTIPLIER bit field, Programmable Clock Mode can be used (for more information, see MMCSD0_CLOCK_CONTROL[5] CLKGEN_SEL bit). If MMCSD0_CAPABILITIES[15-8] BASE_CLK_FREQ bit field is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

Copyright © 2026 Texas Instruments Incorporated

(2) Set MMCSD0_CLOCK_CONTROL[15-8] SDCLK_FRQSEL bit field and MMCSD0_CLOCK_CONTROL[5] CLKGEN_SEL bit in accordance with the calculated result of step (1).

If MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit is set, these bits are set by Host Controller automatically as specified in the Preset Value register (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10).

(3) Set MMCSD0_CLOCK_CONTROL[0] INT_CLK_ENA bit.

(4) Check MMCSD0_CLOCK_CONTROL[1] INT_CLK_STABLE bit. Repeat this step until this status is 1. Clock will be stable in shorter time but timeout of this loop is defined as 150 ms.

(5) Set MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit. This step does not affect Host Controllers which do not support MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit.

(6) If MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit is supported, PLL locked may be checked by this status (if MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit is not supported, this status is supposed to indicate 1 by step (3)). Clock will be stable in shorter time but timeout of this loop is defined as 150 ms.

12.3.5.5.1.2.2 SD Clock Supply and Stop Sequence

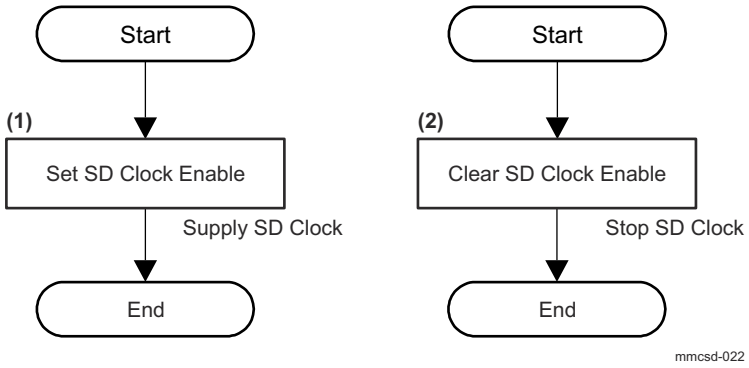


Figure 12-215. SD Clock Supply and Stop Sequence

The flow chart for stopping the SD Clock is shown in Figure 12-215. The Host Driver shall not stop the SD Clock when a SD transaction is occurring on the SD Bus -- namely, when either MMCSD0_PRESENTSTATE[1] INHIBIT_DAT or MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit is set to 1.

(1) Setup Internal Clock (see Figure 12-214). Then set MMCSD0_CLOCK_CONTROL[2] SD_CLK_ENA bit to 1. Then, the Host Controller starts supplying the SD Clock.

(2) Set MMCSD0_CLOCK_CONTROL[2] SD_CLK_ENA bit to 0. Then, the Host Controller stops supplying the SD Clock. Internal Clock is still oscillating.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.2.3 SD Clock Frequency Change Sequence

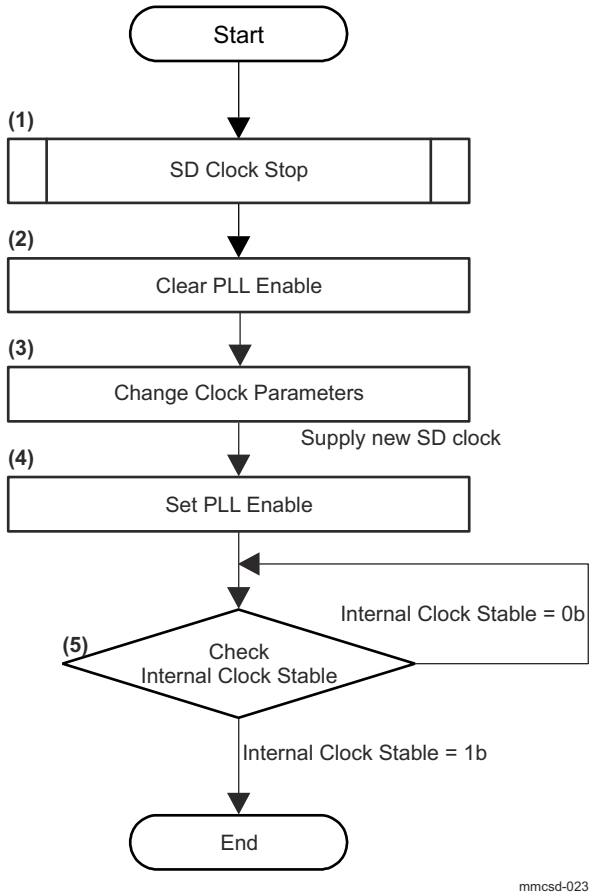


Figure 12-216. SD Clock Change Sequence

The sequence for changing SD Clock frequency is shown in Figure 12-216.

(1) Execute the SD Clock Stop Sequence (see Figure 12-215). If SD Clock supply to card is already stopped, skip this step.

(2) Clear MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit to 0. If MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit is not supported, this step has no effect and may be skipped.

(3) When MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit is set to 0, change clock parameters in the MMCSD0_CLOCK_CONTROL register. When MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit is set to 1, select Bus Speed Mode as described.

(4) Set MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit to 1. If MMCSD0_CLOCK_CONTROL[3] PLL_ENA bit is not supported, this step this step has no effect and may be skipped.

(5) Wait until MMCSD0_CLOCK_CONTROL[1] INT_CLK_STABLE bit is set to 1. Clock will be stable in shorter time but timeout of this loop is defined as 150 ms. SD Clock Supply Sequence is required to provide clock to device (see Figure 12-215).

12.3.5.5.1.3 SD Bus Power Control

The sequence for controlling the SD Bus Power is described in Figure 12-217.

Copyright © 2026 Texas Instruments Incorporated

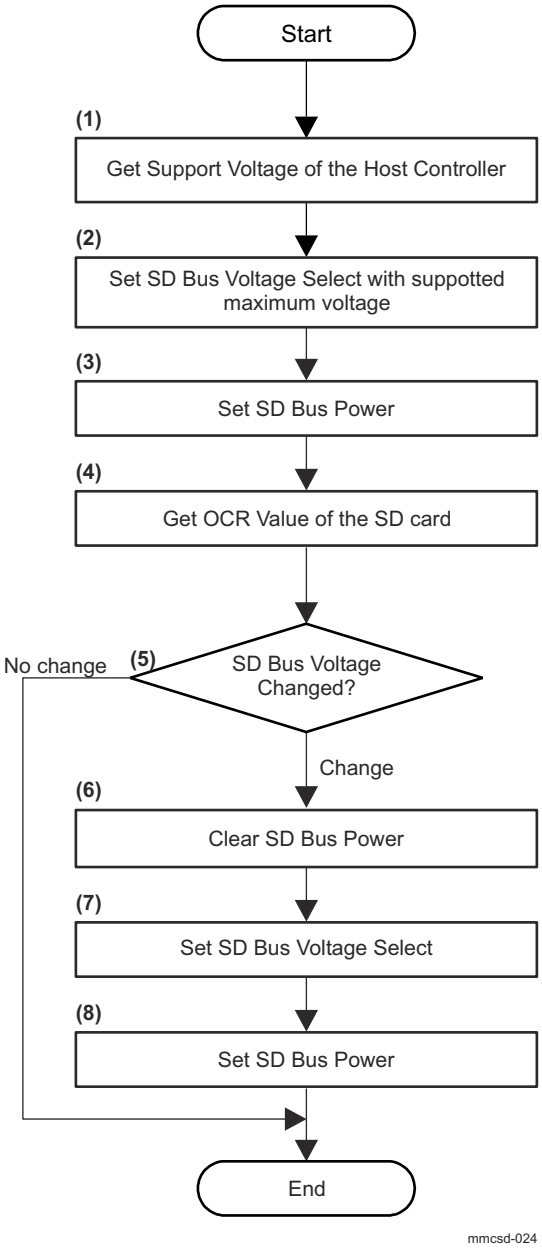


Figure 12-217. SD Bus Power Control Sequence

(1) By reading the MMCSD0_CAPABILITIES register, get the support voltage of the Host Controller.

(2) Set MMCSD0_POWER_CONTROL[3-1] SD_BUS_VOLTAGE bit field with maximum voltage that the Host Controller supports.

(3) Set MMCSD0_POWER_CONTROL[0] SD_BUS_POWER bit to 1.

(4) Get the OCR value of all function internal of SD card.

(5) Judge whether SD Bus voltage needs to be changed or not. In case where SD Bus voltage needs to be changed, go to step (6). In case where SD Bus voltage does not need to be changed, go to "End".

(6) Set MMCSD0_POWER_CONTROL[0] SD_BUS_POWER bit to 0 for clearing this bit. The card requires voltage rising from 0 volt to detect it correctly. The Host Driver shall

Copyright © 2026 Texas Instruments Incorporated

clear MMCSD0_POWER_CONTROL[0] SD_BUS_POWER bit before changing voltage by setting MMCSD0_POWER_CONTROL[3-1] SD_BUS_VOLTAGE bit field.

(7) Set MMCSD0_POWER_CONTROL[3-1] SD_BUS_VOLTAGE bit field.

(8) Set MMCSD0_POWER_CONTROL[0] SD_BUS_POWER to 1.

Note: Step (2) and step (3) can be executed at same time. And also, step (7) and step (8) can be executed at same time.

12.3.5.5.1.4 Changing Bus Width

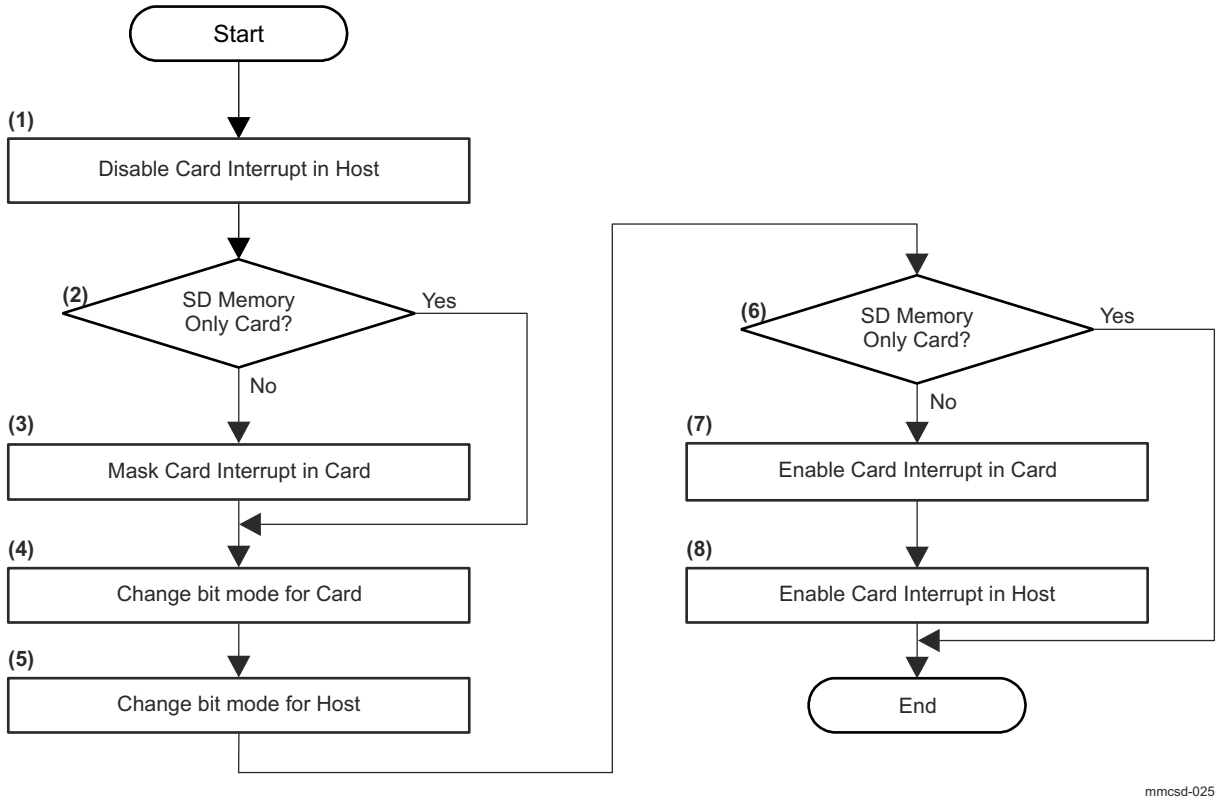


Figure 12-218. Change Bus Width Sequence

The sequence for changing bit mode on SD Bus is shown in Figure 12-218.

(1) Set MMCSD0_NORMAL_INTR_STS_ENA[8] CARD_INTERRUPT bit to 0 for masking incorrect interrupts that may occur while changing the bus width.

(2) In case of SD memory only card, go to step (4). In case of other card, go to step (3).

(3) Set "IENM" of the CCCR in a SDIO or SD combo card to 0 by CMD52.

(4) Change the bus width mode for a SD card. SD Memory Card bus width is changed by ACMD6 and SDIO card bus width is changed by setting Bus Width of Bus Interface Control register in CCCR.

(5) In case of changing to 4-bit mode, set MMCSD0_HOST_CONTROL1[1] DATA_WIDTH bit to 1. In another case (1-bit mode), set this bit to 0.

(6) In case of SD memory only card, go to the "End". In case of other card, go to step (7).

(7) Set "IENM" of the CCCR in a SDIO or SD combo card to 1 by CMD52.

(8) Set MMCSD0_NORMAL_INTR_STS_ENA[8] CARD_INTERRUPT bit to 1.

Copyright © 2026 Texas Instruments Incorporated

Note that if the card is locked, bus width cannot be changed. Unlock the card is required before changing bus width.

12.3.5.5.1.5 Timeout Setting on DAT Line

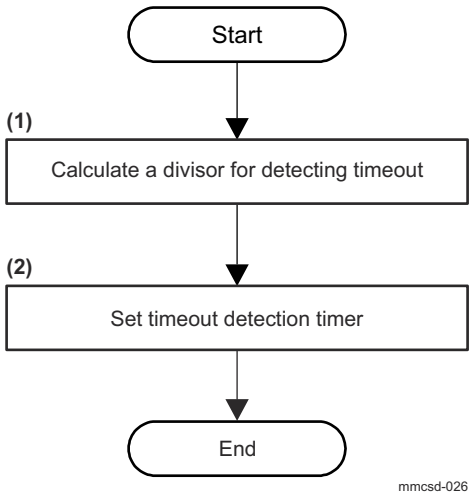


Figure 12-219. Timeout Setting Sequence

In order to detect timeout errors on DAT line, the Host Driver shall execute the following two steps before any SD transaction. For more information regarding SD transactions:

(1) Calculate a divisor to detect timeout errors by reading MMCSD0_CAPABILITIES[5-0] TIMEOUT_CLK_FREQ bit field and MMCSD0_CAPABILITIES[7] TIMEOUT_CLK_UNIT bit. If MMCSD0_CAPABILITIES[5-0] TIMEOUT_CLK_FREQ bit field is 00 0000b, the Host System shall provide this information to the Host Driver by another method.

(2) Set MMCSD0_TIMEOUT_CONTROL[3-0] COUNTER_VALUE bit field in accordance with the value from step (1) above.

12.3.5.5.1.6 Card Initialization and Identification (for SD I/F)

Figure 12-220 and Figure 12-221 shows initialization and card identification sequence for the Standard Capacity SD Memory Card (SDSC), the High Capacity SD Memory Card (SDHC) and the Extended Capacity SD Memory Card (SDXC) that was based on the Physical Layer Version 3.01. Refer to the latest sequence.

Copyright © 2026 Texas Instruments Incorporated

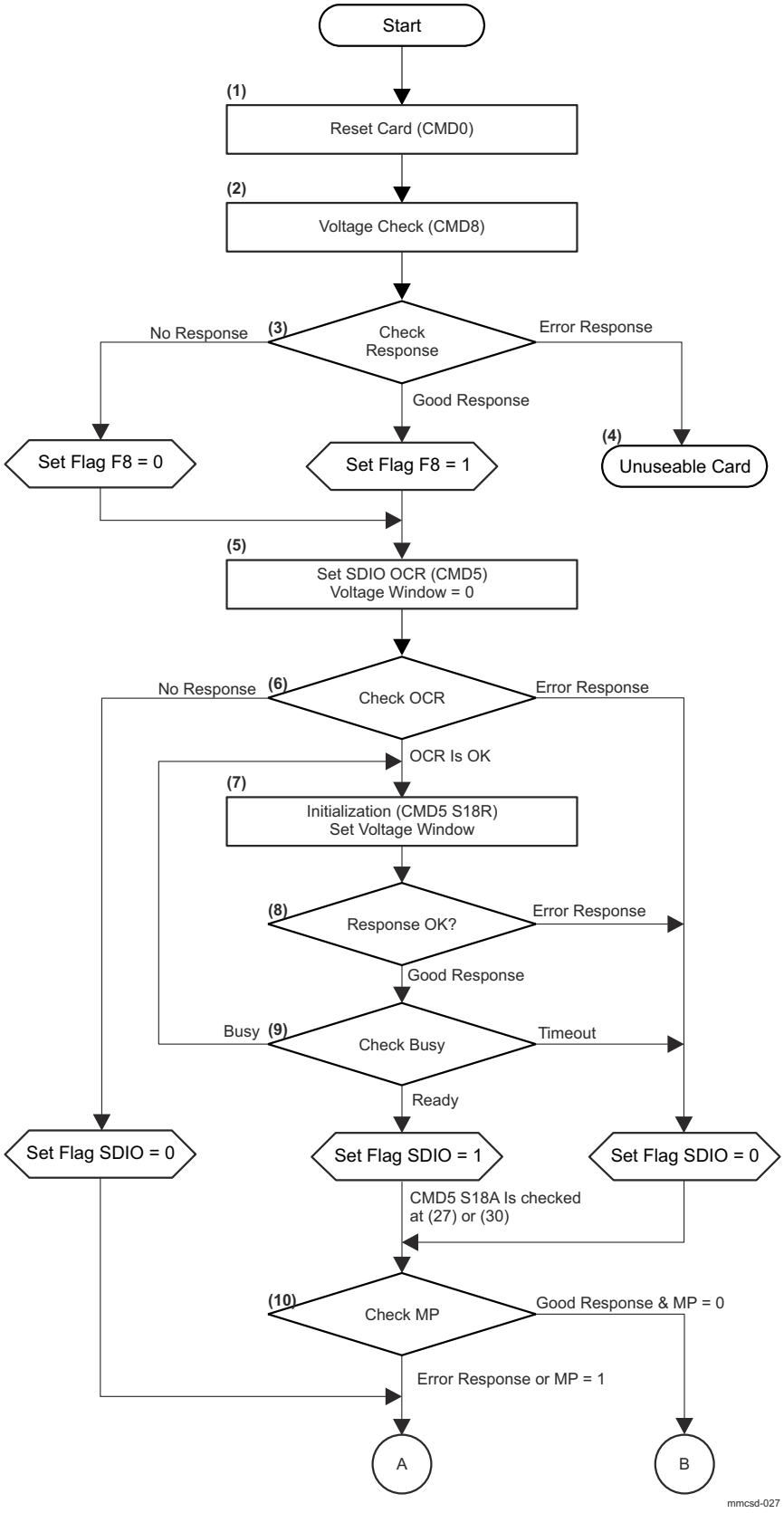


Figure 12-220. Card Initialization and Identification (1)

Copyright © 2026 Texas Instruments Incorporated

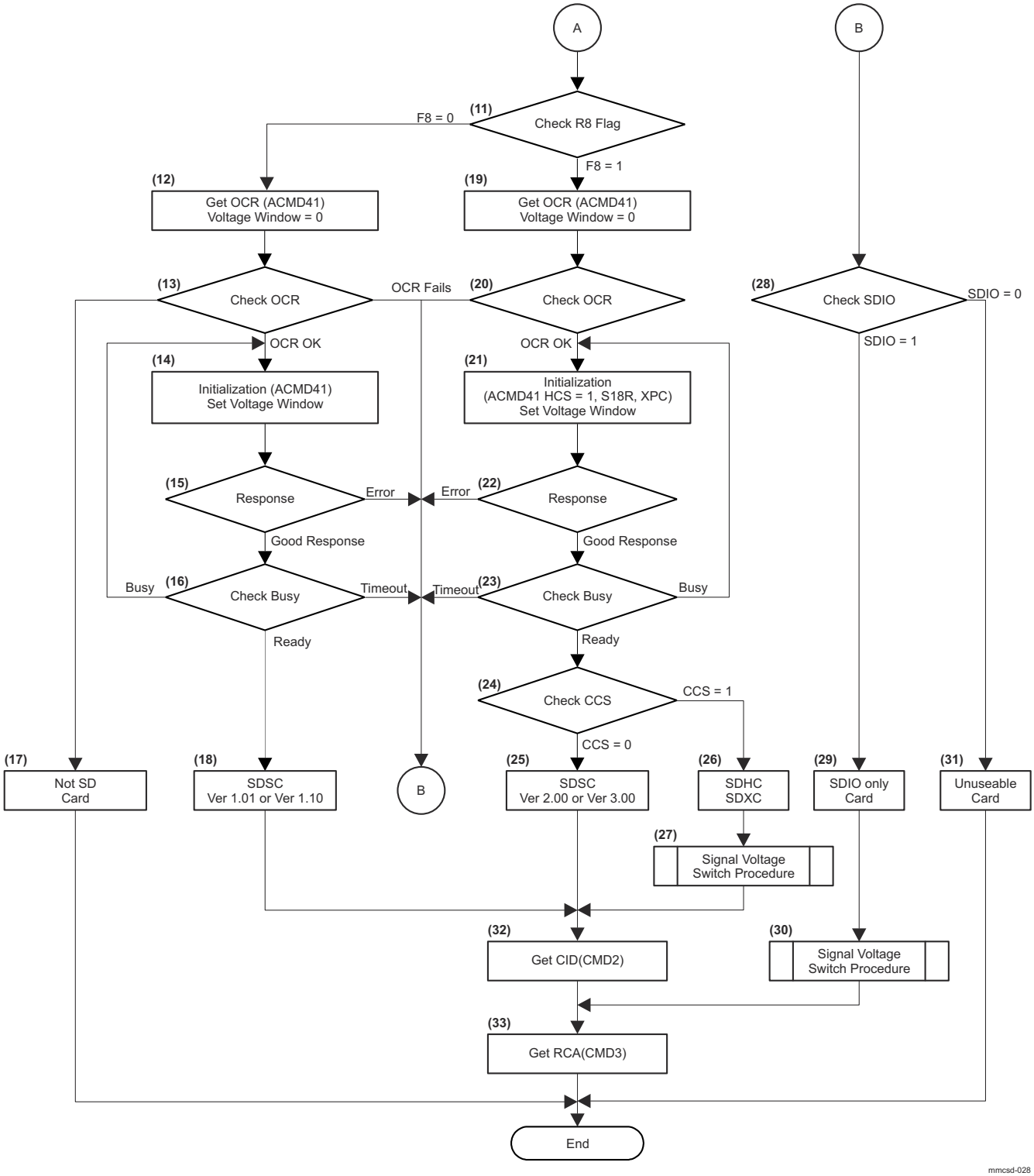


Figure 12-221. Card Initialization and Identification (2)

(1) SD Bus mode is selected by CMD0 (Keep Pin 1 to high during CMD0 execution).

(2) New CMD8 shall be issued after CMD0 to support High Capacity SD Memory Card.

Copyright © 2026 Texas Instruments Incorporated

(3) Voltage check command enables the Hosts to support future low voltage specification. However, at this time, only one voltage is defined. Legacy cards and Not SD cards do not respond to CMD8. In this case, set F8 to 0 (F8 is CMD8 valid flag used in step (11)) and go to Step (5). Only Version 2.00 or higher cards can respond to CMD8. The host needs to check whether CRC of the response is valid and whether VHS and check pattern in the argument are equal to VCA and check pattern in the response. Passing all these checks results in CMD8 response OK. In this case, set F8 to 1 and go to step (5). If one of the checks is failed, go to step (4).

(4) Initialization is stopped by CMD8 fails. The Host Driver should retry step (1) to (3) one more time (this is not described in the figure).

(5) SDIO OCR is available by issuing CMD5 with setting voltage window (bit 23 to 0) in the argument to 0. SDIO initialization is not started.

(6) No response means the card does not have SDIO function. Set SDIO flag to 0 and go to step (11). If the card responds to CMD5 and the response is OK, go to step (7). If the response is error, set SDIO flag to 0 and go to step (10). SDIO flag indicates whether SDIO functions are initialized or not.

(7) The SDIO portion starts initialization by CMD5 with setting the supply voltage to the voltage window. UHS-I supported host sets S18R to 1. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.

(8) If no response or error response is receive, set SDIO flag to 0 and go to step (10). If good response is received, go to step (9).

(9) Check busy status in the response. If busy is released, set SDIO flag to 1 and go to step (10). Repeat from step (7) while busy is indicated. Detecting timeout of 1 second exits the loop. In this case, set SDIO flag to 0 and go to step (10).

(10) Good response in this step means that all responses received at (6) and (8) are valid. When response is good, MP (memory present) flag in the response can be checked. If the response valid and MP = 0, go to step (28). Otherwise, go to step (11).

(11) Check F8 flag set in step (3). If CMD8 is executed correctly (F8 = 1), go to step (19). Otherwise, go to step (12).

(12) OCR is available by issuing ACMD41 with the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. The response of CMD55 (ACMD41) may indicate illegal command error due to some SD cards do not recognized CMD8. The Host Driver should ignore this error or issue CMD0 before ACMD41 to clear this error status.

(13) If response of CMD55 is not received, the card is not SD cards and goes to (17). If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are OK, go to Step (14). Otherwise, go to step (28). Locked card can be detected by the card status in the response of CMD55.

(14) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.

(15) If no response or error response is received, go to step (28). If good response is received, go to step (16).

(16) Check busy status in the response. If busy is released, go to step (18). Repeat from step (14) while busy is indicated. The interval of ACMD41 shall be less than 50 ms. Detecting timeout of 1 second exits the loop and go to step (28).

(17) The host recognizes that the card is not SD memory card and quits SD card initialization.

(18) The host recognizes that the card is Version 1.xx Standard Capacity SD Memory Card. Go to Step (30).

(19) OCR is available by issuing ACMD41 with setting the voltage window (bit 23 to 0) in the argument is set to 0. Memory initialization is not started. Setting of HCS does not affect this operation.

Copyright © 2026 Texas Instruments Incorporated

(20) If the card responds to CMD55, it may also respond to CMD41. If the responses of ACMD41 are OK, go to Step (21). Otherwise, go to step (28). Locked card can be detected by the card status in the response of CMD55.

(21) The memory portion starts initialization by Issuing ACMD41 with setting the supply voltage to the voltage window. UHS-I supported host sets S18R to1. If the host can supply more than 150mA, XPC is set to 1. HCS in the argument is set to 1, which indicates supporting High Capacity Memory Card. If the supplied voltage is not matched with voltage window of card, the card goes into inactive state and does not return the response.

(22) If no response or error response is received, go to step (28). If good response is received, go to step (23).

(23) Check busy status in the response. If busy is released, go to step (24). Repeat from step (21) while busy is indicated. The interval of ACMD41 shall be less than 50 ms. Detecting timeout of 1 second exits the loop and go to step (28).

(24) CCS in the response is valid after busy is released. If CCS = 0, it indicates the Standard Capacity SD Memory Card and go to step (25). If CCS = 1, it indicates the High Capacity SD Memory Card or Extended Capacity Memory Card and go to Step (26).

(25) The host recognizes that the card is Ver2.00 or Ver3.00 Standard Capacity SD Memory Card. Optimal functions defined in Version 2.00 or higher are available. Go to Step (32).

(26) The host recognizes that the card is the High Capacity SD Memory Card or Extended Capacity Memory Card.

(27) Perform the signal voltage switch procedure and go to step (32).

(28) Check SDIO flag. If SDIO = 1, go to step (28). Otherwise, go to step (31).

(29) The host recognizes that the card is SDIO only card and go to step (30).

(30) Perform the signal voltage switch procedure and go to step (33).

(31) The host recognizes that the card is unusable.

(32) In case of memory card, CMD2 is issued to get CID and Go to Step (31).

(33) CMD3 is issued to get RCA. If the RCA number is 0, the Host should issue CMD3 again.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.6.1 Signal Voltage Switch Procedure (for UHS-I)

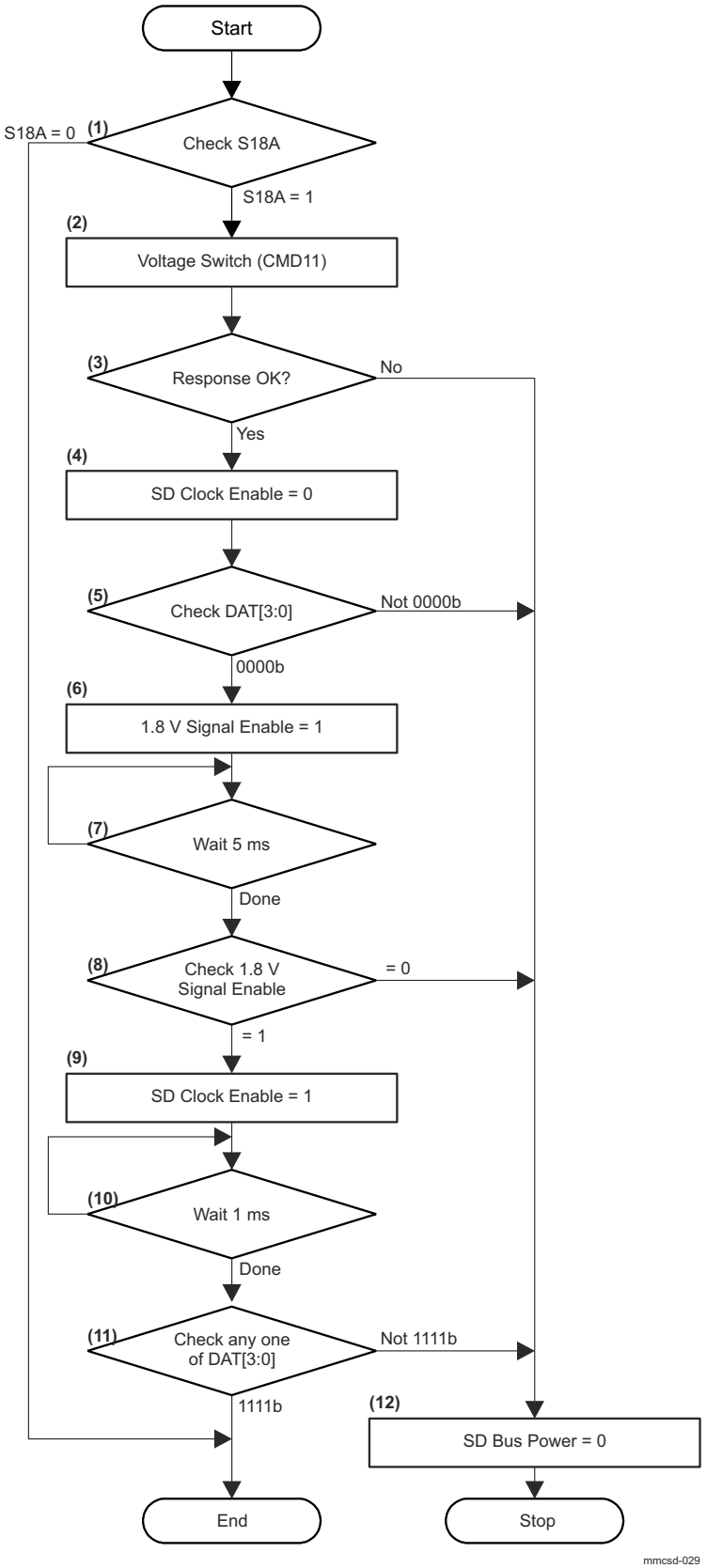


Figure 12-222. Signal Voltage Switch Procedure

Copyright © 2026 Texas Instruments Incorporated

(1) If S18A of CMD5 or S18A of ACMD41 is set to 1, signal voltage switch is performed according to the following steps. Otherwise, exits from this procedure.

(2) Issue CMD11.

(3) Check response and if an error is detected, go to step (12).

(4) Stop providing SD clock to the card.

(5) Check DAT[3:0] level. If the level is 0000b, the card is ready to start voltage switch sequence. Otherwise, go to (12) to quit the sequence.

(6) Set MMCSD0_HOST_CONTROL2[3] V1P8_SIGNAL_ENA bit.

(7) Wait 5 ms. 1.8 V voltage regulator shall be stable within this period.

(8) If MMCSD0_HOST_CONTROL2[3] V1P8_SIGNAL_ENA bit is cleared by Host Controller, go to step (12).

(9) Provide SD Clock to the card again.

(10) Wait 1 ms.

(11) Check DAT[3:0] level. If the level is 1111b, switch to 1.8 V signal level is completed successfully. Otherwise, go to (12).

(12) If an error occurs during voltage switch procedure, stop providing the power to the card. In this case, Host Driver should retry initialization procedure by setting S18R to 0 at step (7) and (21) in Figure 12-220 and Figure 12-221.

12.3.5.5.1.7 SD Transaction Generation

This section describes the sequences how to generate and control various kinds of SD transactions. SD transactions are classified into three cases:

(1) Transactions that do not use the DAT line.

(2) Transactions that use the DAT line only for the busy signal.

(3) Transactions that use the DAT line for transferring data.

In this specification the first and the second case's transactions are classified as "Transaction Control without Data Transfer using DAT Line", the third case's transaction is classified as "Transaction Control with Data Transfer using DAT Line". Refer to the latest SD Physical Layer Specification and SDIO Specification for more detail about SD commands specification.

12.3.5.5.1.7.1 Transaction Control without Data Transfer Using DAT Line

In this section, the sequence for how to issue SD Command and how to complete SD Command is explained. Figure 12-223 shows the sequence to issue a SD Command and Figure 12-224 shows the sequence to finalize a SD Command.

12.3.5.5.1.7.1.1 The Sequence to Issue a SD Command

The sequence to issue the SD Command is detailed in Figure 12-223.

Copyright © 2026 Texas Instruments Incorporated

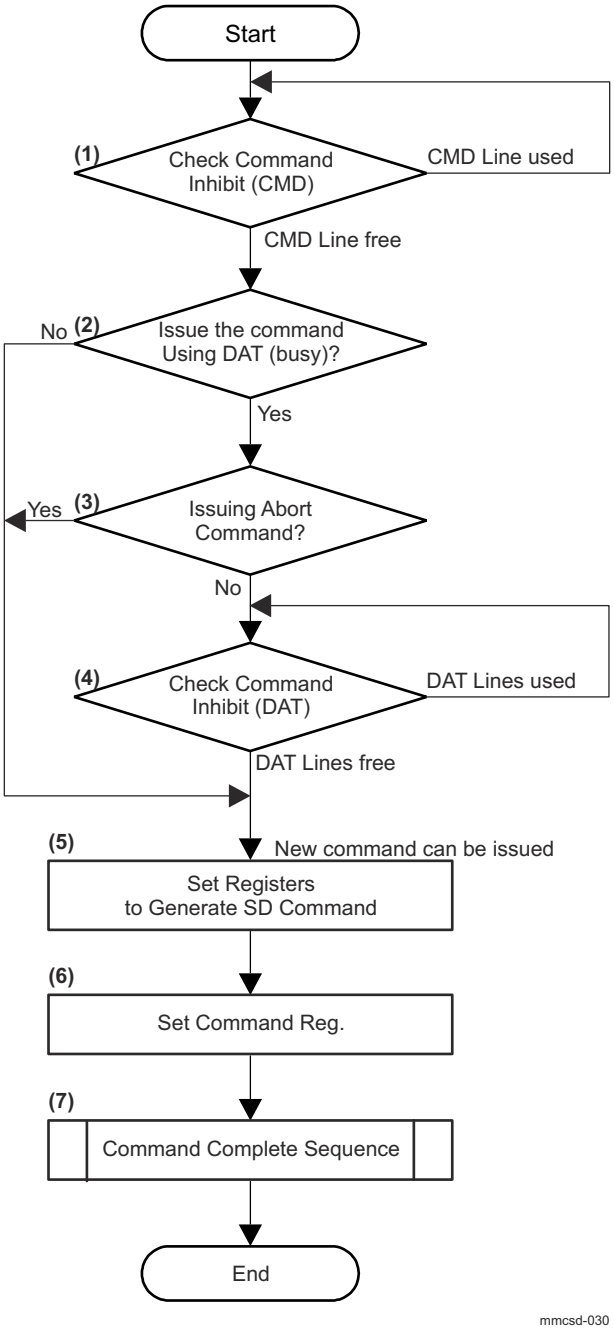


Figure 12-223. SD Command Issue Sequence

(1) Check MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit. Repeat this step until MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit is 0. That is, when MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit is 1, the Host Driver shall not issue a SD Command.

(2) If the Host Driver issues a SD Command using DAT lines including busy signal, go to step (3). If without using DAT lines including busy signal, go to step (5).

(3) If the Host Driver is issuing an abort command, go to step (5). In the case of nonabort command, go to step (4).

(4) Check MMCSD0_PRESENTSTATE[1] INHIBIT_DAT bit. Repeat this step until MMCSD0_PRESENTSTATE[1] INHIBIT_DAT bit is set to 0.

Copyright © 2026 Texas Instruments Incorporated

(5) Set registers except the MMCSD0_COMMAND register.

(6) Set the MMCSD0_COMMAND register.

Note: Writing the upper byte [3] in the MMCSD0_COMMAND register causes the Host Controller to issue a SD command to the SD card.

(7) Perform Command Completion Sequence in accordance.

12.3.5.5.1.7.1.2 The Sequence to Finalize a Command

Figure 12-224 shows the sequence to finalize a SD Command when response check is disabled. There is a possibility that some errors (Command Index/End bit/CRC/Timeout Error) occur during this sequence. If response check is enabled, error is indicated by Response Error Interrupt.

Copyright © 2026 Texas Instruments Incorporated

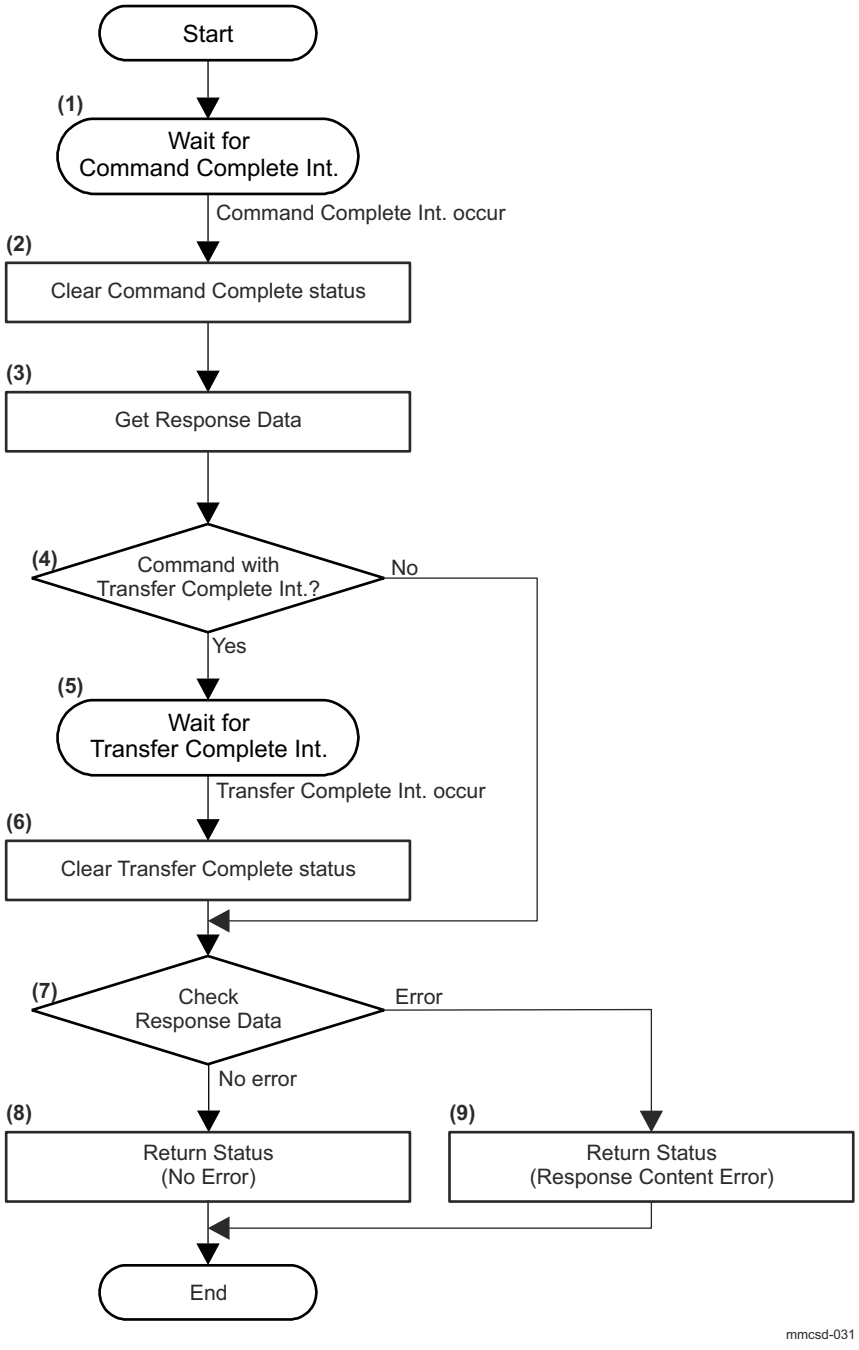


Figure 12-224. Command Complete Sequence

12.3.5.5.1.7.1.3

(1) If MMCSD0_TRANSFER_MODE[8] RESP_INTR_DIS bit is set to 1 (response check is enabled), go to stop (4) else wait for the Command Complete Interrupt (MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE). If the Command Complete Interrupt has occurred, go to step (2).

(2) Write 1 to MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit to clear this bit.

(3) Read the Response register (see MMCSD0_RESPONSE_0 - MMCSD0_RESPONSE_7) and get necessary information of the issued command.

Copyright © 2026 Texas Instruments Incorporated

(4) Judge whether the command uses the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE) or not. If it uses Transfer Complete, go to step (5). If not, go to step (7).

(5) Wait for the Transfer Complete Interrupt. If the Transfer Complete Interrupt has occurred, go to step (6).

(6) Write 1 to MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit to clear this bit.

(7) Check for errors in Response Data (MMCSD0_RESPONSE_0 - MMCSD0_RESPONSE_7). If there is no error, go to step (8). If there is an error, go to step (9).

(8) Return Status of "No Error".

(9) Return Status of "Response Contents Error".

Note1: While waiting for the Transfer Complete interrupt, the Host Driver shall only issue commands that do not use the busy signal.

Note2: The Host Driver shall judge the Auto CMD12 complete by monitoring Transfer Complete.

Note3: When the last block of un-protected area is read using memory multiple block read command (CMD18), OUT_OF_RANGE error may occur even if the sequence is correct. The Host Driver should ignore it. This error will appear in the response of Auto CMD12 or in the response of the next memory command.

12.3.5.5.1.7.2 Transaction Control with Data Transfer Using DAT Line

Depending on whether DMA (optional) is used or not, there are two execution methods.

In addition, the sequences for SD transfers are classified into following three kinds according to how the number of blocks is specified:

(1) Single Block Transfer: The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified is always one.

(2) Multiple Block Transfer: The number of blocks is specified to the Host Controller before the transfer. The number of blocks specified shall be one or more.

(3) Infinite Block Transfer: The number of blocks is not specified to the Host Controller before the transfer. This transfer is continued until an abort transaction is executed. This abort transaction is performed by CMD12 in the case of a SD memory card and by CMD52 in the case of a SDIO card.

12.3.5.5.1.7.2.1 Not using DMA

The sequence for not using DMA is shown in Figure 12-225.

Copyright © 2026 Texas Instruments Incorporated

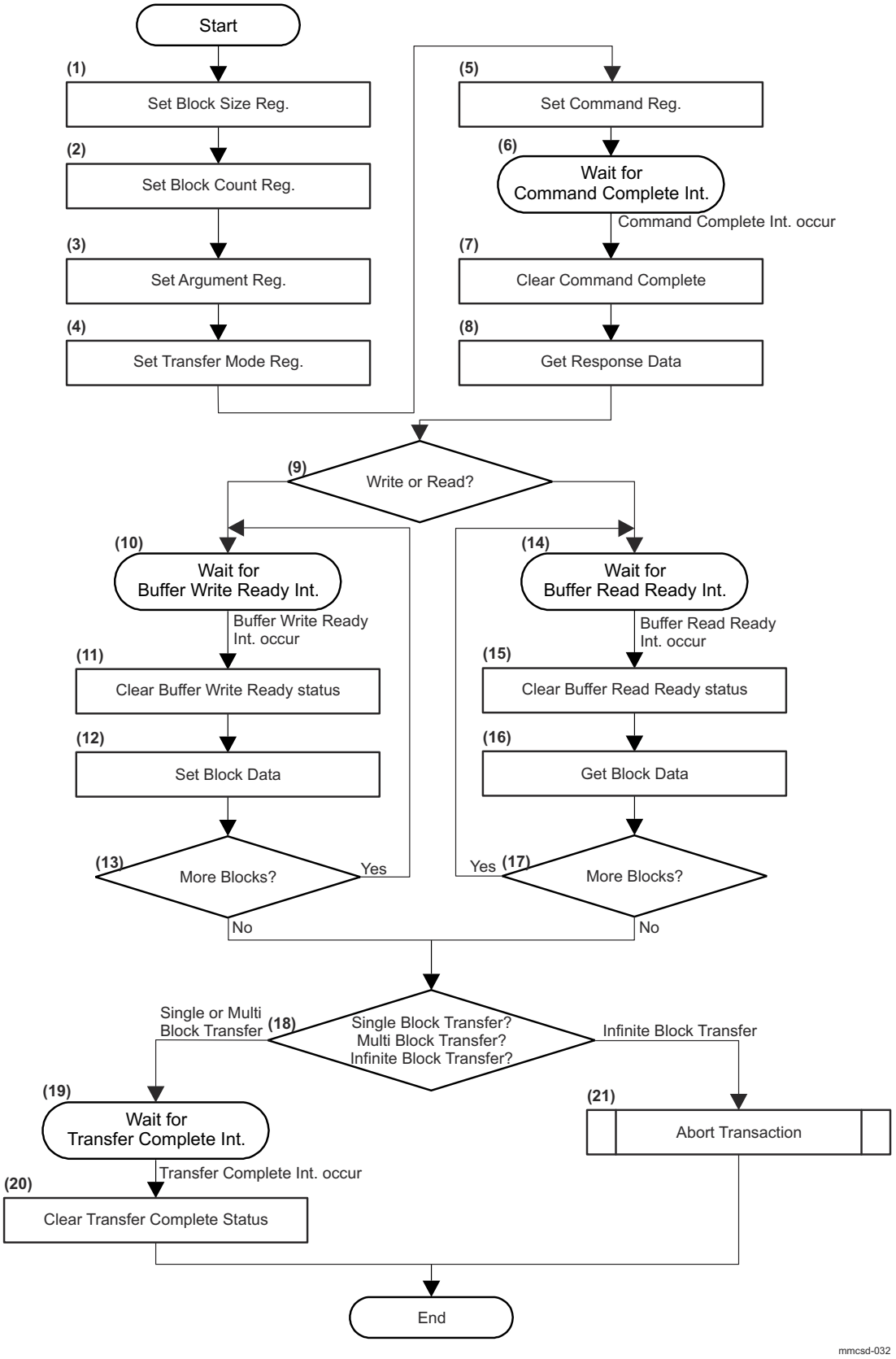


Figure 12-225. Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA)

Copyright © 2026 Texas Instruments Incorporated

(1) Set the value corresponding to the executed data byte length of one block to the MMCSD0_BLOCK_SIZE register.

(2) Set the value corresponding to the executed data block count to the MMCSD0_BLOCK_COUNT register in accordance with *Determination of Transfer Type*.

(3) Set the argument value to Argument register (MMCSD0_ARGUMENT1_LO and MMCSD0_ARGUMENT1_HI).

(4) Set the value to the MMCSD0_TRANSFER_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the MMCSD0_TRANSFER_MODE register. Multi/Single Block Select and Block Count Enable are determined according to *Determination of Transfer Type*.

If response check is enabled (MMCSD0_TRANSFER_MODE[7] RESP_ERR_CHK_ENA = 1), set MMCSD0_TRANSFER_MODE[8] RESP_INTR_DIS bit to 1 and select Response Type R1/R5 (MMCSD0_TRANSFER_MODE[6] RESP_TYPE).

(5) Set the value to MMCSD0_COMMAND register.

Note: When writing the upper byte [3] of MMCSD0_COMMAND register, SD command is issued.

(6) If response check is enabled, go to stop (9) else wait for the Command Complete Interrupt (MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit).

(7) Write 1 to the MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit for clearing this bit.

(8) Read Response register (MMCSD0_RESPONSE_0 - MMCSD0_RESPONSE_7) and get necessary information of the issued command.

(9) In the case where this sequence is for write to a card, go to step (10). In case of read from a card, go to step (14).

(10) Then wait for Buffer Write Ready Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[4] BUF_WR_READY).

(11) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[4] BUF_WR_READY bit for clearing this bit.

(12) Write block data (in according to the number of bytes specified at the step (1)) to MMCSD0_DATA_PORT register.

(13) Repeat until all blocks are sent and then go to step (18).

(14) Then wait for the Buffer Read Ready Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[5] BUF_RD_READY).

(15) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[5] BUF_RD_READY bit for clearing this bit.

(16) Read block data (in according to the number of bytes specified at the step (1)) from the MMCSD0_DATA_PORT register.

(17) Repeat until all blocks are received and then go to step (18).

(18) If this sequence is for Single or Multiple Block Transfer, go to step (19). In case of Infinite Block Transfer, go to step (21).

(19) Wait for Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE).

(20) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit for clearing this bit.

(21) Perform the sequence for Abort Transaction in accordance with Section 12.3.5.5.1.8, *Abort Transaction*.

Note: Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

12.3.5.5.1.7.2.2 Using SDMA

The sequence for using SDMA is shown in Figure 12-226.

Copyright © 2026 Texas Instruments Incorporated

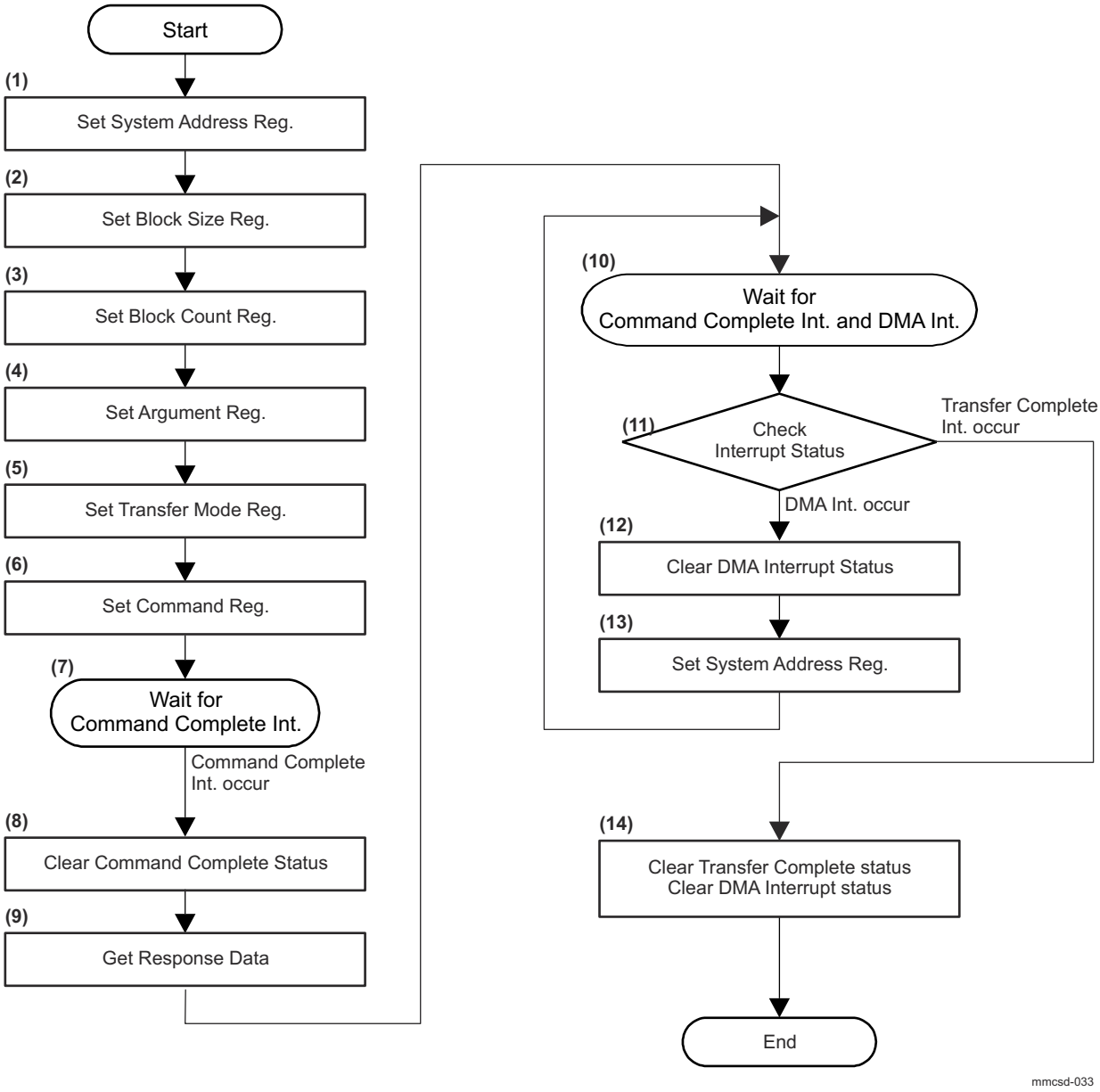


Figure 12-226. Transaction Control with Data Transfer Using DAT Line Sequence (Using SDMA)

(1) Data location of system memory is set to the SDMA System Address register if MMCSD0_HOST_CONTROL2[12] HOST_VER40_ENA = 0 or set to the MMCSD0_ADMA_SYS_ADDRESS register if MMCSD0_HOST_CONTROL2[12] HOST_VER40_ENA = 1.

(2) Set the value corresponding to the executed data byte length of one block in the MMCSD0_BLOCK_SIZE register.

(3) Set the value corresponding to the executed data block count in the MMCSD0_BLOCK_COUNT register in accordance with *Determination of Transfer Type*.

(4) Set the argument value to the Argument register (MMCSD0_ARGUMENT1_LO and MMCSD0_ARGUMENT1_HI).

(5) Set the value to the MMCSD0_TRANSFER_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the

Copyright © 2026 Texas Instruments Incorporated

MMCSD0_TRANSFER_MODE register. Multi/Single Block Select and Block Count Enable are determined according to *Determination of Transfer Type*. If response check is enabled (MMCSD0_TRANSFER_MODE[7] RESP_ERR_CHK_ENA = 1), set MMCSD0_TRANSFER_MODE[8] RESP_INTR_DIS bit to 1 and select Response Type R1/R5.

(6) Set the value to the MMCSD0_COMMAND register.

Note: When writing to the upper byte [3] of the MMCSD0_COMMAND register, the SD command is issued and SDMA is started.

(7) If response check is enabled, go to stop (10) else wait for the Command Complete Interrupt (MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit).

(8) Write 1 to the MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit to clear this bit.

(9) Read Response register (MMCSD0_RESPONSE_0 - MMCSD0_RESPONSE_7) and get necessary information of the issued command.

(10) Wait for the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE) and DMA Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[3] DMA_INTERRUPT).

(11) If MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit is set to 1, go to Step (14) else if MMCSD0_NORMAL_INTR_STS_ENA[3] DMA_INTERRUPT bit is set to 1, go to Step (12). The MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit is higher priority than the MMCSD0_NORMAL_INTR_STS_ENA[3] DMA_INTERRUPT bit.

(12) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[3] DMA_INTERRUPT bit to clear this bit.

(13) Set the next system address of the next data position to the System Address register (MMCSD0_ADMA_SYS_ADDRESS) and go to Step (10).

(14) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit and MMCSD0_NORMAL_INTR_STS_ENA[3] DMA_INTERRUPT bit to clear this bit.

Note: Step (2) and Step (3) can be executed simultaneously. Step (5) and Step (6) can also be executed simultaneously.

12.3.5.5.1.7.2.3 Using ADMA

The sequence for using ADMA is shown in Figure 12-227.

Copyright © 2026 Texas Instruments Incorporated

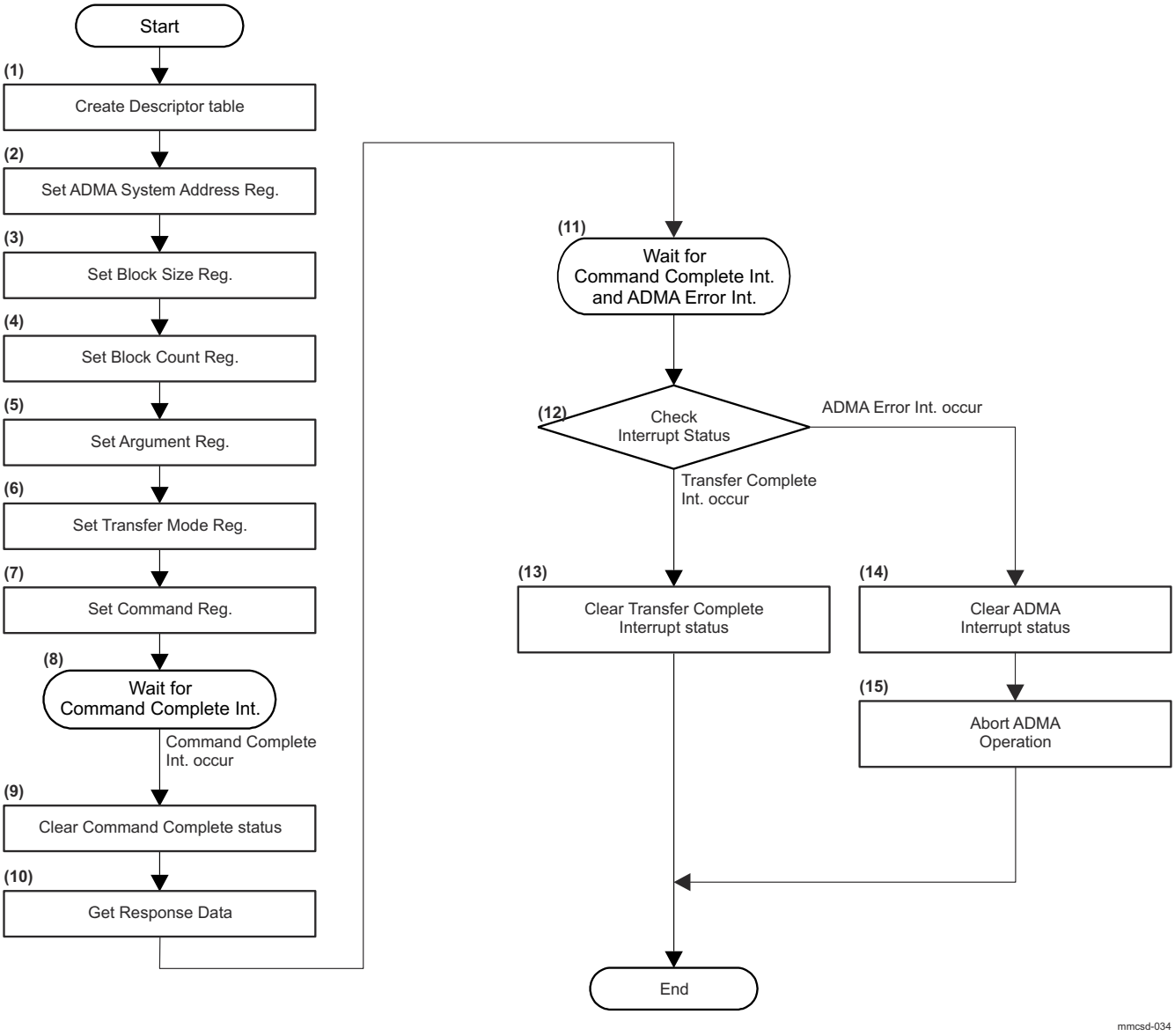


Figure 12-227. Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA)

(1) Create Descriptor table for ADMA in the system memory.

(2) Set the Descriptor address for ADMA in the MMCSD0_ADMA_SYS_ADDRESS register.

(3) Set the value corresponding to the executed data byte length of one block in the MMCSD0_BLOCK_SIZE register.

(4) Set the value corresponding to the executed data block count in the MMCSD0_BLOCK_COUNT register in accordance with *Determination of Transfer Type*.

(5) Set the argument value to the Argument register (MMCSD0_ARGUMENT1_LO and MMCSD0_ARGUMENT1_HI).

(6) Set the value to the MMCSD0_TRANSFER_MODE register. The Host Driver determines Multi/Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable in the MMCSD0_TRANSFER_MODE register. Multi/Single Block Select and Block Count Enable are determined according to *Determination of Transfer Type*. If response check is enabled (MMCSD0_TRANSFER_MODE[7]

Copyright © 2026 Texas Instruments Incorporated

RESP_ERR_CHK_ENA = 1), set MMCSD0_TRANSFER_MODE[8] RESP_INTR_DIS bit to 1 and select Response Type R1/R5 (MMCSD0_TRANSFER_MODE[6] RESP_TYPE).

(7) Set the value to the MMCSD0_COMMAND register.

Note: When writing to the upper byte [3] of the MMCSD0_COMMAND register, the SD command is issued and DMA is started.

(8) If response check is enabled, go to stop (11) else wait for the Command Complete Interrupt (MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit).

(9) Write 1 to the MMCSD0_NORMAL_INTR_STS[0] CMD_COMPLETE bit to clear this bit.

(10) Read Response register (MMCSD0_RESPONSE_0 - MMCSD0_RESPONSE_7) and get necessary information of the issued command.

(11) Wait for the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE) and ADMA Error Interrupt (MMCSD0_ADMA_ERR_STATUS[1-0] ADMA_ERR_STATE).

(12) If MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit is set to 1, go to Step (13) else if MMCSD0_ADMA_ERR_STATUS[1-0] ADMA_ERR_STATE bit field is set to 1, go to Step (14).

(13) Write 1 to the MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit to clear this bit.

(14) Write 1 to the MMCSD0_ADMA_ERR_STATUS[1-0] ADMA_ERR_STATE bit field to clear this bit.

(15) Abort ADMA operation. SD card operation should be stopped by issuing abort command. If necessary, the Host Driver checks MMCSD0_ADMA_ERR_STATUS register to detect why ADMA error is generated.

Note: Step (3) and Step (4) can be executed simultaneously. Step (6) and Step (7) can also be executed simultaneously.

12.3.5.5.1.8 Abort Transaction

An abort transaction is performed by issuing CMD12 for a SD memory card and by issuing CMD52 for a SDIO card. There are two cases where the Host Driver needs to do an Abort Transaction. The first case is when the Host Driver stops Infinite Block Transfers. The second case is when the Host Driver stops transfers while a Multiple Block Transfer is executing.

There are two ways to issue an Abort Command. The first is an asynchronous abort. The second is a synchronous abort. In an asynchronous abort sequence, the Host Driver can issue an Abort Command at any time unless MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit is set to 1. In a synchronous abort, the Host Driver shall issue an Abort Command after the data transfer stopped by using MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

12.3.5.5.1.8.1 Asynchronous Abort

The sequence for Asynchronous Abort is shown in Figure 12-228.

Copyright © 2026 Texas Instruments Incorporated

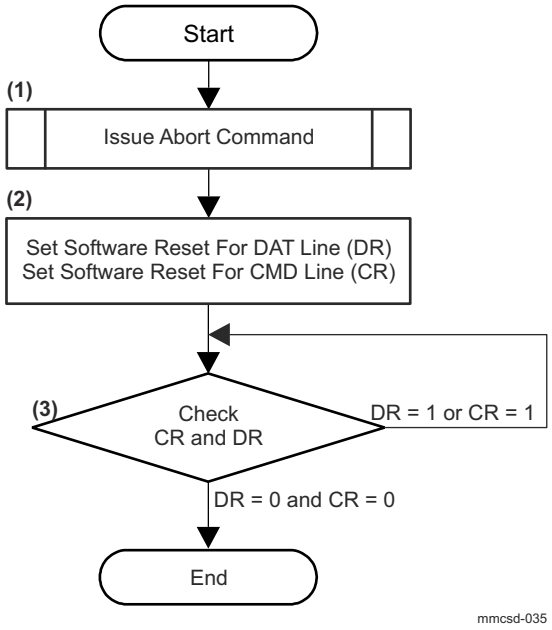


Figure 12-228. Asynchronous Abort Sequence

(1) Issue Abort Command in accordance.

(2) Set both MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit to 1 to do software reset.

(3) Check MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit. If both MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit are 0, go to "End". If either MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit or MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit is 1, go to step (3).

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.8.2 Synchronous Abort

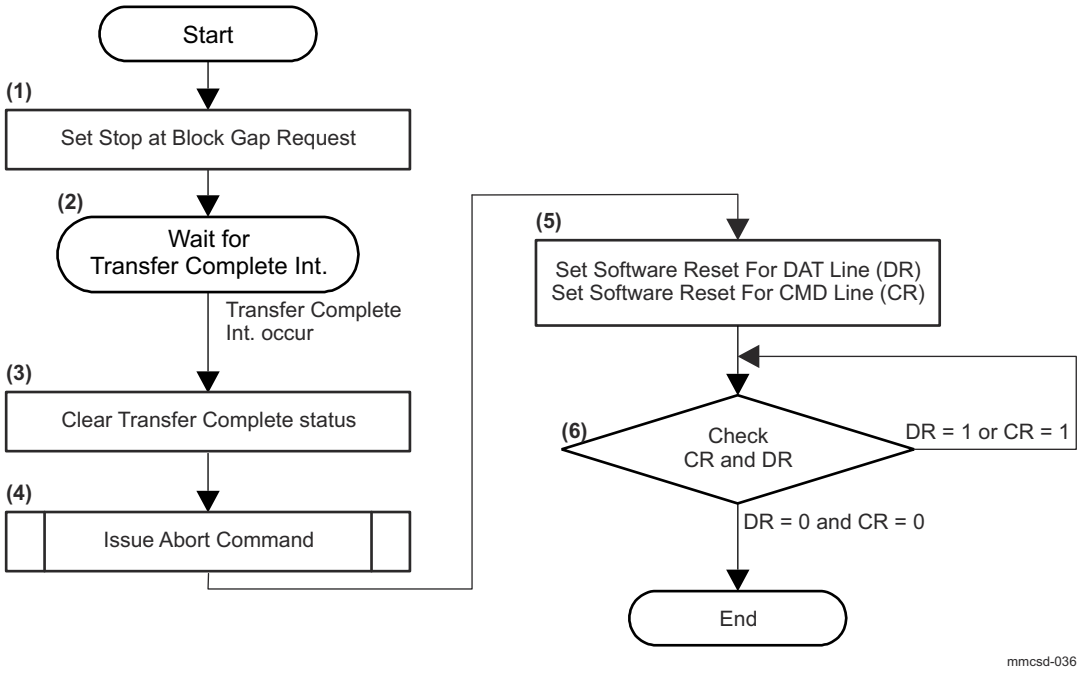


Figure 12-229. Synchronous Abort Sequence

(1) Set the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit to 1 to stop SD transactions.

(2) Wait for the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE).

(3) Set the MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE bit to 1 to clear this bit.

(4) Issue the Abort Command

(5) Set both MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit to 1 to do software reset.

(6) Check both MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit. If both MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit and MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit are 0, go to "End". If either MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit or MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit is 1, go to step (6).

12.3.5.5.1.9 Changing Bus Speed Mode

This section describes the sequence for switching the bus speed mode: Default Speed, High Speed mode and UHS-I mode. The switch command (CMD6) is used to change memory card bus speed mode. The EHS bit (SDIO Version 2.00) or BSS[2-0] bits (SDIO Version 3.00) in the CCCR register is used to change the bus speed mode for SDIO card. In case of Combo card, either of the switch method changes both memory and IO bus speed mode. This means the first switch is effective. Refer to the Physical Layer Specification Version 3.0x and the SDIO Specification Version 3.00 for more information about switching bus speed. Figure 12-230 shows the sequence for switching bus speed mode for Combo Card. Note that if the card is locked, bus width cannot be changed. Unlock the card is required before changing bus width.

Copyright © 2026 Texas Instruments Incorporated

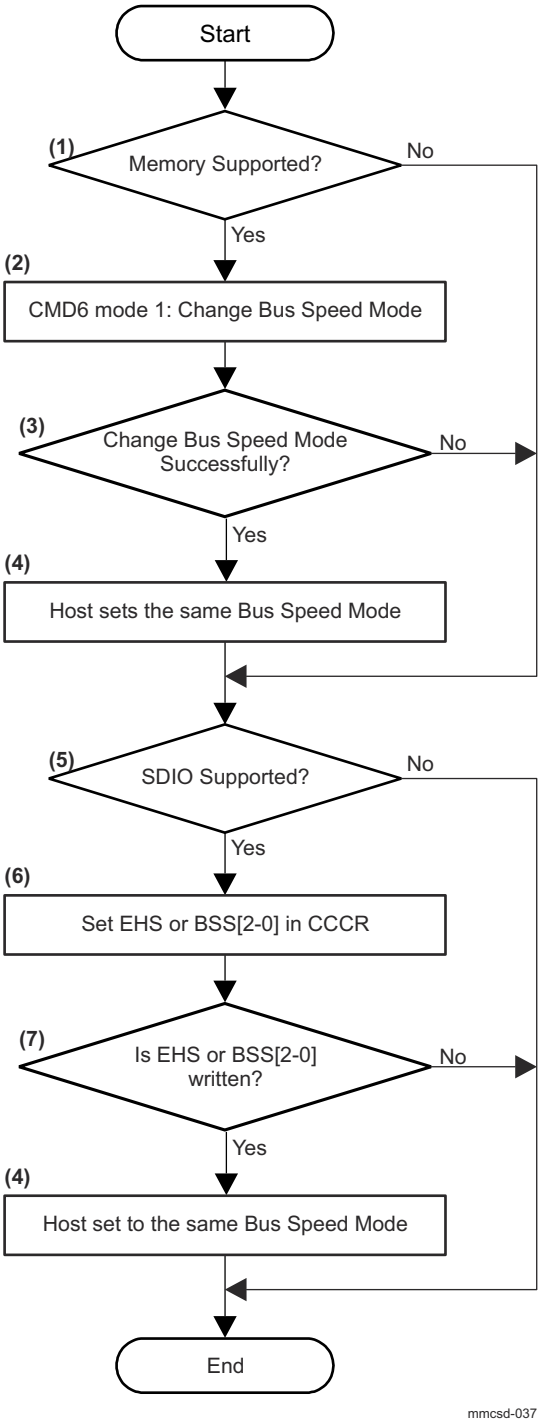


Figure 12-230. Changing Bus Speed Mode

(1) The Host Driver checks if the card supports memory. If not supported, go to (5).

(2) Issue CMD6 with mode 1 to change bus speed mode (Default, High Speed mode or UHS-I mode).

(3) Check the response of CMD6. If the card does not supports CMD6 (no response) or bus speed is not changed successfully, go to step (5). In this case, the card is in Default Speed mode.

(4) The Host Driver changes the Host Controller bus speed mode to the same mode.

Copyright © 2026 Texas Instruments Incorporated

(5) The Host Driver checks if the card supports SDIO. If not supported, go to the end.

(6) Issue CMD52 to write EHS bit or BSS[2-0] bits in CCCR to change bus speed mode (the same bus speed mode of (2) shall be set).

(7) If EHS or BSS[2-0] are not changed successfully, go to the end.

(8) The Host Driver changes the Host Controller bus speed to the same mode. In case of Combo card, bus speed is already changed at step (4) and this step does not affect changing bus speed.

12.3.5.5.1.10 Error Recovery

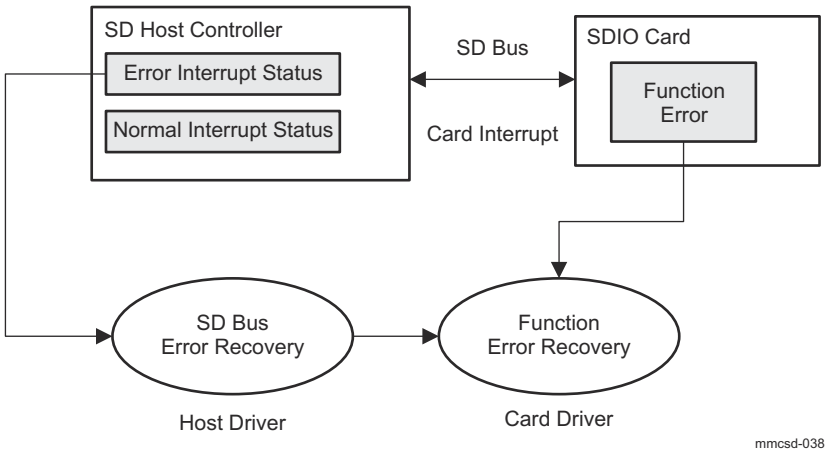


Figure 12-231. Error Report and Recovery

Figure 12-231 shows concept of error report and its recovery. The Host Controller has 2 interrupt status registers. If an error occurs in the SD Bus transaction, one of the bits is set in the MMCSD0_ERROR_INTR_STS register. If the function errors occur in the SDIO card, the card interrupt informs these function errors and the Card interrupt is set in the MMCSD0_NORMAL_INTR_STS_ENA register (the Card Interrupt is used to inform not only error statuses but also normal information. For example, to inform function ready). The Card Driver shall do function error recovery because the Host Driver does not know how to control the function. In the case that function error occurs due to SD Bus error, SD Bus error recovery is required before function error recovery. Abort command is used to recover SD Bus, and then the Host Driver should save error statuses related to SD Bus errors before issuing abort command and transfer these statuses to the Card Driver. These statuses may be used to recover function error. Following explanations are related to SD Bus error recovery. This specification does not specify the function error recovery.

When an error occurs during data transfer in 2L-HD UHS-II mode, there will be the case that Host Controller cannot drive D0 lane in input mode due to DIR LSS for retrieving lane direction is not detected. In this case, Host Driver cannot issue abort command for recovery. Then if DIR LSS is not detected, Host Controller sets MMCSD0_UHS2_ERR_INTR_STS[17] DEADLOCK_TIMEOUT bit. Host Driver should execute power cycle if MMCSD0_UHS2_ERR_INTR_STS[17] DEADLOCK_TIMEOUT bit is detected to recover from this error. Furthermore, if this type of error is detected several times in 2L-HD, Host Driver should use FD mode rather than using 2L-HD.

Implementation Note:

If the Card Driver cannot recover the function errors, the Host Driver should try following methods.

(9) Using IOEx for SDIO card

IOEx may be used as the reset per function basis. Sequence is as follows:

Clear IOEx = 0 and wait until IORx = 0 and then set IOEx = 1 again. SDIO may be recovered when IORx = 1.

(10) Using reset command for memory and SDIO card Re-initialization sequence is required.

Copyright © 2026 Texas Instruments Incorporated

(11) Off and on power supply for the SD Bus.

The card may be recovered by the power on reset. Re-initialization sequence is required.

The two cases where the Host Driver needs the "Error Recovery" sequence are classified as follows:

(1) Error Interrupt Recovery:

If error interrupt is indicated by the MMCSD0_ERROR_INTR_STS register, the Host Driver shall apply this sequence.

(2) Auto CMD12 Error Recovery:

If there are errors in Auto CMD12, the Host Driver shall apply this sequence. In terms of Return Status, Auto CMD12 Error Recovery is classified into 4 cases. It is shown in Figure 12-232. If error occurs during memory write transfer, strongly recommend using ACMD22 and then in the following recovery sequence, retry to send remaining blocks not written.

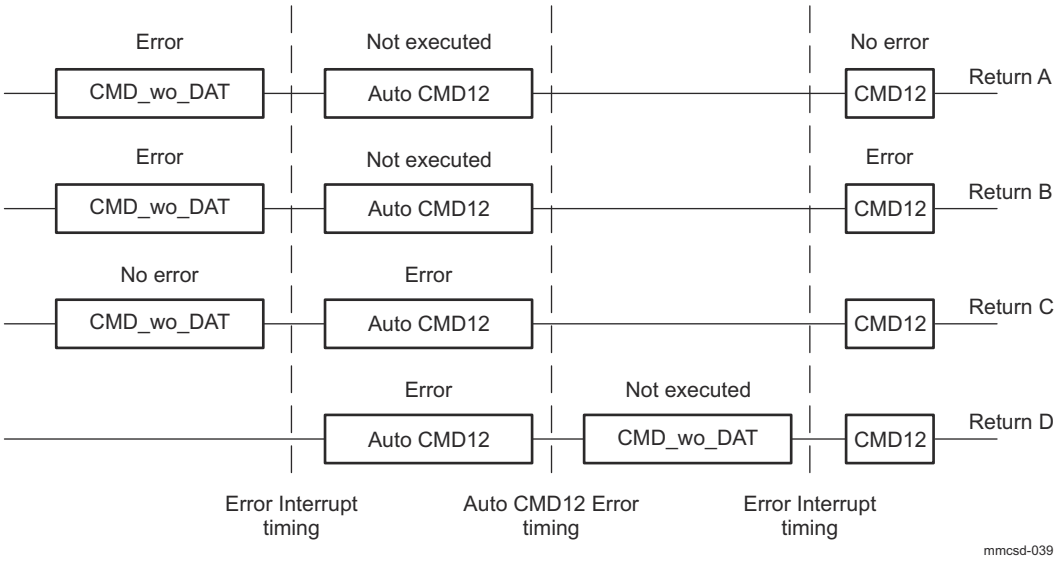


Figure 12-232. Return Status of Auto CMD12 Error Recovery

Implementation Note:

Abort command is used to recover from SD Bus error. SDIO transaction abort using CMD52 returns response but in the case of memory transaction abort using CMD12, response returns depending on the memory card state. If no response returns after issue CMD12, the Host Driver should check card state using CMD13. If the state is "tran" in the CURRENT_STATE, consider CMD12 is successful.

Implementation Note:

The following sequence is one possible error recovery flow. There may be another methods, sometimes using interrupts or polling. It can be possible to use another flows, based on Host System requirements.

In these error recovery sequences, return statuses for the next sequence. When the Host Controller cannot issue the next command due to SD Bus error, the error recovery sequences return "Non-recoverable" status. In this case, the Host System may cut off power to the SD Bus and then power on SD Bus and initialize both the Host Controller and the SD card again.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.10.1 Error Interrupt Recovery

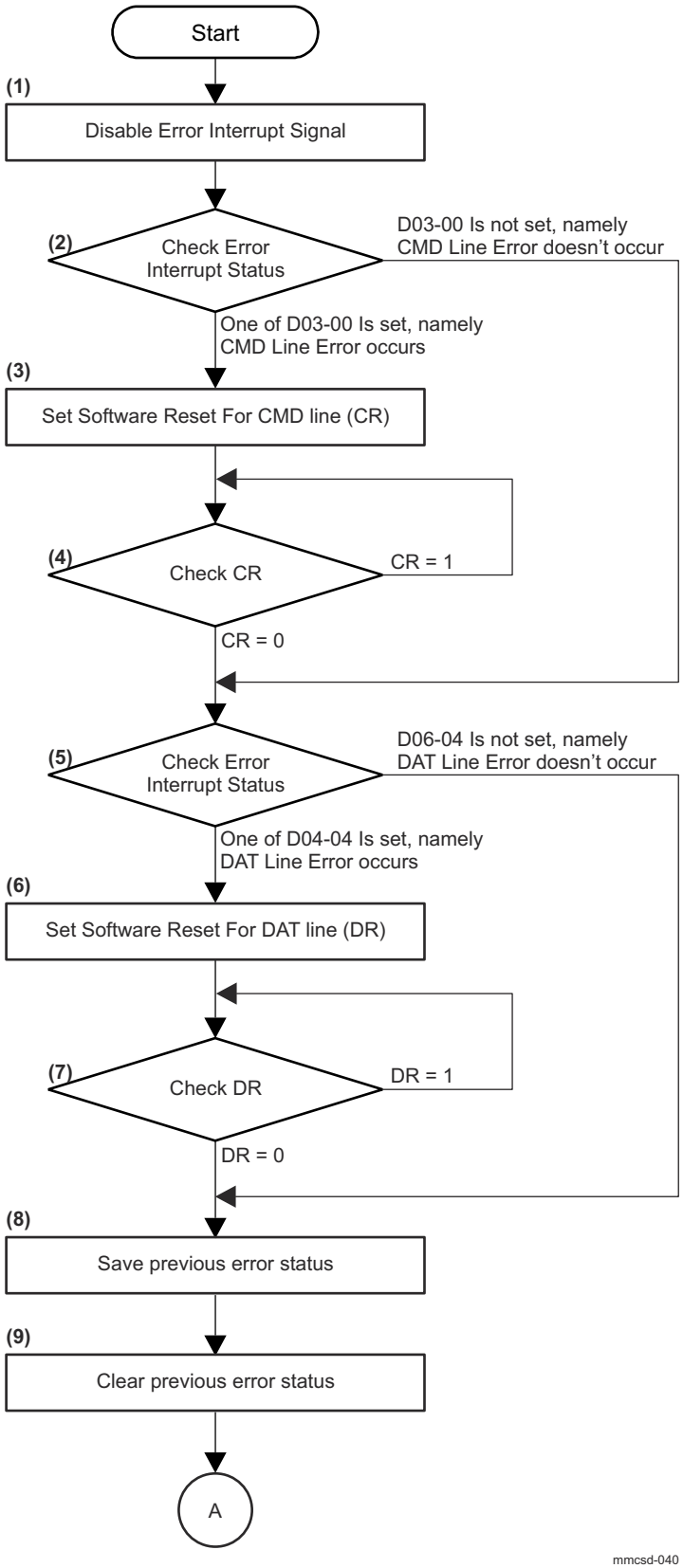


Figure 12-233. Error Interrupt Recovery Sequence (1)

Copyright © 2026 Texas Instruments Incorporated

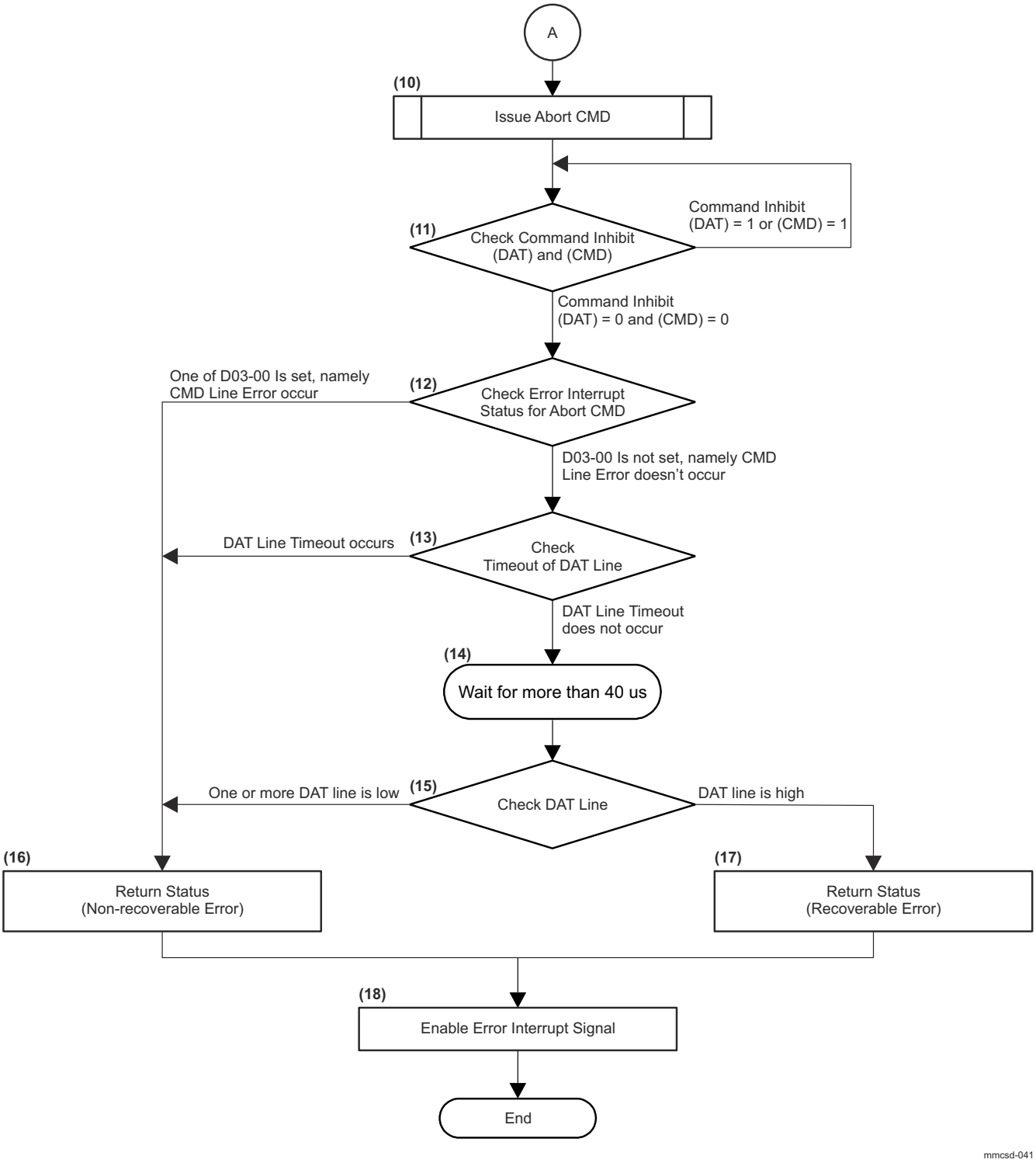


Figure 12-234. Error Interrupt Recovery Sequence (2)

(1) Disable the Error Interrupt Signal (MMCSD0_ERROR_INTR_SIG_ENA).

(2) Check bits D03-00 in the MMCSD0_ERROR_INTR_STS register. If one of these bits (D03-00) is set to 1, go to step (3). If none are set to 1 (all are 0), go to step (5).

(3) Set MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit to 1.

Copyright © 2026 Texas Instruments Incorporated

(4) Check MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit. If MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit is 0, go to step (5). If it is 1, go to step (4).

(5) Check bits D06-04 in the MMCSD0_ERROR_INTR_STS register. If one of these bits (D06-04) is set to 1, go to step (6). If none are set to 1 (all are 0), go to step (8).

(6) Set MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit to 1 for software reset of the DAT line.

(7) Check MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit. If MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit is 0, go to step (8). If it is 1, go to step (7).

(8) Save previous error status.

(9) Clear previous error status with setting them to 1.

(10) Issue Abort Command.

(11) MMCSD0_PRESENTSTATE[1] INHIBIT_DAT bit and MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit. Repeat this step until both MMCSD0_PRESENTSTATE[1] INHIBIT_DAT bit and MMCSD0_PRESENTSTATE[0] INHIBIT_CMD bit are set to 0.

(12) Check bits D03-00 in the MMCSD0_ERROR_INTR_STS register for Abort Command. If one of these bits is set to 1, go to step (16). If none of these bits are set to 1 (all are 0), go to step (13).

(13) Check MMCSD0_ERROR_INTR_STS[4] DATA_TIMEOUT bit. If this bit is set to 1, go to step (16). If it is 0, go to step (14).

(14) Wait for more than 40 us.

(15) By monitoring the DAT [3:0] Line Signal Level in the MMCSD0_PRESENTSTATE register, judge whether the level of the DAT line is low or not. If one or more DAT lines are low, go to step (16). If the DAT lines are high, go to step (17).

(16) Return Status of "Non-recoverable Error".

(17) Return Status of "Recoverable Error".

(18) Enable the Error Interrupt Signal (MMCSD0_ERROR_INTR_SIG_ENA).

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.10.2 Auto CMD12 Error Recovery

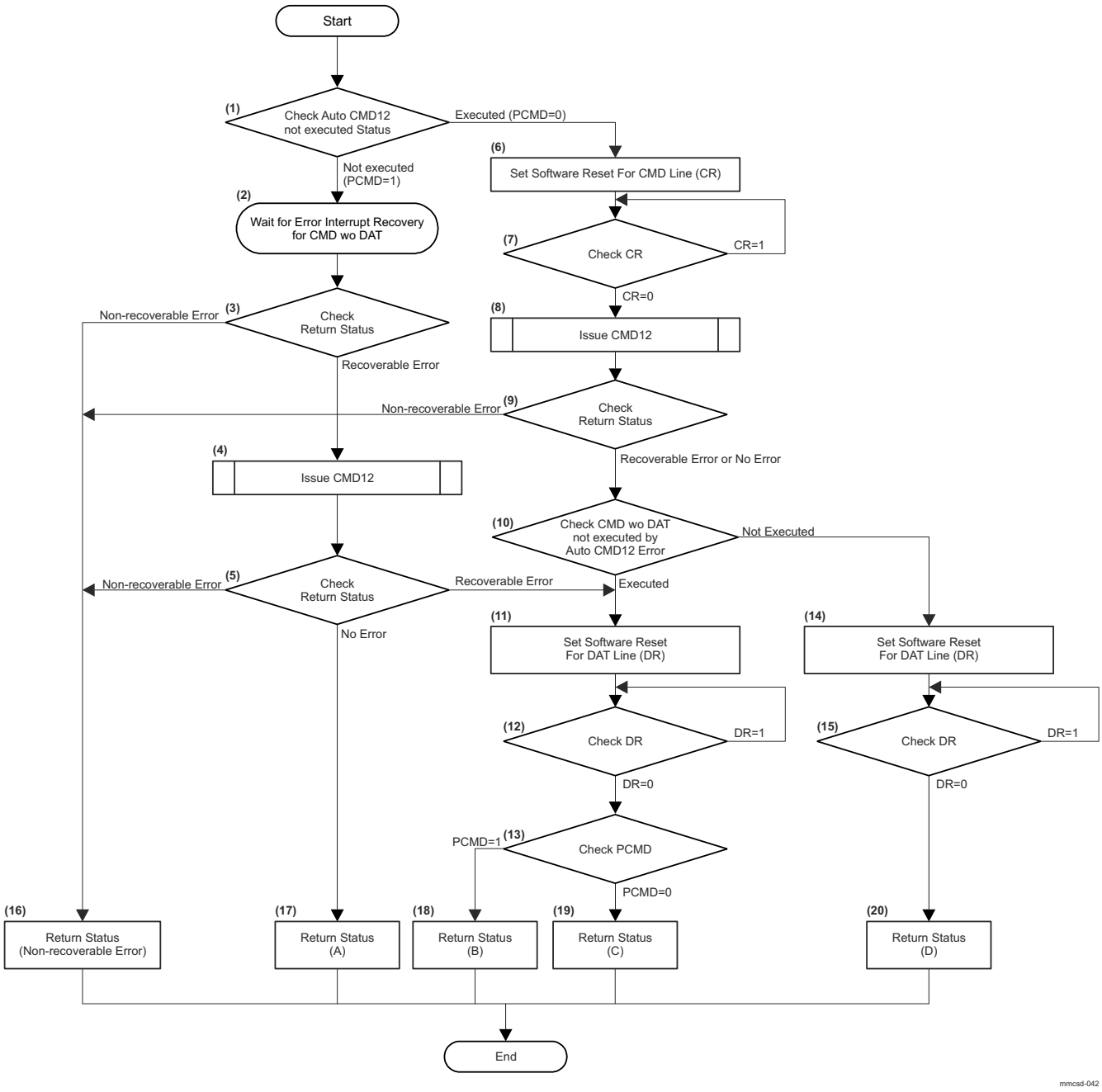


Figure 12-235. Auto CMD12 Error Recovery

The sequence for Auto CMD12 Error Recovery is shown in Figure 12-235. Following four cases A-D shall be covered.

A: An error occurred in CMD_wo_DAT, but not in the SD memory transfer.

B: An error occurred in CMD_wo_DAT, and also occurred in the SD memory transfer.

C: An error did not occur in CMD_wo_DAT, but an error occurred in the SD memory transfer.

D: CMD_wo_DAT was not issued, and an error occurred in the SD memory transfer.

Copyright © 2026 Texas Instruments Incorporated

(1) Check MMCSD0_AUTOCMD_ERR_STS[0] ACMD12_NOT_EXEC bit. If this bit is set to 1, go to step (2). If this bit is set to 0, go to step (6). In addition, the Host Driver shall define PCMD flag, which changes to 1 if MMCSD0_AUTOCMD_ERR_STS[0] ACMD12_NOT_EXEC bit is set to 1.

(2) Wait for Error Interrupt Recovery for CMD_wo_DAT.

(3) Check "Return Status". In the case of "Non-recoverable Error", go to step (16). In the case of "Recoverable Error", go to step (4).

(4) Issue CMD12 .

(5) If the CMD line errors occur for the CMD12 (one of D03-00 is set in the MMCSD0_ERROR_INTR_STS register), "Return Status" is "Non-recoverable Error" and go to step (16). If not CMD line error and busy timeout error occur (D04 is set in the MMCSD0_ERROR_INTR_STS register), "Return Status" is "Recoverable Error" and go to step (11). Otherwise, "Return Status" is "No error" and go to step (17).

(6) Set MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit to 1 for software reset of the CMD line.

(7) Check MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit. If MMCSD0_SOFTWARE_RESET[1] SWRST_FOR_CMD bit is 0, go to step (8). If it is 1, go to step (7).

(8) Issue CMD12 . Acceptance of CMD12 depends on the state of the card. CMD12 may make the card to return to tran state. If the card is already in tran state, the card does not response to CMD12.

(9) Check "Return Status" for CMD12. If "Return Status" returns "Non-recoverable Error", go to step (16). In the case of "Recoverable Error" or "No error", go to step (10).

(10) Check the MMCSD0_AUTOCMD_ERR_STS[0] ACMD12_NOT_EXEC bit. If this bit is 0, go to step (11). If it is 1, go to step (14).

(11) Set MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit to 1 for software reset of the DAT line.

(12) Check MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit. If MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit is 0, go to step (13). If it is 1, go to step (12).

(13) Check the PCMD flag. If PCMD is 1, go to step (18). If it is 0, go to step (19).

(14) Set MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit to 1 for software reset of the DAT line.

(15) Check MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit. If MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit is 0, go to step (20). If it is 1, go to step (15).

(16) Return Status of "Non-recoverable Error".

(17) Return Status that an error has occurred in CMD_wo_DAT, but not in the SD memory transfer.

(18) Return Status that an error has occurred in both CMD_wo_DAT, and the SD memory transfer.

(19) Return Status that an error has not occurred in CMD_wo_DAT, but has occurred in the SD memory transfer.

(20) Return Status that CMD_wo_DAT has not been issued, and an error has occurred in the SD memory transfer.

12.3.5.5.1.11 Wakeup Control (Optional)

After the Host System goes into standby mode, the Host System can resume from standby via a wakeup event initiated by one of the following three events:

(1) Interrupt from a SD card: If an SD card interrupt occurs, the Host System can resume from standby mode. If the Host System uses this wakeup factor, SD Bus power shall be kept on.

(2) Insertion of SD card: If a SD card is inserted, the Host System can resume from standby mode.

(3) Removal of SD card: If a SD card is removed, the Host System can resume from standby mode.

The sequence for preparing wakeup before the Host System goes into standby mode is shown in Figure 12-236.

Copyright © 2026 Texas Instruments Incorporated

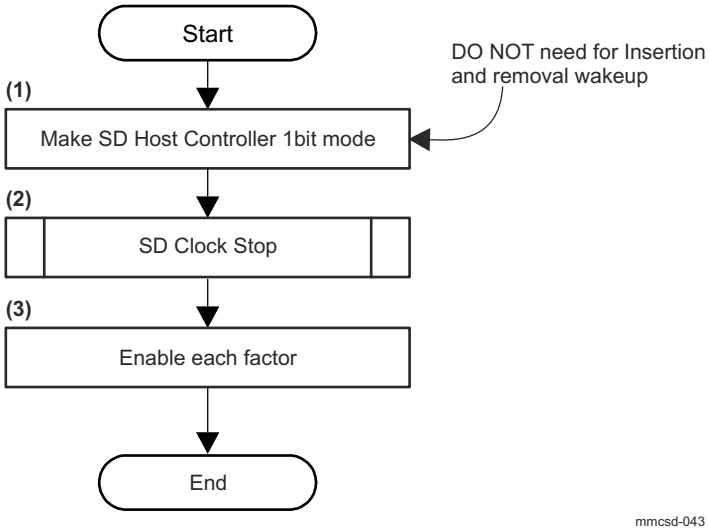


Figure 12-236. Wakeup Control before Standby Mode

(1) Set MMCSD0_HOST_CONTROL1[1] DATA_WIDTH bit to 0.

(2) Execute SD Clock Stop Sequence as described in Section 12.3.5.5.1.2.2, *SD Clock Supply and Stop Sequence*.

(3) Clear the MMCSD0_NORMAL_INTR_STS register and the MMCSD0_NORMAL_INTR_SIG_ENA register, and then set the enable bits of each wakeup event factor to 1 in the MMCSD0_WAKEUP_CONTROL register and set the bits of MMCSD0_ERROR_INTR_STS_ENA register to use wakeup.

The sequence for wakeup once in standby mode is shown in Figure 12-237.

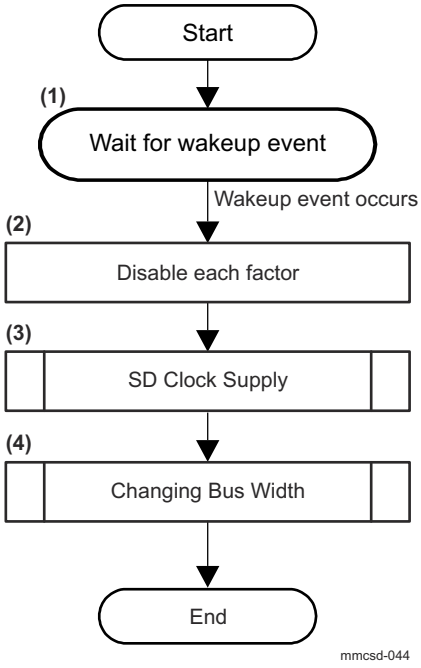


Figure 12-237. Wakeup from Standby

(1) Wait for wakeup event.

Copyright © 2026 Texas Instruments Incorporated

(2) Set the enable bits of each wakeup event factor to 0 in the MMCSD0_WAKEUP_CONTROL register and then clear event statuses in the MMCSD0_NORMAL_INTR_STS register. If necessary, set the MMCSD0_NORMAL_INTR_SIG_ENA register.

(3) Execute SD Clock Supply Sequence (see Section 12.3.5.5.1.2.2, *SD Clock Supply and Stop Sequence*).

(4) Set the SD Bus width (see Section 12.3.5.5.1.4, *Changing Bus Width*).

12.3.5.5.1.12 Suspend/Resume (Optional, Not Supported from Version 4.00)

If a SD card supports suspend and resume functionality, then the Host Controller can initiate suspend and resume. It is necessary for both the Host Controller and the SD card to support the function of "Read Wait". ADMA operation does not support this function.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.12.1 Suspend Sequence

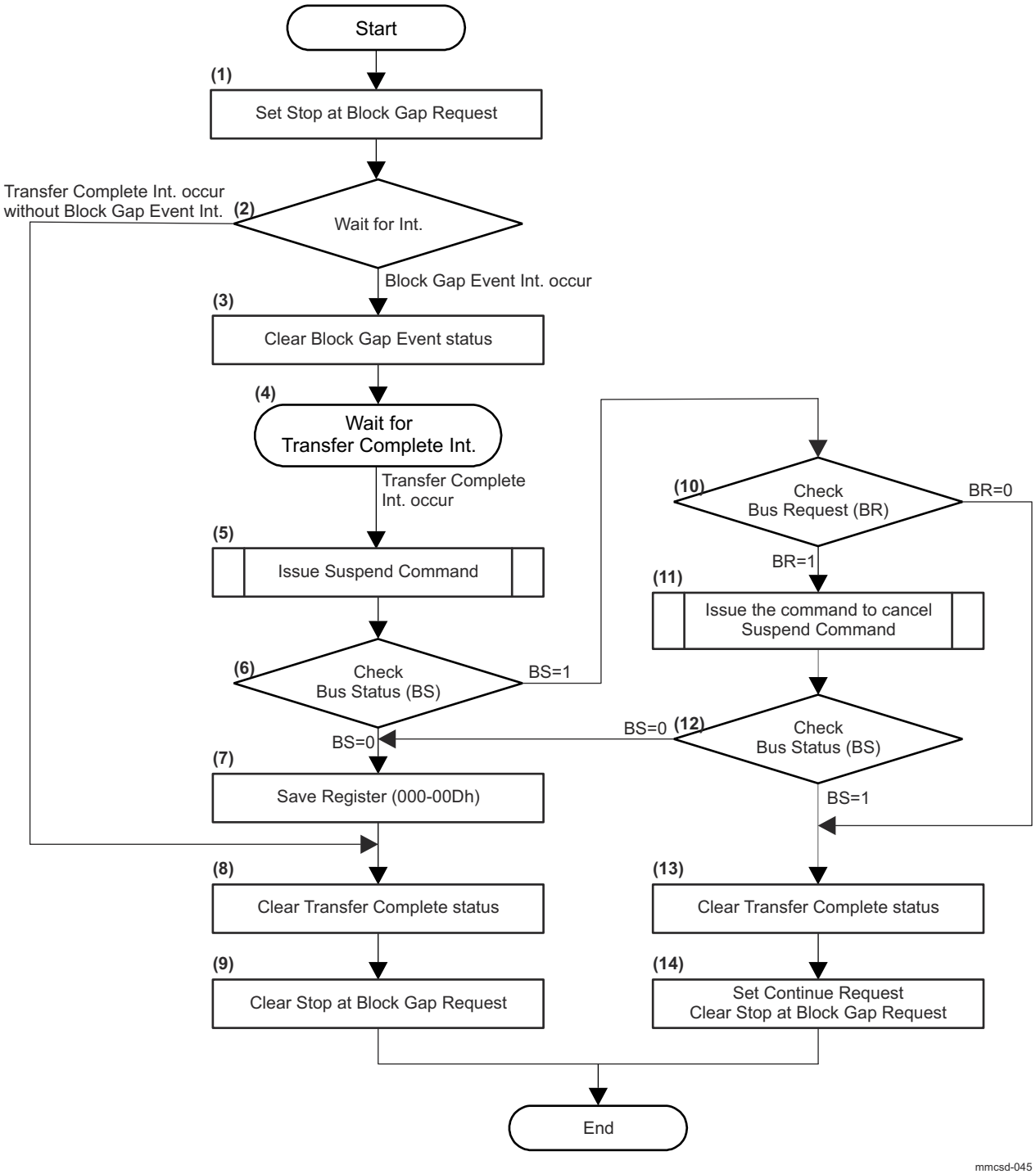


Figure 12-238. The Sequence for Suspend

(1) Set MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit to 1 to stop the SD transaction.

(2) Wait for an Interrupt. If MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit is set to 0 and MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit is set to 1, go to step (8). If MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit is set to 1, go to step (3).

Copyright © 2026 Texas Instruments Incorporated

(3) Set MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit to 1 to clear this bit.

(4) Wait for the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS_ENA[1] XFER_COMPLETE).

(5) Issue the Suspend Command in accordance with Section 12.3.5.5.1.7.1, *Transaction Control without Data Transfer Using DAT Line*.

(6) Check the BS value of the response data. If BS is 0, go to step (7). If BS is 1, go to step (10).

(7) Save the register .

(8) Set MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit to 1 to clear this bit.

(9) Set MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit to 0 to clear this bit.

(10) Check the BR value of the response data. If BR is 1, go to step (11). If BR is 0, go to step (13).

(11) Issues the command to cancel the previous suspend command in accordance with Section 12.3.5.5.1.7.1, *Transaction Control without Data Transfer Using DAT Line*.

(12) Check the BS value of the response data. If BS is 0, go to step (7). If BS is 1, go to step (13).

(13) Set MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit to 1 to clear this bit.

(14) Set MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit to 1 to continue the transaction. At the same time, write 0 to MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit to clear this bit.

Conditions			Suspend/Resume Function	
Host Suspend/Resume Support	**Card Suspend/Resume Support**	**Card Read Wait Support**	**Write Suspend/Resume**	**Read Suspend/Resume**
Not supported	Don’t care	Don’t care	Cannot be used	Cannot be used
Supported	Not supported	Don’t care	Cannot be used	Cannot be used
Supported	Supported	Not supported	Can be used	Cannot be used
Supported	Supported	Supported	Can be used	Can be used

mmcsd-046

Figure 12-239. Suspend/Resume Condition

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.1.12.2 Resume Sequence

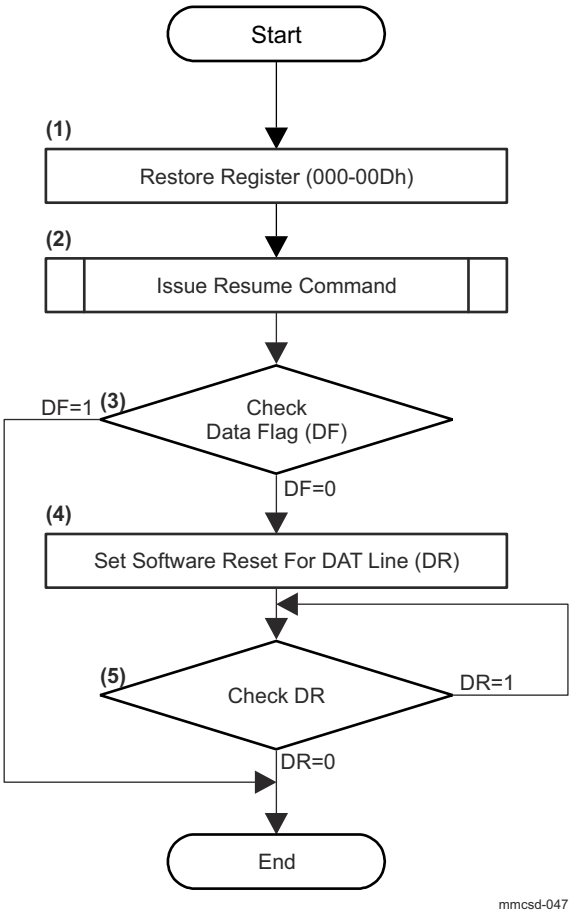


Figure 12-240. The Sequence for Resume

(1) Restore the register .

(2) Issue the Resume Command .

(3) Check the DF value of the response data. If DF is 0, go to step (4). If DF is 1, go to "End".

(4) Set MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit to 1 for software reset of the DAT line.

(5) Check MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit. If MMCSD0_SOFTWARE_RESET[2] SWRST_FOR_DAT bit is 0, go to "End". If it is 1, go to step (5).

12.3.5.5.1.12.3 Stop At Block Gap/Continue Timing for Read Transaction

The timing of Stop At Block Gap Request and Continue Request. The Transfer Complete interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE) is always generated by setting MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit where data transfer is stopped. However, generation of the Block Gap Event interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT) is dependent on whether the last data block is sent or not. Block Gap Event is not generated If all data blocks are transferred (the last block is transferred). It is not necessary to enable Block Gap Event interrupt. The status can be checked when transfer complete interrupt is detected. It is not necessary to use Continue Request (MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE) if Block Gap Event status is not set because there is no further data to be transferred. If Read Wait is not supported, Host Controller stops SD Clock at the block gap.

Implementation Note:

Copyright © 2026 Texas Instruments Incorporated

The MMCSD0_BLOCK_GAP_CONTROL[2] RDWAIT_CTRL, MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE, MMCSD0_PRESENTSTATE[9] RD_XFER_ACTIVE bits shall be set and cleared by the Host Controller.

The MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit shall be set and cleared by the Host Driver. The MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit shall be set by the Host Driver and be cleared by the Host Controller. The MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit and MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit shall be set by the Host Controller and be cleared by the Host Driver.

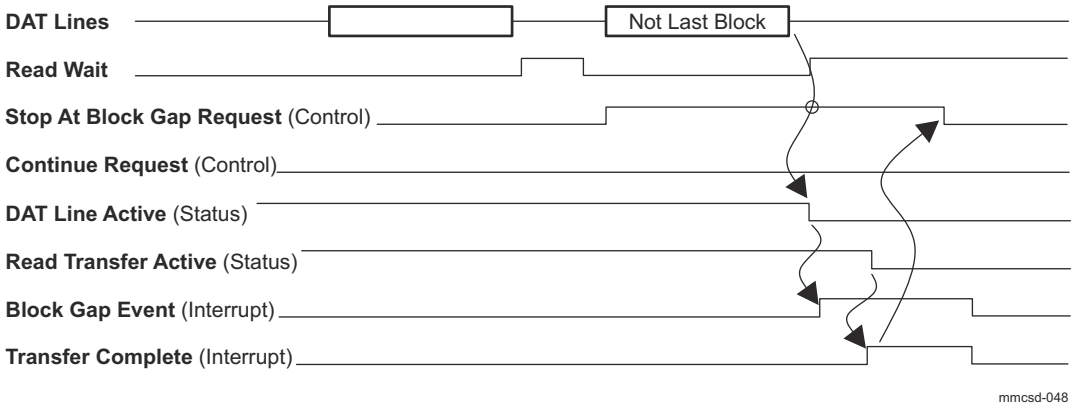


Figure 12-241. Wait Read Transfer by Stop At Block Gap Request

The Host Controller can accept a Stop At Block Gap Request (MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP) when all the following conditions are met.

(1) It is at the block gap.

(2) The Host Controller can assert read wait or it is already asserted.

(3) The MMCSD0_BLOCK_GAP_CONTROL[2] RDWAIT_CTRL bit is set to 1.

After accepting the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit:

(1) Clear MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and generate the Block Gap Event Interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT).

(2) After all valid data has been read (No valid read data remains in the Host Controller), clear the MMCSD0_PRESENTSTATE[9] RD_XFER_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE).

(3) After accepting MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit, clear the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

Copyright © 2026 Texas Instruments Incorporated

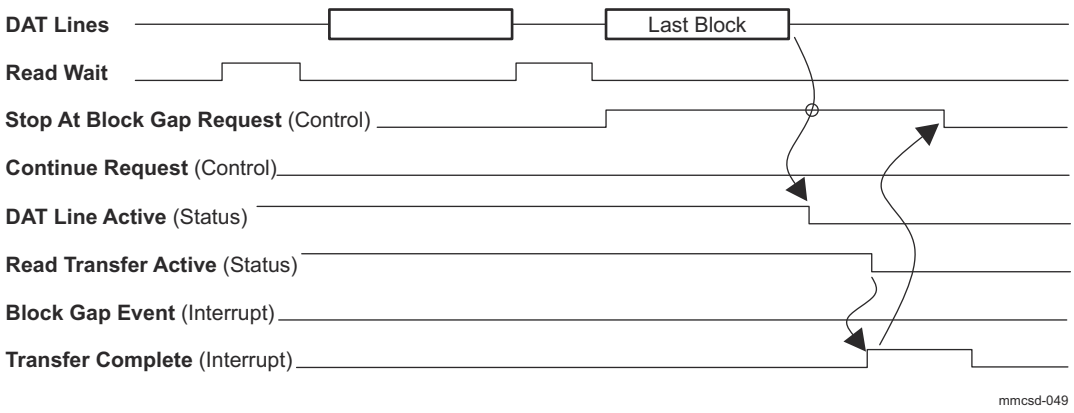


Figure 12-242. Stop At Block Gap Request is Not Accepted at the Last Block of the Read Transfer

If the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit is set to 1 during the last block transfer, the Host Controller shall not accept the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit and stops the transaction normally. The Block Gap Event Interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT) is not generated. When the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE) is generated, and if the MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit status is not set to 1, the driver shall clear the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

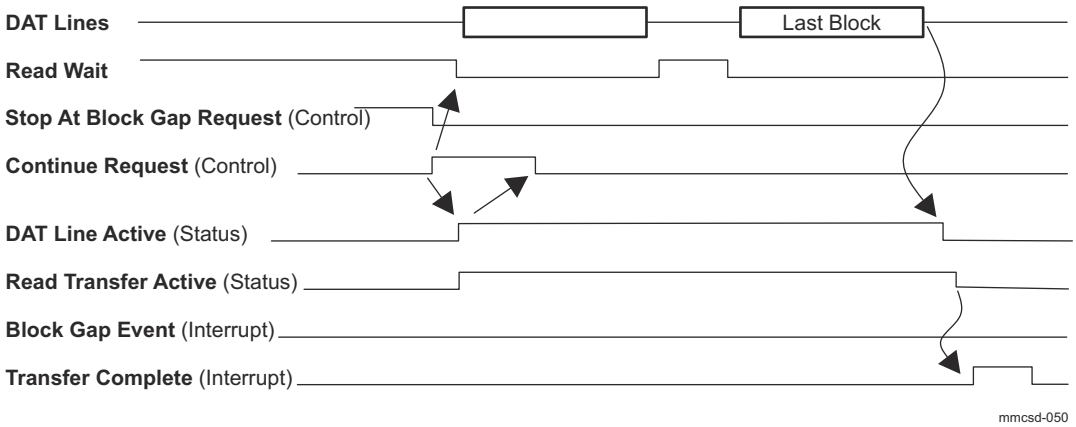


Figure 12-243. Continue Read Transfer by Continue Request

To restart a stopped data transfer, set the MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit to 1 (the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit shall be set to 0).

After accepting the MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit:

(1) Release MMCSD0_BLOCK_GAP_CONTROL[2] RDWAIT_CTRL bit (if the data block can accept the next data).

(2) Set the MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and the MMCSD0_PRESENTSTATE[9] RD_XFER_ACTIVE bit.

(3) The MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit is automatically cleared by (2).

The end of the read transfer is specified by data length.

(1) Clear the MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and do not generate the Block Gap Event Interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit).

Copyright © 2026 Texas Instruments Incorporated

(2) After all valid data has been read (No valid read data remains in the Host Controller), clear the MMCSD0_PRESENTSTATE[9] RD_XFER_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit).

12.3.5.5.1.12.4 Stop At Block Gap/Continue Timing for Write Transaction

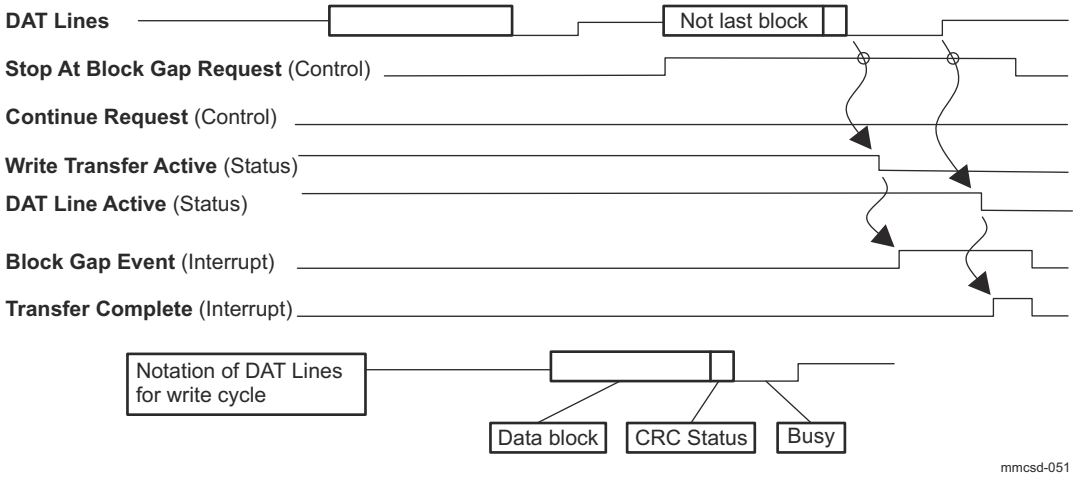


Figure 12-244. Wait Write Transfer by Stop At Block Gap Request

The Host Controller can accept the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit when matches all following conditions:

(1) It is at the block gap.

(2) No valid write data remains in the Host Controller.

After accepting the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

(1) Clear the MMCSD0_PRESENTSTATE[8] WR_XFER_ACTIVE bit and generate the Block Gap Event Interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT bit).

(2) After the busy signal is released, clear the MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and generate the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit).

(3) After accepting the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit), clear the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

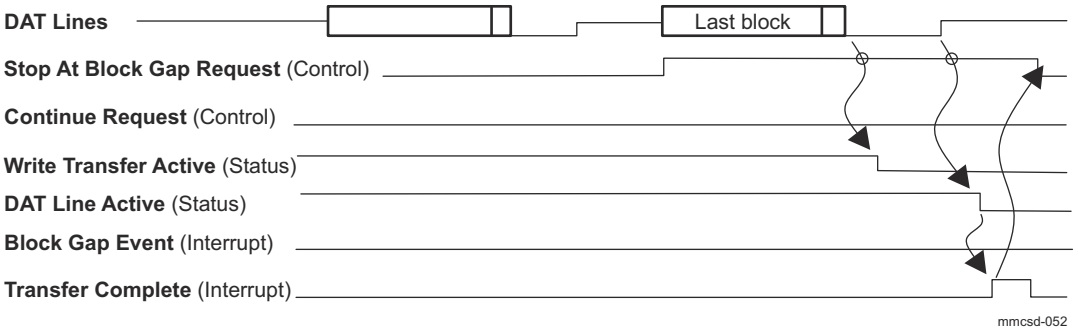


Figure 12-245. Stop At Block Gap Request is Not Accepted at the Last Block of the Write Transfer

If the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit is set to 1 during the last block transfer, the Host Controller shall not accept the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit and terminates the transaction normally. The Block Gap Event Interrupt (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVENT) is not generated. When the Transfer

Copyright © 2026 Texas Instruments Incorporated

Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE) is generated, and if the MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVEN bit is not set to 1, the driver shall clear the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit.

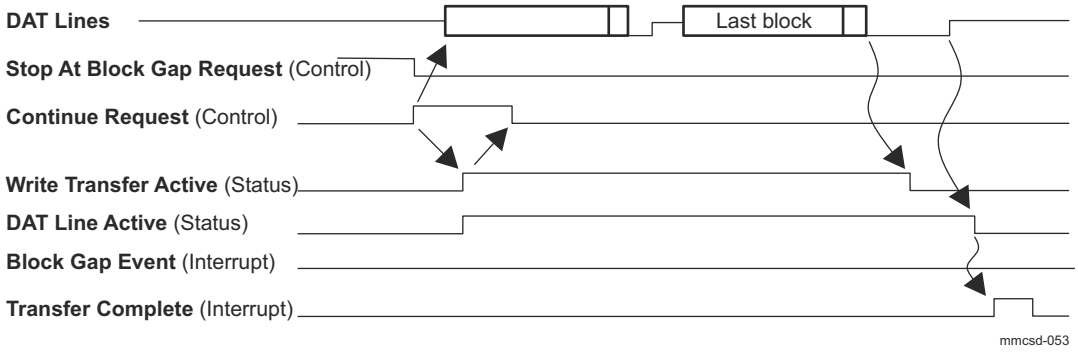


Figure 12-246. Continue Write Transfer by Continue Request

To restart a stopped data transfer, set the MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit to 1 (the MMCSD0_BLOCK_GAP_CONTROL[0] STOP_AT_BLK_GAP bit shall be set to 0).

After accepting the MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit:

(1) Set the MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and MMCSD0_PRESENTSTATE[8] WR_XFER_ACTIVE bit.

(2) The MMCSD0_BLOCK_GAP_CONTROL[1] CONTINUE bit is automatically cleared by (1).

The end of transfer is specified by data length.

(1) Clear the MMCSD0_PRESENTSTATE[8] WR_XFER_ACTIVE bit, and do not generate the (MMCSD0_NORMAL_INTR_STS[2] BLK_GAP_EVEN).

(2) After the busy signal is released, clear the MMCSD0_PRESENTSTATE[2] DATA_LINE_ACTIVE bit and generates the Transfer Complete Interrupt (MMCSD0_NORMAL_INTR_STS[1] XFER_COMPLETE bit).

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.2 Driver Flow Sequence

Note

UHSII is not supported. For more information, see *Not Supported Features*.

12.3.5.5.2.1 Host Controller Setup and Card Detection

12.3.5.5.2.1.1 Host Controller Setup Sequence

Figure 12-247 and Table 12-232 show a Host Controller setup sequence.

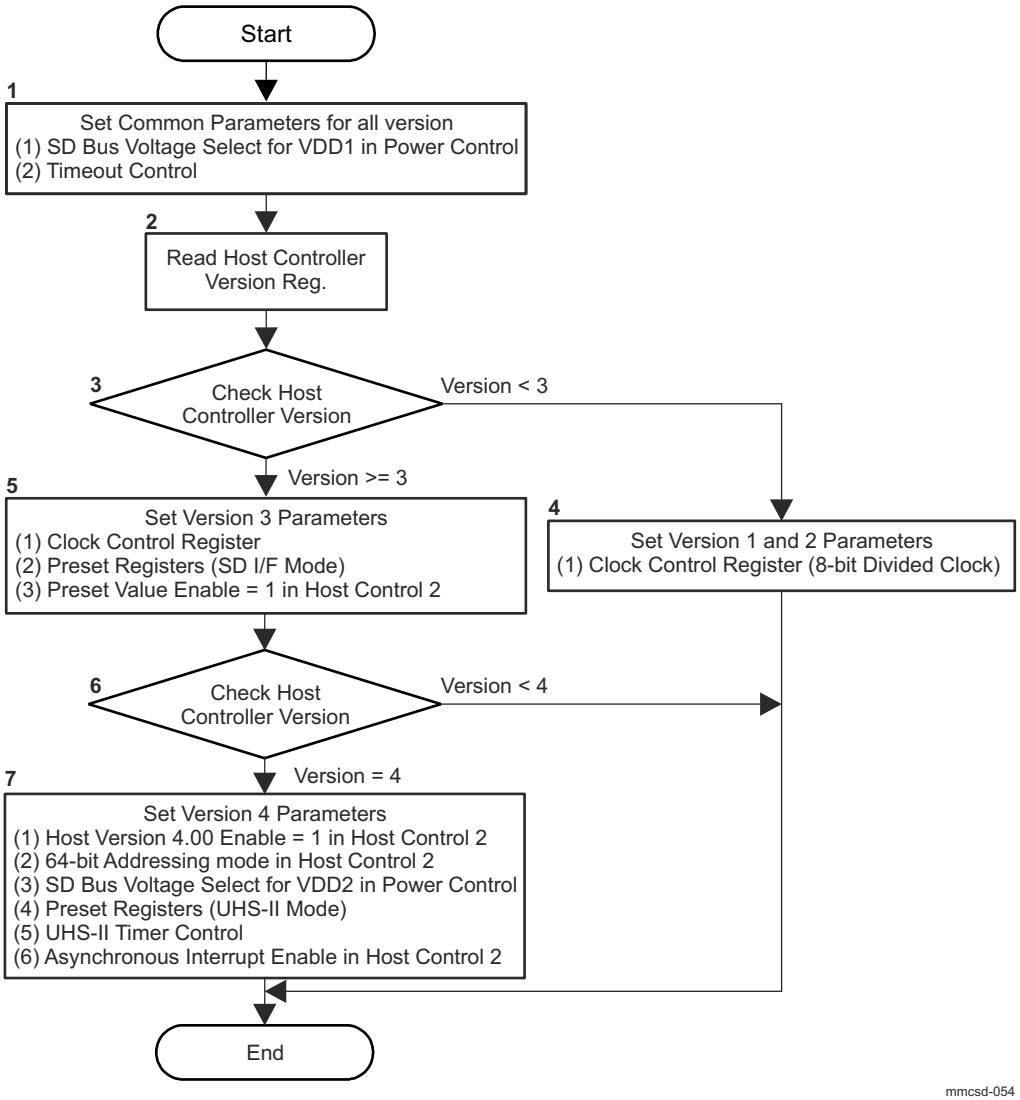


Figure 12-247. Host Controller Setup Sequence

Table 12-232. Host Controller Setup Sequence

Step	Description
1	Set parameters for all Host Controller version. Set the MMCSD0_POWER_CONTROL[3-1] SD_BUS_VOLTAGE and MMCSD0_TIMEOUT_CONTROL[3-0] COUNTER_VALUE bit fields.
2	Read the MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM bit field.
3	If Specification Version number (MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM) is less than version 3, go to step (4), if it is version 3 or later go to step (5).
4	Set the MMCSD0_CLOCK_CONTROL register using 8-bit Divided Clock mode.

Copyright © 2026 Texas Instruments Incorporated

Table 12-232. Host Controller Setup Sequence (continued)

Step	Description
5	Set Version 3 parameters. The MMCSD0_CLOCK_CONTROL register is sets in 10-bit Divided Clock Mode or Programmable Clock Mode. If Clock Multiplier in the MMCSD0_CAPABILITIES register is not zero, Programmable Clock Mode should be used. If Preset Value is used, set Preset Values of SD I/F Modes in the Preset Value register (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10) and set the MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit to 1.
6	If Specification Version number (MMCSD0_HOST_CONTROLLER_VER[7-0] SPEC_VER_NUM) is version 4, go to step (7), if it is less than version 4, exits.
7	Set Version 4 parameters. Set the MMCSD0_HOST_CONTROL2[12] HOST_VER40_ENA bit to 1. If the MMCSD0_CAPABILITIES[27] ADDR_64BIT_SUPPORT_V4 bit is set to 1, set the MMCSD0_HOST_CONTROL2[13] BIT64_ADDRESSING bit to 1. If UHS-II Support is set to 1 and 1.8 V VDD2 Support is set to 1 in the MMCSD0_CAPABILITIES register, set SD Bus Voltage Select for VDD2 to 1.8 V mode in the MMCSD0_POWER_CONTROL register, set preset value for UHS-II Mode to Preset Value register (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10) and set Timeout Counter Value for CMD_RES and Timeout Counter Value for Deadlock in the MMCSD0_UHS2_TIMER_CONTROL register based on Timeout Clock Frequency and Timeout Clock Unit in the MMCSD0_CAPABILITIES register. If Asynchronous Interrupt Support in the MMCSD0_CAPABILITIES register is set to 1, set Asynchronous Interrupt Enable to 1 in the MMCSD0_HOST_CONTROL2 register.

12.3.5.5.2.1.2 Card Interface Detection Sequence

This procedure is invoked by the Card Insertion interrupt. Figure 12-248 and Table 12-233 show a card interface detection sequence.

Copyright © 2026 Texas Instruments Incorporated

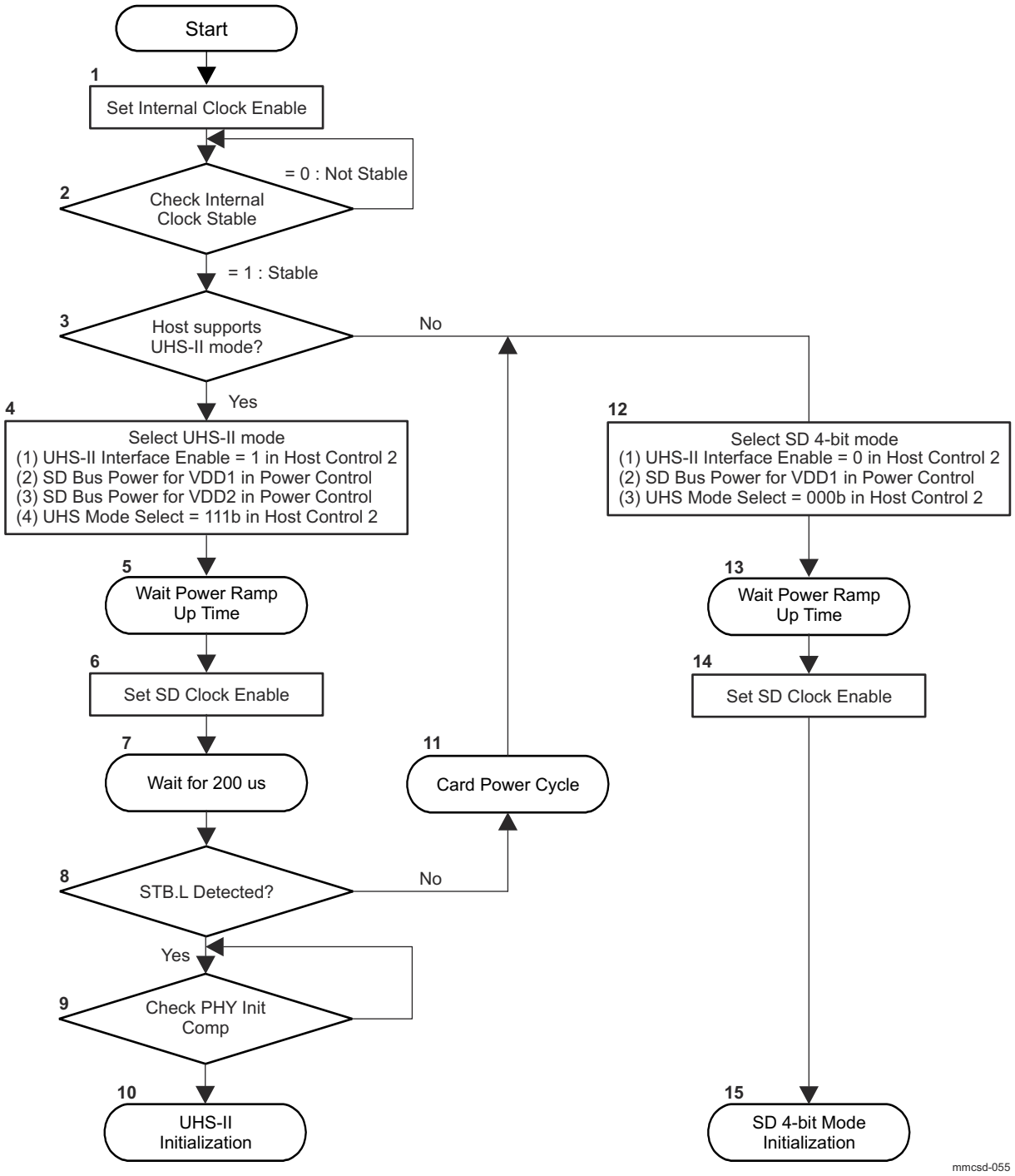


Figure 12-248. Card Interface Detection Sequence

Table 12-233. Card Interface Detection Sequence

Step	Description
1	Set Internal Clock Enable to 1 in the MMCSD0_CLOCK_CONTROL register.
2	Wait until Internal Clock Stable is set to 1 in the MMCSD0_CLOCK_CONTROL register.
3	If Host Supports UHS-II, go to step (4) else go to step (12).
4	Try to initialize a card to UHS-II mode. Set UHS-II Interface Enable to 1 in the MMCSD0_HOST_CONTROL2 register, set SD Bus Power for VDD1 and SD Bus Power for VDD2 to 1 in the MMCSD0_POWER_CONTROL register, and set UHS Mode Select to 111b in the MMCSD0_HOST_CONTROL2 register.

Copyright © 2026 Texas Instruments Incorporated

Table 12-233. Card Interface Detection Sequence (continued)

Step	Description
5	Wait power ramp up time. It is dependent on a Host System.
6	Set SD Clock Enable to 1 in the MMCSD0_CLOCK_CONTROL register.
7	Wait 200 µs to check a card supports UHS-II mode.
8	Check STB.L Detection in the MMCSD0_PRESENTSTATE register. If UHS-II IF is detected, go to step (9). If UHS-II IF is not detected, go to step (11).
9	Wait until Lane Synchronization in the MMCSD0_PRESENTSTATE register is set to 1 (PHY Initialization is completed).
10	Perform UHS-II initialization.
11	Perform card power cycle.
12	Try to initialize a card to SD 4-bit mode. Set UHS-II Interface Enable to 0 in the MMCSD0_HOST_CONTROL2 register, set SD Bus Power for VDD1 to 1 in the MMCSD0_POWER_CONTROL register, and set UHS Mode Select to 000b in the MMCSD0_HOST_CONTROL2 register.
13	Wait power ramp up time. It is dependent on a Host System.
14	Set SD Clock Enable to 1 in the MMCSD0_CLOCK_CONTROL register.
15	Perform SD 4-bit mode initialization. Refer to Section 12.3.5.5.1.6, *Card Initialization and Identification (for SD I/F)*.

12.3.5.5.2.2 Boot Operation

The boot operation sequence is shown in Figure 12-249.

Copyright © 2026 Texas Instruments Incorporated

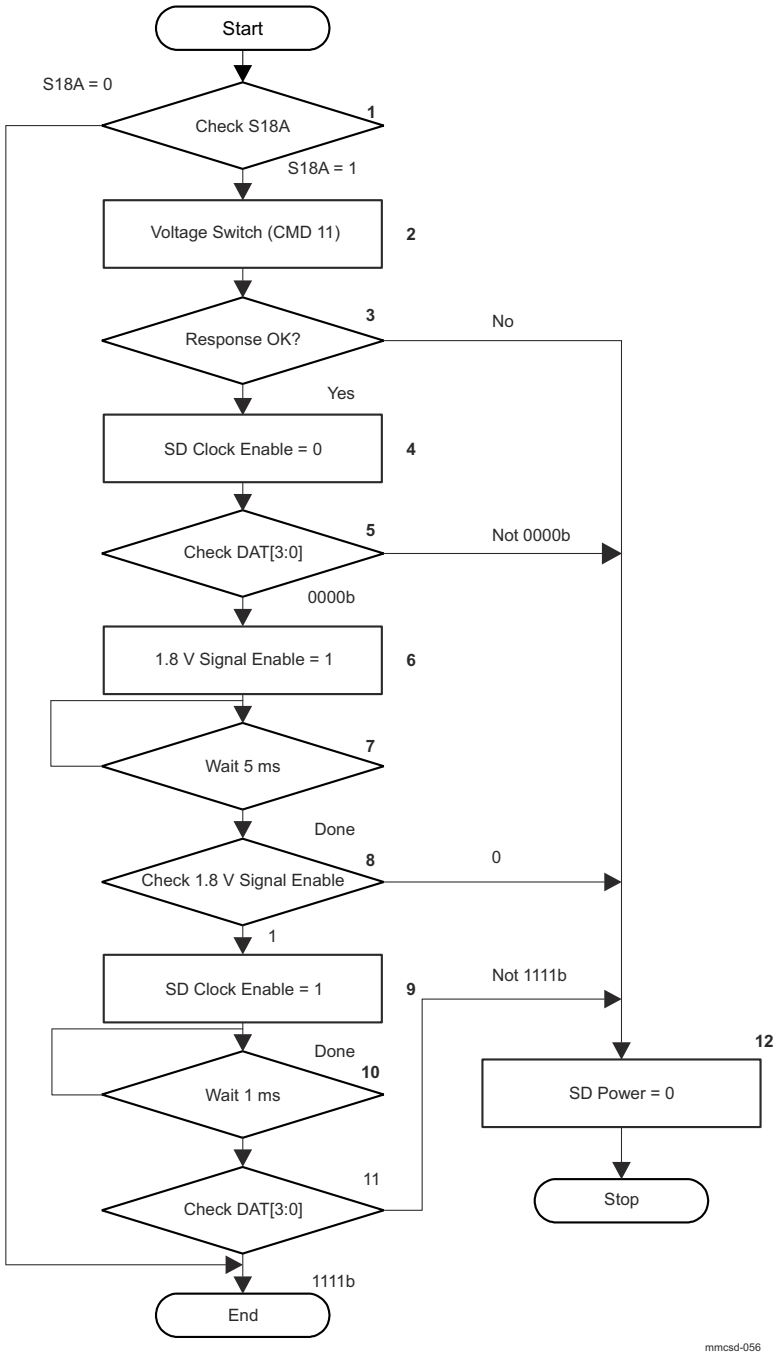


Figure 12-249. Boot Operation Sequence

12.3.5.5.2.2.1 Normal Boot Operation: (For Legacy eMMC 5.0)

The Host driver writes the boot timeout value into MMCSD0_BOOT_TIMEOUT_CONTROL register as per the eMMC4.3+ spec.

When the Host driver sets the "BOOT_ENABLE" to 1 in MMCSD0_BLOCK_GAP_CONTROL register and "DATA_XFER_DIR" bit to "1" in MMCSD0_TRANSFER_MODE register, the Host controller drives the CMD line to "0" for boot operation.

If the Host controller is configured to wait for boot acknowledgement from the eMMC4.3+ device, the controller receives the boot acknowledgement and asserts the "RCV_BOOT_ACK" interrupt to the driver

Copyright © 2026 Texas Instruments Incorporated

(MMCSD0_NORMAL_INTR_STS[13] RCV_BOOT_ACK). If the Host controller doesn't receive the boot acknowledgement from the device within the timeout value, the Host controller will assert the data timeout error interrupt (MMCSD0_ERROR_INTR_STS[4] DATA_TIMEOUT).

After servicing the boot acknowledge interrupt or data timeout error interrupt, the driver programs the MMCSD0_BOOT_TIMEOUT_CONTROL register with boot data timeout value.

The eMMC4.3+ device starts sending the boot data on the data line and Host controller sends the same to system whenever a block of data is received from device. The Host controller terminates the boot operation when the programmed number of blocks is transferred to the System. The Driver can writes "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register.

The boot operation can also be terminated in between the boot transfer (the Driver can set the "BOOT_ENABLE" as "0" anytime to disable the boot operation). After the boot termination, the Host controller asserts soft reset for CMD line and DATA line internally once the driver clears the BOOT_COMPLETE interrupt (MMCSD0_NORMAL_INTR_STS[14] BOOT_COMPLETE), to reset all the state machines to idle state.

There is no need to perform error recovery sequence in case data timeout interrupt occurs during boot operation (ignore sending abort command, soft reset and just clear the timeout interrupt and proceed with the boot flow).

If the device sends wrong acknowledgement to the Host, the Host controller will assert Data CRC Error interrupt (MMCSD0_ERROR_INTR_STS[5] DATA_CRC) on the Driver. In this case, the Host driver has to stop the boot mode operation by setting "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register and write soft reset for CMD and data line.

If the device sends end bit as "0" in acknowledgement to the Host, the Host controller will assert Data End Bit Error interrupt (MMCSD0_ERROR_INTR_STS[6] DATA_ENDBIT) on the Driver.

The Host driver stops the boot mode operation by setting "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register and write soft reset for CMD and data line.

Note: Enable the MMCSD0_BLOCK_GAP_CONTROL[7] BOOT_ACK_ENA bit during boot operation. If failed, the controller will not wait for boot acknowledge from the card and send out data CRC error when the card sends boot acknowledgement first and followed by boot data.

12.3.5.5.2.2.2 Alternate Boot Operation (For Legacy eMMC 5.0):

The Host driver writes the boot timeout value as per the eMMC4.3+ spec into MMCSD0_BOOT_TIMEOUT_CONTROL register.

If the "ALT_BOOT_MODE" and "BOOT_ENABLE" is set to 1 in MMCSD0_BLOCK_GAP_CONTROL register and "DATA_XFER_DIR" bit is set to 1 in MMCSD0_TRANSFER_MODE register, the host controller drives the CMD0 (0xFFFFFFFA) on the CMD line.

The system shall wait for command complete interrupt before "RCV_BOOT_ACK" interrupt (MMCSD0_NORMAL_INTR_STS[13] RCV_BOOT_ACK).

If the Host controller is configured to wait for boot acknowledgement from the eMMC4.3+ device, the controller receives the boot acknowledgement and asserts the "RCV_BOOT_ACK" interrupt to the driver.

The eMMC4.3+ device starts sending the boot data on the data line and Host controller sends the same to system whenever a block of data is received from device. The System needs to send CMD0 to inform the device about the boot operation complete. The system shall program MMCSD0_COMMAND register with argument "00000000h" for the CMD0 and waits for command complete.

The Host controller terminates the boot operation when the programmed number of blocks are transferred to the System and the Driver shall send CMD0 to eMMC card and then program "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register.

The boot operation can be terminated at any time (the Driver can send CMD0 and set the "BOOT_ENABLE" as "0" anytime to disable the boot operation). After the boot termination, all state machines in the Host controller need to move to IDLE state. The Host controller asserts the soft reset for CMD and data line internally when the

Copyright © 2026 Texas Instruments Incorporated

driver clears the BOOT_COMPLETE interrupt (MMCSD0_NORMAL_INTR_STS[14] BOOT_COMPLETE) and all the state machine goes to the idle state.

If the Host controller doesn't receive the acknowledgement from the device with in timeout value the Host controller will assert the data timeout error interrupt (MMCSD0_ERROR_INTR_STS[4] DATA_TIMEOUT).

The driver programs the MMCSD0_BOOT_TIMEOUT_CONTROL register with boot data timeout value.

There is no need to perform error recovery sequence in case of data timeout interrupt as mentioned above. If the device sends wrong acknowledgement to the Host, then Host controller will assert Data CRC Error interrupt to the Driver (MMCSD0_ERROR_INTR_STS[5] DATA_CRC). The Host driver has to stop the boot mode operation by setting "ALT_BOOT_MODE" and "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register and write soft reset for CMD and data line.

If the device sends end bit as "0" in acknowledgement to the Host, the Host controller asserts Data End Bit Error interrupt (MMCSD0_ERROR_INTR_STS[6] DATA_ENDBIT) on the Driver. The Host driver stops the boot mode operation by setting "ALT_BOOT_MODE" and "BOOT_ENABLE" as "0" in MMCSD0_BLOCK_GAP_CONTROL register and write soft reset for CMD and data line.

Note: Enable the MMCSD0_BLOCK_GAP_CONTROL[7] BOOT_ACK_ENA bit during boot operation . If failed, the controller will not wait for boot acknowledge from the card and send out data CRC error when the card sends boot acknowledgement first and followed by boot data.

12.3.5.5.2.2.3 Boot Code Chunk Read Operation (For Legacy eMMC 5.0):

The following figure explains the boot code read done as chunks of data (the Driver can read the boot data as 2K, 4K, 1K and 8K etc. instead of 128K at a time

Copyright © 2026 Texas Instruments Incorporated

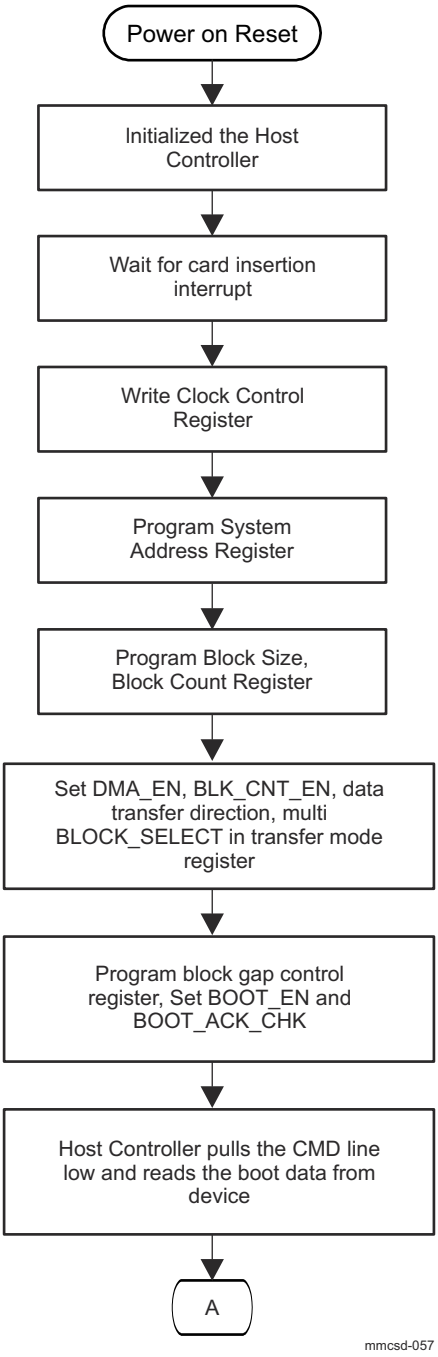


Figure 12-250. Boot Code Access Flow Diagram (1)

Copyright © 2026 Texas Instruments Incorporated

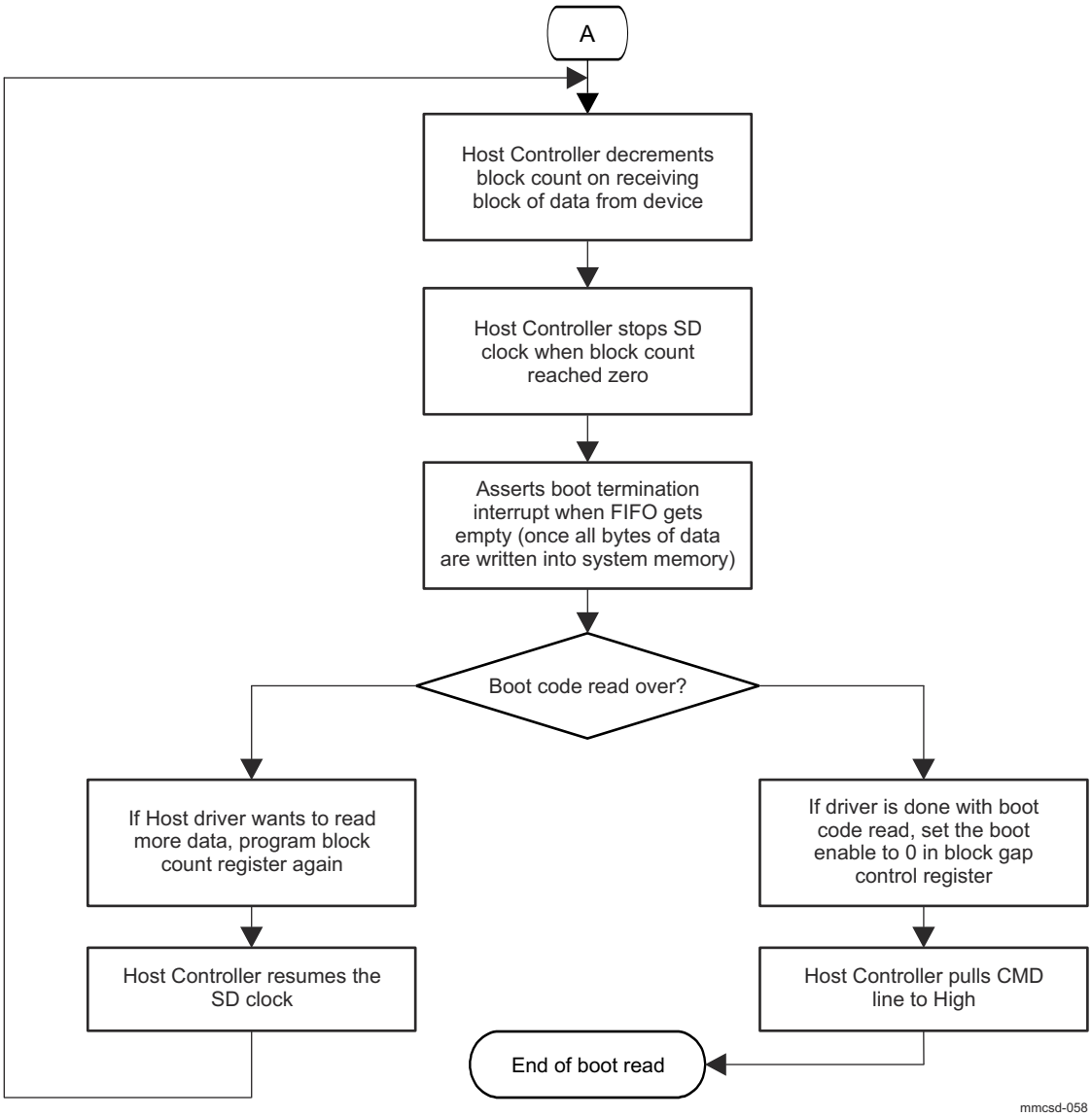


Figure 12-251. Boot Code Access Flow Diagram (2)

12.3.5.5.2.3 Retuning procedure (For Legacy Interface)

12.3.5.5.2.3.1 Sampling Clock Tuning

The SD bus can be operating in high clock frequency mode and then the data window from the card on CMD and DAT[3:0] lines gets smaller. The position of the data window will vary depending on the card and host system implementation. Therefore, the Host Controller shall support a tuning circuit when SDR104 or SDR50.

The Host Controller shall support a tuning circuit when SDR104 or SDR50 (if Use Tuning for SDR50 is set to 1 in the MMCSD0_CAPABILITIES register) is supported by executing the tuning procedure and adjusting the sampling clock. Execute Tuning and Sampling Clock Select in the MMCSD0_HOST_CONTROL2 register are used to control the tuning circuit.

See MMC SW Tuning Algorithm for more information.

12.3.5.5.2.3.2 Tuning Sequence

The Figure 12-252 defines sampling clock tuning procedure supported by Host Controller. In default, for lower frequency operation, fixed sampling clock is used to receive signals on CMD and DAT[3:0]. Before using

Copyright © 2026 Texas Instruments Incorporated

SDR104, sampling clock tuning is required. Start of sampling clock tuning is requested by setting Execute Tuning to 1 and Sampling Clock Select to 0.

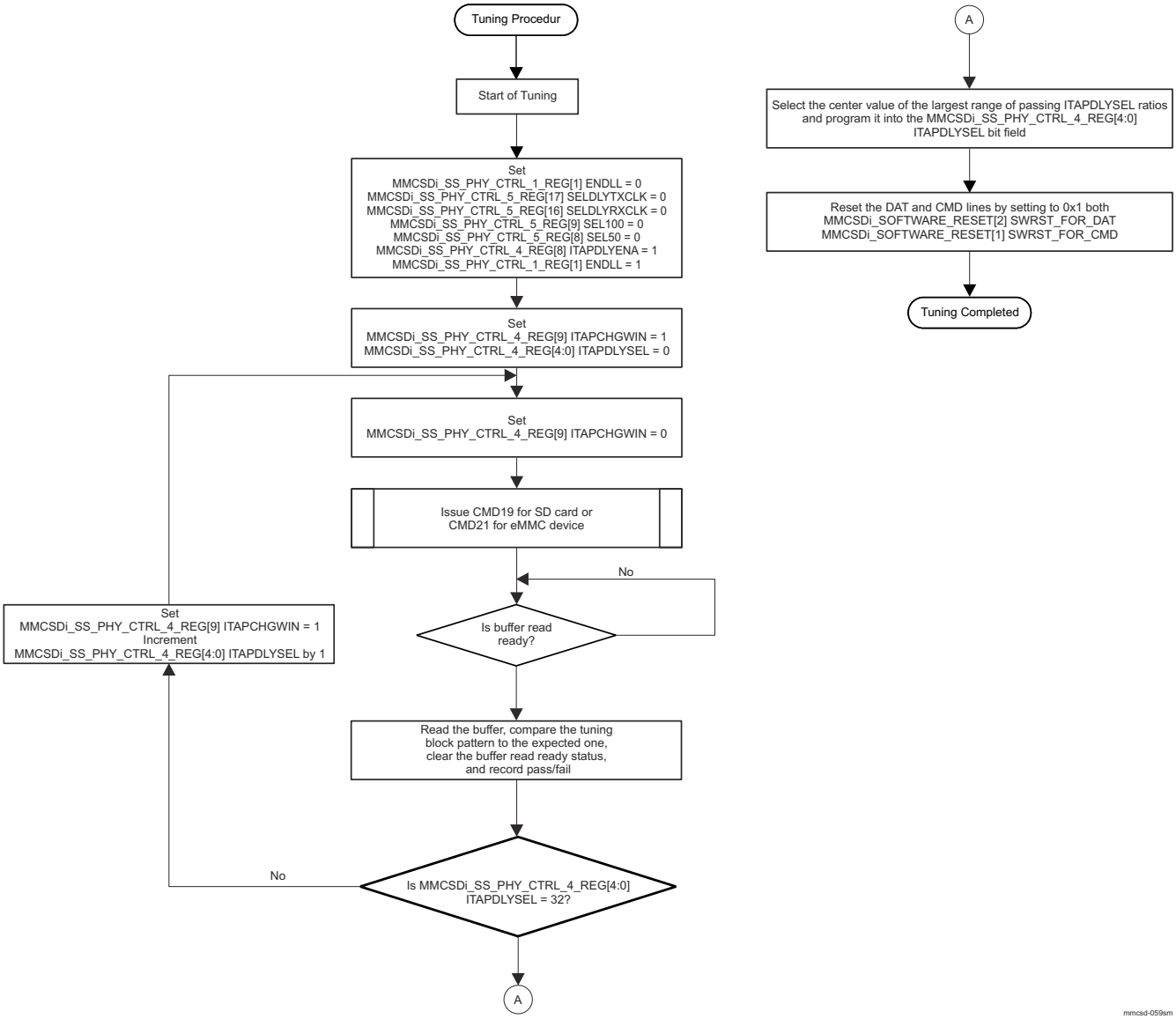


Figure 12-252. MMCSD Clock Tuning

Host driver issue CMD19 repeatedly until the host controller resets Execute Tuning to 0. Host Controller resets Execute Tuning to 0 when tuning is completed or tuning is not completed within 40 times. Host Driver can abort this loop by 40 times CMD19 issue or 150 ms time-out.

If tuning is completed successfully, Host Controller set Sampling Clock Select to 1 and this means the Host Controller start to use tuned sampling clock. If tuning is failed, Host Controller keeps Sampling Clock Select to 0. By writing Sampling Clock Select to 0, sampling clock is switched from tuned sampling clock to fixed sampling clock. Re-tuning time would be smaller than the first tuning time. CMD19 response errors are not indicated while tuning is performed.

The clock tuning tap delay values are selected using Variable sampling point detection. Fixed tap delay value is used for fixed tuning clock method.

Copyright © 2026 Texas Instruments Incorporated

12.3.5.5.2.3.3 Re-Tuning Modes

The re-tuning timing is specified by two methods: Re-Tuning Request generated by Host Controller and expiration of a re-tuning timer prepared by Host Driver.

(1) Re-Tuning Mode 1 The host controller does not have any internal logic to detect when the re-tuning needs to be performed. In this case, the Host Driver should maintain all re-tuning timings by using a Re-Tuning Timer. To enable inserting the re-tuning procedure during data transfers, the data length per read/write command shall be limited up to 4 MB.

(2) Re-Tuning Mode 2 The host controller has the capability to indicate the re-tuning timing by Re-Tuning Request during data transfers. Then the data length per read/write command shall be limited up to 4 MB. During non data transfer, re-tuning timing is determined by either Re-Tuning Request or Re-Tuning Timer. If Re-Tuning Request is used, Re-Tuning Timer should be disabled.

12.3.5.5.2.4 Command Queuing Driver Flow Sequence

12.3.5.5.2.4.1 Command Queuing Initialization Sequence

Figure 12-253 and Table 12-234 show a Command Queuing initialization sequence.

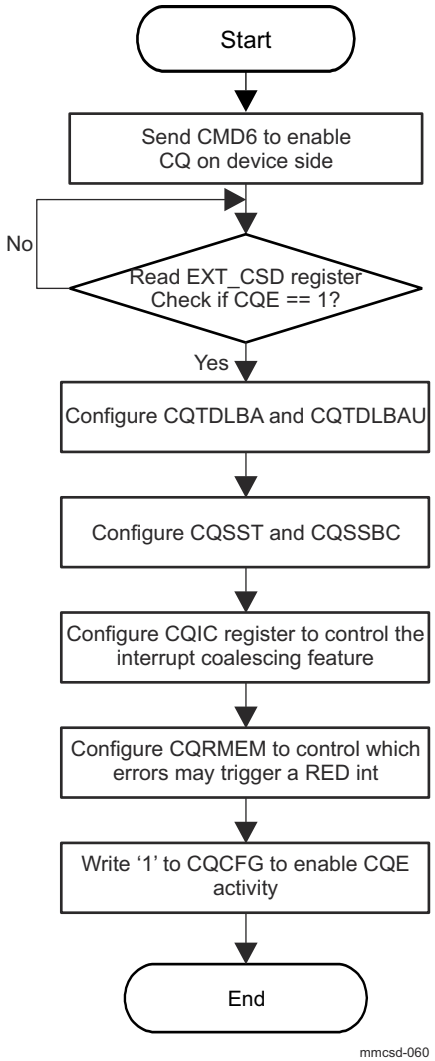


Figure 12-253. Command Queuing Initialization Sequence

Copyright © 2026 Texas Instruments Incorporated

Table 12-234. Command Queuing Initialization Sequence

Step	Description
1	Initialize and enable Command Queueing in the device.
2	Configure Task Descriptor size in MMCSD0_CQ_CONFIG register.
3	Configure MMCSD0_CQ_TDL_BASE_ADDR and MMCSD0_CQ_TDL_BASE_ADDR_UPBITS registers to point to the memory location allocated to the TDL in host memory.
4	Configure CQSST and CQSSBC to control when SEND_QUEUE_STATUS commands are sent to the device by CQE.
5	Configure MMCSD0_CQ_INTR_COALESCING register to control the interrupt coalescing feature: enable/disable, set interrupt count and timer protection.
6	Configure MMCSD0_CQ_RESP_ERR_MASK register to control which errors may trigger a RED interrupt (if different from reset values).
7	Write '1' to MMCSD0_CQ_CONFIG register to enable CQE activity.

12.3.5.5.2.4.2 Task Issuance Sequence

Figure 12-254 and Table 12-235 show a task issuance sequence.

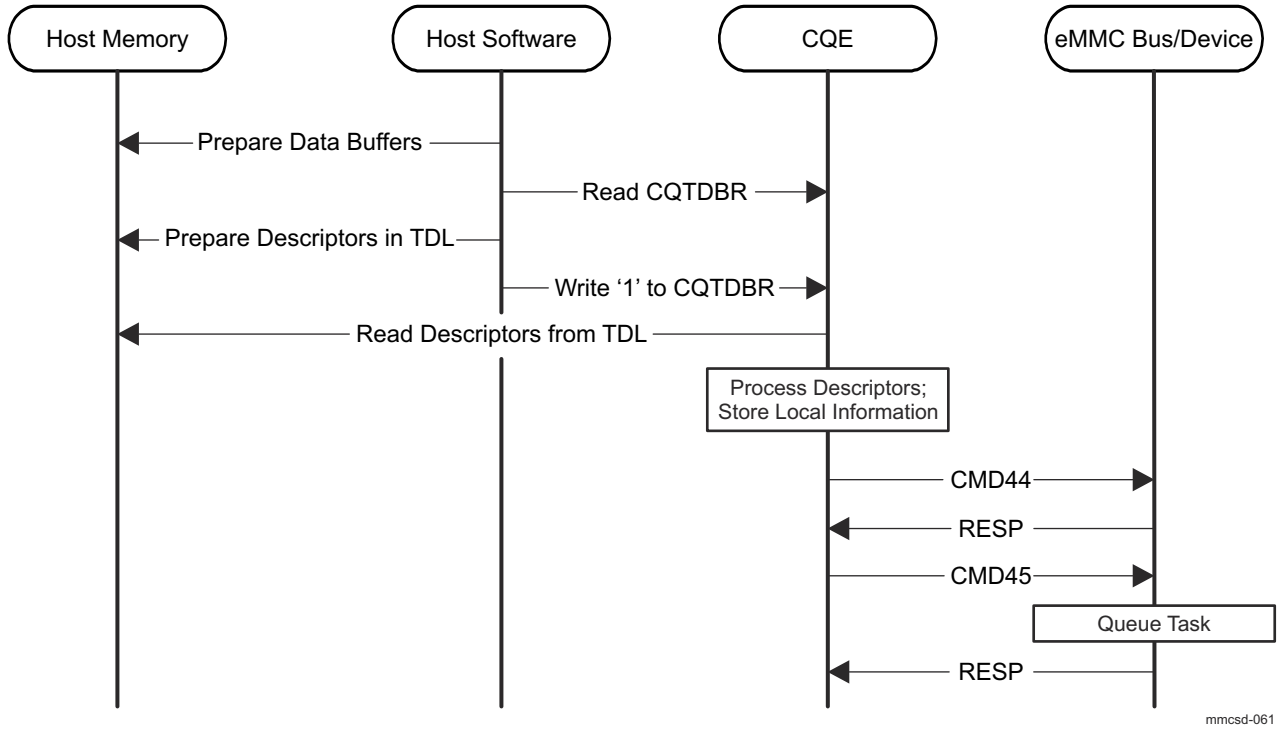


Figure 12-254. Task Issuance Sequence

Table 12-235. Task Issuance Sequence

Step	Description
1	Find an empty transfer request slot by reading the MMCSD0_CQ_TASK_DOOR_BELL register. An empty transfer request slot has its respective bit cleared to '0' in the MMCSD0_CQ_TASK_DOOR_BELL register.

Copyright © 2026 Texas Instruments Incorporated

Table 12-235. Task Issuance Sequence (continued)

Step	Description
2	Build a Task Descriptor at the 1st entry of the empty slot. Task Descriptor field values: • 1. Valid = 1, to indicate the descriptor is effective. • 2. End = 1, as required for Task Descriptors. • 3. Int = 1, if an interrupt on completion is required. Otherwise, Int = 0. • 4. Act = b101, to indicate a Task Descriptor . • 5. Data Direction = 1 for read commands, and 0 for write commands. • 6. Priority = 1 for high priority, and 0 for simple requests. • 7. QBR = 1 if Queue Barrier functionality is required. 0 otherwise. • 8. Forced Programming, Context ID, Tag Request, Reliable Write, Block Count, and Block Address programmed according to application requirements.
3	Build a Transfer Descriptor at the 2nd entry of the empty slot. Transfer Descriptor field values: • 1. Valid = 1, to indicate the descriptor is effective. • 2. End = 1, if a TRAN descriptor is used, to indicate the task only has one data buffer. End = 0 if LINK descriptor is used. • 3. Int = 0. Ignore in Command queueing. • 4. Act = b100, for TRAN descriptors, pointing directly to the task's single data buffer. Act = 401 b110, for LINK descriptors, point to a scatter/gather list (indirect). • 5. Address and Length programmed according to the data buffer supplied by the application.
4	If more than one transfer is requested, repeat step 1-3 for all needed transfers.
5	Set MMCSD0_CQ_TASK_DOOR_BELL register to indicate to the CQE that one or more transfer requests are ready to be sent to the attached device. Host software shall only write a '1' to the bit position that corresponds to new tasks; all other bit positions within MMCSD0_CQ_TASK_DOOR_BELL register should be written with a '0', which indicates no change to their current values.

12.3.5.5.2.4.3 Task Execution and Completion Sequence

The CQE is responsible for task execution, communication with the device and moving the data to the buffers in the host memory.

Figure 12-255 and Table 12-236 show a task execution and completion sequence.

Copyright © 2026 Texas Instruments Incorporated

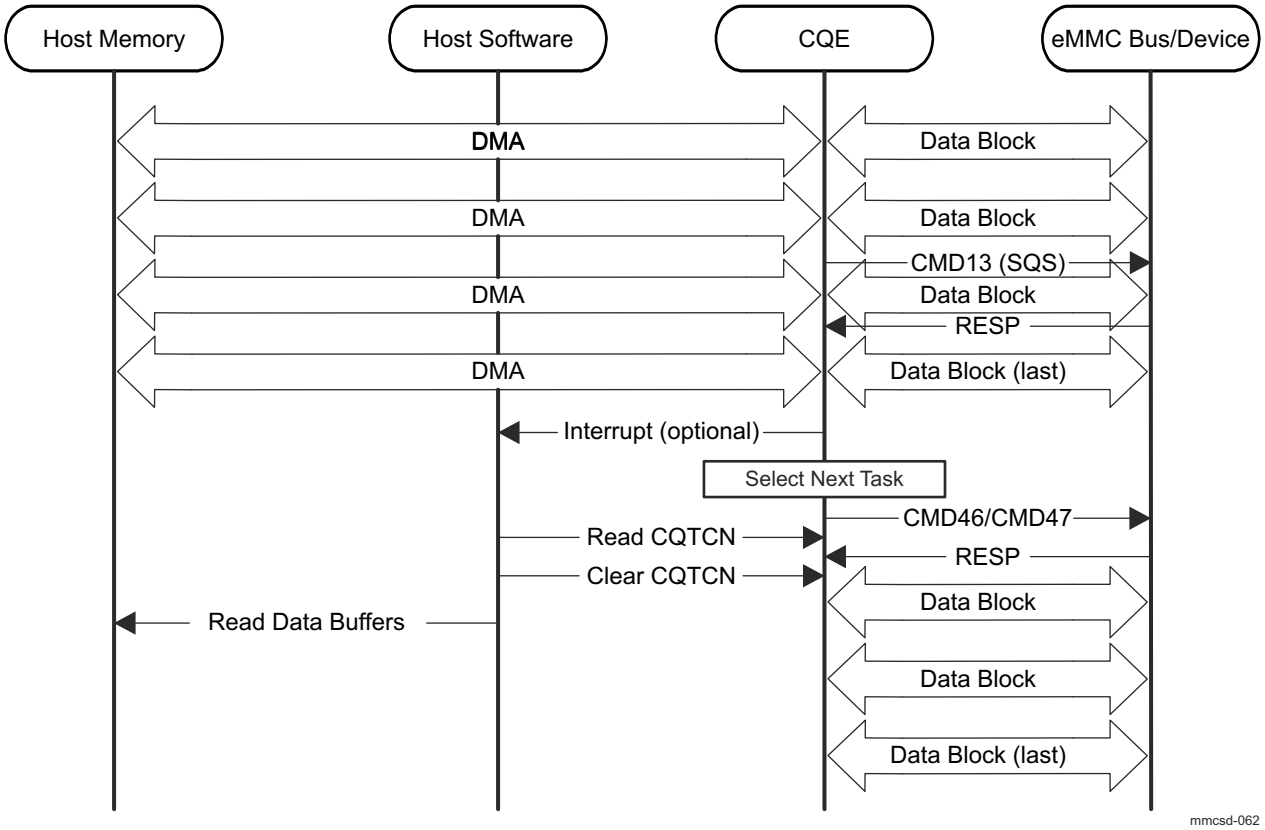


Figure 12-255. Task Execution and Completion Sequence

Table 12-236. Task Execution and Completion Sequence

Step	Description
1	Once the tasks are queued on the device side CQE sends CMD13 to determine the queue status. The queue status lets the CQE know which tasks are ready for execution.
2	The response to CMD13 may indicate no task is ready in which case the CQE is required to send CMD13 again at a later time as controlled CQSST register.
3	If a task is ready for execution then the host sends EXECUTE_READ_TASK (CMD46) or EXECUTE_WRITE_TASK (CMD47) to the device with the task id ordering it to execute a task.
4	When the task execution is completed, an interrupt may be generated, if requested, or as determined by Interrupt Coalescing mechanism.
5	The host software reads MMCSD0_CQ_TASK_COMP_NOTIF register to determine which task(s) has(have) been completed. Each bit set in MMCSD0_CQ_TASK_COMP_NOTIF register represents by index which task has completed but hasn't yet been served by software.
6	For every task completed clear the appropriate MMCSD0_CQ_TASK_COMP_NOTIF register bit.
7	Repeat steps 1-6 for the pending tasks.

12.3.5.5.2.4.4 Task Discard and Clear Sequence

Figure 12-256 and Table 12-237 show a task discard and clear sequence.

Copyright © 2026 Texas Instruments Incorporated

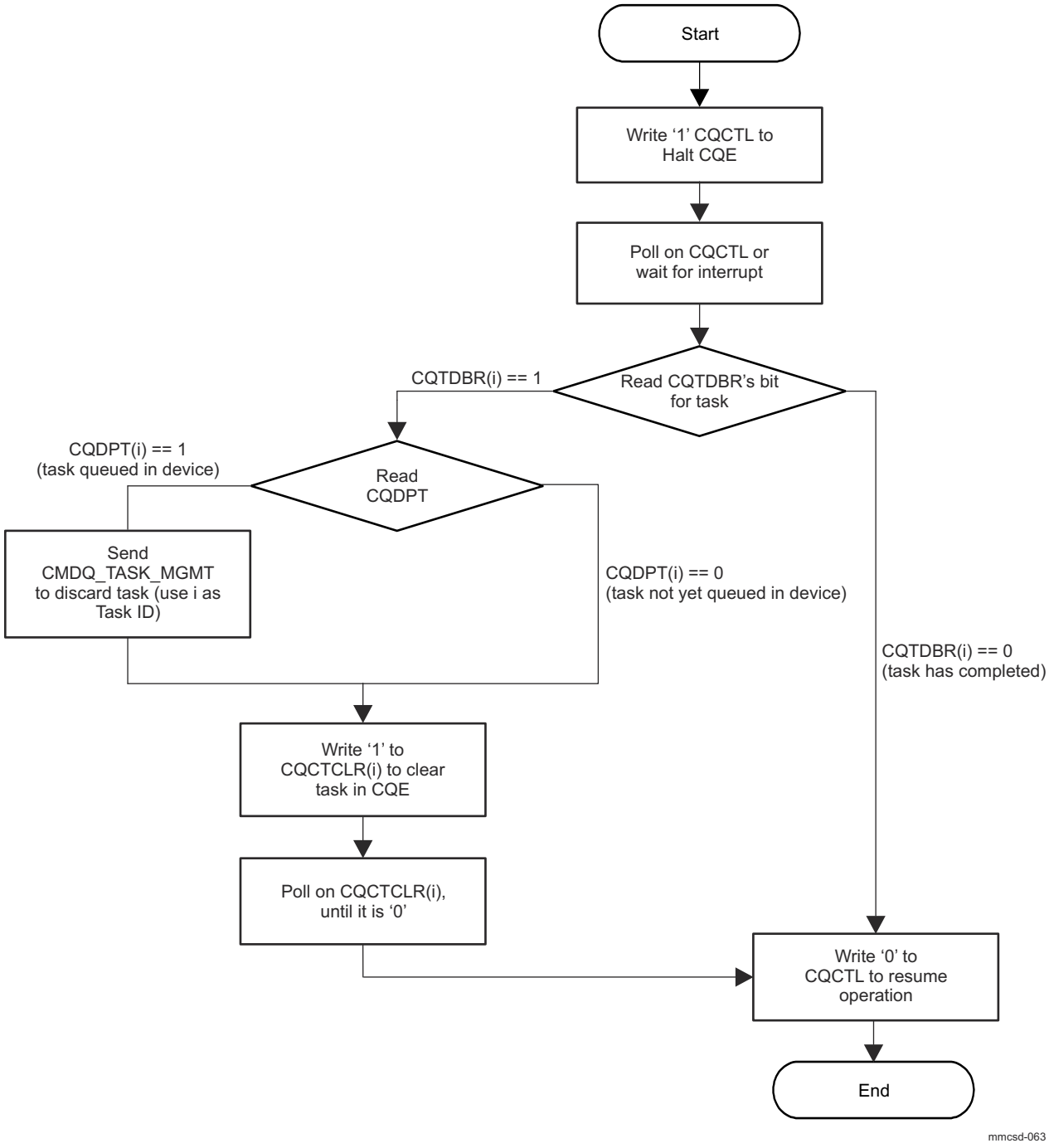


Figure 12-256. Task Discard and Clear Sequence

Table 12-237. Task Discard and Clear Sequence

Step	Description
1	While CQE is enabled write '1' to MMCSD0_CQ_CONTROL[0] HALT_BIT bit to halt CQE.
2	Poll on MMCSD0_CQ_INTR_STS[0] HALT_COMPLETE bit.
3	Read MMCSD0_CQ_TASK_DOOR_BELL register to determine if the task to be dicarded is set to 1.
4	Read MMCSD0_CQ_DEV_PENDING_TASKS register to check if the task is queued in the device.
5	Send CMDQ_TASK_MGMT(CMD48) to discard task using the task id as the argument.
6	Write '1' to CQCTCLR[i] to clear task in CQE.

Copyright © 2026 Texas Instruments Incorporated

Table 12-237. Task Discard and Clear Sequence (continued)

Step	Description
7	Poll on CQCTCLR[i] until it is '0'.
8	Write '0' to MMCSD0_CQ_CONTROL register to resume CQE.

12.3.5.5.2.4.5 Error Detect and Recovery when CQ is enabled

Figure 12-257 and Table 12-238 show an error detect and recovery sequence.

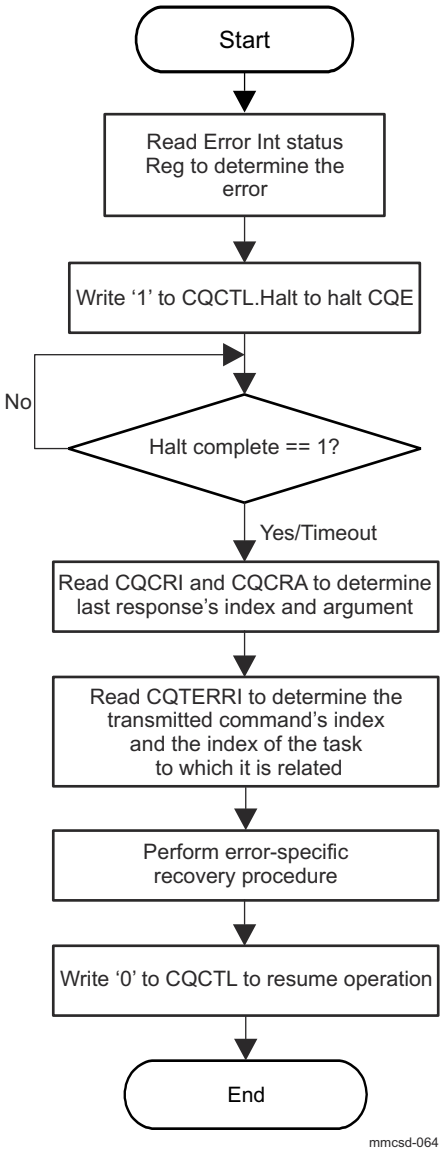


Figure 12-257. Error Detect and Recovery Sequence

Table 12-238. Error Detect and Recovery Sequence

Step	Description
1	Read eMMC host controller MMCSD0_ERROR_INTR_STS register and determine error is related to CQE.
2	Write '1' to MMCSD0_CQ_CONTROL[0] HALT_BIT bit to halt CQE.
3	Wait for MMCSD0_CQ_CONTROL[0] HALT_BIT bit to read '1'. In some error cases, this may not happen, so software should proceed to the next step after a sufficient time-out.

Copyright © 2026 Texas Instruments Incorporated

Table 12-238. Error Detect and Recovery Sequence (continued)

Step	Description
4	Read MMCSD0_CQ_CMD_RESP_INDEX and MMCSD0_CQ_CMD_RESP_ARG registers to determine last response's index and argument.
5	Read MMCSD0_CQ_TASK_ERR_INFO register to determine the transmitted command's index and the index of the task to which it is related.
6	Perform error-specific recovery procedure.
7	Write '0' to MMCSD0_CQ_CONTROL register to resume operation.

12.4 Industrial and Communication Interfaces

Copyright © 2026 Texas Instruments Incorporated

12.4.1 Modular Controller Area Network (MCAN)

This section describes the Modular Controller Area Network (MCAN) modules in the device.

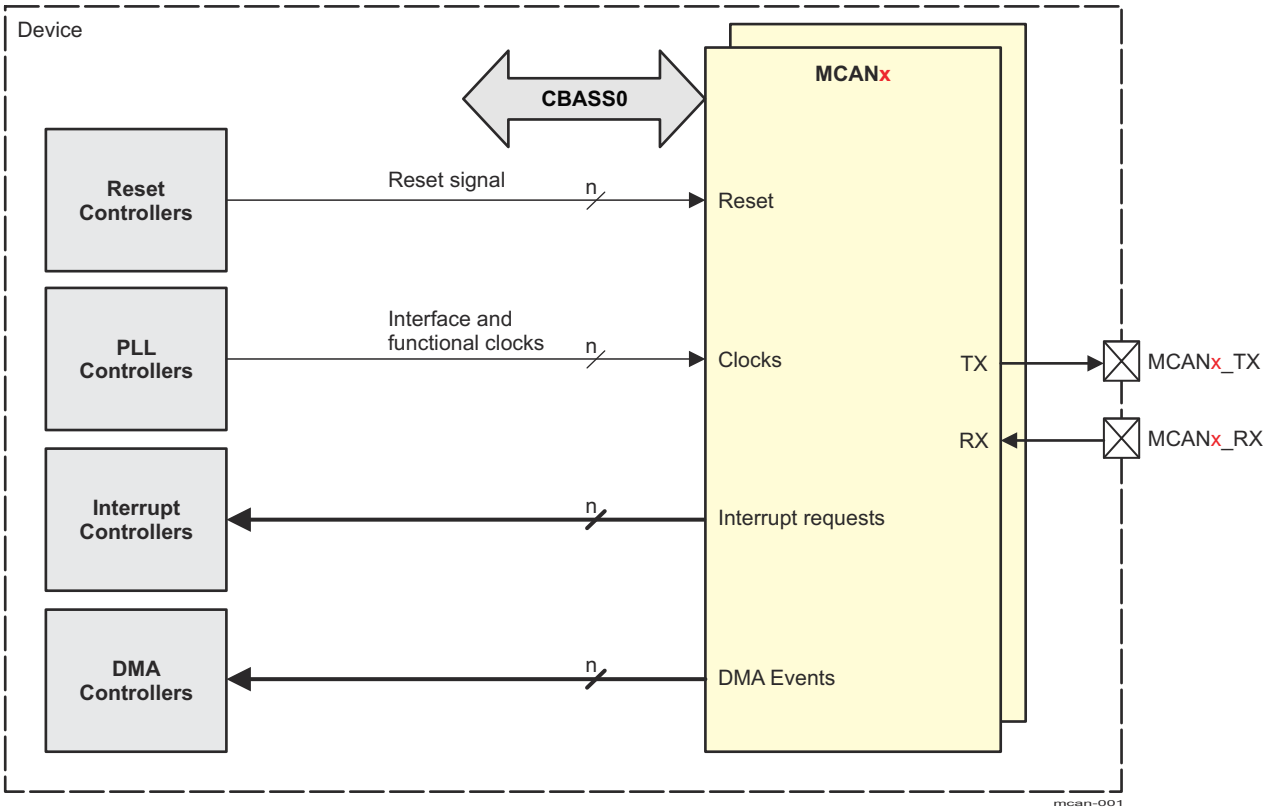
12.4.1.1 MCAN Overview

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real-time control. CAN has high immunity to electrical interference. In a CAN network, many short messages are broadcast to the entire network, which provides for data consistency in every node of the system.

The MCAN module supports both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications. CAN FD feature allows high throughput and increased payload per data frame. The classic CAN and CAN FD devices can coexist on the same network without any conflict.

They connect to the physical layer of the CAN network through external (for the device) transceivers. Each MCAN module supports flexible bit rates greater than 1 Mbps and is compliant to ISO 11898-1:2015.

Figure 12-258 shows the MCAN modules overview.



A. x = 0 to number of MCAN in a domain -1

Figure 12-258. MCAN Modules Overview

12.4.1.1.1 MCAN Features

Each MCAN module implements the following features:

- Conforms with CAN Protocol 2.0 A, B and ISO 11898-1:2015
- Full CAN FD support (up to 64 data bytes)
- SAE J1939 support
- AUTOSAR support
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO, up to 32 elements

Copyright © 2026 Texas Instruments Incorporated

- Configurable Transmit Queue, up to 32 elements
- Configurable Transmit Event FIFO, up to 32 elements
- Up to 64 dedicated Receive Buffers
- Two configurable Receive FIFOs, up to 64 elements each
- Up to 128 filter elements
- Internal Loopback mode for self-test
- Maskable interrupts, two interrupt lines
- Two clock domains (CAN clock/Host clock)
- Parity/ECC support - Message RAM single error correction and double error detection (SECDED) mechanism
- Local power-down and wakeup support
- Timestamp Counter

12.4.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.2 MCAN Environment

This section describes the MCAN external connections (environment).

CAN network physical layer consists of two-wire differential bus, usually twisted pair, and provides high level of interference immunity. External CAN transceiver IC is needed to access a CAN bus by the MCAN.

Figure 12-259 shows the MCAN typical application.

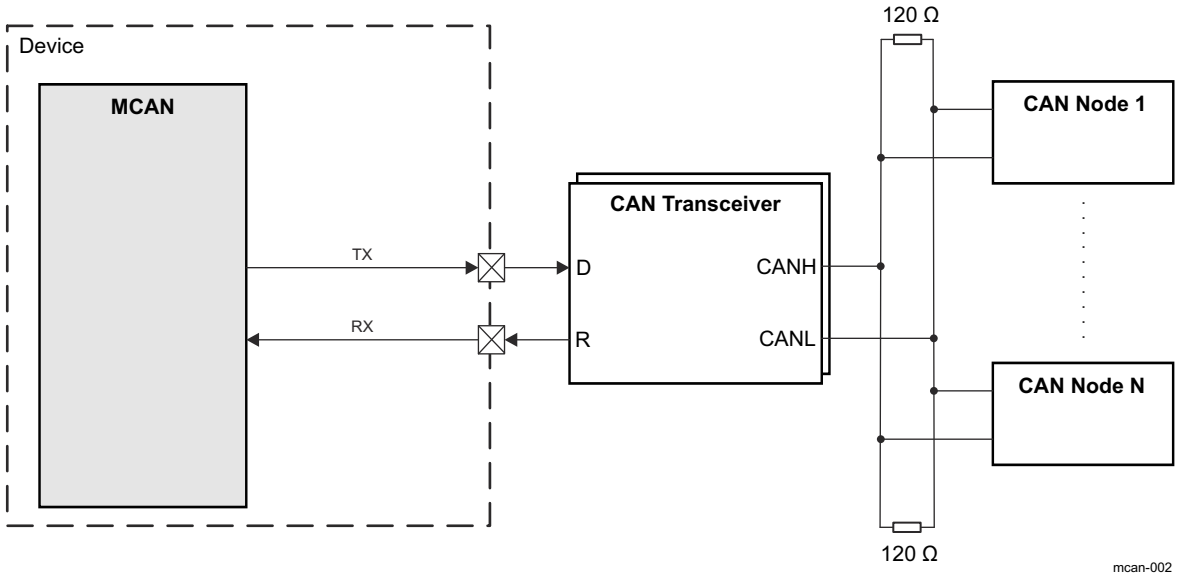


Figure 12-259. MCAN Typical Application

Table 12-239 describes the MCAN I/O signals.

Table 12-239. MCAN I/O Signals

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[1]
		MCANi[2]		
RX	MCANi[2]_RX	I	Serial data input from external CAN transceiver	HiZ
TX	MCANi[2]_TX	O	Serial data output to external CAN transceiver	1

(1) I = Input; O = Output; HiZ = High Impedance
(2) i represents an MCAN instance. See the device datasheet for available domains and MCAN instances.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.2.1 CAN Network Basics

- CAN bus is a 2-wire differential bus using Non-Return-to-Zero (NRZ) encoding and has two states:
  - Recessive state (logical 1)
  - Dominant state (logical 0)
- The network is multicontroller. When two or more nodes (ECUs) attempt to transmit at the same time, a non-destructive arbitration technique guarantees messages are sent in order of priority and no messages are lost.
- The message transmission is multicast. Data messages transmitted are identifier based, not address based.
- Content of message is labeled by the identifier that is unique throughout the network (for example: rpm, temperature, position, pressure, and so forth).
- All nodes on network receive the message and each performs an acceptance test on the identifier. If message is relevant, it is processed, otherwise it is ignored.
- The unique identifier also determines the priority of the message (the lower the numerical value of the identifier, the higher the priority is).
- Data is transmitted and received using message frames, consisting of the following basic fields:
  - Arbitration field
  - Control field
  - Data field (up to 8 bytes for Classical CAN and up to 64 bytes for CAN FD)
  - CRC field
  - ACK field

For more information, see *ISO 11898-1:2015: CAN data link layer and physical signaling*.

12.4.1.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4 MCAN Functional Description

The MCAN module performs CAN protocol communication according to ISO 11898-1:2015. The bit rate can be programmed to values greater than 1 Mbps. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

For communication on a CAN network, individual message frames can be configured. The message frames and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the Message Handler.

The register set of the MCAN module can be accessed directly via the module interface. These registers are used to control and configure the CAN core and the Message Handler, and to access the Message RAM.

Figure 12-260 shows the MCAN module block diagram.

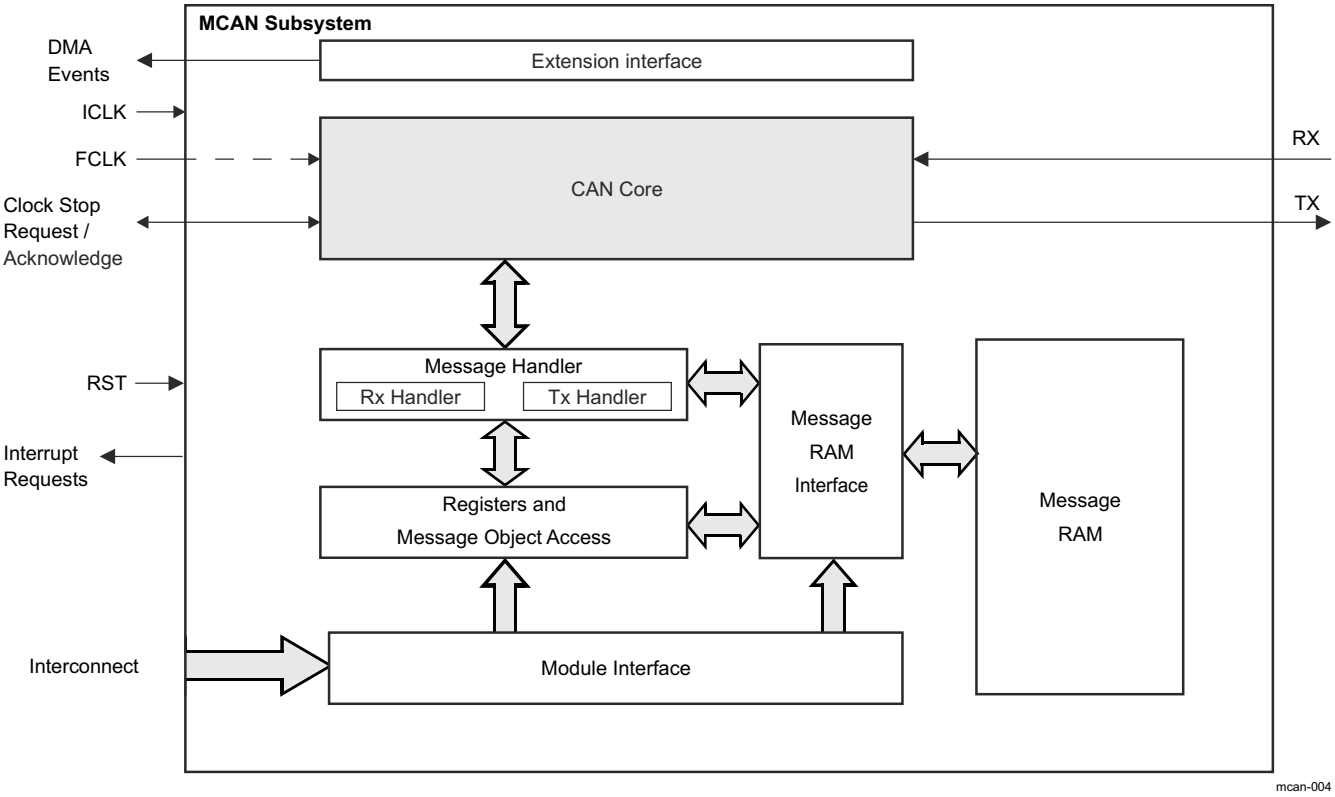


Figure 12-260. MCAN Block Diagram

The MCAN module blocks description:

- **CAN Core**: the CAN core consists of the CAN protocol controller and the Rx/Tx shift register. It handles all ISO 11898-1:2015 protocol functions and supports 11-bit and 29-bit identifiers.
- **Message Handler**: the Message Handler (Rx Handler and Tx Handler) is a state machine that controls the data transfer between the single-ported Message RAM and the CAN core's Rx/Tx shift register. It also handles the acceptance filtering and the Interrupt/DMA request generation as programmed in the control registers.
- **Message RAM**: the main purpose of the Message RAM is to store Rx/Tx messages, Tx Event elements, and Message ID Filter elements (for more information, see Section 12.4.1.4.10, *Message RAM*).
- **Message RAM Interface**: enables connection between the Message RAM and the other blocks in the MCAN module.
- **Registers and Message Object Access**: data consistency is ensured by indirect accesses to the message objects. The interface registers have the same word-length as the Message RAM.

Copyright © 2026 Texas Instruments Incorporated

- **Module Interface**: provides connection to the Registers and Message Object Access block and Message RAM Interface block
- **Clocking**: two clocks are provided to the MCAN module: the peripheral synchronous clock (interface clock ICLK) and the peripheral asynchronous clock (functional clock - FCLK). For more information, see *Module Integration*.
- **Extension Interface**: this interface is used for DMA requests signaling (see Section 12.4.1.4.2.2).

12.4.1.4.1 Module Clocking Requirements

Two clocks are provided to the MCAN module:

- the peripheral synchronous clock (ICLK) as the general module clock source
- and the peripheral asynchronous clock (FCLK) provided to the CAN core for generating the CAN bit timing.

Within the MCAN module there is a synchronization mechanism implemented to ensure safe data transfer between the two clock domains. There is synchronization between the signals from the Host clock domain to the CAN clock domain and vice versa and between the reset signal to the Host clock domain and to the CAN clock domain.

Note

ICLK must always be higher or equal to FCLK, in order to achieve a stable functionality of the MCAN module. Here, also the frequency shift of the modulated ICLK has to be considered:

$f_{0,ICLK} \pm \Delta f_{FM,ICLK} \geq f_{FCLK}$

For more information on how to configure the relevant clock source registers, see *Clocking* and the device-specific Datasheet.

12.4.1.4.2 Interrupt and DMA Requests

The MCAN module provides interrupt and DMA requests. They are configured via the Host CPU. The Suspend Mode is requesting or forcing (based on MCANSS_CTRL[3] DBGSUSP_FREE bit) the MCAN module to go into initialization mode (see MCAN_CCCR[0] INIT bit) in which new interrupts and DMA requests will not be issued, that is to prevents the interrupt and DMA requests from propagating to the Host CPU (for more information, see Section 12.4.1.4.3.8.2, *Suspend Mode*).

12.4.1.4.2.1 Interrupt Requests

The MCAN module has two interrupt lines. There are 30 internal interrupt sources. Each source can be configured to drive one of the two interrupt lines. The interrupts are 'level high' interrupts.

The MCAN core provides two interrupt requests (for Line 0 and Line 1).

For more information, see the following registers:

- Interrupt Register (MCAN_IR)
- Interrupt Enable (MCAN_IE)
- Interrupt Line Select (MCAN_ILS)
- Interrupt Line Enable (MCAN_ILE)

The MCAN module is capable of issuing ECC interrupts. After clearing the ECC interrupt source, the application software must also write 1 to EOI register (MCANSS_ECC_SEC_EOI_REG/MCANSS_ECC_DED_EOI_REG). For more information, see *ECC Aggregator*.

The MCAN module supports External Timestamp Counter. When the External Timestamp Counter rolls over it produces an interrupt (see Section 12.4.1.4.4.1, *External Timestamp Counter*).

For more information, see the following registers:

- Interrupt Clear Shadow Register (MCANSS_ICS)
- Interrupt Raw Status Register (MCANSS_IRS)
- Interrupt Enable Clear Shadow Register (MCANSS_IECS)

Copyright © 2026 Texas Instruments Incorporated

- Interrupt Enable Register (MCANSS_IE)
- Interrupt Enable Status Register (MCANSS_IES)
- End Of Interrupt Register (MCANSS_EOI)
- External Timestamp Prescaler Register (MCANSS_EXT_TS_PRESCALER)
- External Timestamp Unserviced Interrupts Counter Register (MCANSS_EXT_TS_UNSERVICED_INTR_CNTR)

For more information about available Interrupt Requests, see *MCAN Hardware Requests*.

12.4.1.4.2.2 DMA Requests

Functional transmit and Filter DMA requests are generated by the MCAN module based on the signaling in the Extension Interface. The DMA signaling uses a simple DMA request active high pulse.

The active high pulse indicates a pending message is transmitted (see MCAN_TXBRP). This pulse can be used to transfer another message to the Tx Buffer, which would need to be followed by writing 1 to the corresponding MCAN_TXBAR[0] AR bit to mark a new Tx message pending transmission.

The Parity on Tx DMA Events is available using an EDC Controller which can be accessed through the ECC Aggregator.

Standard and Extended message filters can be set to issue a pulse when a filter match occurs. These 'Filter Events' can be used to DMA messages from the Rx FIFO. The events are high level single clock cycle pulses (ICLK).

12.4.1.4.3 Operating Modes

12.4.1.4.3.1 Software Initialization

Setting the MCAN_CCCR[0] INIT bit to 1 starts a software initialization. This is done either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off state. While the MCAN_CCCR[0] INIT bit is set, the message transfer is stopped and the status of the output TX pin is recessive (high). The counters of the Error Management Logic (EML) are unchanged. Setting the MCAN_CCCR[0] INIT bit does not change any configuration register. Resetting the MCAN_CCCR[0] INIT bit finishes the software initialization. After waiting for the occurrence of a sequence of 11 consecutive recessive bits (indication for Bus_Idle state) the message transfer starts.

Access to the MCAN configuration registers is only enabled when both MCAN_CCCR[0] INIT and MCAN_CCCR[1] CCE bits are set (write protection).

The MCAN_CCCR[1] CCE bit can only be set/reset while the MCAN_CCCR[0] INIT = 1. The MCAN_CCCR[1] CCE bit is automatically reset when the MCAN_CCCR[0] INIT bit is reset.

The following registers are reset when the MCAN_CCCR[1] CCE bit is set:

- MCAN_HPMS - High Priority Message Status
- MCAN_RXF0S - Rx FIFO 0 Status
- MCAN_RXF1S - Rx FIFO 1 Status
- MCAN_TXFQS - Tx FIFO/Queue Status
- MCAN_TXBRP - Tx Buffer Request Pending
- MCAN_TXBTO - Tx Buffer Transmission Occurred
- MCAN_TXBCF - Tx Buffer Cancellation Finished
- MCAN_TXEFS - Tx Event FIFO Status

The Timeout Counter value MCAN_TOCV[15-0] TOC field is preset to the value configured by the MCAN_TOCC[31-16] TOP field when the MCAN_CCCR[1] CCE bit is set.

In addition the Tx Handler and Rx Handler are held in idle state while MCAN_CCCR[1] CCE = 1.

The following registers are only writeable while MCAN_CCCR[1] CCE = 0

- MCAN_TXBAR - Tx Buffer Add Request
- MCAN_TXBCR - Tx Buffer Cancellation Request

Copyright © 2026 Texas Instruments Incorporated

MCAN_CCCR[7] TEST and MCAN_CCCR[5] MON bits can only be set by the Host CPU while MCAN_CCCR[0] INIT = 1 and MCAN_CCCR[1] CCE = 1. Both bits may be reset at any time. The MCAN_CCCR[6] DAR bit can only be set/reset while MCAN_CCCR[0] INIT = 1
and MCAN_CCCR[1] CCE = 1.

Table 12-240 shows the steps to configure the MCAN module.

Table 12-240. Steps to Configure MCAN Module

Step	Operation	Description	Pseudo Code
1	Initialize MCAN_CCCR	Set MCAN_CCCR[0] INIT bit and check that it has been set	INIT = 1; If INIT ≠ 1, wait until it is
2	Unlock protected registers	Set MCAN_CCCR[1] CCE bit	CCE = 1;
3	Configure CAN mode	Set MCAN_CCCR[8] FDOE bit to CAN FD	FDOE = 1 for CAN FD FDOE = 0 for CAN
4	Configure Bit Rate Switching	Set MCAN_CCCR[9] BRSE bit	BRSE = 1 with bit rate switching BRSE = 0 without bit rate switching
5	Set bit timing	Set MCAN_NBTP register	
6	Lock protected registers	Clear MCAN_CCCR[1] CCE bit	CCE = 0;
7	Return MCAN module to normal operation	Clear MCAN_CCCR[0] INIT bit and check it has been cleared	INIT = 0; If INIT ≠ 0, wait until it is

12.4.1.4.3.2 Normal Operation

Once the MCAN module is initialized and the MCAN_CCCR[0] INIT bit is reset to zero, the MCAN module synchronizes itself to the CAN bus and is ready for communication. After passing the acceptance filtering, received messages including Message Identifier (ID) and Data Length Code (DLC) are stored into a dedicated Rx Buffer or into Rx FIFO 0/Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated.

Note

Automated transmission on reception of remote frames is not supported.

12.4.1.4.3.3 CAN FD Operation

The CAN FD standard allows extended frames to be sent, up to 64 data bytes in a single frame at a higher bit rate for the data phase of a frame, up to 8 Mbps. The CAN FD standard introduces the ability to switch from one bit rate to another. Extended Data Length (EDL), as shown in Figure 12-261, sets a data length of up to 8 or up to 64 data bytes. Bit Rate Switching (BRS) indicates whether two bit rates (the data phase is transmitted at a different bit rate to the arbitration phase) are enabled.

Copyright © 2026 Texas Instruments Incorporated

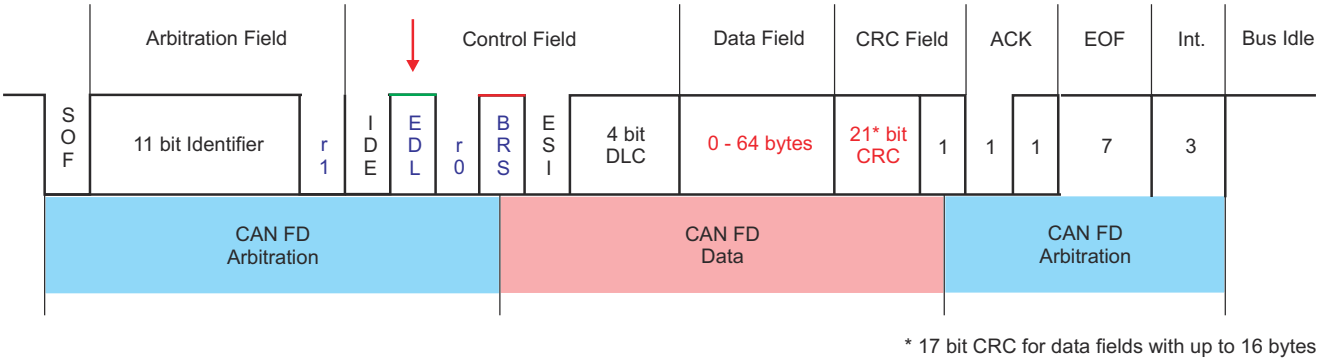


Figure 12-261. CAN FD Frame

Note

Figure 12-261 presents CAN FD frame according to the Non-ISO CAN FD (legacy) protocol. In the new ISO CAN FD protocol the CRC Field includes additional 5 bits (three stuff bit counter (SBC) bits and two parity bits). With these additional bits, a weakness identified in the error detection scheme chosen by the original protocol is removed. By setting MCAN_CCCR[15] NISO bit, the ISO or Non-ISO CAN FD format can be chosen. In CAN network ISO CAN FD and non-ISO CAN FD devices should never mix.

There are two variants of CAN FD frame transmission:

- CAN FD frame transmission without bit rate switching
- CAN FD frame transmission where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame

In the CAN frames FDF = recessive (logical 1) signifies a CAN FD frame, FDF = dominant (logical 0) signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF - res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. Note that the coding of res = recessive is reserved for future expansion of the protocol. In case the MCAN module receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting the MCAN_PSR[14] EXE bit. When Protocol Exception Handling is enabled (MCAN_CCCR[12] PXHD = 0), this causes the operation state to change from Receiver (MCAN_PSR[4-3] ACT = 10) to Integrating (MCAN_PSR[4-3] ACT = 00) at the next sample point. In case Protocol Exception Handling is disabled (MCAN_CCCR[12] PXHD = 1), the MCAN will treat a recessive res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming the MCAN_CCCR[8] FDOE bit. In case MCAN_CCCR[8] FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx Buffer element.

With MCAN_CCCR[8] FDOE = 0, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx Buffer element is set. The MCAN_CCCR[8] FDOE and MCAN_CCCR[9] BRSE bits can only be changed while the MCAN_CCCR[0] INIT and MCAN_CCCR[1] CCE bits are both set. With MCAN_CCCR[8] FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format.

With MCAN_CCCR[8] FDOE = 1 and MCAN_CCCR[9] BRSE = 0, only FDF bit of a Tx Buffer element is evaluated. With MCAN_CCCR[8] FDOE = 1 and MCAN_CCCR[9] BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

Copyright © 2026 Texas Instruments Incorporated

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wakeup messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the DLC coding differs from the standard CAN format (see Table 12-241).

Table 12-241. DLC Coding

DLC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Number of Data Bytes in Standard CAN	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8
Number of Data Bytes in CAN FD	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

For CAN FD frames, the bit timing will be switched inside the frame after the BRS (Bit Rate Switch) bit in case this bit is recessive. In the CAN FD arbitration phase, before the BRS bit, the nominal CAN bit timing (see Figure 12-262) is used as configured by the Nominal Bit Timing and Prescaler Register MCAN_NBTP. In the following CAN FD data phase, the data phase bit timing is used as configured by the Data Bit Timing and Prescaler Register MCAN_DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

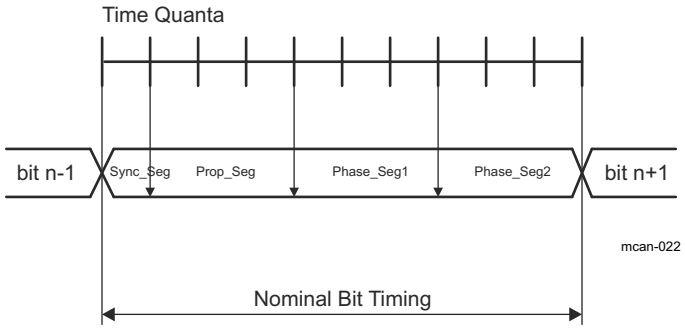


Figure 12-262. CAN Bit Timing

The maximum configurable data phase bit timing depends on the CAN clock frequency (FCLK). Example: with FCLK = 20 MHz and the shortest configurable bit time of 4 t_q (time quanta), the bit rate in the data phase is 5 Mbps.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the ESI (Error Status Indicator) bit depends on transmitter's error state (see MCAN_PSR[11] RESI bit) monitored at the start of the transmission. If the transmitter has error passive flag the ESI bit is transmitted recessive, else it is transmitted dominant.

12.4.1.4.3.4 Transmitter Delay Compensation

12.4.1.4.3.4.1 Description

When only one CAN FD node is transmitting and all others are receivers the length of the bus line has no impact. When transmitting via the TX pin the MCAN module receives the transmitted data from its local CAN transceiver via the RX pin. The received data is delayed. If the transmitter delay is greater than TSEG1 (time segment before sample point), a bit error is detected.

The MCAN module provides a delay compensation mechanism to compensate the transmitter delay. The compensation mechanism enables transmission with higher bit rates during the CAN FD data phase

Copyright © 2026 Texas Instruments Incorporated

independent of the delay of a specific CAN transceiver. Without transmitter delay compensation the bit rate in the data phase is limited by the transmitter delay.

The mechanism enables configurations where the data bit time is shorter than the transmitter delay (it is described in detail in ISO 11898-1:2015). The transmitter delay compensation is enabled by setting the MCAN_DBTP[23] TDC bit to 1.

The delayed transmit data is compared against the received data at the Secondary Sample Point (SSP) in order to check for bit errors during the data phase of transmitting nodes. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the MCAN's transmit output TX pin through the transceiver to the receive input RX pin plus the transmitter delay compensation offset configured by the MCAN_TDCR[14-8] TDCO field (see Figure 12-263). The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (example: half of the bit time in the data phase). The position of the SSP is rounded down to the next integer number of mtq.

The actual transmitter delay compensation value can be checked by reading the MCAN_PSR[22-16] TDCV field. This field is cleared when the MCAN_CCCR[0] INIT bit is set and is updated at each transmission of CAN FD frame while the MCAN_DBTP[23] TDC bit is set.

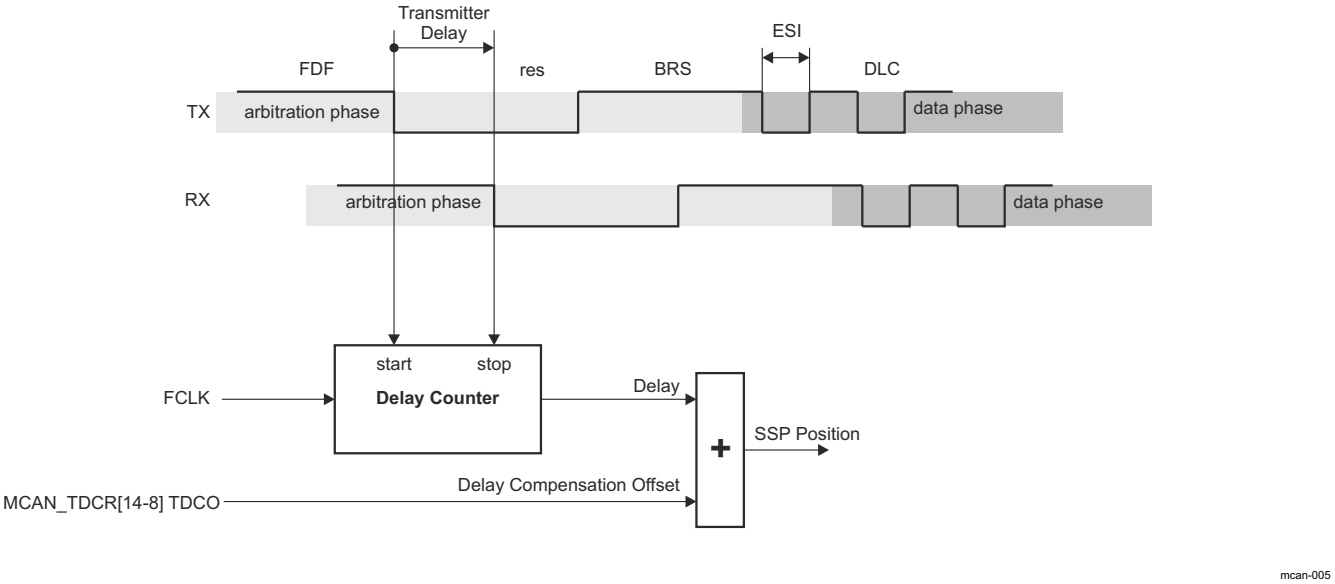


Figure 12-263. Transmitter Delay Measurement

12.4.1.4.3.4.2 Transmitter Delay Compensation Measurement

When transmitter delay compensation is enabled (by programming MCAN_DBTP[23] TDC = 1), the measurement is started within each transmitted CAN FD frame at the falling edge of FDF bit to bit res. The measurement is stopped when this edge is seen at the receive input RX pin of the transmitter. The resolution of this measurement is one mtq (see Figure 12-263). The mtq (minimum time quantum) dimension is equal to the CAN clock period (FCLK).

The use of a transmitter delay compensation filter window can be enabled by programming MCAN_TDCR[6-0] TDCF field. This filter feature defines a minimum value for the SSP position to avoid the case in which a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in an early taken SSP position. Dominant edges on the RX pin, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least MCAN_TDCR[6-0] TDCF field and the RX pin is low.

Copyright © 2026 Texas Instruments Incorporated

The following boundary conditions have to be considered:

- The sum of the measured delay from the TX pin to the RX pin and the configured transmitter delay compensation offset (MCAN_TDCR[14-8] TDCO field) has to be less than 6 bit times in the data phase.
- The sum of the measured delay from the TX pin to the RX pin and the configured transmitter delay compensation offset (MCAN_TDCR[14-8] TDCO) field has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs.

12.4.1.4.3.5 Restricted Operation Mode

In Restricted Operation Mode the CAN node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The recieve and transmit error counters (MCAN_ECR[14-8] REC and MCAN_ECR[7-0] TEC) are frozen while CAN error logging (MCAN_ECR[23-16] CEL) is active. The Host CPU can set the MCAN module into Restricted Operation Mode by setting MCAN_CCCR[2] ASM bit. The bit can only be set by the Host CPU at any time when both MCAN_CCCR[2] CCE and MCAN_CCCR[0] INIT bits are set to 1.

The Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset MCAN_CCCR[2] ASM bit. This mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

Note

The Restricted Operation Mode must not be combined with the Internal Loopback Mode .

12.4.1.4.3.6 Bus Monitoring Mode

Entering Bus Monitoring Mode is done by setting the MCAN_CCCR[5] MON bit to 1. In this mode (see ISO 11898-1:2015, *Bus Monitoring* section), the MCAN module is able to receive valid data and remote frames, but cannot start a transmission. The MCAN module sends only recessive bits on the CAN bus. If the MCAN module is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MCAN module monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode the MCAN_TXBRP register is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 12-264 shows the connection of the TX and RX signals to the MCAN module in Bus Monitoring Mode.

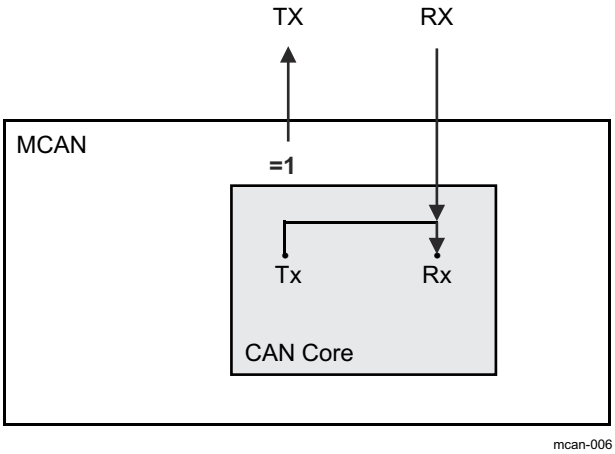


Figure 12-264. Connection of Signals in Bus Monitoring Mode

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.3.7 Disabled Automatic Retransmission (DAR) Mode

According to the CAN Specification (see ISO11898-1:2015, *Recovery Management* section), the MCAN module provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled (see the MCAN_CCCR[6] DAR bit).

12.4.1.4.3.7.1 Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending MCAN_TXBRP[xx] TRPx bit is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

Successful transmission:

- Corresponding Tx Buffer Transmission Occurred MCAN_TXBTO[xx] TOx bit is set
- Corresponding Tx Buffer Cancellation Finished MCAN_TXBCF[xx] CFx bit is not set

Successful transmission in spite of cancellation:

- Corresponding Tx Buffer Transmission Occurred MCAN_TXBTO[xx] TOx bit is set
- Corresponding Tx Buffer Cancellation Finished MCAN_TXBCF[xx] CFx bit is set

Arbitration lost or frame transmission disturbed:

- Corresponding Tx Buffer Transmission Occurred MCAN_TXBTO[xx] TOx bit is not set
- Corresponding Tx Buffer Cancellation Finished MCAN_TXBCF[xx] CFx bit is set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

12.4.1.4.3.8 Power Down (Sleep Mode)

The entering in Power Down mode is controlled via two sources:

- PSC (via clock stop request signal)
- Software (by writing to the MCAN_CCCR[4] CSR bit)

As long as the clock stop request signal is active, the MCAN_CCCR[4] CSR bit is read as 1.

When all pending transmission requests have completed, the MCAN module waits until bus idle state is detected. Then the MCAN module sets the MCAN_CCCR[0] INIT bit to 1 to prevent any further CAN transfers.

The MCAN module acknowledges that it is ready for power down:

- By asserting clock stop acknowledge signal to the PSC (in case of PSC source).
- By setting the MCAN_CCCR[3] CSA flag bit to 1 (in case of Software source).

In this state, before the clocks are switched off, further register accesses can be made. Now the module clock inputs ICLK and FCLK may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting the input clock stop request signal respectively the MCAN_CCCR[4] CSR flag bit. The MCAN will acknowledge this by resetting the output clock stop acknowledge signal respectively the MCAN_CCCR[3] CSA flag bit. Afterwards, the application can restart CAN communication by resetting the MCAN_CCCR[0] INIT bit.

Restoring the clocks from clock stop mode, needs to be done according to how the clock stop was initiated:

- If Software asserts the MCAN_CCCR[3] CSA flag bit, once the MCAN module goes idle, the MCAN_CCCR[0] INIT bit is set. To get it started again, Software needs to write 0 to the MCAN_CCCR[0] INIT bit.
- If PSC is issuing a clock stop request, than there are two options for waking up:
  - After removing clock stop request signal, Software would need to write 0 to the MCAN_CCCR[0] INIT bit, or
  - If the MCANSS_CTRL[5] AUTOWAKEUP bit is set, than after removing clock stop request signal, an FSM inside the MCAN module will reset the MCAN_CCCR[0] INIT bit (without Software).

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.3.8.1 External Clock Stop Mode

The MCAN module supports two external clock stop modes:

- Immediate
- Graceful

In a graceful clock stop mode, when the clock stop request is asserted, the MCAN core will respond with clock stop acknowledge when all pending Tx messages have been processed and an Idle line had been detected. The MCAN_CCCR[0] INIT bit will be set, the MCAN core will go and stay Idle.

The automatic wakeup feature is enabled by setting the MCANSS_CTRL[5] AUTOWAKEUP and MCANSS_CTRL[4] WAKEUPREQEN bits to 1 (for more information, see Section 12.4.1.4.3.8.3, *Wakeup request*). When external clock stop request is removed and no suspend request is active, a read-modify-write to the MCAN_CCCR[0] INIT bit is performed to clear it.

12.4.1.4.3.8.2 Suspend Mode

The MCAN module supports two suspend modes:

- Immediate
- Graceful

In a graceful suspend mode (see the MCANSS_CTRL[3] DBGSUSP_FREE bit), when the suspend request is asserted, a clock stop request to the MCAN core is performed. The MCAN core will respond with clock stop acknowledge when all pending Tx messages have been processed and an Idle line had been detected. At that point the MCAN_CCCR[0] INIT bit will be set, the MCAN core will go and stay Idle. The suspend state can be verified by reading MCAN_CCCR[0] INIT bit.

The automatic wakeup feature is enabled by setting the MCANSS_CTRL[5] AUTOWAKEUP and MCANSS_CTRL[4] WAKEUPREQEN bits to 1 (for more information, see Section 12.4.1.4.3.8.3, *Wakeup request*). When suspend request is removed, if no external clock stop request is active, a read-modify-write to the MCAN_CCCR[0] INIT bit is performed to clear it.

During suspend mode the auto-clear feature is disabled. The following register fields have an auto-clear feature:

- MCAN_ECR[23-16] CEL
- MCAN_PSR[2-0] LEC
- MCAN_PSR[10-8] DLEC
- MCAN_PSR[11] RESI
- MCAN_PSR[12] RBRS
- MCAN_PSR[13] RFDF
- MCAN_PSR[14] PXE

12.4.1.4.3.8.3 Wakeup request

Issuing a clock stop request puts the MCAN module into Power Down mode (Sleep Mode). During transition from IDLE to ACTIVE, if the MCANSS_CTRL[5] AUTOWAKEUP and MCANSS_CTRL[4] WAKEUPREQEN bits are enabled, after the MCAN Core respond to the removal of the clock stop request with removing the clock stop acknowledge, a read-modify-write will be issued to clear the MCAN_CCCR[0] INIT bit and the MCAN core will resume operation.

If the MCANSS_CTRL[4] WAKEUPREQEN bit is set, the MCAN module provides a wakeup request on the following wakeup event:

- The receive RX pin is dominant (logical 0)

The wakeup request is de-asserted when any of the following conditions occur:

- Clock stop request is removed and clock stop acknowledge is de-asserted
- A reset is applied to the MCAN module

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.3.9 Test Modes

The MCAN_TEST register write access is enabled by setting the test mode enable MCAN_CCCR[7] TEST bit to 1. The MCAN_TEST register allows the configuration of the test modes and test functions.

The CAN transmit TX pin has four output functions. One of those functions can be selected by programming the MCAN_TEST[6-5] TX field. Additionally to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the MCAN's bit timing and it can drive constant dominant or recessive values.

The actual value of the CAN receive RX pin can be monitored from MCAN_TEST[7] RX bit. Both functions can be used to check the CAN bus physical layer. Due to the synchronization mechanism between CAN clock (FCLK) and Host clock (ICLK) domain, there may be a delay of several Host clock periods between writing to the MCAN_TEST[6-5] TX field until the new configuration is visible at the output TX pin. This applies also when reading input RX pin via the MCAN_TEST[7] RX bit.

Note

Test modes should be used for self test only. The software control for TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

12.4.1.4.3.9.1 Internal Loopback Mode

The MCAN module can be set into Internal Loopback Mode by programming MCAN_TEST[4] LBCK and MCAN_CCCR[5] MON bits to 1. The Internal Loopback Mode is used for a 'Hot Selftest'. The 'Hot Selftest' allows the MCAN module to be tested without affecting a running CAN system connected to the TX and RX pins. In this mode RX pin is disconnected from the MCAN module and TX pin is held recessive. Figure 12-265 shows the connection of the TX and RX pins to the MCAN module in case of Internal Loopback Mode.

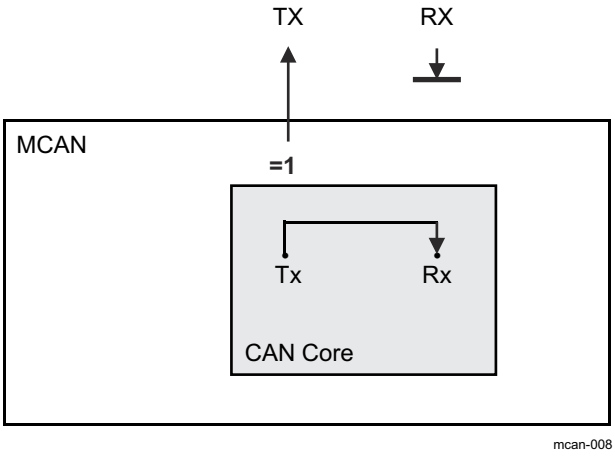


Figure 12-265. Internal Loopback Mode

12.4.1.4.4 Timestamp Generation

The MCAN module has integrated a 16-bit wrap-around counter for timestamp generation. The timestamp counter prescaler MCAN_TSCC[19-16] TCP field can be configured to clock the counter in multiples of CAN bit times (1-16). The counter is readable via the MCAN_TSCV[15-0] TSC field. A write access to the MCAN_TSCV register resets the counter to zero. When the timestamp counter wraps around the interrupt MCAN_IR[16] TSW flag is set. On start of a frame reception/transmission the counter value is captured and stored into the timestamp section of an Rx Buffer/Rx FIFO (RXTS[15-0]) or Tx Event FIFO (TXTS[15-0]) element. For more information, see Section 12.4.1.4.10, *Message RAM*.

12.4.1.4.4.1 External Timestamp Counter

For CAN FD operation mode the MCAN core requires an External Timestamp Counter. An externally generated 16-bit vector may substitute the integrated 16-bit CAN bit time counter (internal timestamp counter) for receive

Copyright © 2026 Texas Instruments Incorporated

and transmit timestamp generation. An external 16-bit timestamp counter can be used by programming the MCAN_TSCC[1-0] TSS field.

The External Timestamp Counter uses the interface clock (ICLK) as a reference clock. The MCAN Core accepts a 16-bit timestamp. A 24-bit prescaler provides a programmable resolution for the timestamp (see MCANSS_EXT_TS_PRESCALER[23-0] PRESCALER bit field). The External Timestamp Counter counter can be enabled or disabled through the MCANSS_CTRL[6] EXT_TS_CNTR_EN bit. When disabled the counter is reset back to zero. While enabled the counter keeps incrementing. When the timestamp rolls over the MCAN timestamp interrupt is generated.

When the timestamp rolls over the MCANSS_IRS register is set (see Figure 12-266). The MCANSS_IE register can be affected by writing to the *MCANSS_IESS* register to set or to the MCANSS_IECS register to clear. The *MCANSS_IESS* register is a shadow register mapped to the same address as the MCANSS_IE register. The level interrupt is a reflection of both MCANSS_IRS and MCANSS_IE being set. The MCANSS_IES register reflects the level interrupt. When an rollover event occurs the interrupt counter is incremented. Writing to the MCANSS_ICS register to clear the MCANSS_IRS register will also decrement the interrupt counter. Writing to the MCANSS_EOI register will issue another pulse if the interrupt counter is not zero.

The rollover event can be artificially simulated by software through writing to the Interrupt Set Shadow register (*MCANSS_ISS*). The *MCANSS_ISS* register is a shadow register mapped to the same address as the MCANSS_IRS register.

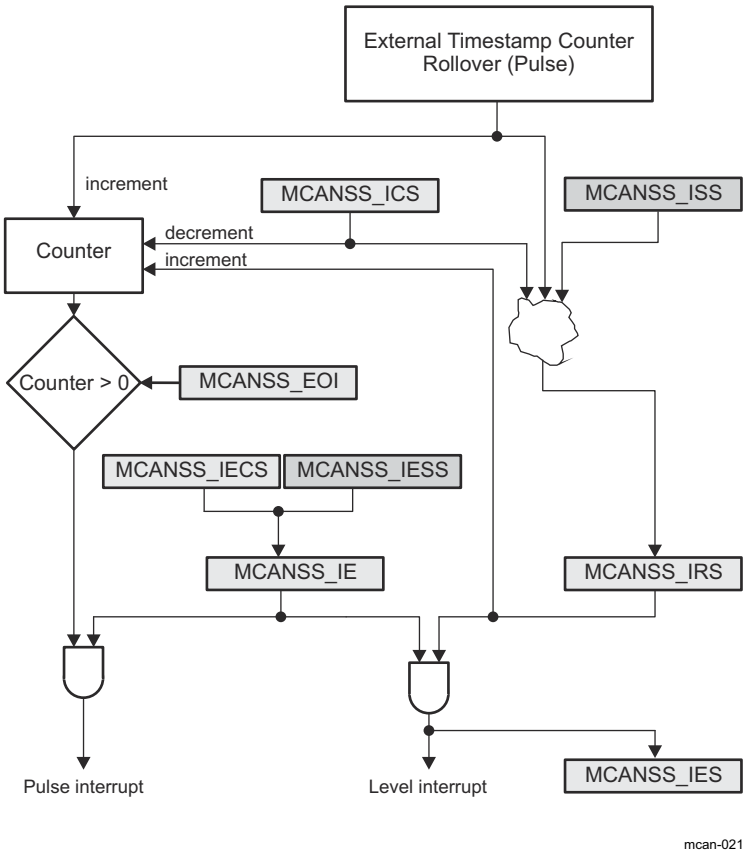


Figure 12-266. External Timestamp Counter Interrupt

12.4.1.4.5 Timeout Counter

The MCAN module has integrated a 16-bit Timeout Counter. It is used to signal timeout conditions for the Rx FIFO 0, Rx FIFO 1, and Tx Event FIFO Message RAM elements. The Timeout Counter is configured via the MCAN_TOCC register. It is enabled via the MCAN_TOCC[0] ETOC bit. The Timeout Counter operates

Copyright © 2026 Texas Instruments Incorporated

as down-counter and uses the same prescaler programmed by the MCAN_TSCC[19-16] TCP field as the Timestamp Counter. The actual counter value can be monitored from the MCAN_TOCV[15-0] TOC field. The Timeout Counter can be started only when MCAN_CCCR[0] INIT = 0 and stopped when MCAN_CCCR[0] INIT = 1 (example: when the MCAN enters Bus_Off state). The operation mode is selected by the MCAN_TOCC[2-1] TOS field. When Continuous Mode is selected, the counter starts when MCAN_CCCR[0] INIT = 0, a write to the MCAN_TOCV register presets the counter to the value configured by the MCAN_TOCC[31-16] TOP field and continues down-counting.

In case the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by the MCAN_TOCC[31-16] TOP field. Down-counting is started when the first FIFO element is stored. Writing to the MCAN_TOCV register has no effect. When the counter reaches zero, the interrupt MCAN_IR[18] TOO flag is set.

In Continuous Mode, the counter is immediately restarted at the value configured by the MCAN_TOCC[31-16] TOP field.

12.4.1.4.6 ECC Support

The Message Memory is wrapped in an ECC wrapper providing SECDED parity functionality. The ECC wrapper is controlled by an ECC Aggregator.

12.4.1.4.6.1 ECC Wrapper

The ECC wrapper provides Single Error Correction (SEC) and Double Error Detection (DED) parity to the Message Memory content. It has side band signals for error notification. The ECC Wrapper implements an error injection test mode.

The error correction is done using a lazy write back. When an error is detected, it is noted in a FIFO Queue which waits for an access gap to write the data back and refresh the memory. If a transaction writes new data to the compromised entry before the lazy write back completes, the write back is discarded.

12.4.1.4.6.2 ECC Aggregator

Note

For more information about ECC Aggregator, refer to *ECC Aggregator*.

12.4.1.4.7 Rx Handling

The Rx Handler controls the following operations:

- Acceptance filtering
- The transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs (Rx FIFO 0 or Rx FIFO 1)
- Rx FIFO Put and Get Index operations

12.4.1.4.7.1 Acceptance Filtering

The MCAN module is capable to configure two sets of acceptance filters - one set for standard and one set for extended identifiers. These filters can be assigned to an Rx Buffer or to one of the two Rx FIFOs.

The main features of the filter elements are:

- Each filter element can be configured as:
  - Range Filter (from - to)
  - Filter for specific IDs (for one or two dedicated IDs)
  - Classic Bit Mask Filter
- Each filter element can be enabled/disabled individually
- Each filter element can be configured for acceptance or rejection filtering
- Filters are checked sequentially and execution (acceptance filtering procedure) stops at the first matching filter element or when the end of the filter list is reached

Copyright © 2026 Texas Instruments Incorporated

Related configuration registers are:

- Global Filter Configuration (MCAN_GFC) register
- Standard ID Filter Configuration (MCAN_SIDFC) register
- Extended ID Filter Configuration (MCAN_XIDFC) register
- Extended ID AND Mask (MCAN_XIDAM) register

Depending on the configuration of the filter element (see SFEC/EFEC in Section 12.4.1.4.10, *Message RAM*) if filter matches, one of the following actions is performed:

- Received frame is stored in FIFO 0 or FIFO 1
- Received frame is stored in Rx Buffer
- Received frame is stored in Rx Buffer and generation of pulse at filter event pin is performed. This is high level single ICLK pulse. For more information, see Section 12.4.1.4.2.1, *DMA Requests*.
- Received frame is rejected
- Set High Priority Message interrupt flag MCAN_IR[8] HPM
- Set High Priority Message interrupt flag MCAN_IR[8] HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering starts when complete Message ID is received. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If a filter element matches - the Rx Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If an error condition occurs (for example: CRC error), this message is rejected with the following impact on the affected Rx Buffer or Rx FIFO:

- Rx Buffer:
  New Data flag (MCAN_NDAT1/MCAN_NDAT2) of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data (for error type see MCAN_PSR[2-0] LEC respectively MCAN_PSR[10-8] DLEC fields).
- Rx FIFO:
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data (for error type see MCAN_PSR[2-0] LEC respectively MCAN_PSR[10-8] DLEC fields). If matching Rx FIFO is configured to operate in overwrite mode, the boundary conditions described in Section 12.4.1.4.7.2.2 have to be considered.

12.4.1.4.7.1.1 Range Filter

Each filter element can be configured to operate as Range Filter (Standard Filter Type SFT = 00/Extended Filter Type EFT = 00). The filter matches for all received message frames with IDs in the range from SFID1 to SFID2 (SFID2 ≥ SFID1) respectively in the range from EFID1 to EFID2 (EFID2 ≥ EFID1). For more information see Section 12.4.1.4.10.5, *Standard Message ID Filter Element* and Section 12.4.1.4.10.6, *Extended Message ID Filter Element*.

There are two options for range filtering of extended frames:

- Extended Filter Type EFT = 00: The Extended ID AND Mask (MCAN_XIDAM) is used for Range Filtering. The Message ID of received frames is ANDed with the Extended ID AND Mask (MCAN_XIDAM) before the range filter is applied.
- Extended Filter Type EFT = 11: The Extended ID AND Mask (MCAN_XIDAM) is not used for Range Filtering.

12.4.1.4.7.1.2 Filter for specific IDs

Each filter element can be configured to filter one or two dedicated Message IDs (Standard Filter Type SFT =01/ Extended Filter Type EFT =01). To filter only one specific Message ID, the filter element has to be configured with SFID1 = SFID2 respectively EFID1 = EFID2. For more information see Section 12.4.1.4.10.5, *Standard Message ID Filter Element* and Section 12.4.1.4.10.6, *Extended Message ID Filter Element*.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.7.1.3 Classic Bit Mask Filter

Classic bit mask filtering can filter groups of Message IDs (Standard Filter Type SFT =10/Extended Filter Type EFT =10). This is done by masking single bits of a received Message ID. In this case SFID1/EFID1 element is used as Message ID filter, while SFID2/EFID2 element is used as filter mask.

A 0 bit at the filter mask (SFID2/EFID2) will mask out the corresponding bit position of the configured Message ID filter (SFID1/EFID1) and the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

There are two interesting cases:

- All mask bits are 1: a match occurs only when the received Message ID and the configured Message ID filter are identical.
- All mask bits are 0: all Message IDs match.

12.4.1.4.7.1.4 Standard Message ID Filtering

The standard Message ID (11-bit ID) filtering flow is shown in Figure 12-267. Section 12.4.1.4.10.5, *Standard Message ID Filter Element* describes the standard Message ID filter element.

The Remote Transmission Request (RTR) and Extended Identifier (XTD) bits of the received frames are compared against the list of configured filter elements. This is controlled by the following registers:

- Global Filter Configuration (MCAN_GFC) register
- Standard ID Filter Configuration (MCAN_SIDFC) register

Copyright © 2026 Texas Instruments Incorporated

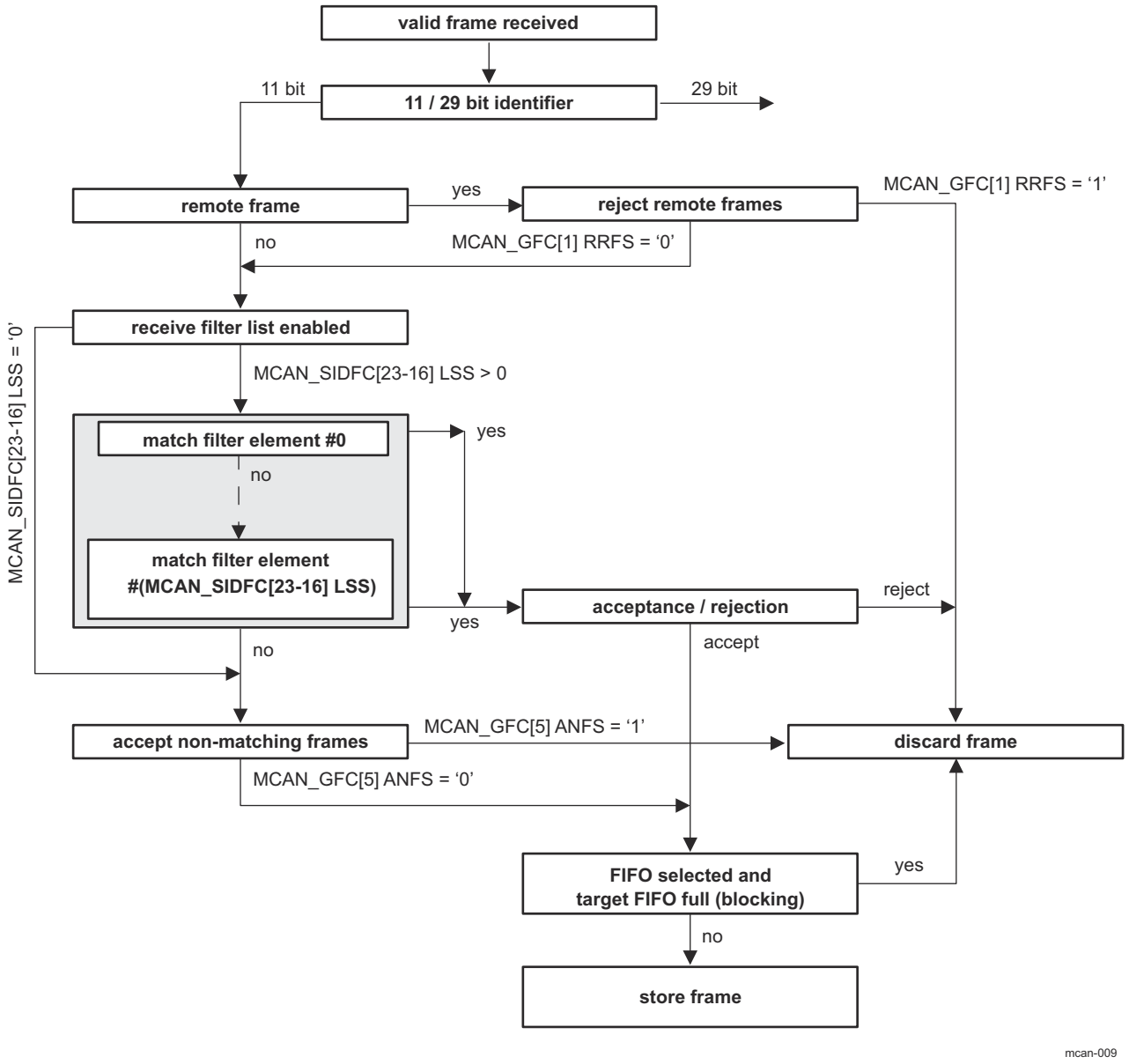


Figure 12-267. Standard Message ID Filter Path

12.4.1.4.7.1.5 Extended Message ID Filtering

The extended Message ID (29-bit ID) filtering flow is shown in Figure 12-268. Section 12.4.1.4.10.6, *Extended Message ID Filter Element* describes the extended Message ID filter element.

The Remote Transmission Request (RTR) and Extended Identifier (XTD) bits of the received frames are compared against the list of configured filter elements. This is controlled by the following registers:

- Global Filter Configuration (MCAN_GFC) register
- Extended ID Filter Configuration (MCAN_XIDFC) register

Note that before the filter list is executed the received identifier is ANDed with the Extended ID AND Mask (MCAN_XIDAM).

Copyright © 2026 Texas Instruments Incorporated

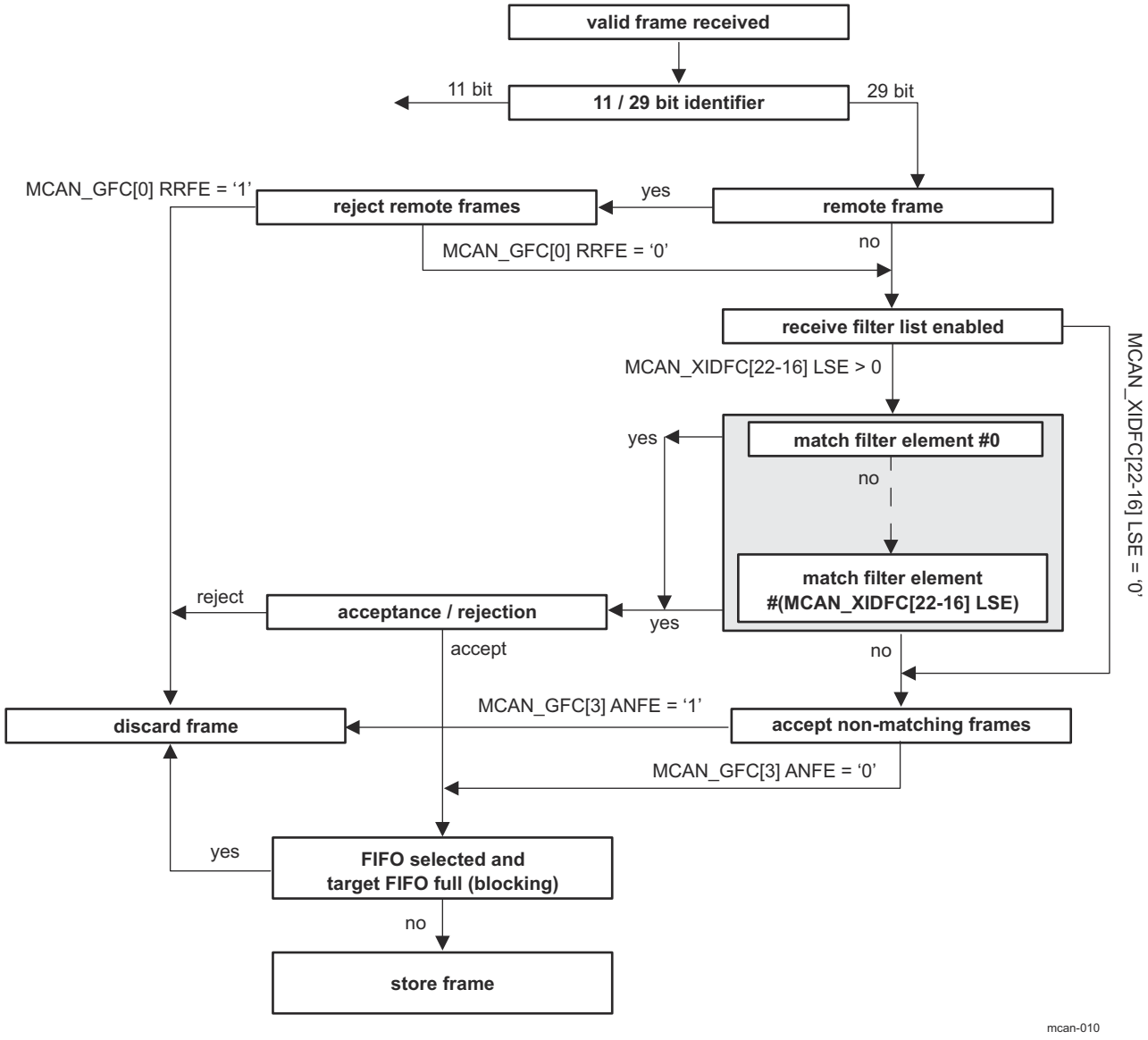


Figure 12-268. Extended Message ID Filter Path

12.4.1.4.7.2 Rx FIFOs

The configuration of the Rx FIFOs (Rx FIFO 0 and Rx FIFO 1) can be done via the MCAN_RXF0C and MCAN_RXF1C registers. Each Rx FIFO can be configured to store up to 64 received messages.

After acceptance filtering the received messages that passed are transferred to the Rx FIFO. The filter mechanisms available for the Rx FIFO 0 and Rx FIFO 1 is described in Section 12.4.1.4.7.1, *Acceptance Filtering*. Section 12.4.1.4.10.2, *Rx Buffer and FIFO Element* describes the Rx FIFO element.

The Rx FIFO watermark can be used to prevent an Rx FIFO overflow. If the Rx FIFO fill level reaches the Rx FIFO watermark configured by the MCAN_RXFnC[30-24] FnWM field (where: n = 0 or 1) an interrupt flag MCAN_IR[1] RF0W/MCAN_IR[5] RF1W is set.

When the Rx FIFO Put Index reaches the Rx FIFO Get Index (MCAN_RXFnS[21-16] FnPI = MCAN_RXFnS[13-8] FnGI) an Rx FIFO Full condition is signalled by the MCAN_RXFnS[24] FnF status bit and interrupt flag MCAN_IR[2] RF0F/MCAN_IR[6] RF1F is set. Figure 12-269 shows Rx FIFO Status. The FIFOs fill level is presented in the MCAN_RXFnS[6-0] FnFL field (the number of elements stored in Rx FIFO).

Copyright © 2026 Texas Instruments Incorporated

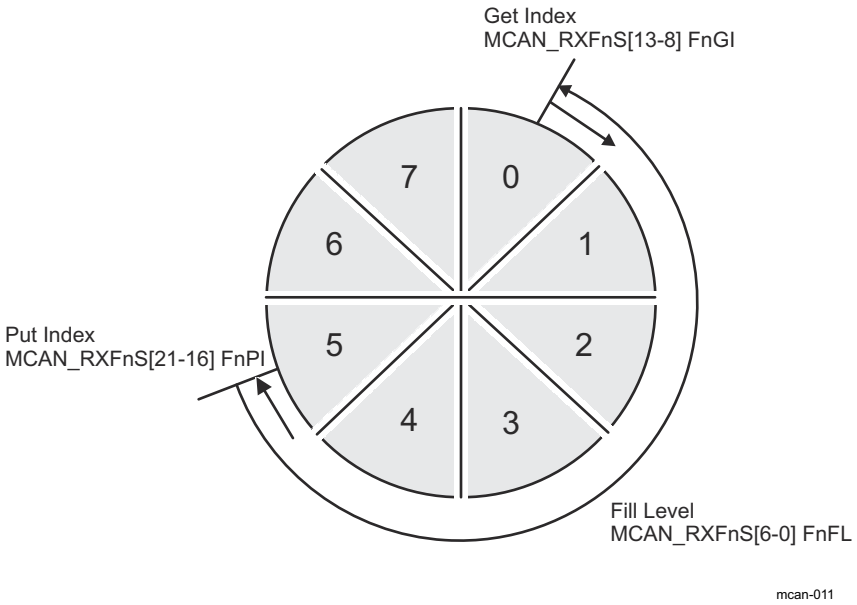


Figure 12-269. Rx FIFO Status

Rx FIFOs start address in the Message RAM (MCAN_RXFnC[15-2] FnSA field) have to be configured when reading from an Rx FIFO (Rx FIFO Get Index - MCAN_RXFnS[13-8] FnGI). Table 12-242 presents Rx Buffer/Rx FIFO Element Size for different Rx Buffer/Rx FIFO Data Field Size which is configured via the MCAN_RXESC register.

Table 12-242. Rx Buffer/Rx FIFO Element Size

MCAN_RXESC[10-8] RBDS MCAN_RXESC[2-0] F0DS/ MCAN_RXESC[6-4] F1DS	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

12.4.1.4.7.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is the default operation mode for the Rx FIFOs. It is configured by the MCAN_RXFnC[31] FnOM = 0.

If an Rx FIFO full condition is reached (MCAN_RXFnS[21-16] FnPI = MCAN_RXFnS[13-8] FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by the MCAN_RXFnS[24] FnF = 1 and interrupt flag MCAN_IR[2] RF0F/MCAN_IR[6] RF1F is set.

In case a message is received while the corresponding Rx FIFO is full, this message is rejected and the message lost condition is signalled by MCAN_RXFnS[25] RFnL = 1 and interrupt flag MCAN_IR[3] RFnL/ MCAN_IR[25] RFnL is set.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.7.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by the MCAN_RXFnC[31] FnOM = 1. When an Rx FIFO full condition is reached (MCAN_RXFnS[21-16] FnPI = MCAN_RXFnS[13-8] FnGI) signalled by MCAN_RXFnS[24] FnF = 1, the next accepted message for the FIFO will overwrite the oldest FIFO message. Put index/Get index are both incremented by one.

In overwrite mode if an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (Put index) while the Host CPU is reading from the Message RAM (Get index). In this case inconsistent data may be read from the respective Rx FIFO element. The problem is solved by adding an offset to the Get index when reading from the Rx FIFO. The offset depends on how fast the Host CPU accesses the Rx FIFO. Figure 12-270 shows an offset of two with respect to the Get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

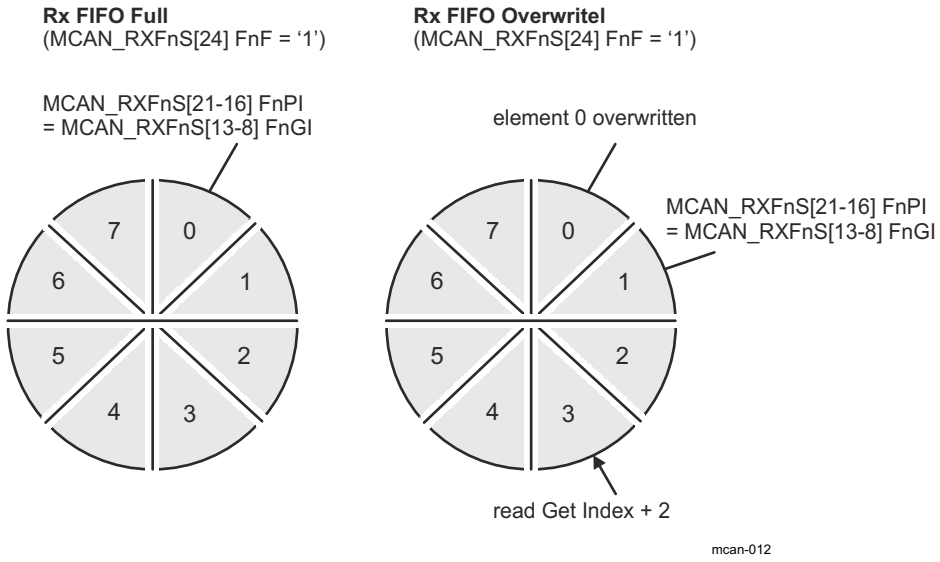


Figure 12-270. Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index MCAN_RXFnA[5-0] FnAI. This increments the get index to that element number. In case the Put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (MCAN_RXFnS[24] FnF = 0).

12.4.1.4.7.3 Dedicated Rx Buffers

The MCAN supports up to 64 dedicated Rx Buffers. The start address of the Rx Buffers section in the Message RAM is configured via MCAN_RXBC[15-2] RBSA field. To store in an Rx Buffer a Standard or Extended Message ID Filter Element with SFEC/EFEC = 111 and SFID2/EFID2[10-9] = 00 has to be configured (see Section 12.4.1.4.10.5, *Standard Message ID Filter Element* and Section 12.4.1.4.10.6, *Extended Message ID Filter Element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element (the format is the same as for an Rx FIFO element). In addition the flag MCAN_IR[19] DRX (Message stored in Dedicated Rx Buffer) is set.

Table 12-243 shows Example Filter Configuration for Rx Buffers.

Table 12-243. Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10-0] EFID1[28-0]	SFID2[10-9] EFID2[10-9]	SFID2[5-0] EFID2[5-0]
0	ID message 1	00	00 0000

Copyright © 2026 Texas Instruments Incorporated

Table 12-243. Example Filter Configuration for Rx Buffers (continued)

Filter Element	SFID1[10-0] EFID1[28-0]	SFID2[10-9] EFID2[10-9]	SFID2[5-0] EFID2[5-0]
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register MCAN_NDAT1/MCAN_NDAT2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host CPU by writing a 1 to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

12.4.1.4.7.3.1 Rx Buffer Handling

Rx Buffer Handling include the following steps:

- Reset interrupt flag MCAN_IR[19] DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

12.4.1.4.7.4 Debug on CAN Support

Debug DMA is not supported feature. Debug messages can be traced through the RX FIFO (see Section 12.4.1.4.7.2).

12.4.1.4.8 Tx Handling

The Tx Handler is used to handle the Tx requests. It controls the transfer of transmit messages from the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue to the CAN Core, the Tx Event FIFO, and the Put and Get Index operations. The MCAN module supports up to 32 Tx Buffers. These Tx Buffers can be configured as dedicated Tx Buffers, Tx FIFO, or Tx Queue and as combination of dedicated Tx Buffers/Tx FIFO or dedicated Tx Buffers/Tx Queue. For each Tx Buffer element Classical CAN or CAN FD transmission mode can be configured. Section 12.4.1.4.10.3 describes the Tx Buffer Element. Table 12-244 shows the possible configurations for message transmission.

Table 12-244. Possible Configurations for Message Transmission

MCAN_CCCR		Tx Buffer Element		Frame Transmission
MCAN_CCCR[9] BRSE	MCAN_CCCR[8] FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	CAN FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	CAN FD without bit rate switching
1	1	1	1	CAN FD with bit rate switching

When the Tx Buffer Request Pending MCAN_TXBRP register is updated, or when a transmission has been started the Tx Handler starts scanning to check for the highest priority pending Tx request. The Tx Buffer with lowest Message ID has highest priority.

Copyright © 2026 Texas Instruments Incorporated

Note

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

12.4.1.4.8.1 Transmit Pause

The transmit pause feature is intended for use in CAN networks where the CAN Message IDs are specific and cannot easily be changed. These Message IDs may have a higher priority than other defined Message IDs, while in a specific application their relative priority should be inverse. This allows for a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed (paused).

The transmit pause feature is enabled by the MCAN_CCCR[14] TXP bit. By default this bit is disabled (MCAN_CCCR[14] TXP = 0). Each time after successfully transmitted message, a pause for two CAN bit times occurs before the start of the next transmission. This allows the other CAN nodes in the network to transmit messages even if their Message IDs have lower priority.

12.4.1.4.8.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU.

There are two options:

- Each dedicated Tx Buffer is configured with a specific Message ID.
- Two or more dedicated Tx Buffers are configured with the same Message ID. In this case the Tx Buffer with the lowest buffer number is transmitted first.

After the data section has been updated, a transmission is requested by an Add Request. This is done via the MCAN_TXBAR[x]ARn bit (where x = 0 - 31). The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

Table 12-245 shows Tx Buffer/Tx FIFO/Tx Queue Element Size. A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM. The start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index from 0 to 31 (MCAN_TXFQS[20-16] TFQPI) × Element Size to the Tx Buffer Start Address MCAN_TXBC[15-2] TBSA field.

Table 12-245. Tx Buffer/Tx FIFO/Tx Queue Element Size

MCAN_TXESC[2-0] TBDS	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

12.4.1.4.8.3 Tx FIFO

Tx FIFO mode is configured by setting bit MCAN_TXBC[30] TFQM = 0. The stored in the Tx FIFO messages are transmitted starting with the message referenced by the Get Index MCAN_TXFQS[12-8] TFGI field. After each transmission the Get Index is incremented until the Tx FIFO is empty. The Tx FIFO Free Level MCAN_TXFQS[5-0] TFFL field indicates the number of the available free Tx FIFO elements. The Tx FIFO allows transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN_TXFQS[20-16] TFQPI field. After each Add Request (MCAN_TXBAR[x] ARn = 1) the

Copyright © 2026 Texas Instruments Incorporated

Put Index is incemented to the next free Tx FIFO element. When the Put Index reaches the Get Index (MCAN_TXFQS[20-16] TFQPI = MCAN_TXFQS[12-8] TFGI), Tx FIFO Full condition is signalled by bit MCAN_TXFQS[21] TFQF = 1. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level MCAN_TXFQS[5-0] TFFL field.

In case a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level MCAN_TXFQS[5-0] TFFL field is recalculated. In case transmission cancellation is applied to any other Tx Buffer - the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see Table 12-245). The start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS[20-16] TFQPI (from 0 to 31) × Element Size to the Tx Buffer Start Address MCAN_TXBC[15-2] TBSA field.

12.4.1.4.8.4 Tx Queue

Tx Queue mode is configured by setting bit MCAN_TXBC[30] TFQM = 1. The stored in the Tx Queue messages are transmitted starting with the highest priority message (lowest Message ID). In case two or more Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN_TXFQS[20-16] TFQPI field. Each Add Request cyclically increments the Put Index to the next free Tx Buffer. In case of Tx Queue Full condition (MCAN_TXFQS[21] TFQF = 1), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use the MCAN_TXBRP register instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see Table 12-245). The start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS[20-16] TFQPI (from 0 to 31) × Element Size to the Tx Buffer Start Address MCAN_TXBC[15-2] TBSA field.

12.4.1.4.8.5 Mixed Dedicated Tx Buffers/Tx FIFO

For this combination the Tx Buffers section in the Message RAM is separated in two parts:

- Dedicated Tx Buffers: the number of Dedicated Tx Buffers is configured by the MCAN_TXBC[21-16] NDTB field
- Tx FIFO: the number of Tx Buffers assigned to the Tx FIFO is configured by the MCAN_TXBC[29-24] TFQS field

If the MCAN_TXBC[29-24] TFQS field is empty (zero) - only Dedicated Tx Buffers are used.

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by the MCAN_TXFQS[12-8] TFGI field)
- Buffer with lowest Message ID gets highest priority and is transmitted next

Figure 12-271 shows Mixed Dedicated Tx Buffers/Tx FIFO example.

Copyright © 2026 Texas Instruments Incorporated

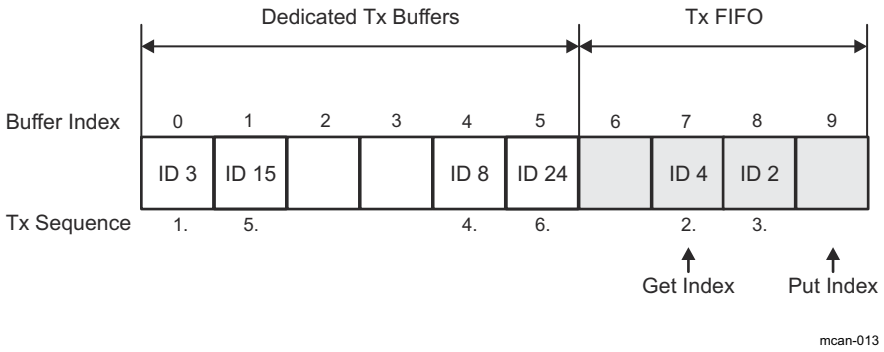


Figure 12-271. Mixed Dedicated Tx Buffers/Tx FIFO (example)

12.4.1.4.8.6 Mixed Dedicated Tx Buffers/Tx Queue

For this combination the Tx Buffers section in the Message RAM is separated in two parts:

- Dedicated Tx Buffers: the number of Dedicated Tx Buffers is configured by the MCAN_TXBC[21-16] NDTB field
- Tx Queue: the number of Tx Buffers assigned to the Tx Queue is configured by the MCAN_TXBC[29-24] TFQS field

If MCAN_TXBC[29-24] TFQS field is empty (zero) - only Dedicated Tx Buffers are used.

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

Figure 12-272 shows Mixed Dedicated Tx Buffers/Tx Queue example.

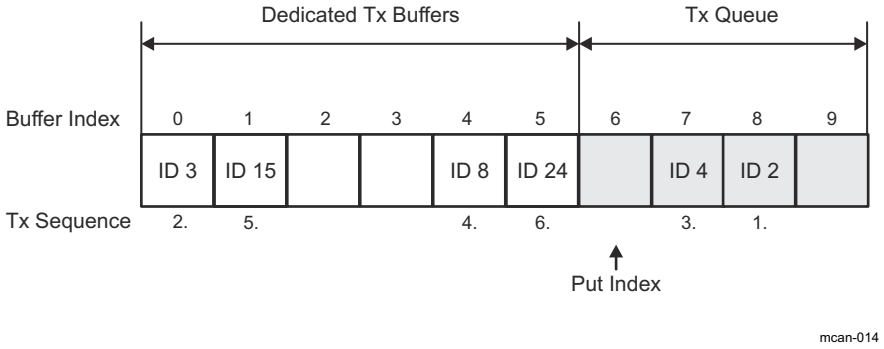


Figure 12-272. Mixed Dedicated Tx Buffers/Tx Queue (example)

12.4.1.4.8.7 Transmit Cancellation

This feature is especially intended for gateway and AUTOSAR based applications. The Host CPU can cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer by setting bit MCAN_TXBCR[n] CRn = 1 (where n = 0 - 31). The corresponding bit position n is equivalent to the number of the Tx Buffer.

Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of the MCAN_TXBCF register (MCAN_TXBCF[n] CFn = 1).

If transmission from a Tx Buffer is already ongoing and a transmit cancellation is requested, the corresponding MCAN_TXBRP[n] TRPn bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding MCAN_TXBTO[n] TOn and MCAN_TXBCF[n] CFn bits are set. If the transmission was not successful, only the corresponding bit MCAN_TXBCF[n] CFn = 1.

Copyright © 2026 Texas Instruments Incorporated

Note

If pending transmission is cancelled immediately before this transmission could have been started, a short time window occurs where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

12.4.1.4.8.8 Tx Event Handling

To support Tx Event Handling the Message RAM has implemented a Tx Event FIFO section. Up to 32 Tx Event FIFO elements can be configured. Section 12.4.1.4.10.4 describes the Tx Event FIFO element. After message transmission on the CAN bus, Message ID and Timestamp are stored in a Tx Event FIFO element. To link a Tx Event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

A Tx Event FIFO full condition is signalled by the MCAN_IR[14] TEFF bit. In this case no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented (MCAN_TXEFS[12-8] EFGI). In case a Tx Event occurs while the Tx Event FIFO is full, this event is rejected and interrupt flag MCAN_IR[15] TEFL bit is set.

The Tx Event FIFO watermark can be configured to avoid a Tx Event FIFO overflow. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by the MCAN_TXEFC[29-24] EFWM field, interrupt flag MCAN_IR[13] TEFW is set. When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index MCAN_TXEFS[12-8] EFGI field has to be added to the Tx Event FIFO start address MCAN_TXEFC[15-2] EFSA field.

12.4.1.4.9 FIFO Acknowledge Handling

The Get Indices of the two Rx FIFOs (Rx FIFO 0 or Rx FIFO 1) and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see MCAN_RXF0A, MCAN_RXF1A, and MCAN_TXEFA). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level.

There are two use cases:

- A single element has been read from the FIFO: the Get Index value is written to the FIFO Acknowledge Index.
- A sequence of elements has been read from the FIFO: the Get Index value (Index of the last element read) is written to the FIFO Acknowledge Index at the end of that read sequence.

The Host CPU has free access to the Message RAM. The special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This can be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also changes the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The MCAN module does not check for erroneous values.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.10 Message RAM

The MCAN module has implemented Message RAM. The main purpose of the Message RAM is to store:

- Receive Messages
- Transmit Messages
- Tx Event Elements
- Message ID Filter Elements

12.4.1.4.10.1 Message RAM Configuration

The MCAN module is configured to allocate 4352 words in the Message RAM. The Message RAM has a width of 32 bits.

The Message RAM is capable to include each of the sections listed in Figure 12-273. It is not necessary to configure each of the sections (a section in the Message RAM may be 0) and there is not restriction with respect to the sequence of the sections. For parity checking or ECC a respective number of bits has to be added to each word.

When the MCAN module addresses the Message RAM it addresses 32-bit words. The start addresses are configurable and they are 32-bit word addresses.

The element size can be configured for:

- Rx FIFO 0 via the MCAN_RXESC[2-0] F0DS field
- Rx FIFO 1 via the MCAN_RXESC[6-4] F1DS field
- Rx Buffers via the MCAN_RXESC[10-8] RBDS field
- Tx Buffers via the MCAN_TXESC[2-0] TBDS field

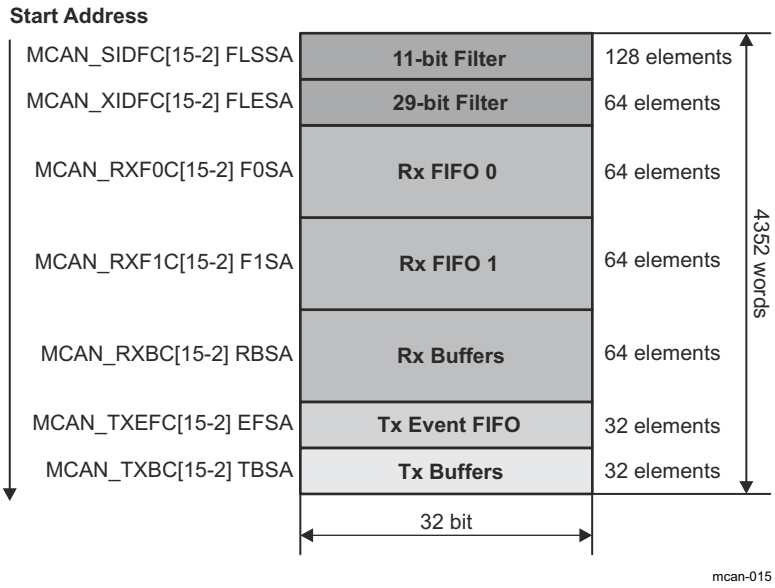


Figure 12-273. Message RAM Configuration

The Host CPU configures the following information in the Message RAM:

- Start addresses of the memory sections
- Number of elements in each section
- The size of the elements in some sections

Copyright © 2026 Texas Instruments Incorporated

Note

The MCAN module does not check for errors in the Message RAM configuration. The configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully. This will prevent falsification or loss of data.

12.4.1.4.10.2 Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the MCAN_RXESC register.

Figure 12-274 shows Rx Buffer/Rx FIFO element structure.

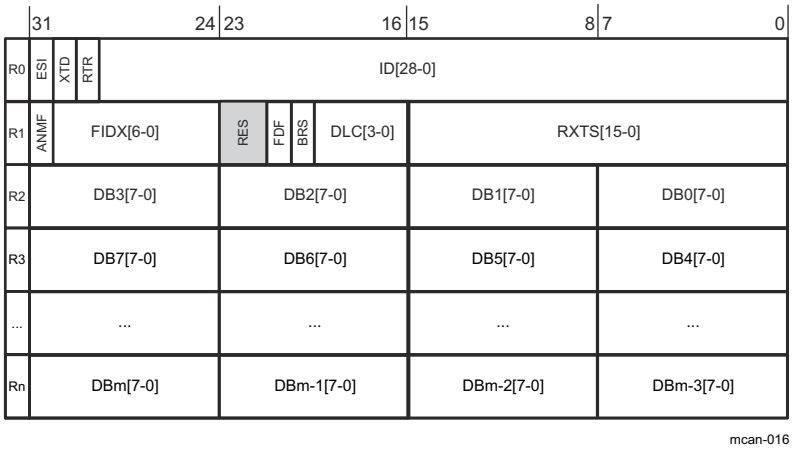


Figure 12-274. Rx Buffer/Rx FIFO Element Structure

Table 12-246 shows Rx Buffer/Rx FIFO element field descriptions.

Table 12-246. Rx Buffer/Rx FIFO Element Field Descriptions

Word	Bits	Field Name	Description
R0	31	ESI	Error State Indicator • 0h = Transmitting node is error active • 1h = Transmitting node is error passive
	30	XTD	Extended Identifier Signals to the Host CPU whether the received frame has a standard or extended identifier. • 0h = 11-bit standard identifier • 1h = 29-bit extended identifier
	29	RTR	Remote Transmission Request Signals to the Host CPU whether the received frame is a data frame or a remote frame. • 0h = Received frame is a data frame • 1h = Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (FDF = 1), RTR bit reflects the state of the reserved r1 bit (RES[23]).
	28-0	ID[28-0]	Identifier Standard or extended identifier depending on XTD bit. A standard identifier is stored into ID[28-18].

Copyright © 2026 Texas Instruments Incorporated

Table 12-246. Rx Buffer/Rx FIFO Element Field Descriptions (continued)

Word	Bits	Field Name	Description
R1	31	ANMF	Accepted Non-matching Frame Acceptance of non-matching frames may be enabled via the MCAN_GFC[5-4] ANFS and MCAN_GFC[3-2] ANFE fields. • 0h = Received frame matching filter index FIDX field • 1h = Received frame did not match any Rx filter element
	30-24	FIDX[6-0]	Filter Index 0h-7Fh (0-127): Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to MCAN_SIDFC[23-16] LSS - 1 respectively MCAN_XIDFC[22-16] LSE - 1.
	23-22	RES	Reserved
	21	FDF	FD Format • 0h = Standard frame format • 1h = CAN FD frame format (new DLC-coding and CRC)
	20	BRS	Bit Rate Switch • 0h = Frame received without bit rate switching • 1h = Frame received with bit rate switching
	19-16	DLC[3-0]	Data Length Code • 0h-8h (0-8) = CAN + CAN FD: received frame has 0-8 data bytes • 9h-Fh (9-15) = CAN: received frame has 8 data bytes • 9h-Fh (9-15) = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
	15-0	RXTS[15-0]	Rx Timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC[19-16] TCP.
R2	31-24	DB3[7-0]	Data Byte 3
	23-16	DB2[7-0]	Data Byte 2
	15-8	DB1[7-0]	Data Byte 1
	7-0	DB0[7-0]	Data Byte 0
R3	31-24	DB7[7-0]	Data Byte 7
	23-16	DB6[7-0]	Data Byte 6
	15-8	DB5[7-0]	Data Byte 5
	7-0	DB4[7-0]	Data Byte 4
...	...	...	...
Rn	31-24	DBm[7-0]	Data Byte m
	23-16	DBm-1[7-0]	Data Byte m-1
	15-8	DBm-2[7-0]	Data Byte m-2
	7-0	DBm-3[7-0]	Data Byte m-3

Note: Depending on the configuration of the element size (MCAN_RXESC), between two and sixteen 32-bit words (Rn = 3-17) are used for storage of a CAN message's data field.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.10.3 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO/Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO/Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler makes difference between dedicated Tx Buffers and Tx FIFO/Tx Queue via the MCAN_TXBC[29-24] TFQS and MCAN_TXBC[21-16] NDTB fields. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the MCAN_TXESC register.

Figure 12-275 shows Tx Buffer element structure.

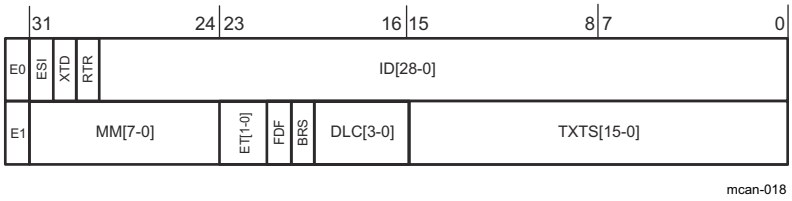


Figure 12-275. Tx Buffer Element Structure

Table 12-247 shows Tx Buffer element field descriptions.

Table 12-247. Tx Buffer Element Field Descriptions

Word	Bits	Field Name	Description
T0	31	ESI	Error State Indicator • 0h = ESI bit in CAN FD format depends only on error passive flag • 1h = ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted CAN FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.
	30	XTD	Extended Identifier • 0h = 11-bit standard identifier • 1h = 29-bit extended identifier
	29	RTR	Remote Transmission Request • 0h = Transmit data frame • 1h = Transmit remote frame **Note:** When RTR = 1, the MCAN module transmits a remote frame according to ISO11898-1:2015, even if the MCAN_CCCR[8] FDOE bit enables the transmission in CAN FD format.
	28-0	ID[28-0]	Identifier Standard or extended identifier depending on XTD bit. A standard identifier has to be written to ID[28-18].

Copyright © 2026 Texas Instruments Incorporated

Table 12-247. Tx Buffer Element Field Descriptions (continued)

Word	Bits	Field Name	Description
T1	31-24	MM[7-0]	Message Marker Written by Host CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status (see also MM[7-0] field in Table 12-248).
	23	EFC	Event FIFO Control • 0h = Don't store Tx events • 1h = Store Tx events
	22	RES	Reserved
	21	FDF	FD Format • 0h = Frame transmitted in Classic CAN format • 1h = Frame transmitted in CAN FD format
	20	BRS	Bit Rate Switch • 0h = CAN FD frames transmitted without bit rate switching • 1h = CAN FD frames transmitted with bit rate switching **Note:** ESI, FDF, and BRS bits are only evaluated when CAN FD operation is enabled vie the MCAN_CCCR[8] FDOE bit. BRS bit is only evaluated when in addition the MCAN_CCCR[9] BRSE = 1.
	19-16	DLC[3-0]	Data Length Code • 0h-8h (0-8) = CAN + CAN FD: transmit frame has 0-8 data bytes • 9h-Fh (9-15) = CAN: transmit frame has 8 data bytes • 9h-Fh (9-15) = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
	15-0	RES	Reserved
T2	31-24	DB3[7-0]	Data Byte 3
	23-16	DB2[7-0]	Data Byte 2
	15-8	DB1[7-0]	Data Byte 1
	7-0	DB0[7-0]	Data Byte 0
T3	31-24	DB7[7-0]	Data Byte 7
	23-16	DB6[7-0]	Data Byte 6
	15-8	DB5[7-0]	Data Byte 5
	7-0	DB4[7-0]	Data Byte 4
...	...	...	...
Tn	31-24	DBm[7-0]	Data Byte m
	23-16	DBm-1[7-0]	Data Byte m-1
	15-8	DBm-2[7-0]	Data Byte m-2
	7-0	DBm-3[7-0]	Data Byte m-3

Note: Depending on the configuration of the element size (MCAN_TXESC), between two and sixteen 32-bit words (Tn = 3-17) are used for storage of a CAN message's data field.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.10.4 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from the MCAN_TXEFS register.

Figure 12-276 shows Tx Event FIFO element structure.

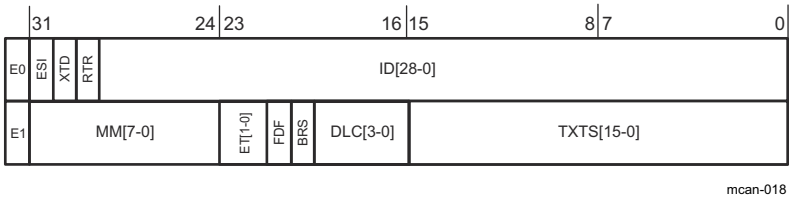


Figure 12-276. Tx Event FIFO Element Structure

Table 12-248 shows Tx Event FIFO element field descriptions.

Table 12-248. Tx Event FIFO Element Field Descriptions

Word	Bits	Field Name	Description
E0	31	ESI	Error State Indicator • 0h = Transmitting node is error active • 1h = Transmitting node is error passive
	30	XTD	Extended Identifier • 0h = 11-bit standard identifier • 1h = 29-bit extended identifier
	29	RTR	Remote Transmission Request • 0h = Data frame transmitted • 1h = Remote frame transmitted
	28-0	ID[28-0]	Identifier Standard or extended identifier depending on XTD bit. A standard identifier has to be written to ID[28-18].

Copyright © 2026 Texas Instruments Incorporated

Table 12-248. Tx Event FIFO Element Field Descriptions (continued)

Word	Bits	Field Name	Description
E1	31-24	MM[7-0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status (see also MM[7-0] field in Table 12-247).
	23-22	ET[1-0]	Event Type • 0h = Reserved • 1h = Tx event • 2h = Transmission in spite of cancellation (always set for transmissions in DAR mode) • 3h = Reserved
	21	FDF	FD Format • 0h = Standard frame format • 1h = CAN FD frame format (new DLC-coding and CRC)
	20	BRS	Bit Rate Switch • 0h = Frame transmitted without bit rate switching • 1h = Frame transmitted with bit rate switching
	19-16	DLC[3-0]	Data Length Code • 0h-8h (0-8) = CAN + CAN FD: frame with 0-8 data bytes transmitted • 9h-Fh (9-15) = CAN: frame with 8 data bytes transmitted • 9h-Fh (9-15) = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
	15-0	TXTS[15-0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler MCAN_TSCC[19-16] TCP field.

12.4.1.4.10.5 Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address MCAN_SIDFC[15-2] FLSSA field plus the index of the filter element (0-127).

Figure 12-277 shows Standard Message ID Filter element structure.

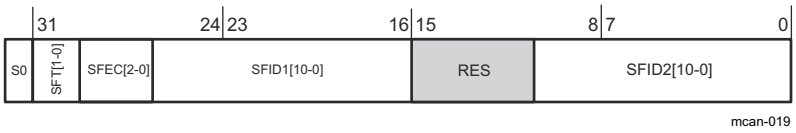


Figure 12-277. Standard Message ID Filter Element Structure

Table 12-249 shows Standard Message ID Filter element field descriptions.

Copyright © 2026 Texas Instruments Incorporated

Table 12-249. Standard Message ID Filter Element Field Descriptions

Word	Bits	Field Name	Description
S0	31-30	SFT[1-0]	Standard Filter Type • 0h = Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) • 1h = Dual ID filter for SFID1 or SFID2 • 2h = Classic filter: SFID1 = filter; SFID2 = mask • 3h = Filter element disabled **Note:** With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = 000)
	29-27	SFEC[2-0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101, or 110 a match sets interrupt flag MCAN_IR[8]HPM and, if enabled, an interrupt is generated. In this case the MCAN_HPMS register is updated with the status of the priority match. • 0h = Disable filter element • 1h = Store in Rx FIFO 0 if filter matches • 2h = Store in Rx FIFO 1 if filter matches • 3h = Reject ID if filter matches • 4h = Set priority if filter matches • 5h = Set priority and store in FIFO 0 if filter matches • 6h = Set priority and store in FIFO 1 if filter matches • 7h = Store into Rx Buffer , configuration of SFT[1-0] ignored
	26-16	SFID1[10-0]	Standard Filter ID 1 When filtering for Rx Buffers this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
	15-11	RES	Reserved
	10-0	SFID2[10-0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: • 1) SFEC = 001 - 110 Second ID of standard ID filter element • 2) SFEC = 111 Filter for Rx Buffers
		SFID2[10-9]	This field is decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. • 0h = Store message into an Rx Buffer • 1h = Debug Message A • 2h = Debug Message B • 3h = Debug Message C **Note:** Debug feature is not supported.
		SFID2[8-6]	This field is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN_ICKL period in case the filter matches. **Note:** Only three filter event pins are supported.
		SFID2[5-0]	This field defines the offset to the Rx Buffer Start Address MCAN_RXBC[15-2] RBSA field for storage of a matching message.

Copyright © 2026 Texas Instruments Incorporated

12.4.1.4.10.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCAN_XIDFC[15-2] FLESA field plus two times the index of the filter element (0-63).

Figure 12-278 shows Extended Message ID Filter element structure.

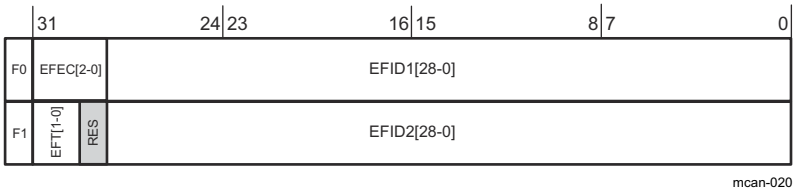


Figure 12-278. Extended Message ID Filter Element Structure

Table 12-250 shows Extended Message ID Filter element field descriptions.

Table 12-250. Extended Message ID Filter Element Field Descriptions

Word	Bits	Field Name	Description
F0	31-29	EFEC[2-0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag MCAN_IR[8]HPM and, if enabled, an interrupt is generated. In this case the MCAN_HPMS register is updated with the status of the priority match. • 0h = Disable filter element • 1h = Store in Rx FIFO 0 if filter matches • 2h = Store in Rx FIFO 1 if filter matches • 3h = Reject ID if filter matches • 4h = Set priority if filter matches • 5h = Set priority and store in FIFO 0 if filter matches • 6h = Set priority and store in FIFO 1 if filter matches • 7h = Store into Rx Buffer or as debug message, configuration of EFT[1-0] ignored
	28-0	EFID1[28-0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see Section 12.4.1.4.7.1.5, *Extended Message ID Filtering*) is used.

Copyright © 2026 Texas Instruments Incorporated

Table 12-250. Extended Message ID Filter Element Field Descriptions (continued)

Word	Bits	Field Name	Description
F1	31-30	EFT[1-0]	Extended Filter Type • 0h = Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1) • 1h = Dual ID filter for EFID1 or EFID2 • 2h = Classic filter: EFID1 = filter, EFID2 = mask • 3h = Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied
	29	RES	Reserved
	28-0	EFID2[28-0]	Extended Filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: • 1) EFEC = 001 - 110 Second ID of extended ID filter element • 2) EFEC = 111 Filter for Rx Buffers
		EFID2[10-9]	This field decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. • 0h = Store message into an Rx Buffer • 1h = Debug Message A • 2h = Debug Message B • 3h = Debug Message C **Note:** Debug feature is not supported.
		EFID2[8-6]	This field is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN_ICKL period in case the filter matches. **Note:** Only three filter event pins are supported.
		EFID2[5-0]	This field defines the offset to the Rx Buffer Start Address MCAN_RXBC[15-2] RBSA field for storage of a matching message.

12.4.2 Enhanced Capture (ECAP) Module

This section describes the Enhanced Capture (ECAP) module in the device.

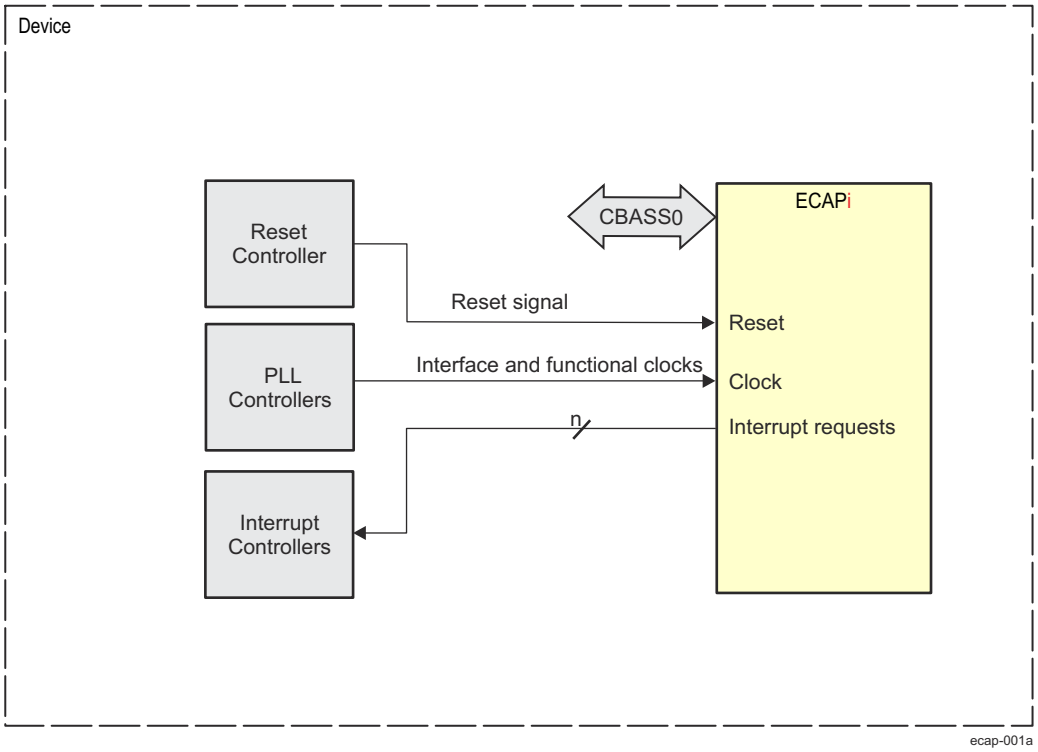
12.4.2.1 ECAP Overview

The Enhanced Capture (ECAP) module can be used for:

- Sample rate measurements of audio inputs
- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

Figure 12-279 shows the ECAP modules overview.

Copyright © 2026 Texas Instruments Incorporated



A. i = 0 to number of instances - 1.

Figure 12-279. ECAP Overview

Copyright © 2026 Texas Instruments Incorporated

12.4.2.1.1 ECAP Features

The ECAP module includes the following features:

- 32-bit time base counter
- 4 × 32 bits event time-stamp capture registers (ECAP0_CAP1 through ECAP0_CAP4)
- 4-stage sequencer (Mod4 counter), synchronized to external events (ECAPx pin edges)
- Independent edge polarity (rising / falling edge) selection for all 4 sequenced time-stamp capture events
- Input capture signal pre-scaling (from 1 to 16)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Interrupt capabilities on any of the 4 capture events
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

12.4.2.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

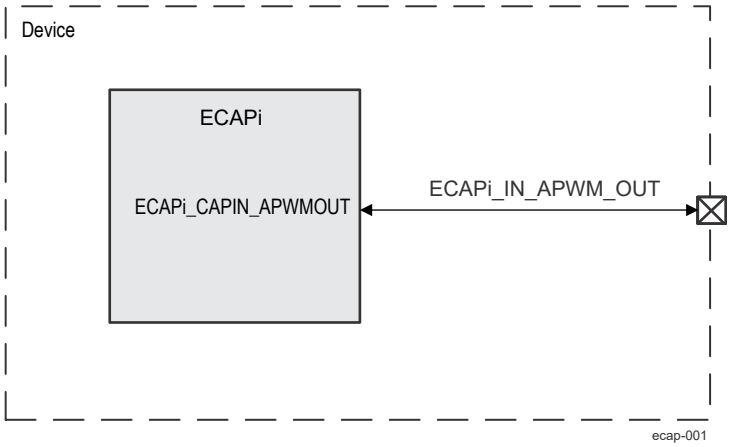
Copyright © 2026 Texas Instruments Incorporated

12.4.2.2 ECAP Environment

12.4.2.2.1 ECAP I/O Interface

This section describes the ECAP external connections (environment).

Figure 12-280 shows the ECAP interface signals.



Note

i represents an ECAP instance. See the device datasheet for available domains and ECAP instances

Figure 12-280. ECAP External Interface I/Os

Table 12-251 describes the ECAP I/O signals.

Table 12-251. ECAP I/O Signals

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
		ECAPi(3)		
ECAPi(3)_CAPIN_APWMOUT	ECAPi(3)_IN_APWM_OUT	I/O	ECAPi(3) Capture input / PWM output	HiZ

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) i represents an ECAP instance. See the device datasheet for available domains and ECAP instances

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.4.2.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4 ECAP Functional Description

The ECAP module represents one complete capture channel, which has the following independent key resources:

- Dedicated input capture pin
- 32 events selection mapping capability to the input capture pin
- 32-bit time base counter
- 4 × 32-bit time-stamp capture registers (ECAP0_CAP1 through ECAP0_CAP4)
- 4-stage sequencer (Mod4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (ECAP0_CAP1 through ECAP0_CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

Multiple identical ECAP modules can be contained in a system as shown in Figure 12-281. For actual number of the ECAP modules integrated in the device, refer to *ECAP Integration*. The letter x within a signal or module name is used to indicate a generic ECAP instance on a device. For example, output interrupt request, ECAP1INT belongs to ECAP1, ECAP2INT belongs to ECAP2, and so forth.

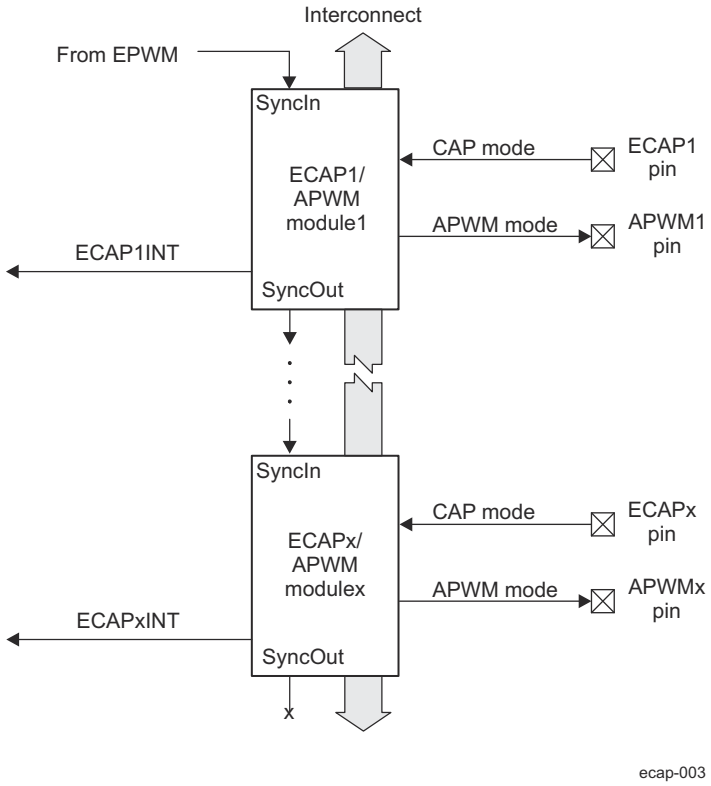
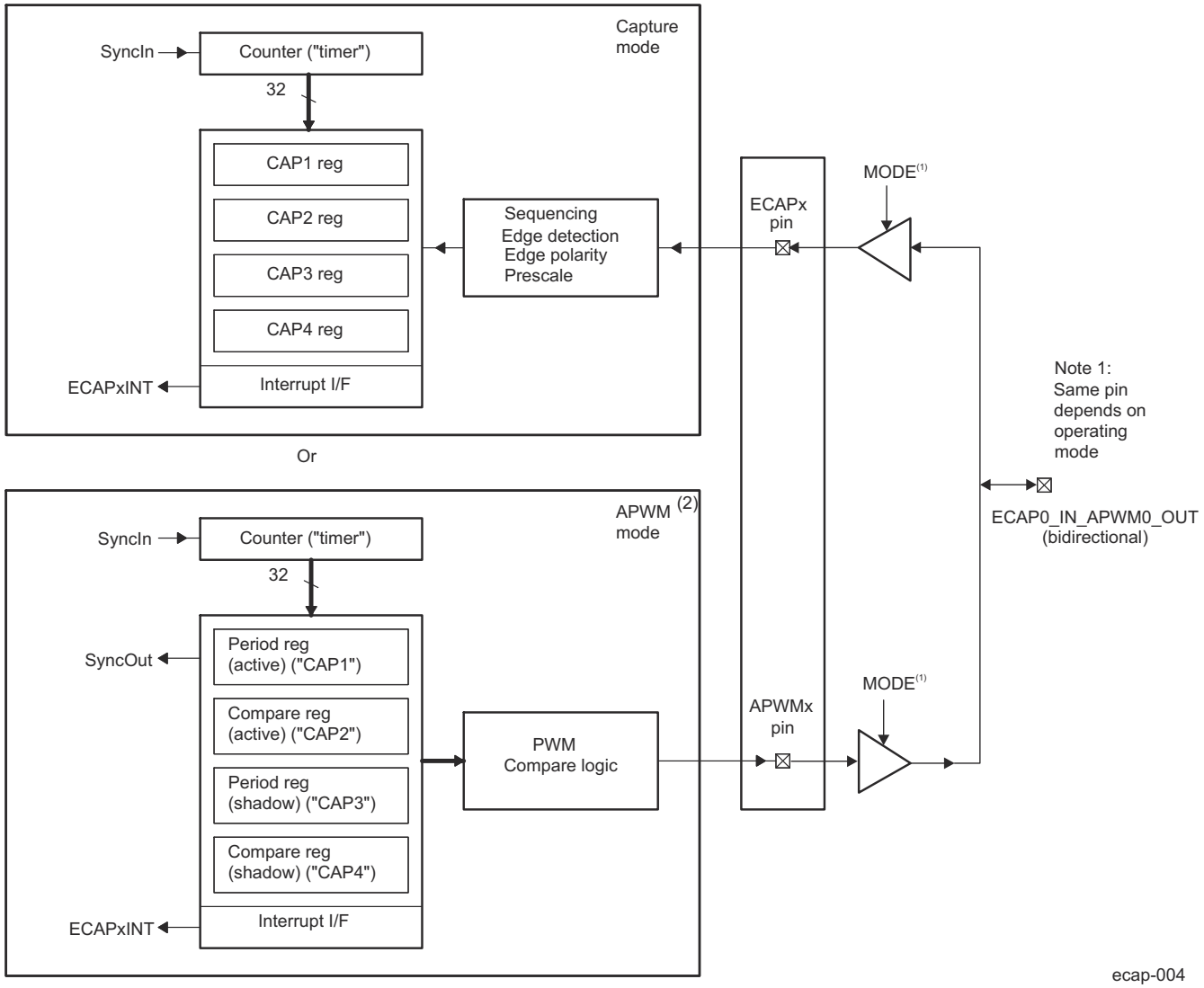


Figure 12-281. Multiple ECAP Modules

12.4.2.4.1 Capture and APWM Operating Modes

The ECAP module resources can be used to implement a single-channel PWM generator (with 32-bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The ECAP0_CAP1 and ECAP0_CAP2 registers become the active period and compare registers, respectively, while the ECAP0_CAP3 and ECAP0_CAP4 registers become the period and capture shadow registers, respectively. Figure 12-282 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Copyright © 2026 Texas Instruments Incorporated



A. A single pin is shared between CAP and APWM functions. In capture mode, it is an input. In APWM mode, it is an output.

B. In APWM mode, writing any value to the ECAP0_CAP1/ECAP0_CAP2 active registers also writes the same value to the corresponding shadow registers (ECAP0_CAP3/ECAP0_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP0_CAP3/ ECAP0_CAP4) invokes the shadow mode.

Figure 12-282. Capture and APWM Modes of Operation

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4.1.1 ECAP Capture Mode Description

Figure 12-283 shows the various components that implement the capture function.

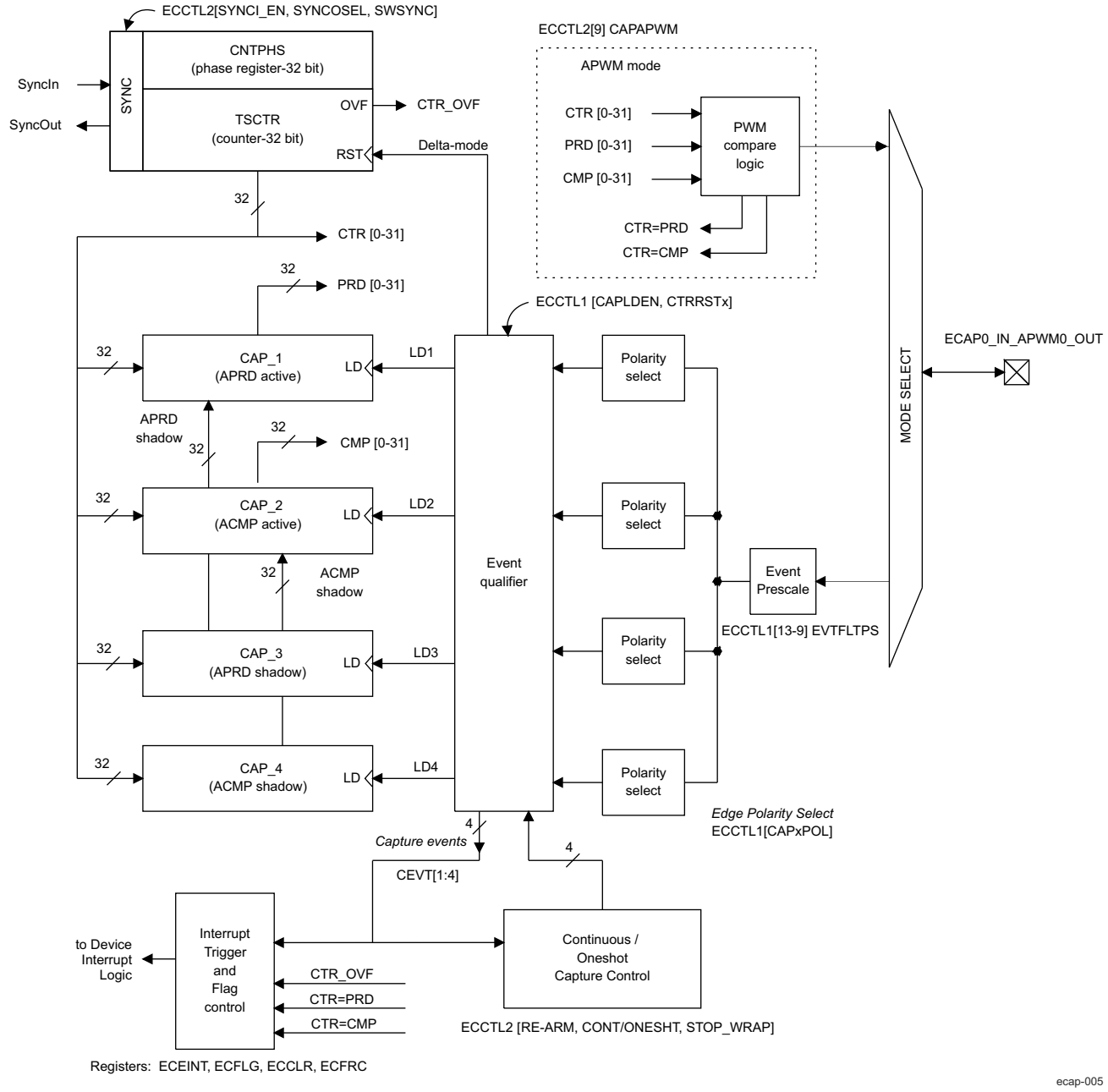
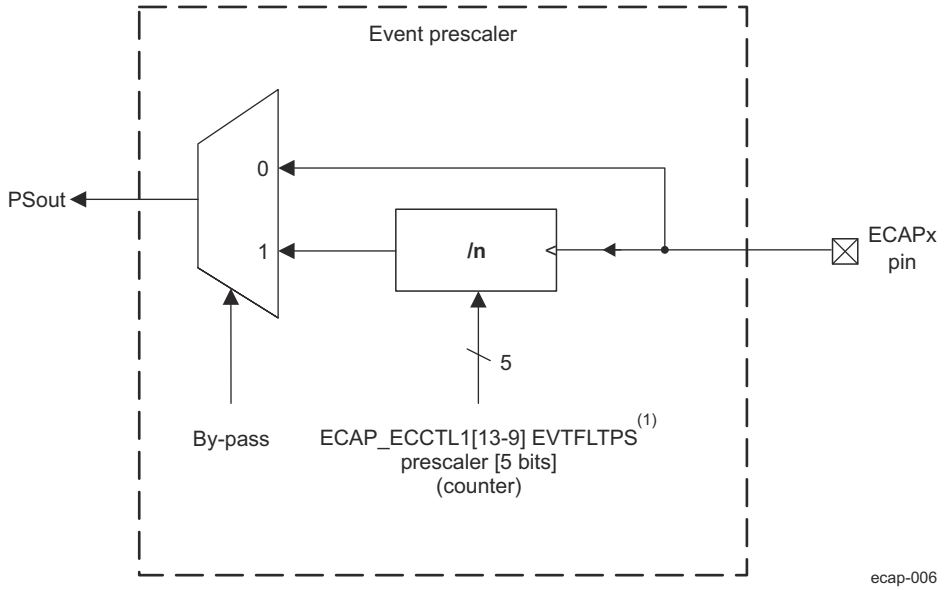


Figure 12-283. Capture Function Diagram

12.4.2.4.1.1.1 ECAP Event Prescaler

An input capture signal (pulse train) can be prescaled by N = 2-62 (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. Figure 12-284 shows a functional diagram and Figure 12-285 shows the operation of the prescale function.

Copyright © 2026 Texas Instruments Incorporated



A. When a prescale value of 1 is chosen (ECAP0_ECCTL[13-9] EVTFLTPS = 0b0000) the input capture signal by-passes the prescale logic completely.

Figure 12-284. Event Prescale Control

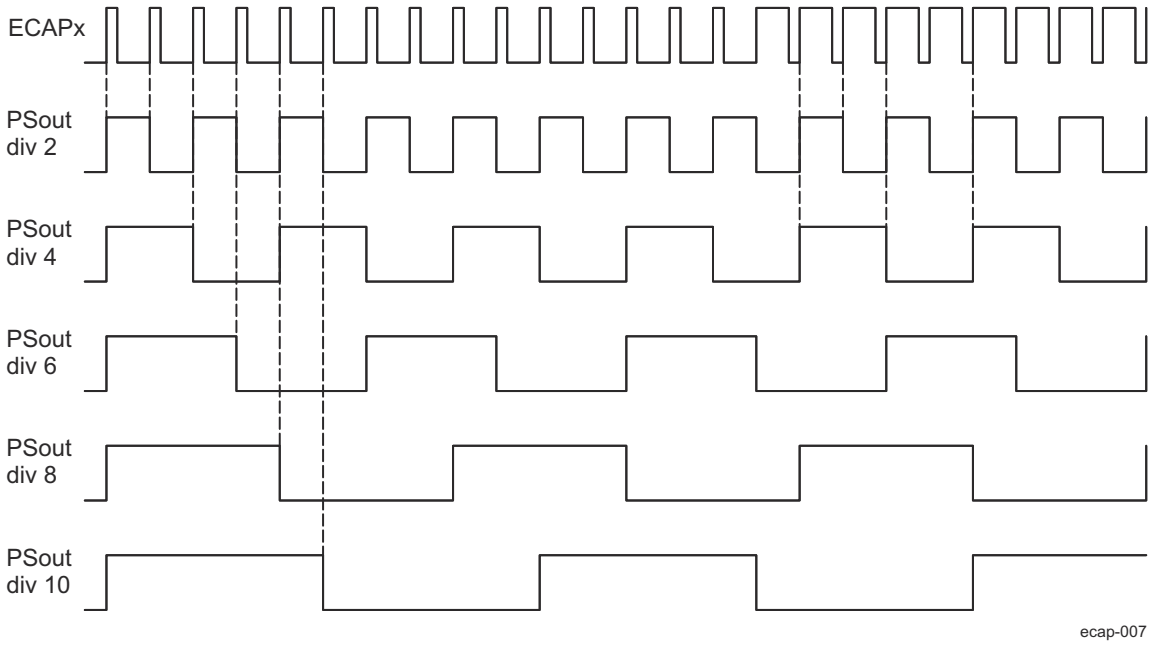


Figure 12-285. Prescale Function Waveforms

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4.1.1.2 ECAP Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection multiplexers are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Mod4 sequencer.
- The edge event is gated to its respective CAP*n* register by the Mod4 counter. The CAP*n* register is loaded on the falling edge.

12.4.2.4.1.1.3 ECAP Continuous/One-Shot Control

- The Mod4 (2-bit) counter is incremented via edge qualified events CEVT1 through CEVT4 (see the ECAP0_ECINT_EN_FLG register).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output; when equal the Mod4 counter stops and inhibits further loads of the ECAP0_CAP1 through ECAP0_CAP4 registers. This occurs during one-shot operation.

The continuous/one-shot block (Figure 12-286) controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the ECAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of the ECAP0_CAP1 through ECAP0_CAP4 registers (time-stamps).

Re-arming prepares the ECAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of the ECAP0_CAP1 through ECAP0_CAP4 registers again, providing the ECAP0_ECCTL[8] CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0), the one-shot action is ignored, and capture values continue to be written to the ECAP0_CAP1 through ECAP0_CAP4 registers in a circular buffer sequence.

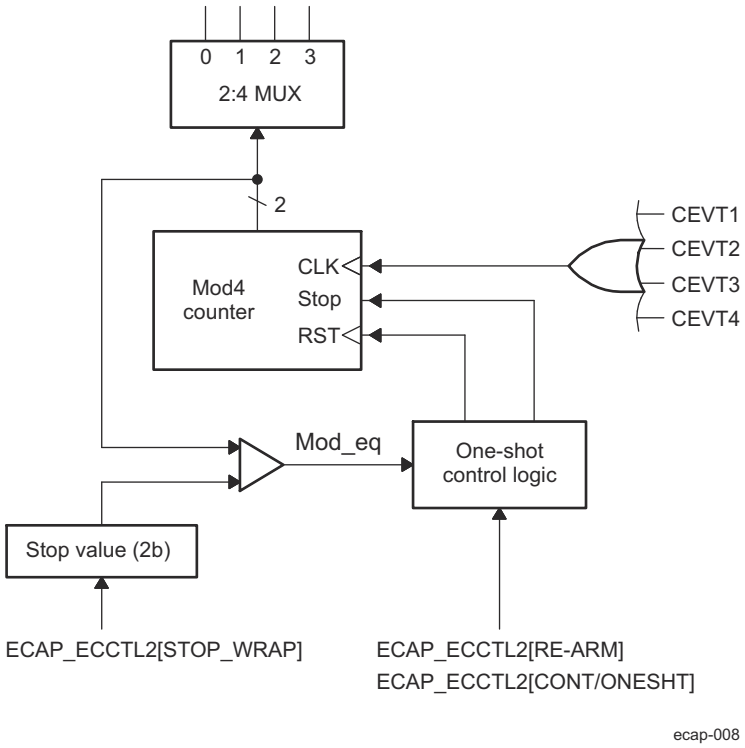


Figure 12-286. ECAP Continuous/One-shot Block Diagram

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4.1.1.4 ECAP 32-Bit Counter and Phase Control

This counter (Figure 12-287) provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1 through LD4 signals.

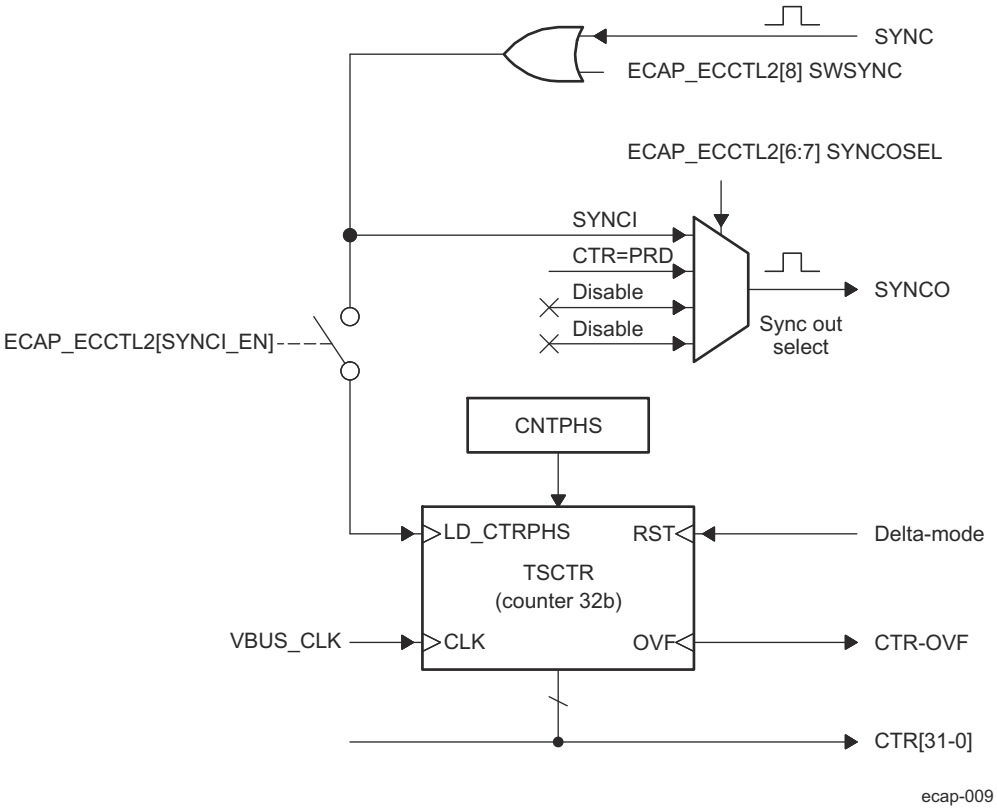


Figure 12-287. ECAP Counter and Synchronization Block Diagram

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4.1.1.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via the control ECAP0_ECCTL[8] CAPLDEN bit. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

The ECAP0_CAP1 and ECAP0_CAP2 registers become the active period and compare registers, respectively, in APWM mode.

The ECAP0_CAP3 and ECAP0_CAP4 registers become the respective shadow registers (APRD and ACMP) for the ECAP0_CAP1 and ECAP0_CAP2 registers during APWM operation.

12.4.2.4.1.1.6 ECAP Interrupt Control

An interrupt can be generated on capture events CEVT1 through CEVT4, CNTOVF (see the ECAP0_ECINT_EN_FLG[21] CNTOVF_FLG bit) or APWM events (TSCNT = PRD, TSCNT = CMP). See Figure 12-288.

A counter overflow event (FFFF FFFFh->0000 0000h) is also provided as an interrupt source (CNTOVF).

The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the ECAP*n* module) going to the interrupt controller.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, TSCNT = PRD, TSCNT = CMP) can be generated. The interrupt enable register (ECAP0_ECINT_EN_FLG) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECAP0_ECINT_EN_FLG) indicates if any interrupt event has been latched and contains the global interrupt flag ECAP0_ECINT_EN_FLG[16] INT_FLG bit. An interrupt pulse is generated to the interrupt controller only if any of the interrupt events are enabled, the flag bit is 1h, and the INT_FLG flag bit is 0h. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECAP0_ECINT_CLR_FRC) before any other interrupt pulses are generated. The interrupt force register (ECAP0_ECINT_CLR_FRC) can force an interrupt event. This is useful for test purposes.

12.4.2.4.1.1.7 ECAP Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of the ECAP0_CAP1 or ECAP0_CAP2 registers from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to the ECAP0_CAP1 or ECAP0_CAP2 register immediately upon writing a new value.
- On period equal, CTR[31-0] = PRD[31-0]

Note

The CEVT1_FLG, CEVT2_FLG, CEVT3_FLG, CEVT4_FLG flags are only active in capture mode (ECAP0_ECCTL[25] CAP_APWM == 0h). The TSCNT = PRD, TSCNT = CMP flags are only valid in APWM mode (ECAP0_ECCTL[25] CAP_APWM == 1h). CNTOVF_FLG flag is valid in both modes.

Copyright © 2026 Texas Instruments Incorporated

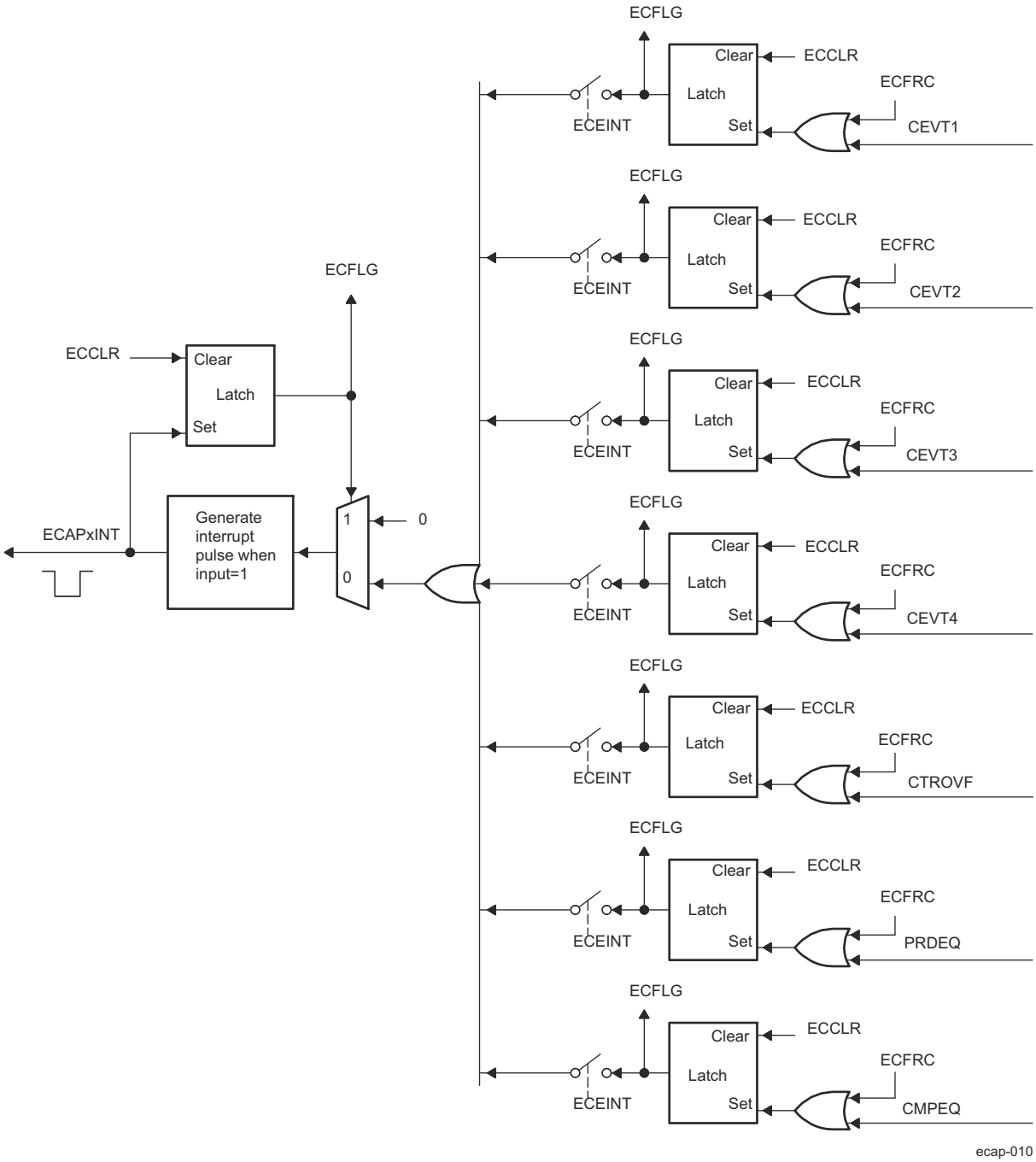


Figure 12-288. Interrupts in ECAP Module

Copyright © 2026 Texas Instruments Incorporated

12.4.2.4.1.2 ECAP APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When the ECAP0_CAP1/ECAP0_CAP2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via the shadow registers - APRD and ACMP (ECAP0_CAP3/ECAP0_CAP4). The shadow register contents are transferred over to ECAP0_CAP1/ECAP0_CAP2 registers either immediately upon a write, or on a TSCNT = PRD trigger.
- In APWM mode, writing to the ECAP0_CAP1/ECAP0_CAP2 active registers will also write the same value to the corresponding shadow registers (ECAP0_CAP3/ECAP0_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP0_CAP3/ECAP0_CAP4) will invoke the shadow mode.
- During initialization, software must write the PRD and CMP values to the active registers. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, software should use only shadow registers.

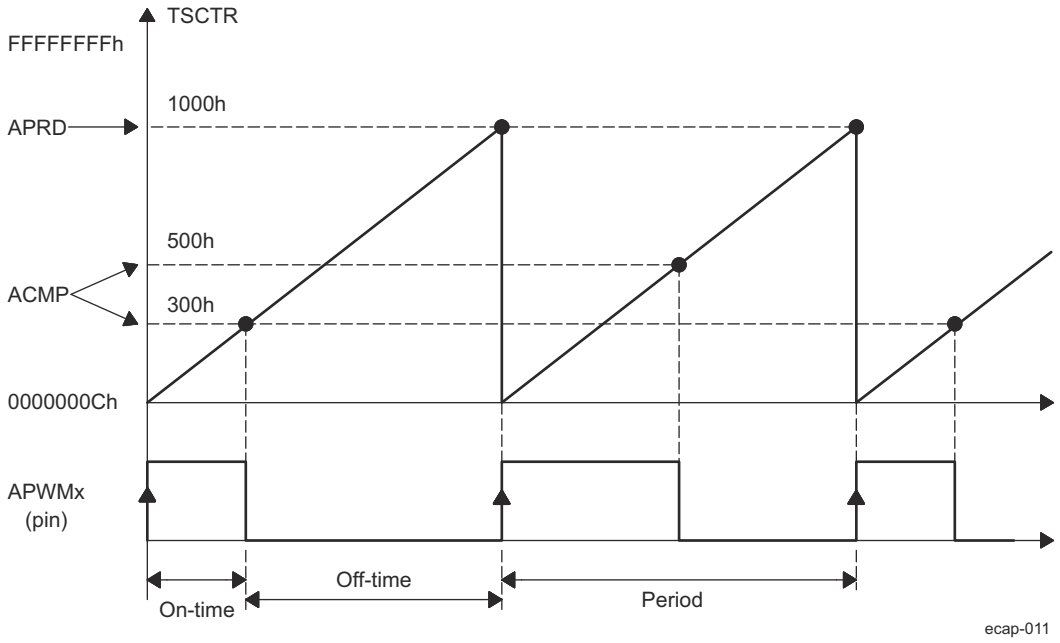


Figure 12-289. PWM Waveform Details of ECAP APWM Mode Operation

The behavior of APWM active-high mode (APWMPOL == 0) is:

```
CMP = 0x00000000, output low for duration of period (0% duty)
```

```
CMP = 0x00000001, output high 1 cycle
```

```
CMP = 0x00000002, output high 2 cycles
```

```
CMP = PERIOD, output high except for 1 cycle (<100% duty)
```

```
CMP = PERIOD+1, output high for complete period (100% duty)
```

```
CMP > PERIOD+1, output high for complete period
```


Copyright © 2026 Texas Instruments Incorporated

The behavior of APWM active-low mode (APWMPOL == 1) is:

```
CMP = 0x00000000, output high for duration of period (0% duty)
```

```
CMP = 0x00000001, output low 1 cycle
```

```
CMP = 0x00000002, output low 2 cycles
```

```
CMP = PERIOD, output low except for 1 cycle (<100% duty)
```

```
CMP = PERIOD+1, output low for complete period (100% duty)
```

```
CMP > PERIOD+1, output low for complete period
```

12.4.2.4.2 Summary of ECAP Functional Registers

Table 12-252 shows the ECAP module control and status register set. All 32-bit registers are aligned on even address boundaries and are organized in little-endian mode.

Note

In APWM mode, writing to the ECAP0_CAP1/ECAP0_CAP2 active registers also writes the same value to the corresponding shadow registers (ECAP0_CAP3/ECAP0_CAP4). This emulates immediate mode. Writing to the shadow registers (ECAP0_CAP3/ECAP0_CAP4) invokes the shadow mode.

Table 12-252. ECAP Control and Status Functional Registers

Offset	Register Name	Description	Size (×16)
0h	ECAP0_TSCNT	Time-Stamp Counter Register	2
4h	ECAP0_CNTPHS	Counter Phase Offset Value Register	2
8h	ECAP0_CAP1	Capture 1 Register	2
Ch	ECAP0_CAP2	Capture 2 Register	2
10h	ECAP0_CAP3	Capture 3 Register	2
14h	ECAP0_CAP4	Capture 4 Register	2
28h	ECAP0_ECCTL	Capture Control Register	2
2Ch	ECAP0_ECINT_EN_FLG	Capture Interrupt Enable and Flag Register	2
30h	ECAP0_ECINT_CLR_FRC	Capture Interrupt Clear and Forcing Register	2
5Ch	ECAP0_PID	Revision ID Register	2

Note

For more information on the ECAP registers, see *ECAP Registers*.

Copyright © 2026 Texas Instruments Incorporated

12.4.2.5 ECAP Use Cases

The following sections will provide applications examples and code snippets to show how to configure and operate the ECAP module. For clarity and ease of use, below are useful #defines which will help in the understanding of the examples.

```
// ECCTL1 ( ECAP Control Reg 1)
//==========================
// CAPxPOL bits
#defineEC_RISING      0x0
#defineEC_FALLING     0x1
// CTRRSTx bits
#defineEC_ABS_MODE    0x0
#defineEC_DELTA_MODE      0x1
// PRESCALE bits
#defineEC_BYPASS      0x0
#defineEC_DIV1        0x0
#defineEC_DIV2        0x1
#defineEC_DIV4        0x2
#defineEC_DIV6        0x3
#defineEC_DIV8        0x4
#defineEC_DIV10       0x5
// ECCTL2 ( ECAP Control Reg 2)
//==========================
// CONT/ONESHOT bit
#defineEC_CONTINUOUS      0x0
#defineEC_ONESHOT     0x1
// STOPVALUE bit
#defineEC_EVENT1      0x0
#defineEC_EVENT2      0x1
#defineEC_EVENT3      0x2
#defineEC_EVENT4      0x3
// RE-ARM bit
#defineEC_ARM         0x1
// TSCTRSTOP bit
#defineEC_FREEZE      0x0
#defineEC_RUN         0x1
// SYNCO_SEL bit
#defineEC_SYNCIN      0x0
#defineEC_CTR_PRD     0x1
#defineEC_SYNCO_DIS       0x2
// CAP/APWM mode bit
#defineEC_CAP_MODE    0x0
#defineEC_APWM_MODE       0x1
// APWMPOL bit
#defineEC_ACTV_HI     0x0
#defineEC_ACTV_LO     0x1
// Generic
#defineEC_DISABLE     0x0
#defineEC_ENABLE      0x1
#defineEC_FORCE       0x1
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.1 Absolute Time-Stamp Operation Rising Edge Trigger Example

Figure 12-290 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFF FFFFh (maximum value), it wraps around to 0000 0000h (not shown in Figure 12-290), if this occurs, the CNTOVF_FLG (counter overflow) flag is set, and an interrupt (if enabled) occurs. Captured time-stamps are valid at the point indicated by the diagram, after the 4-th event, hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAP*n* registers.

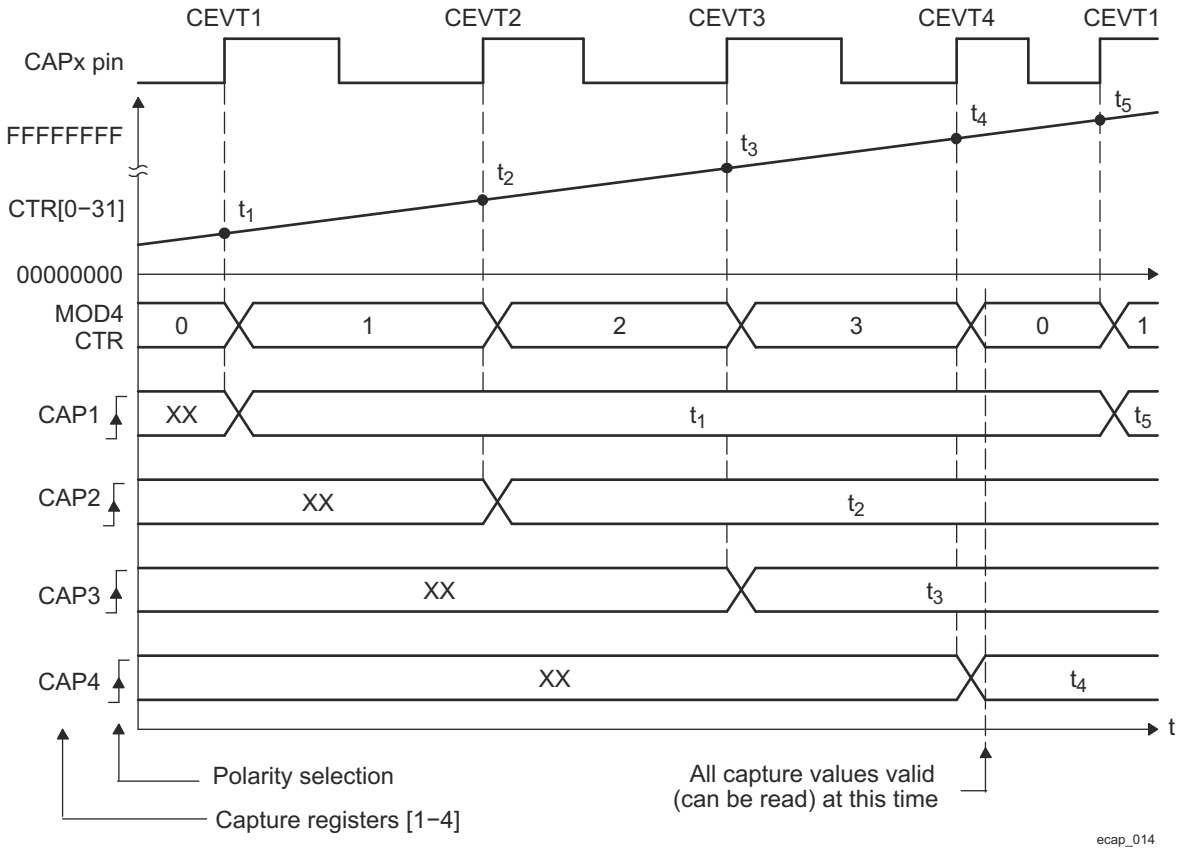


Figure 12-290. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect

Copyright © 2026 Texas Instruments Incorporated

Table 12-253. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-1. Code Snippet for CAP Mode Absolute Time, Rising Edge Trigger

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger
// Run Time ( e.g. CEVT4 triggered ISR call)
//=======================================
TSt1 = ECAPxRegs.CAP1;// Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;// Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;// Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;// Fetch Time-Stamp captured at t4
Period1 = TSt2-TSt1;// Calculate 1st period
Period2 = TSt3-TSt2;// Calculate 2nd period
Period3 = TSt4-TSt3;// Calculate 3rd period
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example

In Figure 12-291 the ECAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, …etc. Duty Cycle1 (on-time %) = $(t_2 - t_1)$ / Period1 x 100%, etc. Duty Cycle1 (off-time %) = $(t_3 - t_2)$ / Period1 x 100%, etc.

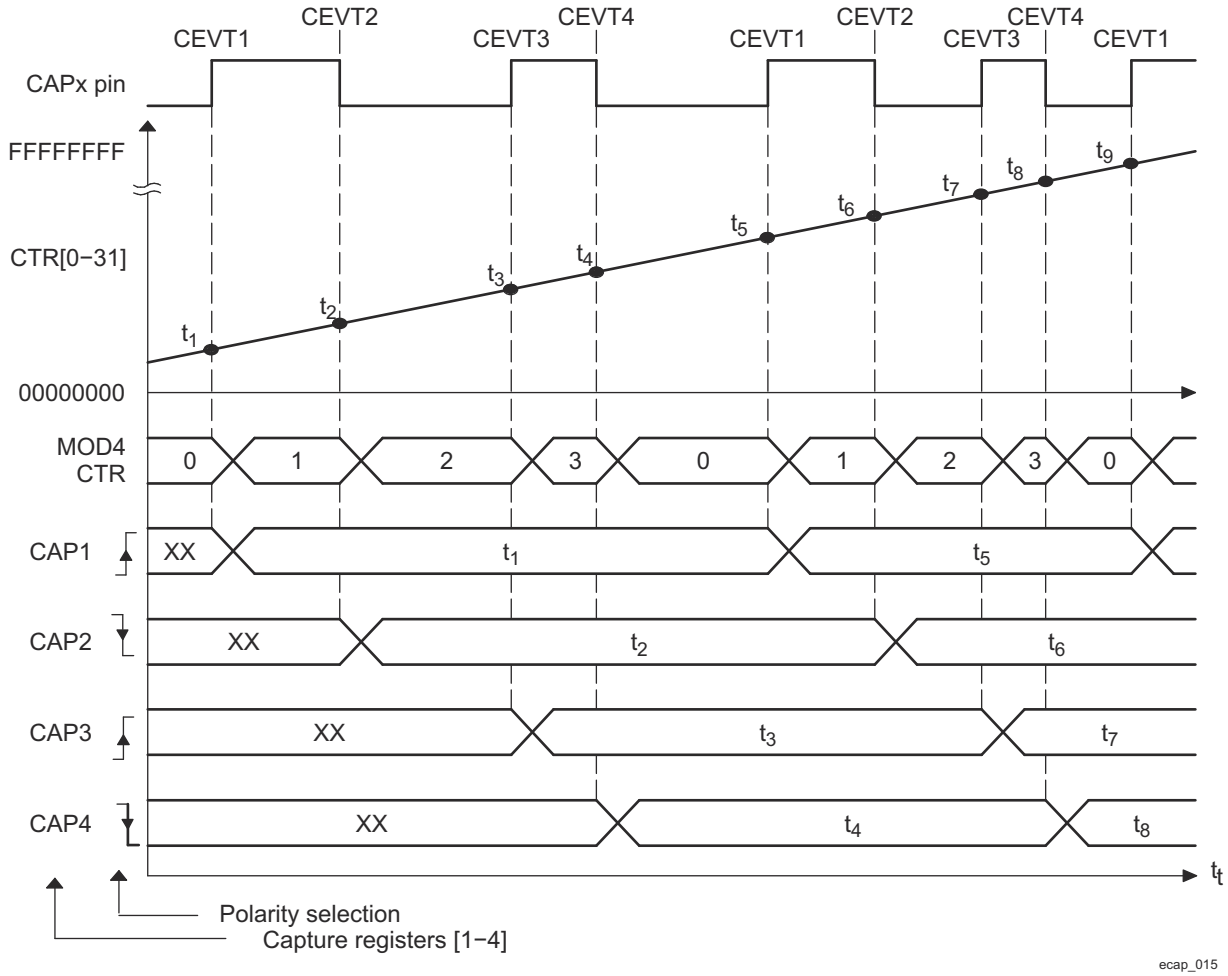


Figure 12-291. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect

Copyright © 2026 Texas Instruments Incorporated

Table 12-254. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-2. Code Snippet for CAP Mode Absolute Time, Rising and Falling Edge Trigger

```
// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers
// Run Time ( e.g. CEVT4 triggered ISR call)
//=======================================
TSt1 = ECAPxRegs.CAP1;// Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;// Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;// Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;// Fetch Time-Stamp captured at t4
Period1 = TSt3-TSt1;// Calculate 1st period
DutyOnTime1 = TSt2-TSt1;// Calculate On time
DutyOffTime1 = TSt3-TSt2;// Calculate Off time
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.3 Time Difference (Delta) Operation Rising Edge Trigger Example

Figure 12-292 shows how the ECAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (time-stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFF FFFFh (maximum value), before the next event, it wraps around to 0000 0000h and continues, a CNTOVF_FLG (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAP*n* contents directly give timing data without the need for CPU calculations: Period1 = T_1, Period2 = T_2,…etc. As shown in Figure 12-292, the CEVT1 event is a good trigger point to read the timing data, T_1, T_2, T_3, T_4 are all valid here.

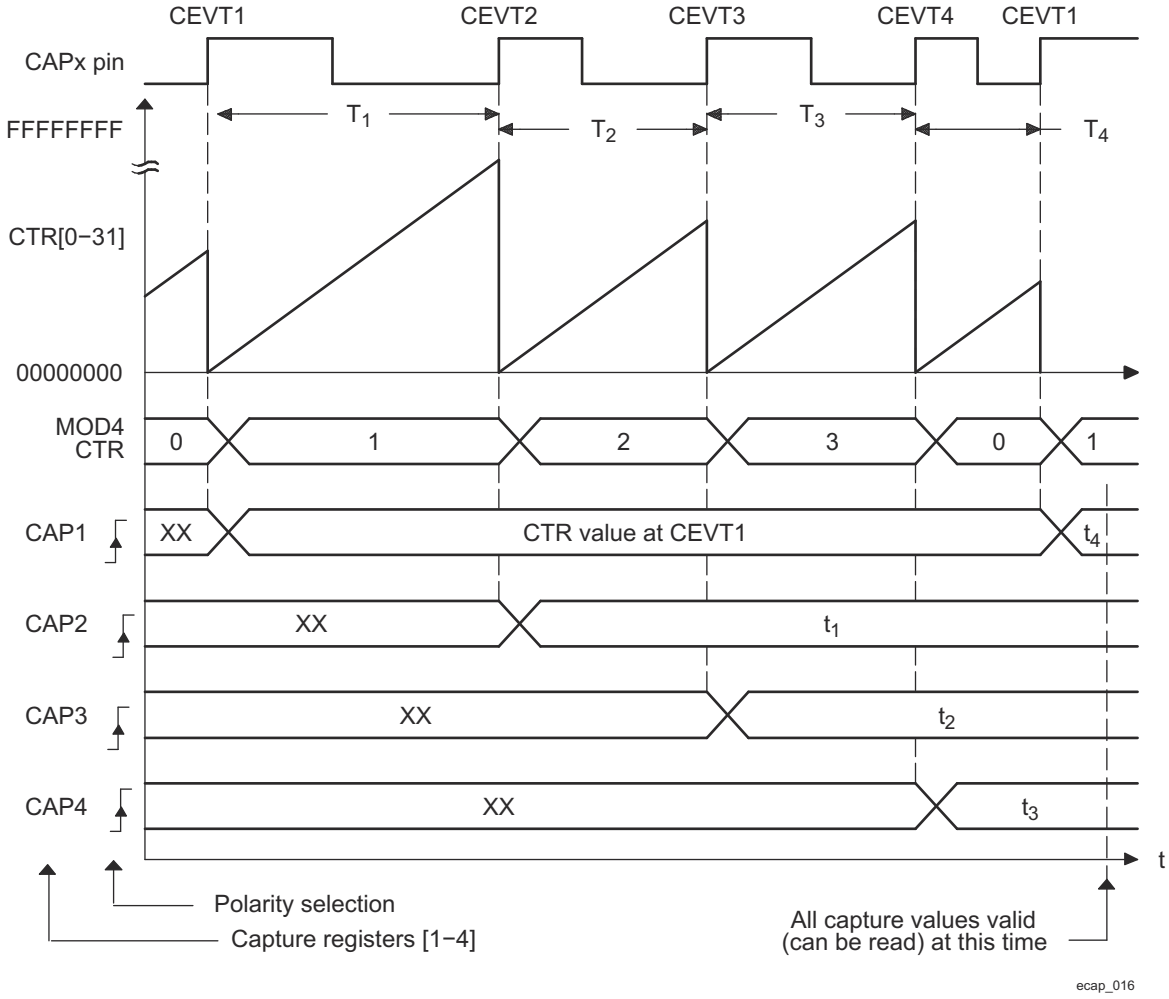


Figure 12-292. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect

Copyright © 2026 Texas Instruments Incorporated

Table 12-255. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-3. Code Snippet for CAP Mode Delta Time, Rising Edge Trigger

```
// Code snippet for CAP mode Delta Time, Rising edge trigger
// Run Time ( e.g. CEVT1 triggered ISR call)
//=======================================
// Note: here Time-stamp directly represents the Period value.
Period4 = ECAPxRegs.CAP1;// Fetch Time-Stamp captured at T1
Period1 = ECAPxRegs.CAP2;// Fetch Time-Stamp captured at T2
Period2 = ECAPxRegs.CAP3;// Fetch Time-Stamp captured at T3
Period3 = ECAPxRegs.CAP4;// Fetch Time-Stamp captured at T4
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example

In Figure 12-293 the ECAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information: Period1 = T_1 + T_2, Period2 = T_3 + T_4, …etc Duty Cycle1 (on-time %) = T_1 / Period1 × 100%, etc Duty Cycle1 (off-time %) = T_2 / Period1 × 100%, etc.

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, that is, during run-time, only the shadow registers must be used.

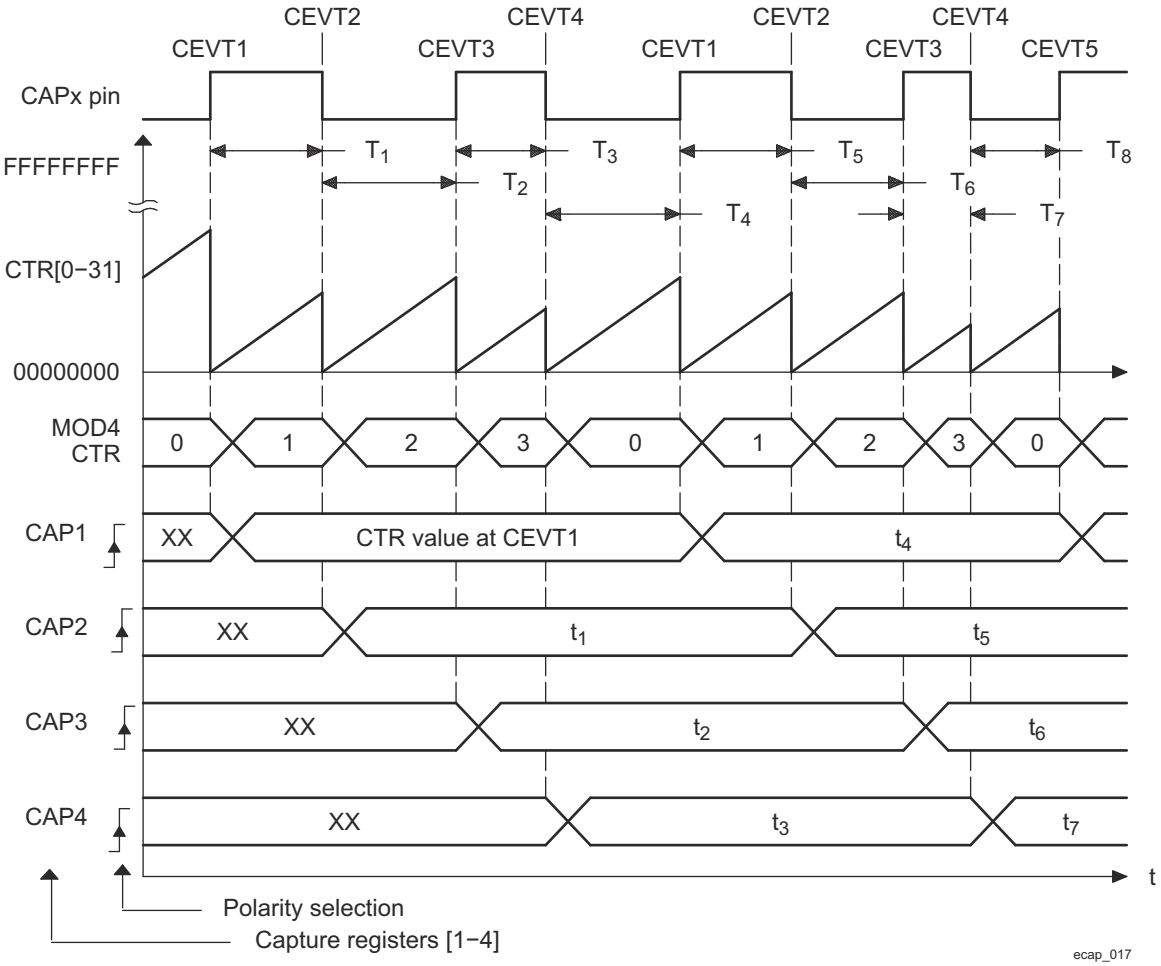


Figure 12-293. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect

Copyright © 2026 Texas Instruments Incorporated

Table 12-256. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-4. Code Snippet for CAP Mode Delta Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers
// Run Time ( e.g. CEVT1 triggered ISR call)
//=======================================
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 =  ECAPxRegs.CAP2;// Fetch Time-Stamp captured at T2
DutyOffTime1 = ECAPxRegs.CAP3;// Fetch Time-Stamp captured at T3
DutyOnTime2 =  ECAPxRegs.CAP4;// Fetch Time-Stamp captured at T4
DutyOffTime2 = ECAPxRegs.CAP1;// Fetch Time-Stamp captured at T1
Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.5 Application of the APWM Mode

12.4.2.5.5.1 Simple PWM Generation (Independent Channel/s) Example

In this example, the ECAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWM*n*. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.

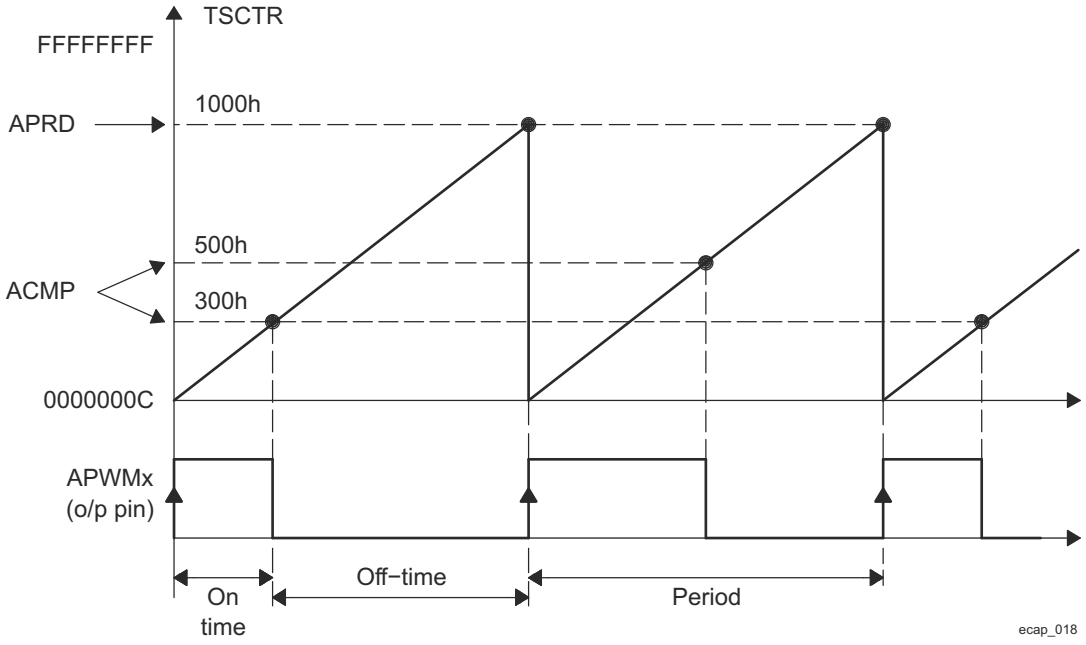


Figure 12-294. PWM Waveform Details of APWM Mode Operation

Copyright © 2026 Texas Instruments Incorporated

Table 12-257. ECAP Initialization for APWM Mode

Register	Bit	Value
CAP1	CAP1	0x1000
CTRPHS	CTRPHS	0x0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-5. Code Snippet for APWM Mode

```
// Code snippet for APWM mode Example 1
// Run Time (Instant 1, e.g. ISR call)
//=====================
   ECAPxRegs.CAP2 = 0x300;      // Set Duty cycle i.e. compare value
// Run Time (Instant 2, e.g. another ISR call)
//=====================
   ECAPxRegs.CAP2 = 0x500;      // Set Duty cycle i.e. compare value
```

12.4.2.5.5.2 Multichannel PWM Generation with Synchronization Example

Figure 12-295 takes advantage of the synchronization feature between the ECAP modules. Here 4 independent PWM channels are required with different frequencies, but at integer multiples of each other to avoid "beat" frequencies. Hence one ECAP module is configured as the Controller and the remaining 3 are Targets all receiving their synch pulse (CTR = PRD) from the controller. Note the Controller is chosen to have the lower frequency (F1 = 1/20,000) requirement. Here Target2 Freq = 2 × F1, Target3 Freq = 4 × F1 and Target4 Freq = 5 × F1. Note here values are in decimal notation. Also, only the APWM1 output waveform is shown.

Copyright © 2026 Texas Instruments Incorporated

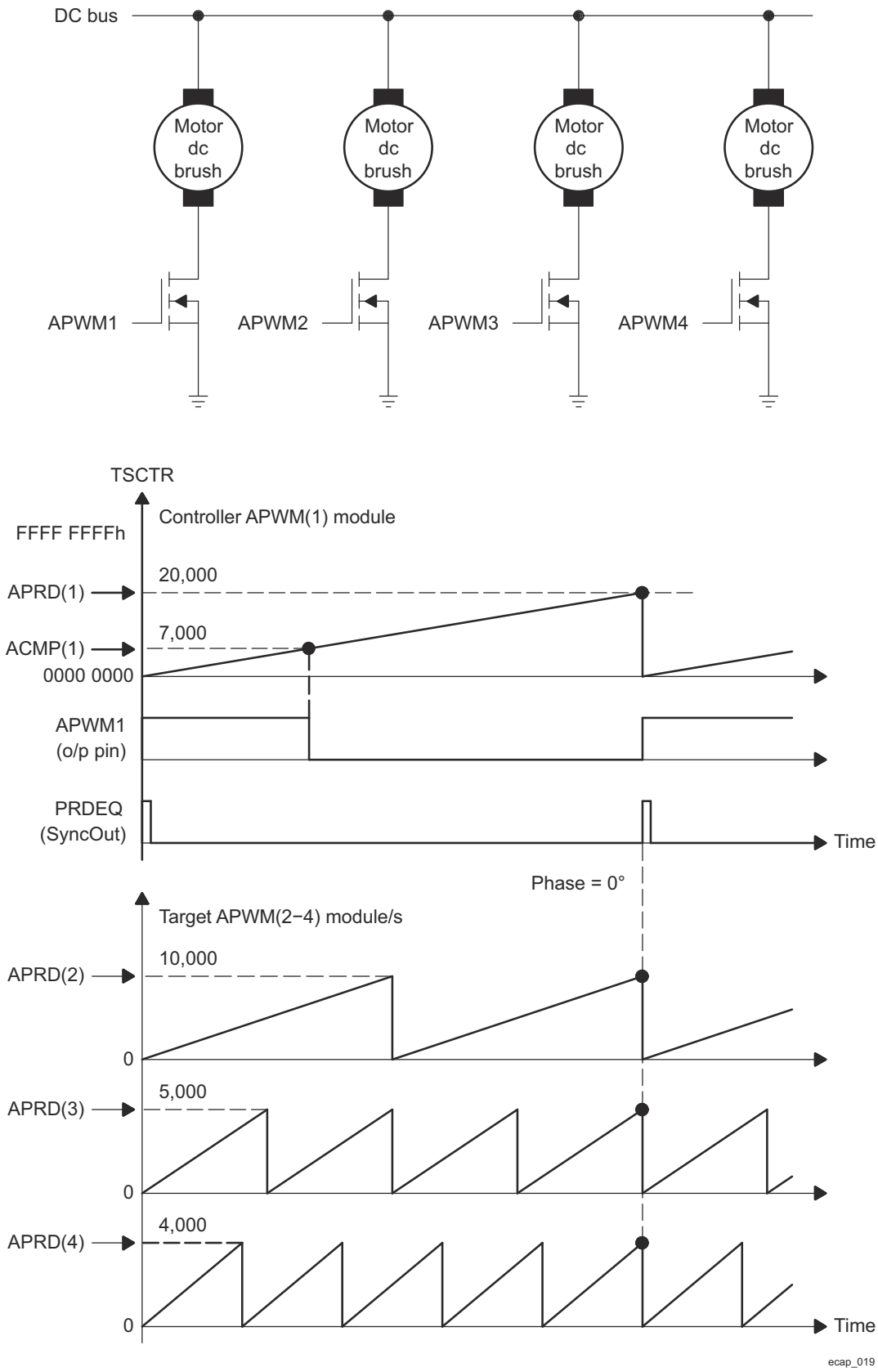


Figure 12-295. Multichannel PWM Example Using 4 ECAP Modules

Copyright © 2026 Texas Instruments Incorporated

Table 12-258. ECAP1 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	20000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

Table 12-259. ECAP2 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	10000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 12-260. ECAP3 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	5000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 12-261. ECAP4 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	4000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-6. Code Snippet for Multichannel PWM Generation with Synchronization

```
// Code snippet for APWM mode Example 2
// Run Time (Note: Example execution of one run-time instant)
//=====================================================
   ECAP1Regs.CAP2 = 7000;    // Set Duty cycle i.e., compare value = 7000
   ECAP2Regs.CAP2 = 2000;    // Set Duty cycle i.e., compare value = 2000
   ECAP3Regs.CAP2 = 550;     // Set Duty cycle i.e., compare value = 550
   ECAP4Regs.CAP2 = 6500;    // Set Duty cycle i.e., compare value = 6500
```


Copyright © 2026 Texas Instruments Incorporated

12.4.2.5.5.3 Multichannel PWM Generation with Phase Control Example

In Figure 12-296, the Phase control feature of the APWM mode is used to control a 3 phase Interleaved DC/DC converter topology. This topology requires each phase to be off-set by 120° from each other. Hence if "Leg" 1 (controlled by APWM1) is the reference Leg (or phase), that is, 0°, then Leg 2 need 120° off-set and Leg 3 needs 240° off-set. The waveforms in Figure 12-296 show the timing relationship between each of the phases (Legs). Note ECAP1 module is the Controller and issues a SyncOut pulse to the targets (modules 2, 3) whenever TSCTR = Period value.

Copyright © 2026 Texas Instruments Incorporated

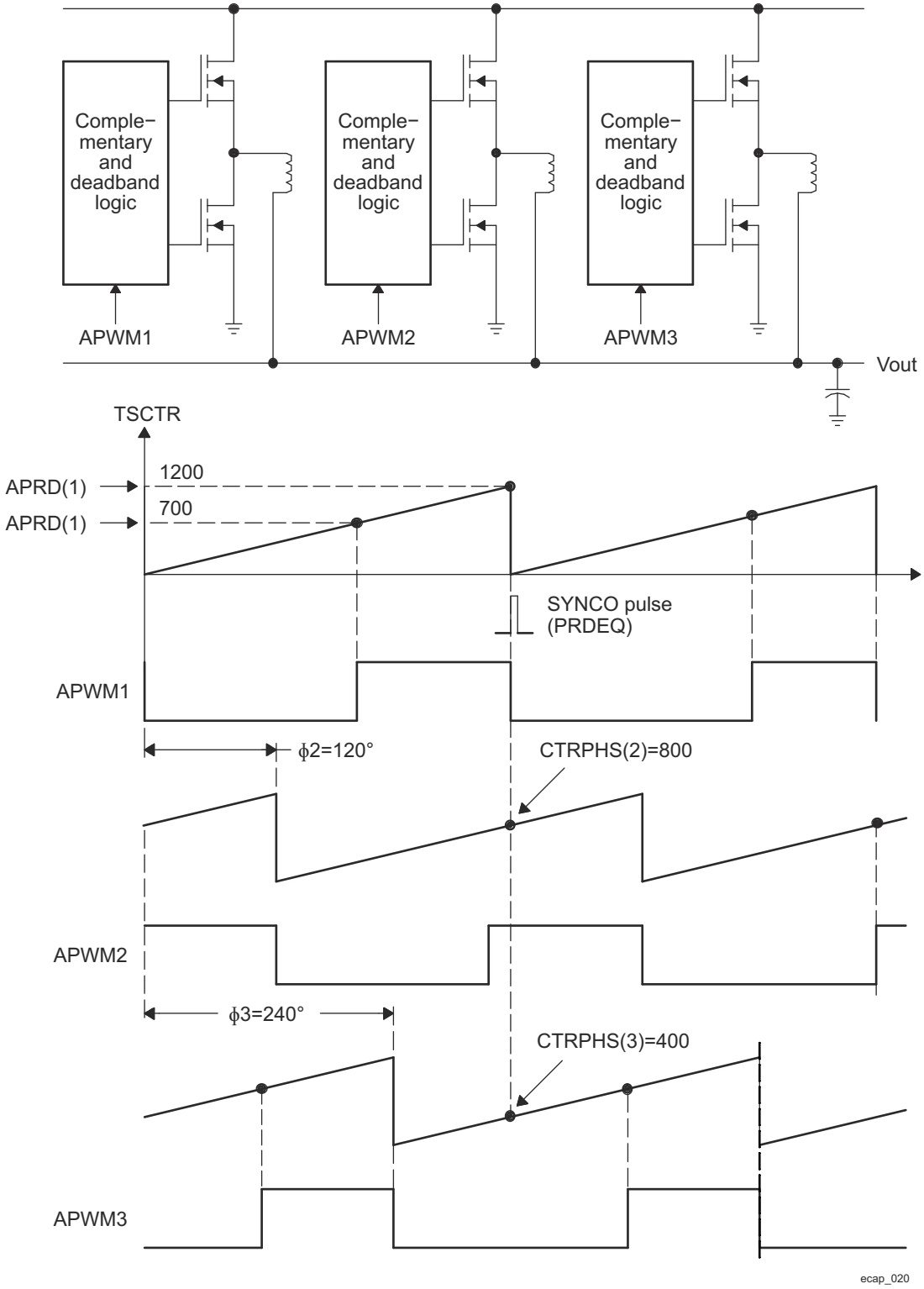


Figure 12-296. Multiphase (channel) Interleaved PWM Example Using 3 ECAP Modules

Copyright © 2026 Texas Instruments Incorporated

Table 12-262. ECAP1 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

Table 12-263. ECAP2 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	800
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 12-264. ECAP3 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	400
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Example 12-7. Code Snippet for Multichannel PWM Generation with Phase Control

```
// Code snippet for APWM mode Example 3
// Run Time (Note: Example execution of one run-time instant)
//=============================================================
// All phases are set to the same duty cycle
  ECAP1Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
  ECAP2Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
  ECAP3Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
```

12.4.3 Enhanced Pulse Width Modulation (EPWM) Module

This chapter describes the Enhanced Pulse Width Modulation (EPWM) module in the device.

12.4.3.1 EPWM Overview

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The EPWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross-coupling or sharing of resources has been avoided; instead, the EPWM is built up from smaller single-channel modules with separate resources and that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

Copyright © 2026 Texas Instruments Incorporated

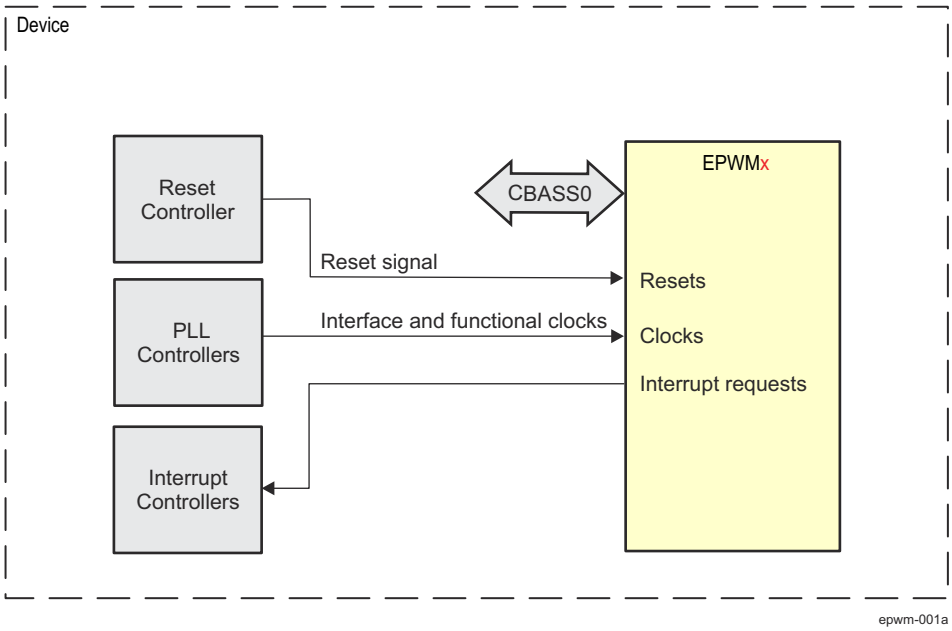
In the further description, the letter x within a signal or module name is used to indicate a generic EPWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the EPWMx instance. Thus, EPWM1A and EPWM1B belong to EPWM1, EPWM2A and EPWM2B belong to EPWM2, and so forth.

The EPWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB.

As also described in *Daisy-Chain Connectivity between EPWM Modules*, the EPWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the Enhanced Capture (ECAP) peripheral module. The EPWM modules can also operate stand-alone.

Copyright © 2026 Texas Instruments Incorporated

Figure 12-297 shows the EPWM module overview.



A. x = 0 to number of instances

Figure 12-297. EPWM Overview

12.4.3.1.1 EPWM Features

Each EPWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software
- Programmable phase-control support for lag or lead operation relative to other EPWM modules
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis
- Dead-band generation with independent rising and falling edge delay control
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs
- Allows events to trigger both CPU interrupts and ADC start of conversions
- Programmable event prescaling minimizes CPU overhead on interrupts
- PWM chopping by a high-frequency carrier signal, useful for pulse transformer gate drives

12.4.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

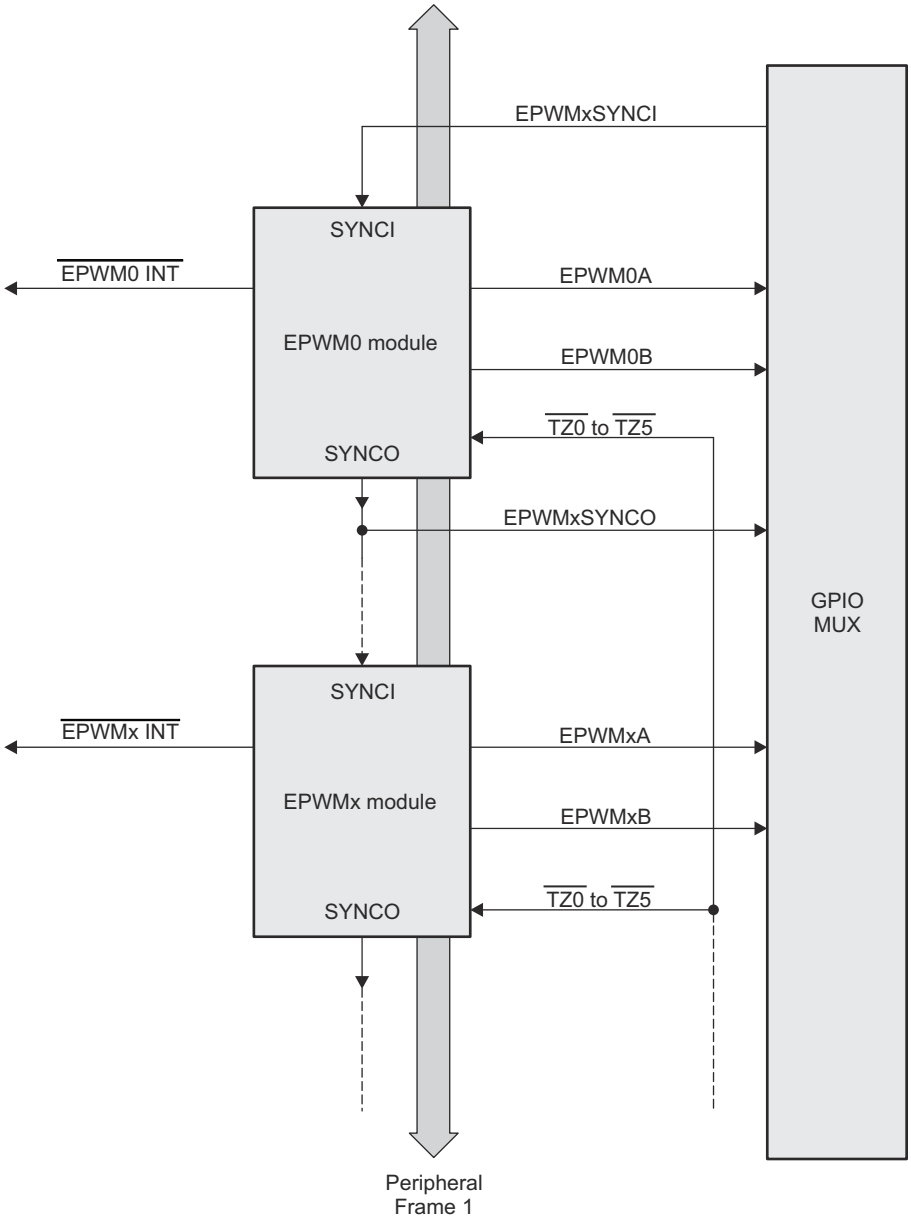
Some features may not be available. See *Module Integration* for more information.

12.4.3.1.3 Multiple EPWM Module Details

Copyright © 2026 Texas Instruments Incorporated

Each EPWM module is connected to the input/output signals shown in Figure 12-298.

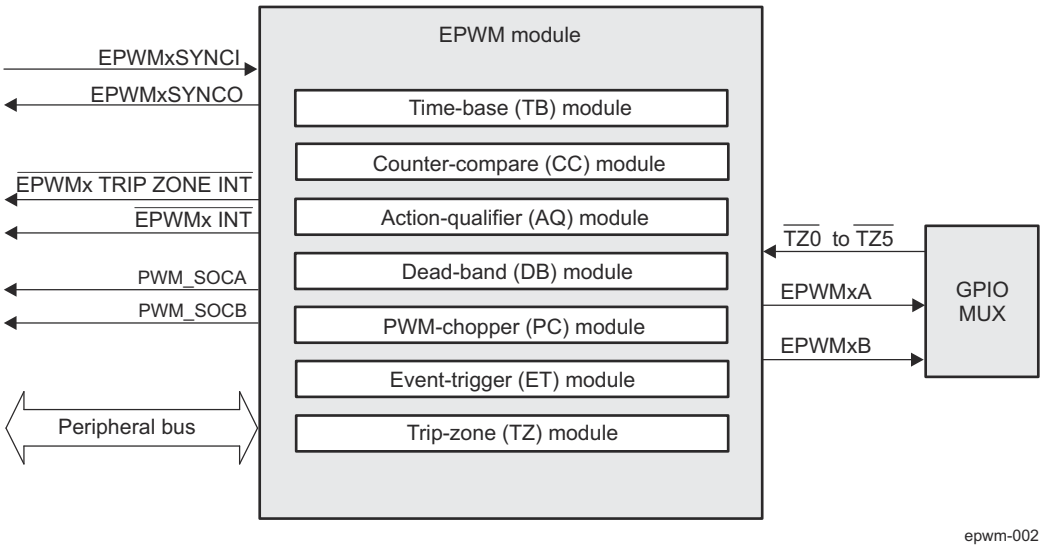
The order in which the EPWM modules are connected may differ from what is shown in Figure 12-298. See *Daisy-Chain Connectivity between EPWM Modules* for the actual synchronization scheme implemented in the device. Each EPWM module consists of seven submodules and is connected within a system via the signals shown in Figure 12-299.



A. x = 0 to number of instances

Figure 12-298. Multiple EPWM Modules

Copyright © 2026 Texas Instruments Incorporated



A. x = 0 to number of instances

Figure 12-299. Submodules and Signal Connections for an EPWM Module

Figure 12-300 shows more internal details of a single EPWM module.

The main signals used by the EPWM module are:

- **PWM output signals (EPWMxA and EPWMxB)**
  The PWM output signals are available external to the device through the GPIO peripheral.
- **Trip-zone signals ($\overline{\text{TZ0}}$ to $\overline{\text{TZ5}}$)**
  These input signals alert the EPWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral.
- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals**
  The synchronization signals daisy chain the EPWM modules together. Each module can be configured to either use or ignore its synchronization input. For more information see *Daisy-Chain Connectivity between EPWM Modules*.
- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB)**
  Each EPWM module has two ADC start of conversion signals (one for each sequencer). Any EPWM module can trigger a start of conversion for either sequencer. Which event triggers the start of conversion is configured in the Event-Trigger submodule of the EPWM module.
- **Peripheral Bus**
  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the EPWM registers.

Copyright © 2026 Texas Instruments Incorporated

Figure 12-300 also shows the key internal submodule interconnect signals. Each submodule is described in *EPWM Functional Description*.

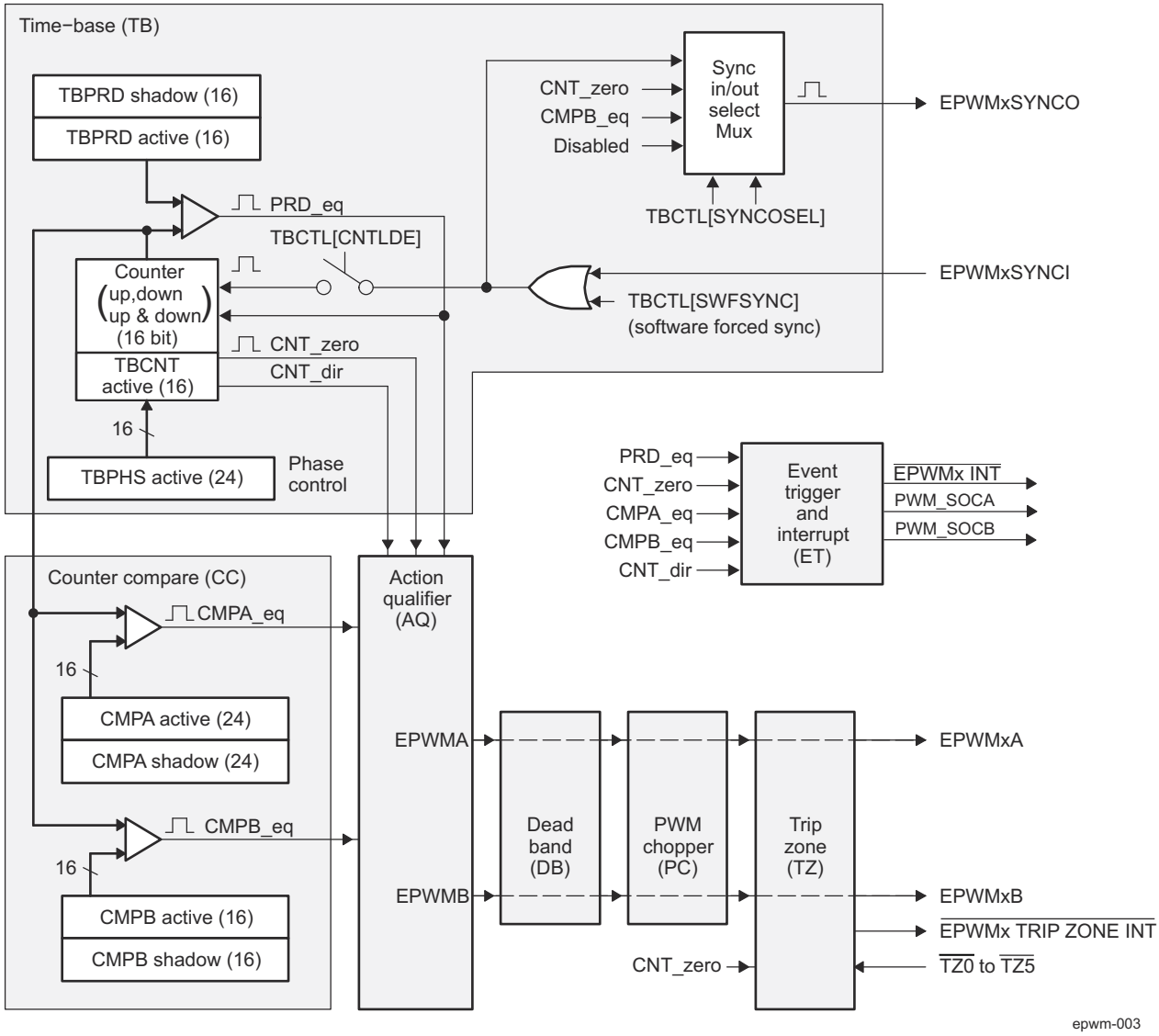


Figure 12-300. EPWM Submodule Interconnect Signals

Copyright © 2026 Texas Instruments Incorporated

12.4.3.2 EPWM Environment

The EPWMx (where x = 0 to number of instances) module is hereinafter referred to as EPWM module.

This section describes the EPWM external connections (environment).

Figure 12-301 shows the EPWM I/O interface signals.

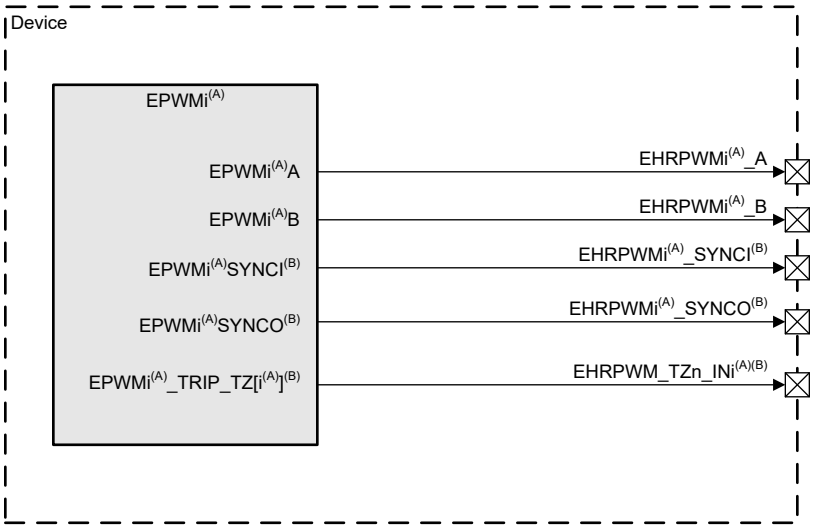


Figure 12-301. EPWM I/O Interface Signals

A. i represents a valid instance of EPWM in a domain. See the device datasheet for specifics.
B. Pin only available on specific instances. See the device datasheet for specifics.

Table 12-265 describes the EPWM I/O signals.

Table 12-265. EPWM I/O Signals

Module Pin	Device Level Signal	I/O(1)	Description	Module Pin Reset Value(2)
EPWMi(3)				
EPWMi(3)A	EHRPWMi(3)_A	O	EPWMi(3) output A	0x0
EPWMi(3)B	EHRPWMi(3)_B	O	EPWMi(3) output B	0x0
EPWMi(3)SYNCI(4)	EHRPWMi(3)_SYNCI	I	EPWMi(3) Sync input	HiZ
EPWMi(3)SYNCO(4)	EHRPWMi(3)_SYNCO	O	EPWMi(3) Sync output	0x0
$\overline{\text{EPWMi(3)_TRIP_TZ[0](4)}}$	EHRPWM_TZn_INi(3)	I	EPWMi(3) TripZone input	HiZ
EPWM Start of Conversion				
PWM_SOCA	EHRPWM_SOCA	O	EPWM start of conversion output A	HiZ
PWM_SOCB	EHRPWM_SOCB	O	EPWM start of conversion output B	HiZ

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) i represents a valid instance of EPWM in a domain. See the device datasheet for specifics.
(4) Pin only available on specific instances. See the device datasheet for specifics.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

Copyright © 2026 Texas Instruments Incorporated

12.4.3.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4 EPWM Functional Description

Seven submodules are included in each EPWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

12.4.3.4.1 EPWM Submodule Features

Table 12-266 lists the seven key submodules together with a list of their main configuration parameters.

Table 12-266. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option
Time-base (TB)	• Scale the time-base clock (TBCLK) relative to the system clock (FICLK). • Configure the PWM time-base counter (TBCNT) frequency or period. • Time-base counter mode selection: – count-up mode: used for asymmetric PWM – count-down mode: used for asymmetric PWM – count-up-and-down mode: used for symmetric PWM • Configure the time-base phase relative to another EPWM module. • Synchronize the time-base counter between modules through hardware or software. • Configure the direction (up or down) of the time-base counter after a synchronization event. • Configure how the time-base counter will behave when the device is halted by an emulator. • Specify the source for the synchronization output of the EPWM module: – Synchronization input signal – Time-base counter equal to zero – Time-base counter equal to counter-compare B (CMPB) – No output synchronization signal generated
Counter-compare (CC)	• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB • Specify the time at which switching events occur on the EPWMxA or EPWMxB output
Action-qualifier (AQ)	• Specify the type of action taken when a time-base or counter-compare submodule event occurs: – No action taken – Output EPWMxA and/or EPWMxB switched high – Output EPWMxA and/or EPWMxB switched low – Output EPWMxA and/or EPWMxB toggled • Force the PWM output state through software control • Configure and control the PWM dead-band through software
Dead-band (DB)	• Control of traditional complementary dead-band relationship between upper and lower switches • Specify the output rising-edge-delay value • Specify the output falling-edge delay value • Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification
PWM-chopper (PC)	• Create a chopping (carrier) frequency • Pulse width of the first pulse in the chopped pulse train • Duty cycle of the second and subsequent pulses • Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.

Copyright © 2026 Texas Instruments Incorporated

Table 12-266. Submodule Configuration Parameters (continued)

Submodule	Configuration Parameter or Option
Trip-zone (TZ)	• Configure the EPWM module to react to one, all, or none of the trip-zone pins • Specify the tripping action taken when a fault occurs: – Force EPWMxA and/or EPWMxB high – Force EPWMxA and/or EPWMxB low – Force EPWMxA and/or EPWMxB to a high-impedance state – Configure EPWMxA and/or EPWMxB to ignore any trip condition • Configure how often the EPWM will react to the trip-zone pin: – One-shot – Cycle-by-cycle • Enable the trip-zone to initiate an interrupt • Bypass the trip-zone module entirely
Event-trigger (ET)	• Enable the EPWM events that will trigger an interrupt. • Specify the rate at which events cause triggers (every occurrence or every second or third occurrence) • Poll, set, or clear event flags

Some examples on various EPWM module configurations are shown below. These examples use the constant definitions shown in Example 12-8.

Constant Definitions Used in the EPWM Code Examples

```
// TBCTL (Time-Base Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TBCNT MODE bits
#define         TB_COUNT_UP         0x0
#define         TB_COUNT_DOWN       0x1
#define         TB_COUNT_UPDOWN     0x2
#define         TB_FREEZE           0x3
// PHSEN bit
#define         TB_DISABLE          0x0
#define         TB_ENABLE           0x1
// PRDLD bit
#define         TB_SHADOW           0x0
#define         TB_IMMEDIATE        0x1
// SYNCOSEL bits
#define         TB_SYNC_IN          0x0
#define         TB_CTR_ZERO         0x1
#define         TB_CTR_CMPB         0x2
#define         TB_SYNC_DISABLE     0x3
// HSPCLKDIV and CLKDIV bits
#define         TB_DIV1             0x0
#define         TB_DIV2             0x1
#define         TB_DIV4             0x2
// PHSDIR bit
#define         TB_DOWN             0x0
#define         TB_UP               0x1
```

```
// CMPCTL (Compare Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// LOADAMODE and LOADBMODE bits
#define         CC_CTR_ZERO         0x0
#define         CC_CTR_PRD          0x1
```


Copyright © 2026 Texas Instruments Incorporated

```
#define         CC_CTR_ZERO_PRD    0x2
#define         CC_LD_DISABLE      0x3
// SHDWAMODE and SHDWBMODE bits
#define         CC_SHADOW          0x0
#define         CC_IMMEDIATE       0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define         AQ_NO_ACTION       0x0
#define         AQ_CLEAR           0x1
#define         AQ_SET             0x2
#define         AQ_TOGGLE          0x3
// DBCTL (Dead-Band Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// MODE bits
#define         DB_DISABLE         0x0
#define         DBA_ENABLE         0x1
#define         DBB_ENABLE         0x2
#define DB_FULL_ENABLE 0x3
// POLSEL bits
#define         DB_ACTV_HI         0x0
#define         DB_ACTV_LOC        0x1
#define         DB_ACTV_HIC        0x2
#define         DB_ACTV_LO         0x3
// PCCTL (chopper control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CHPEN bit
#define         CHP_ENABLE         0x0
#define CHP_DISABLE 0x1
// CHPFREQ bits
#define         CHP_DIV1           0x0
#define         CHP_DIV2           0x1
#define         CHP_DIV3           0x2
#define         CHP_DIV4           0x3
#define         CHP_DIV5           0x4
#define         CHP_DIV6           0x5
#define         CHP_DIV7           0x6
#define         CHP_DIV8           0x7
// CHPDUTY bits
#define         CHP1_8TH           0x0
#define         CHP2_8TH           0x1
#define         CHP3_8TH           0x2
#define         CHP4_8TH           0x3
#define         CHP5_8TH           0x4
#define         CHP6_8TH           0x5
#define         CHP7_8TH           0x6
// TZSEL (Trip-zone Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CBCn and OSHTn bits
#define         TZ_ENABLE          0x0
#define         TZ_DISABLE         0x1
// TZCTL (Trip-zone Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TZA and TZB bits
#define         TZ_HIZ             0x0
#define         TZ_FORCE_HI        0x1
#define         TZ_FORCE_LO        0x2
#define         TZ_DISABLE         0x3
// ETSEL (Event-trigger Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTSEL, SOCASEL, SOCBSEL bits
#define         ET_CTR_ZERO        0x1
#define         ET_CTR_PRD         0x2
#define         ET_CTRU_CMPA       0x4
#define         ET_CTRD_CMPA       0x5
#define         ET_CTRU_CMPB       0x6
#define         ET_CTRD_CMPB       0x7
// ETPS (Event-trigger Prescale)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTPRD, SOCAPRD, SOCBPRD bits
#define         ET_DISABLE         0x0
#define         ET_1ST             0x1
```


Copyright © 2026 Texas Instruments Incorporated

```
#define         ET_2ND          0x2
#define         ET_3RD          0x3
```

12.4.3.4.2 EPWM Time-Base (TB) Submodule

This section describes the Time-Base (TB) submodule in the PWM module.

12.4.3.4.2.1 Overview

Each EPWM module has its own time-base submodule that determines all of the timing events. Built-in synchronization logic allows the time-base of multiple EPWM modules (EPWMx) to work together as a single system. Figure 12-302 illustrates the time-base module's place within the EPWM.

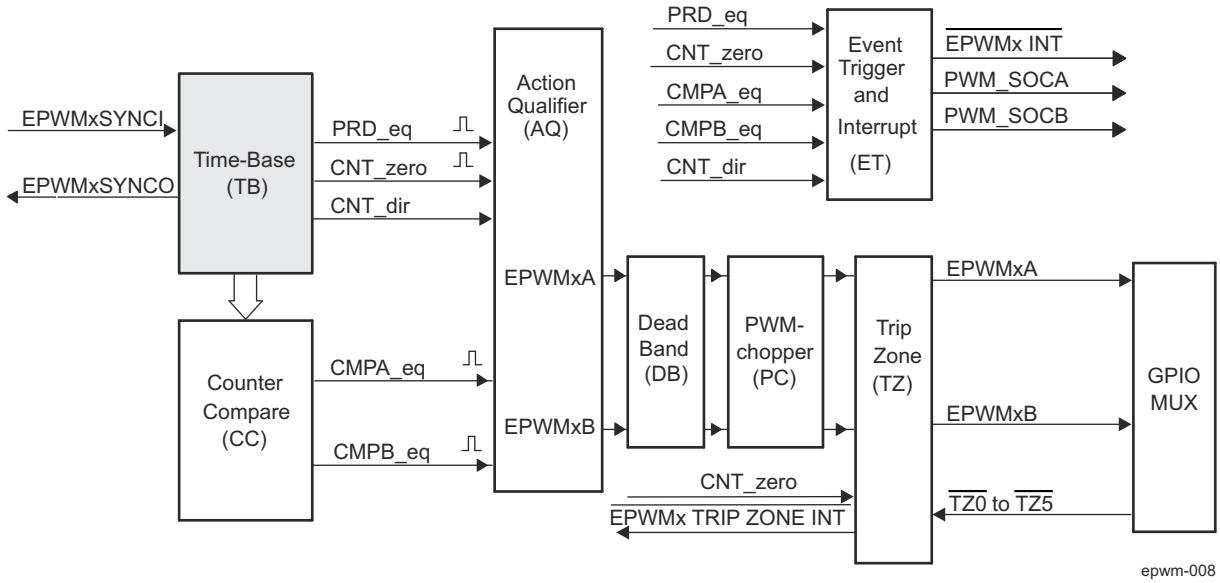


Figure 12-302. EPWM Time-Base Submodule

12.4.3.4.2.2 Controlling and Monitoring the EPWM Time-Base Submodule

Table 12-267 lists the registers used to control and monitor the time-base submodule.

Table 12-267. EPWM Time-Base Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_TBCTL	Time-Base Control Register	0h	No
EPWM_TBSTS	Time-Base Status Register	2h	No
EPWM_TBPHS	Time-Base Phase Register	6h	No
EPWM_TBCNT	Time-Base Counter Register	8h	No
EPWM_TBPRD	Time-Base Period Register	Ah	Yes

Figure 12-303 shows the critical signals and registers of the time-base submodule. Table 12-268 provides descriptions of the key signals associated with the time-base submodule.

Copyright © 2026 Texas Instruments Incorporated

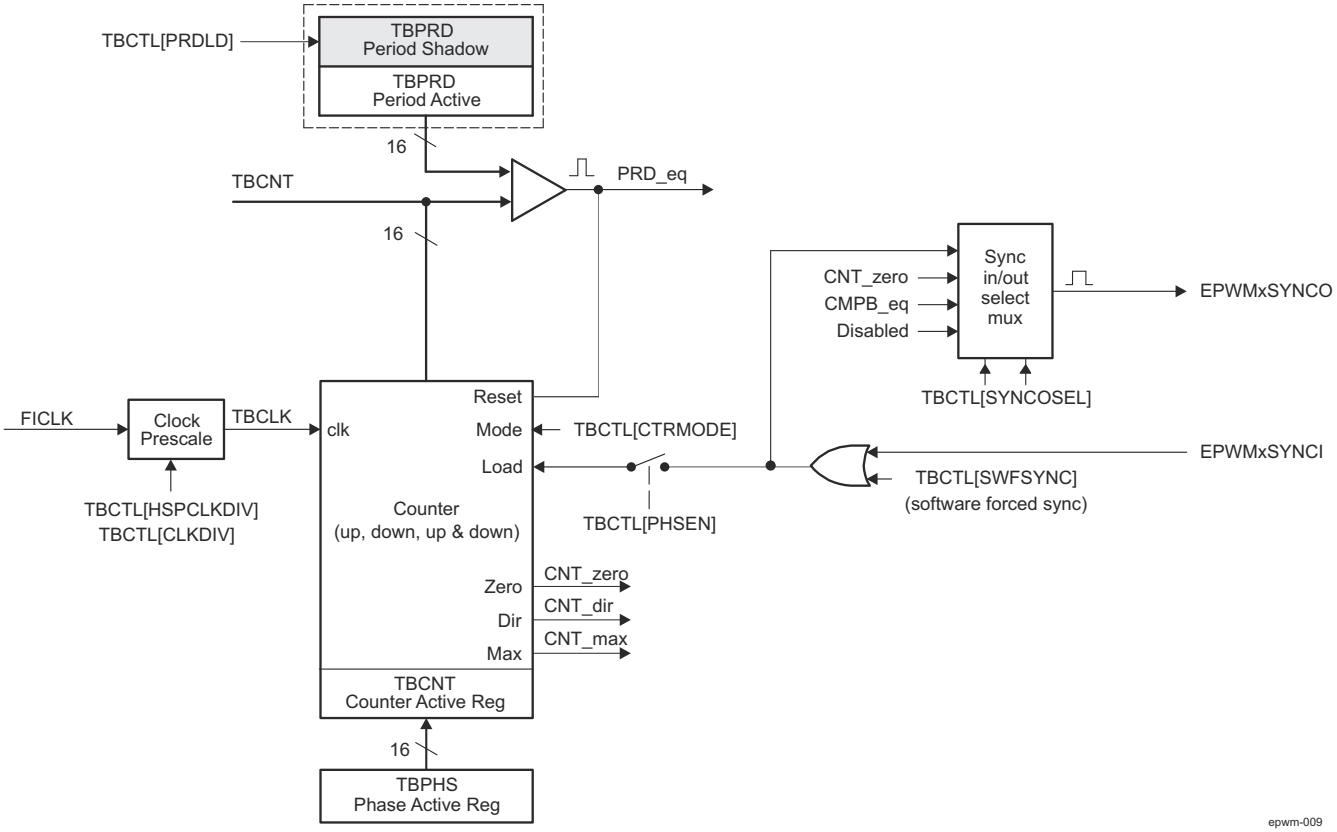


Figure 12-303. EPWM Time-Base Submodule Signals and Registers

Table 12-268. EPWM Time-Base Submodule Key Signals

Signal	Description
EPWMxSYNCI	Time-base synchronization input. Input pulse used to synchronize the time-base counter with the counter of another EPWM module earlier in the synchronization chain. An EPWM peripheral can be configured to use or ignore this signal. For the first EPWM module (EPWM0) this signal comes from a device pin. For subsequent EPWM modules this signal is passed from another EPWM peripheral. For example, EPWM2SYNCI is generated by the EPWM1 peripheral and the EPWM3SYNCI is generated (optional through internal multiplexer) by EPWM2 or from a device pin. See Section 12.4.3.4.2.3.2 for information on the synchronization order of a particular device.
EPWMxSYNCO	Time-base synchronization output. This output pulse is used to synchronize the counter of an EPWM module later in the synchronization chain. The EPWM module generates this signal from one of three event sources: 1. EPWMxSYNCI (Synchronization input pulse) 2. TBCNT = 0: The time-base counter (EPWM_TBCNT register) equal to zero (TBCNT = 0000h). 3. TBCNT = CMPB: The time-base counter (EPWM_TBCNT register) equal to the counter-compare B register — EPWM_CMPB (that is bitfield TBCNT = bitfield CMPB).
PRD_eq	Time-base counter equal to the specified period. This signal is generated whenever the counter value is equal to the active period register value. That is when TBCNT = TBPRD.
CNT_zero	Time-base counter equal to zero. This signal is generated whenever the counter value is zero. That is when TBCNT equals 0000h.
CMPB_eq	Time-base counter equal to active counter-compare B register (TBCNT = CMPB). This event is generated by the counter-compare submodule and used by the synchronization out logic.

Copyright © 2026 Texas Instruments Incorporated

Table 12-268. EPWM Time-Base Submodule Key Signals (continued)

Signal	Description
CNT_dir	Time-base counter direction.
	Indicates the current direction of the EPWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CNT_max	Time-base counter equal max value (TBCNT = FFFFh).
	Generated event when the EPWM_TBCNT register value reaches its maximum value. This signal is only used only as a status bit.
TBCLK	Time-base clock.
	This is a prescaled version of the system clock — FICLK and is used by all submodules within the EPWMn. This clock determines the rate at which time-base counter increments or decrements.

12.4.3.4.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period register (EPWM_TBPRD) and the mode of the time-base counter. Figure 12-304 shows the period (T_{pwm}) and frequency (F_{pwm}) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (EPWM_TBPRD[15-0] TBPRD = 0x4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (FICLK).

The time-base counter has three modes of operation selected by the time-base control register (EPWM_TBCTL):

- **Up-Down-Count Mode:** In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.
- **Down-Count Mode:** In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset back to the period value and it repeats this pattern.

Copyright © 2026 Texas Instruments Incorporated

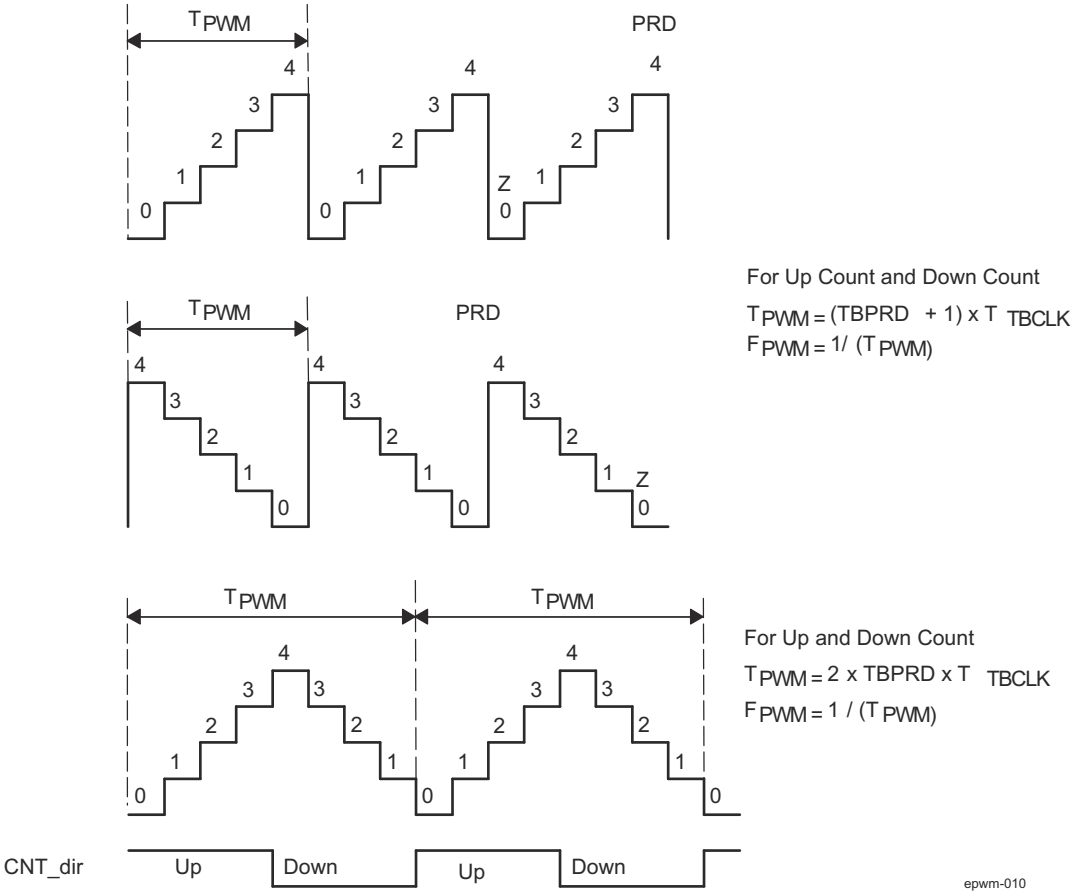


Figure 12-304. EPWM Time-Base Frequency and Period

12.4.3.4.2.3.1 EPWM Time-Base Period Shadow Register

The time-base period register (EPWM_TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the EPWM module:

- **Active Register:** The active register controls the hardware and is responsible for actions that the hardware causes or invokes.
- **Shadow Register:** The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the EPWM_TBCTL[3] PRDLD bit. This bit enables and disables the EPWM_TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:** The EPWM_TBPRD shadow register is enabled when EPWM_TBCTL[3] PRDLD = 0h. Reads from and writes to the EPWM_TBPRD register memory address go to the shadow register. The shadow register contents are transferred to the active register (EPWM_TBPRD (Active) ← EPWM_TBPRD (shadow)) when the time-base counter register (EPWM_TBCNT) equals zero (TBCNT = 0000h). By default the EPWM_TBPRD shadow register is enabled.
- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (EPWM_TBCTL[3] PRDLD = 1h), then a read from or a write to the TBPRD memory address goes directly to the active register.

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.2.3.2 EPWM Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the EPWM modules on a device. Each EPWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (EPWM0) comes from an external pin. For the device EPWM environment sync pin details refer to the Section 12.4.3.2. The possible synchronization connections for the remaining EPWM modules is shown in Figure 12-305.

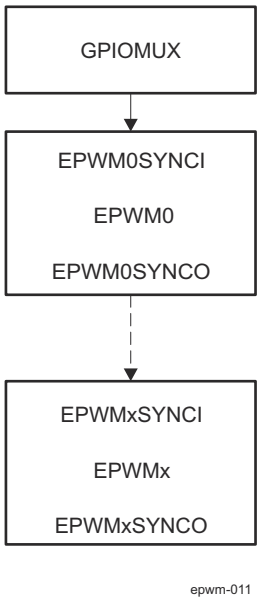


Figure 12-305. EPWM Time-Base Counter Synchronization Scheme 1

Each EPWM module can be configured to use or ignore the synchronization input. If the EPWM_TBCTL[2] PHSEN bit is set, then the time-base counter (TBCNT) of the EPWM module (EPWM_TBCNT register) will be automatically loaded with the phase register (EPWM_TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:** The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (EPWM_TBPHS → EPWM_TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.
- **Software Forced Synchronization Pulse:** Writing a 1h to the EPWM_TBCTL[6] SWFSYNC control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.

This feature enables the EPWM module to be automatically synchronized to the time base of another EPWM module. Lead or lag phase control can be added to the waveforms generated by different EPWM modules to synchronize them. In up-down-count mode, the EPWM_TBCTL[13] PHSDIR bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The TBPHS bit is ignored in count-up or count-down modes. See Figure 12-306 through Figure 12-309 for examples.

Clearing the EPWM_TBCTL[2] PHSEN bit configures the EPWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other EPWM modules. In this way, a controller time-base (for example: EPWM1) and downstream modules (EPWM2–EPWMx) can be set and they may select to run in synchronization with the controller.

12.4.3.4.2.4 Phase Locking the Time-Base Clocks of Multiple EPWM Modules

As already described in the *EPWM Modules Time-Base Clock Gating*, TB_CLKEN bit in the EPWMn_CTRL (where n = 0 to number of instances on the device) register of the device CTRL_MMR can be used to individually control or globally synchronize the time-base clocks of all enabled EPWM modules on a device.

Copyright © 2026 Texas Instruments Incorporated

When all TB_CLKEN bits are set to 0h, the time-base clocks of all EPWMx (where x = 0 to 8) modules are stopped (default). When all TB_CLKEN bits are simultaneously set in software to 1h, all EPWMx modules time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the EPWM_TBCTL register of each EPWM module must be set identically. The proper procedure for enabling the EPWM clocks is as follows:

1. Enable the EPWM module clocks.
2. Set TB_CLKEN = 0. This will stop the time-base clock within any enabled EPWMx module.
3. Configure the prescaler values and desired EPWM modes per each involved EPWMx.
4. Simultaneously set TB_CLKEN bits to 1h.

12.4.3.4.2.5 EPWM Time-Base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical.
- Frozen where the time-base counter is held constant at the current value.

To illustrate the operation of the first three modes, Figure 12-306 to Figure 12-309 show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

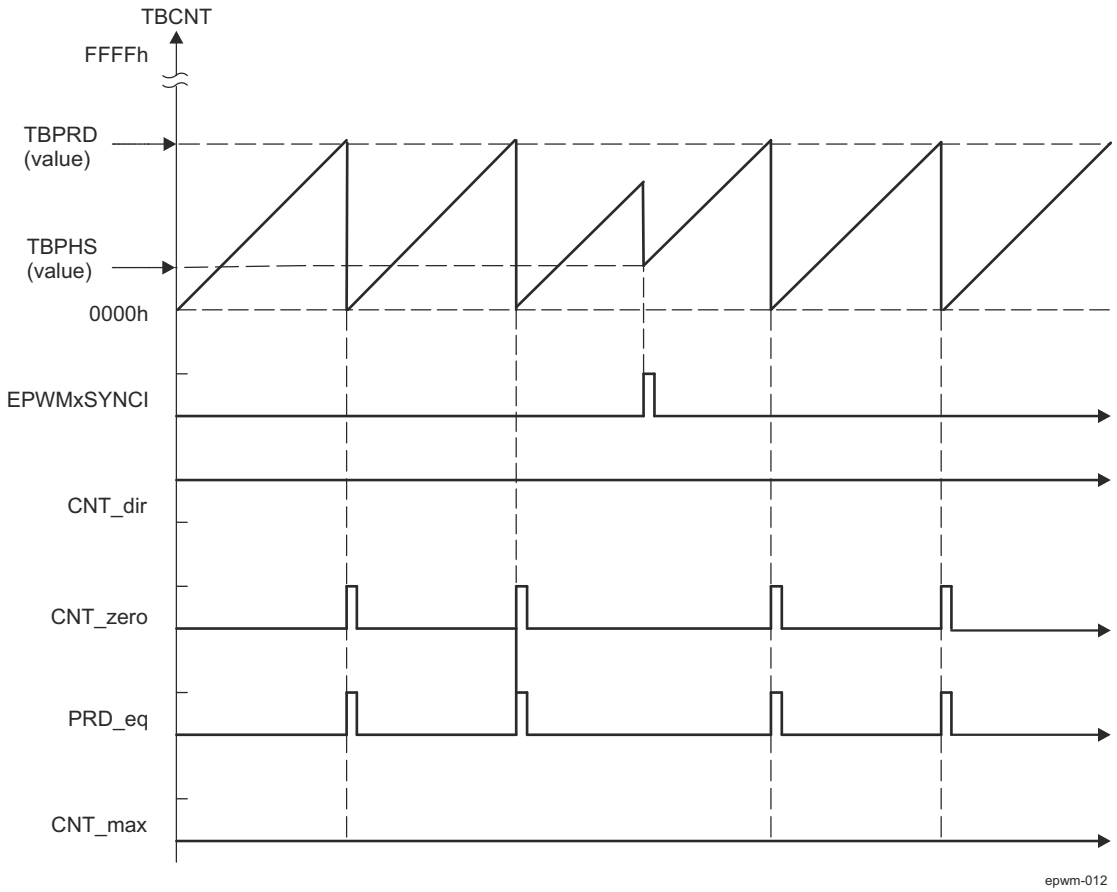


Figure 12-306. EPWM Time-Base Up-Count Mode Waveforms

Copyright © 2026 Texas Instruments Incorporated

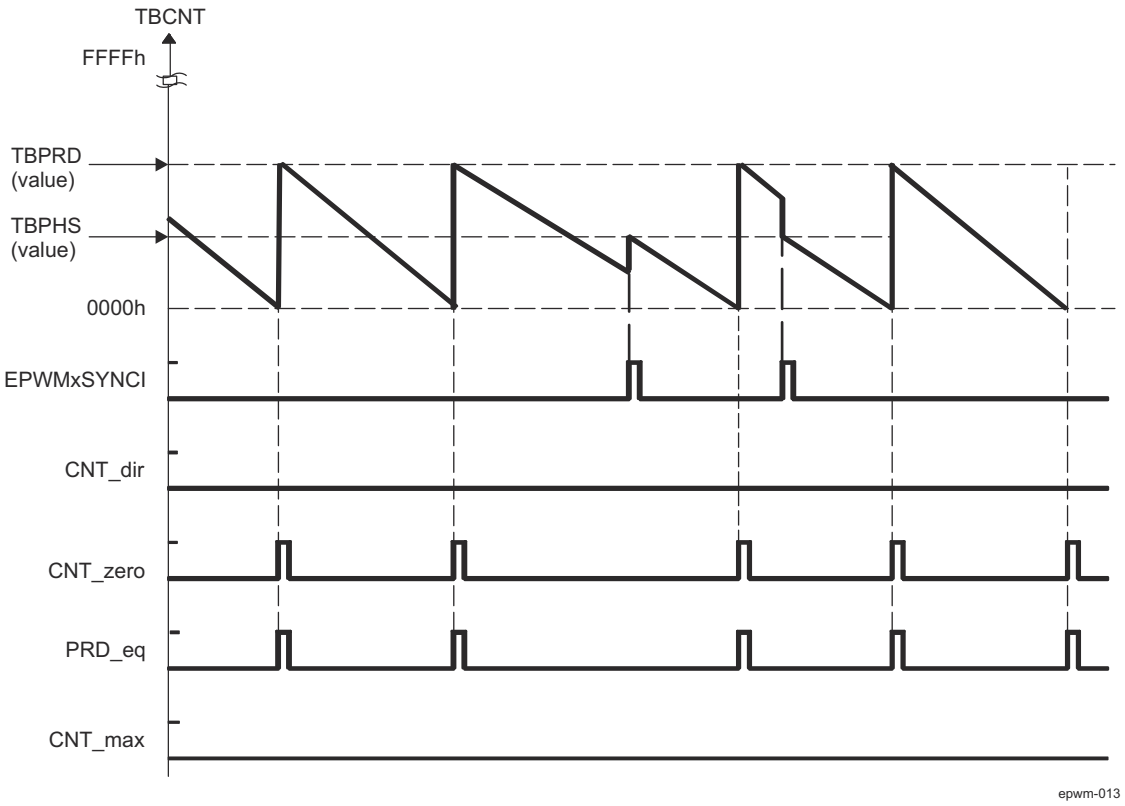


Figure 12-307. EPWM Time-Base Down-Count Mode Waveforms

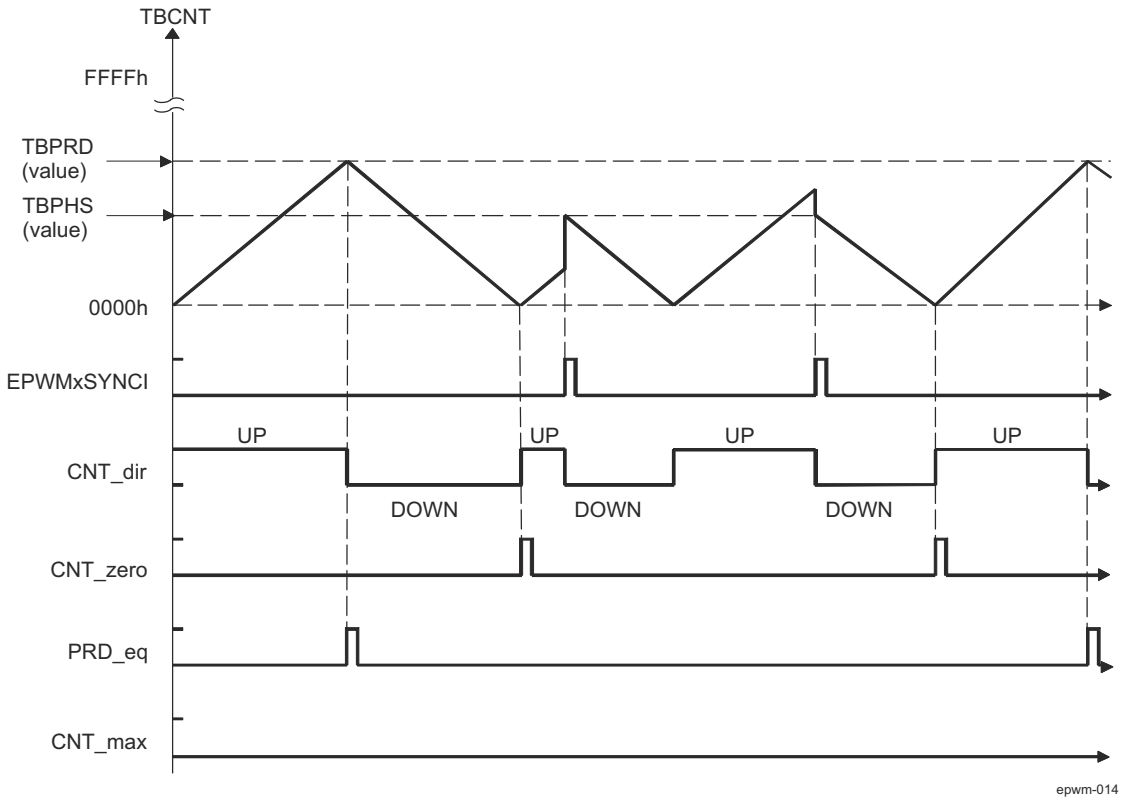


Figure 12-308. EPWM Time-Base Up-Down-Count Waveforms, EPWM_TBCTL[13] PHSDIR = 0 Count Down on Synchronization Event

Copyright © 2026 Texas Instruments Incorporated

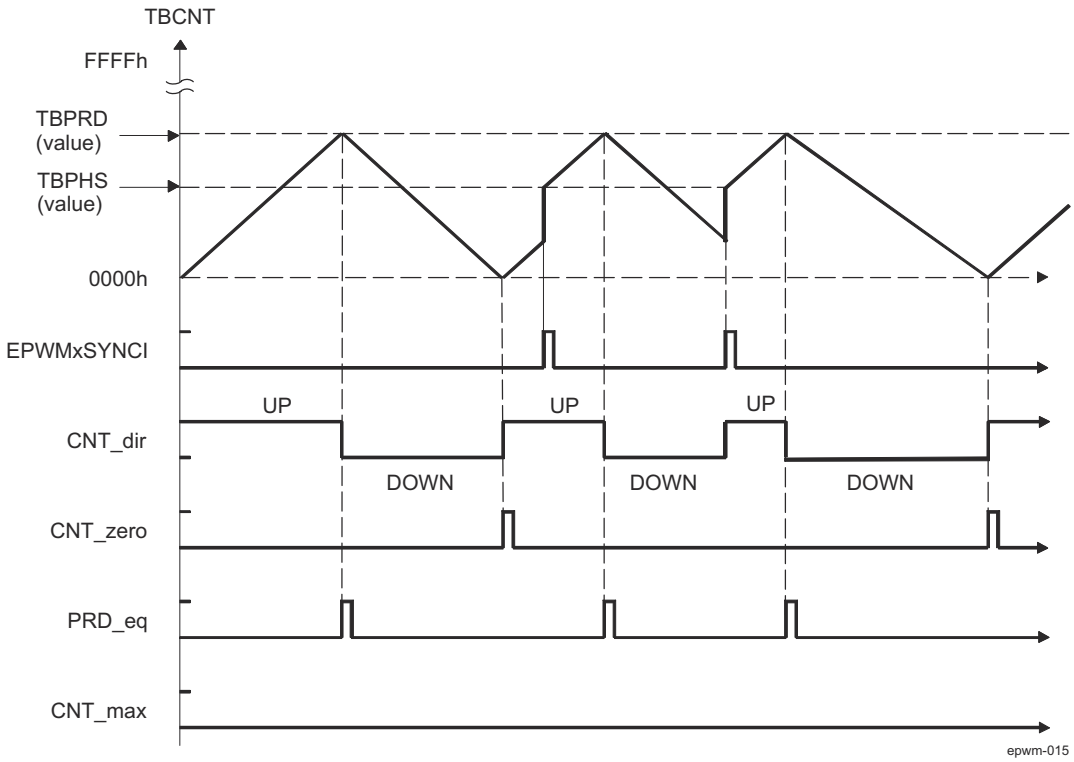


Figure 12-309. EPWM Time-Base Up-Down Count Waveforms, EPWM_TBCTL[13] PHSDIR = 1 Count Up on Synchronization Event

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.3 EPWM Counter-Compare (CC) Submodule

This section describes the Counter-Compare (CC) submodule in the PWM module.

12.4.3.4.3.1 Overview

Figure 12-310 illustrates the counter-compare submodule within the EPWM. Figure 12-311 shows the basic structure of the counter-compare submodule.

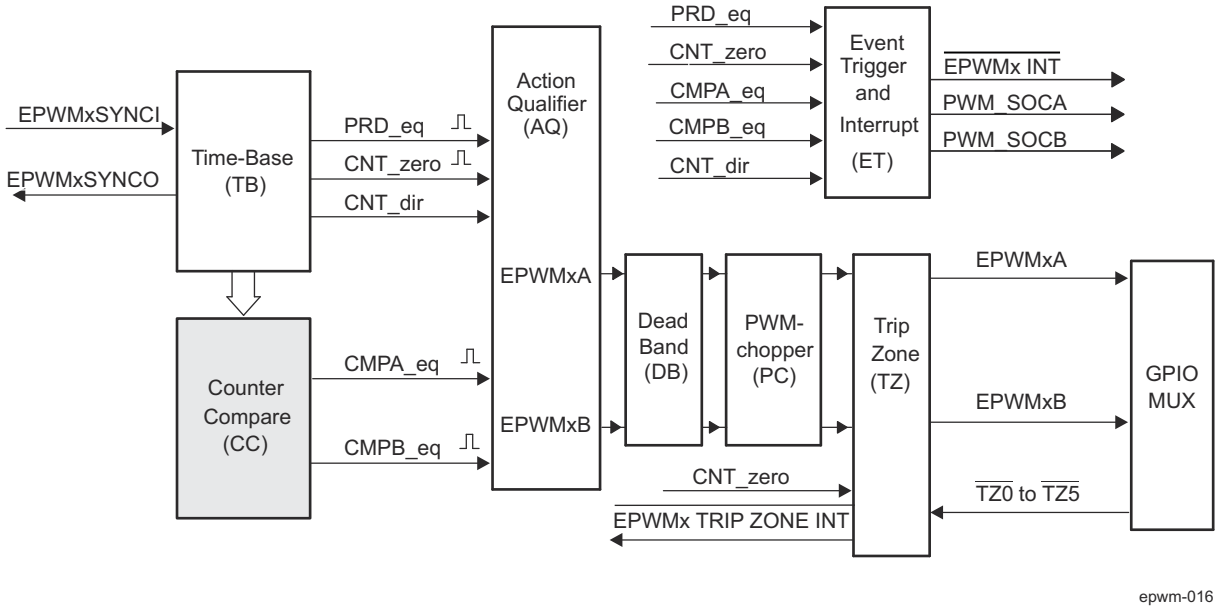


Figure 12-310. EPWM Counter-Compare Submodule

CC module features:

- Generates events based on programmable time stamps using the EPWM_CMPA and EPWM_CMPB registers
  - CMPA_eq (Time-base counter equals counter-compare A register (TBCNT = CMPA)).
  - CMPB_eq (Time-base counter equals counter-compare B register (TBCNT = CMPB)).
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle.

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (EPWM_CMPA) and counter-compare B (EPWM_CMPB) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

12.4.3.4.3.2 Controlling and Monitoring the EPWM Counter-Compare Submodule

Table 12-269 lists the registers used to control and monitor the counter-compare submodule. Table 12-270 lists the key signals associated with the counter-compare submodule.

Table 12-269. EPWM Counter-Compare Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_CMPCTL	Counter-Compare Control Register.	Eh	No
EPWM_CMPA	Counter-Compare A Register	12h	Yes
EPWM_CMPB	Counter-Compare B Register	14h	Yes

Copyright © 2026 Texas Instruments Incorporated

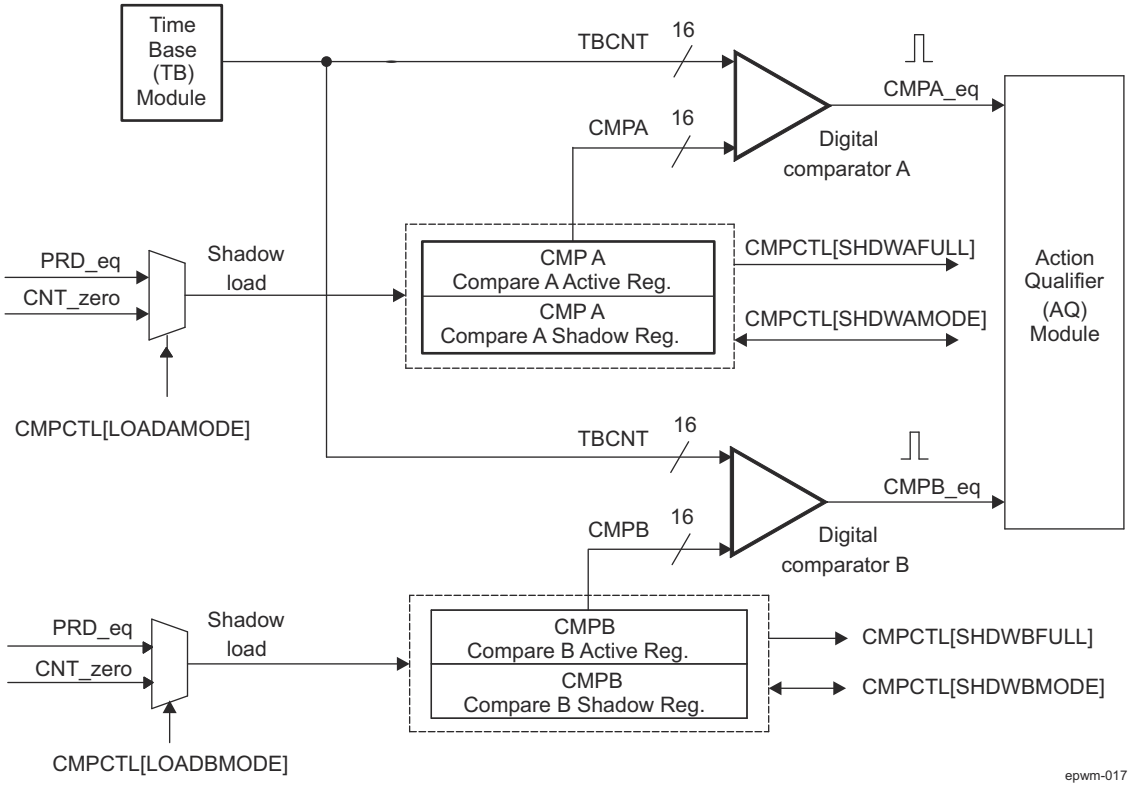


Figure 12-311. EPWM Counter-Compare Submodule Signals and Registers

Table 12-270. EPWM Counter-Compare Submodule Key Signals

Signal	Description of Event	Register Bitfields Compared
CMPA_eq	Time-base counter equal to the active counter-compare A value	TBCNT = CMPA
CMPB_eq	Time-base counter equal to the active counter-compare B value	TBCNT = CMPB
PRD_eq	Time-base counter equal to the active period. Used to load active counter-compare A and B registers from the shadow register	TBCNT = TBPRD
CNT_zero	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCNT = 0000h

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.3.3 Operational Highlights for the EPWM Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CMPA: Time-base counter equal to counter-compare A register (EPWM_TBCNT = EPWM_CMPA).
2. CMPB: Time-base counter equal to counter-compare B register (EPWM_TBCNT = EPWM_CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle, if the compare value is between 0000h and TBPRD; and occurs once per cycle, if the compare value is equal to 0000h or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to Section 12.4.3.4.4 for more details.

The counter-compare EPWM_CMPA and EPWM_CMPB registers each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occurs at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the EPWM_CMPCTL[4] SHDWAMODE and EPWM_CMPCTL[6] SHDWBMODE bits. These bits enable and disable the EPWM_CMPA shadow register and EPWM_CMPB shadow register respectively. The behavior of the two load modes is described below:

- **Shadow Load Mode:**
  The shadow mode for the EPWM_CMPA register is enabled by clearing the EPWM_CMPCTL[4] SHDWAMODE bit and the shadow register for EPWM_CMPB register is enabled by clearing the EPWM_CMPCTL[6] SHDWBMODE bit. Shadow mode is enabled by default for both EPWM_CMPA and EPWM_CMPB register.

  If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events:
  - PRD_eq: Time-base counter equal to the period (TBCNT = TBPRD).
  - CNT_zero: Time-base counter equal to zero (TBCNT = 0000h)
  - Both PRD_eq and CNT_zero events occurrence

  Which of these three events will drive the counter compare module is specified by the EPWM_CMPCTL[1-0] LOADAMODE and EPWM_CMPCTL[3-2] LOADBMODE register bit fields. Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.
- **Immediate Load Mode:**
  If immediate load mode is selected (EPWM_CMPCTL[4] SHDWAMODE = 1h or EPWM_CMPCTL[6] SHDWBMODE = 1h), then a read from or a write to the register will go directly to the active register.

Figure 12-312 and Figure 12-313 show Compare A and B Dual Shadow registers.

Copyright © 2026 Texas Instruments Incorporated

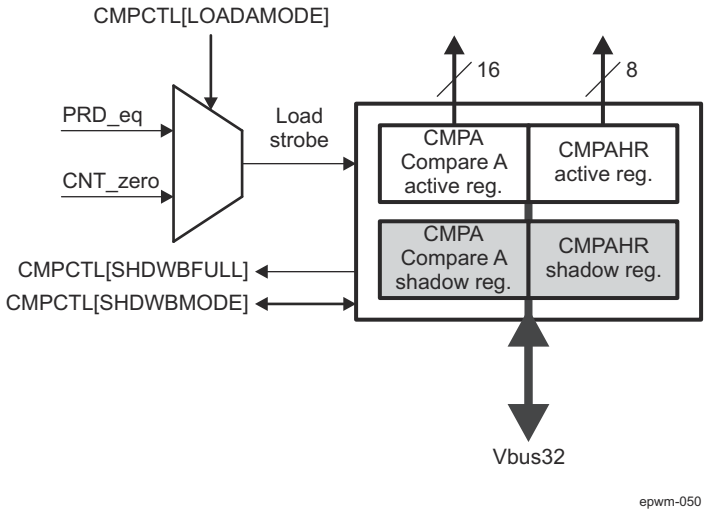


Figure 12-312. Compare A Dual Shadow register

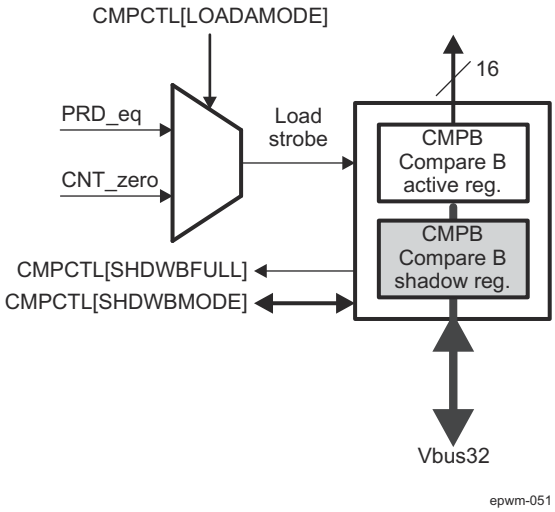


Figure 12-313. Compare B Dual Shadow register

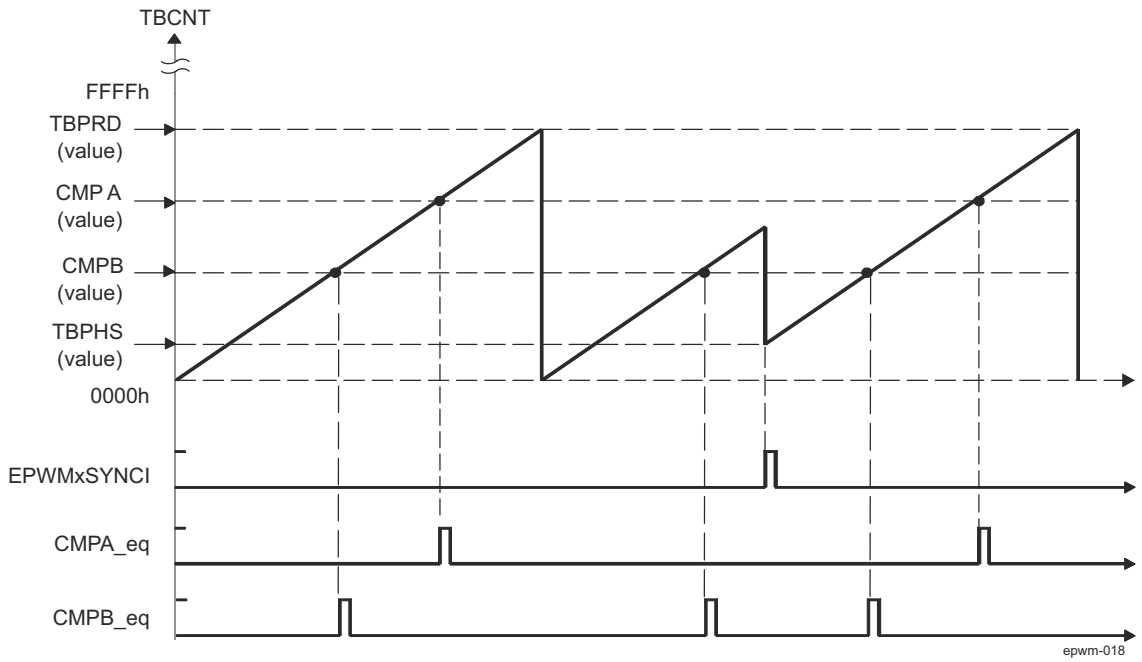
12.4.3.4.3.4 EPWM Counter-Compare Submodule Timing Waveforms

As described in Section 12.4.3.4.2, the Time Base (TB) module can be configured to operate in 3 distinct count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

The timing diagrams in Figure 12-314 to Figure 12-317 show how CMPA and CMPB events are generated in each of the 3 count modes and how the EPWMxSYNCI signal interacts.

Copyright © 2026 Texas Instruments Incorporated



An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCNT count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

Figure 12-314. EPWM Counter-Compare Event Waveforms in Up-Count Mode

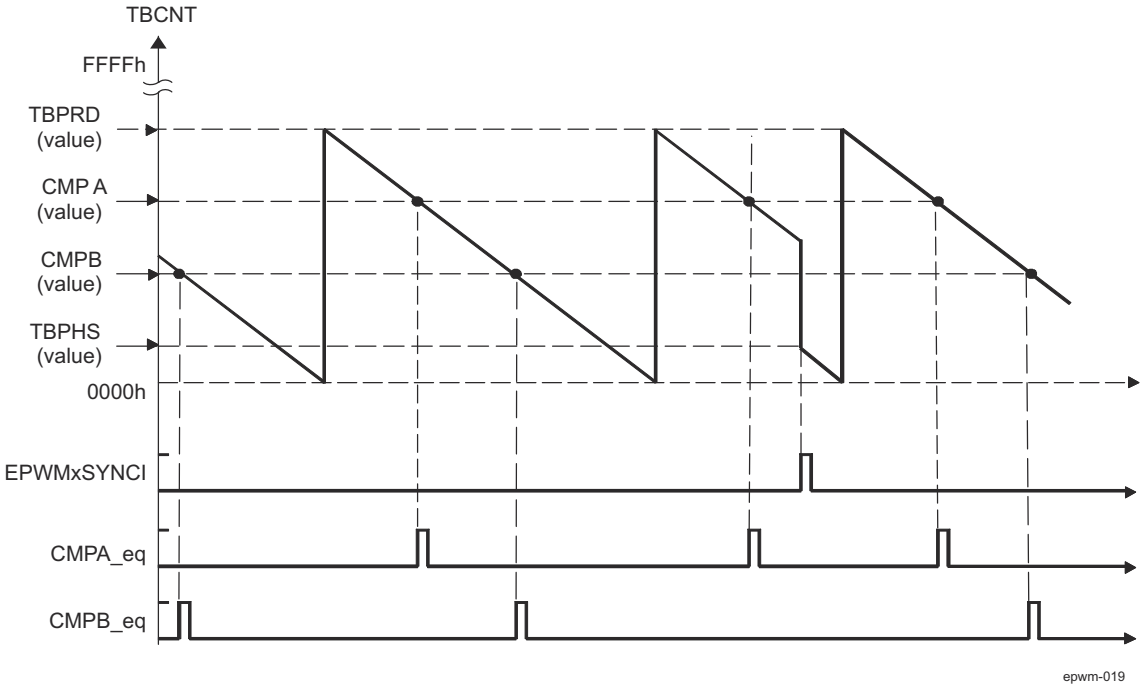


Figure 12-315. EPWM Counter-Compare Events in Down-Count Mode

Copyright © 2026 Texas Instruments Incorporated

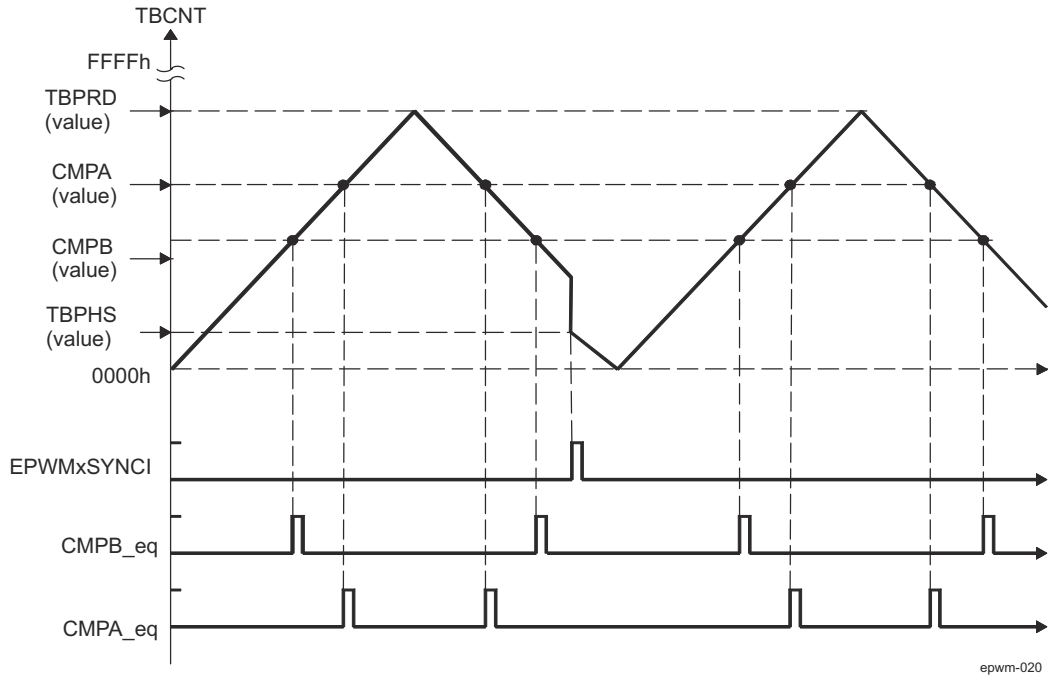


Figure 12-316. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM_TBCTL[13] PHSDIR = 0 Count Down on Synchronization Event

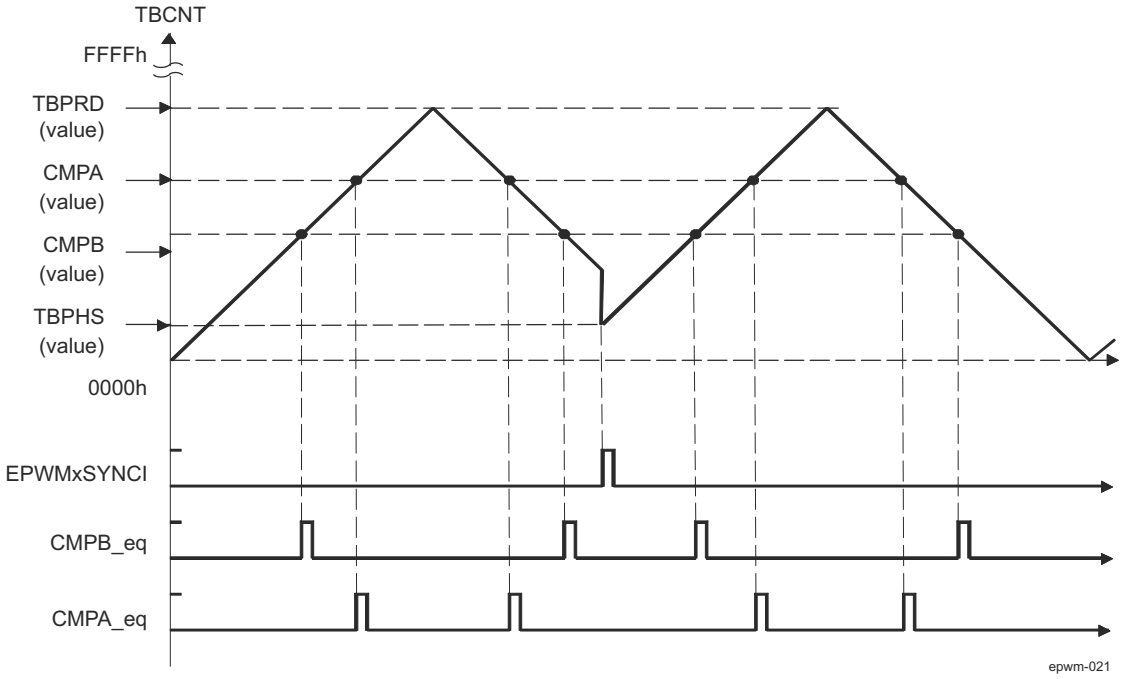


Figure 12-317. EPWM Counter-Compare Events in Up-Down-Count Mode, EPWM_TBCTL[13] PHSDIR = 1 Count Up on Synchronization Event

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.4 EPWM Action-Qualifier (AQ) Submodule

This section describes the Action-Qualifier (AQ) submodule in the PWM module.

12.4.3.4.4.1 Overview

Figure 12-318 shows the action-qualifier (AQ) submodule in the EPWM system. This submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

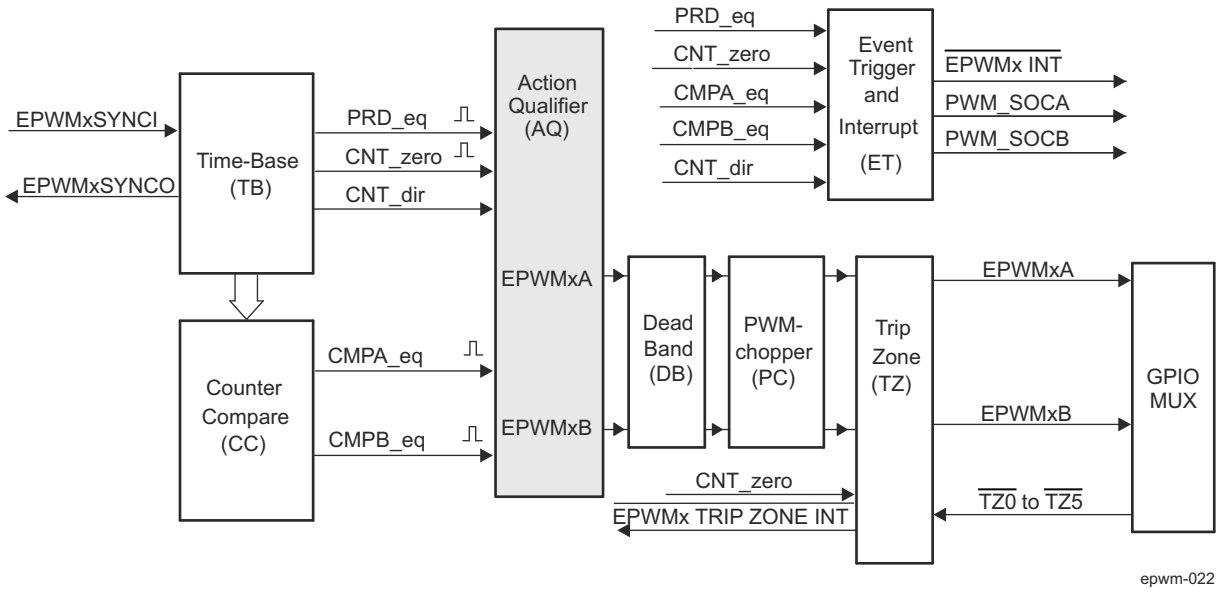


Figure 12-318. EPWM Action-Qualifier Submodule

AQ module features:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - PRD_eq: Time-base counter equal to the period (TBCNT = TBPRD)
  - CNT_zero: Time-base counter equal to zero (TBCNT = 0000h)
  - CMPA_eq: Time-base counter equal to the counter-compare A register (TBCNT = CMPA)
  - CMPB_eq: Time-base counter equal to the counter-compare B register (TBCNT = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

12.4.3.4.4.2 Controlling and Monitoring the EPWM Action-Qualifier Submodule

Table 12-271 lists the registers used to control and monitor the action-qualifier submodule.

Table 12-271. EPWM Action-Qualifier Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_AQCTLA	Action-Qualifier Control Register For Output A (EPWMxA)	16h	No
EPWM_AQCTLB	Action-Qualifier Control Register For Output B (EPWMxB)	18h	No
EPWM_AQSFRC	Action-Qualifier Software Force Register	1Ah	No
EPWM_AQCSFRC	Action-Qualifier Continuous Software Force	1Ch	Yes

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers as shown in Figure 12-319. The possible input events are summarized again in Table 12-272.

Copyright © 2026 Texas Instruments Incorporated

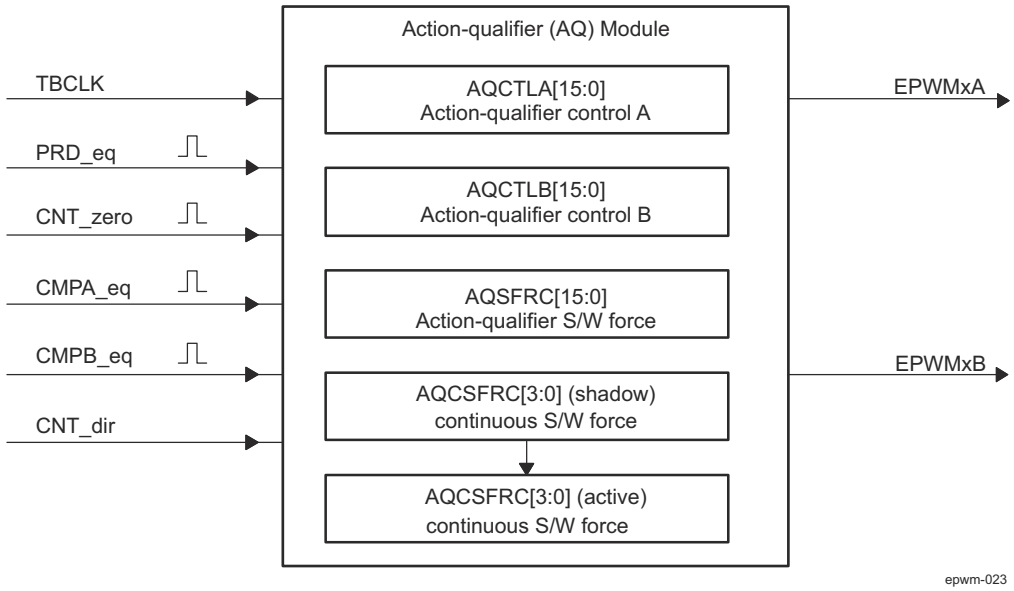


Figure 12-319. EPWM Action-Qualifier Submodule Inputs and Outputs

Table 12-272. EPWM Action-Qualifier Submodule Possible Input Events

Signal	Description	Register Bitfield Compared
PRD_eq	Time-base counter equal to the period value	TBCNT = TBPRD
CNT_zero	Time-base counter equal to zero	TBCNT = 0000h
CMPA_eq	Time-base counter equal to the counter-compare A	TBCNT = CMPA
CMPB_eq	Time-base counter equal to the counter-compare B	TBCNT = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by EPWM_AQSFRC and EPWM_AQCSFRC registers.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts. See the Event_Trigger (ET) submodule description in Section 12.4.3.4.8 for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this chapter use a set of symbolic actions. These symbols are summarized in Figure 12-320. Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

Copyright © 2026 Texas Instruments Incorporated

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
SW ╳	Z ╳	CA ╳	CB ╳	P ╳	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

epwm-024

Figure 12-320. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.4.3 EPWM Action-Qualifier Event Priority

It is possible for the EPWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is: events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in Table 12-273. A priority level 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCNT.

Table 12-273. EPWM Action-Qualifier Event Priority for Up-Down-Count Mode

Priority Level	Event if TBCNT is Incrementing TBCNT = 0 up to TBCNT = TBPRD	Event if TBCNT is Decrementing TBCNT = TBPRD down to TBCNT = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD in EPWM_TBPRD active register)
5	Counter equals CMPB on down-count (CBD) (1)	Counter equals CMPB on up-count (CBU) (1)
6 (Lowest)	Counter equals CMPA on down-count (CAD) (1)	Counter equals CMPA on up-count (CBU) (1)

(1) To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

Table 12-274 shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

Table 12-274. EPWM Action-Qualifier Event Priority for Up-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	Counter equal to CMPB on up-count (CBU)
4	Counter equal to CMPA on up-count (CAU)
5 (Lowest)	Counter equal to Zero

Table 12-275 shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

Table 12-275. EPWM Action-Qualifier Event Priority for Down-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

Copyright © 2026 Texas Instruments Incorporated

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 12-276.

Table 12-276. Behavior if CMPA/CMPB is Greater than the Period

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAU/CBU
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB > TBPRD, then the event will not occur.	Never occurs.
Down-Count Mode	Never occurs.	If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCNT = TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCNT = TBPRD).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCNT = TBPRD).

12.4.3.4.4.4 Waveforms for Common EPWM Configurations

Note

The waveforms in this chapter show the EPWMs behavior for a static compare register value. In a running system, the active compare registers (EPWM_CMPA and EPWM_CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place — either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

Use up-down-count mode to generate a symmetric PWM:

- If EPWM_CMPA / EPWM_CMPB is loaded on zero, then use EPWM_CMPA / EPWM_CMPB values greater than or equal to 1.
- If EPWM_CMPA / EPWM_CMPB is loaded on period, then use EPWM_CMPA / EPWM_CMPB values less than or equal to TBPRD - 1.

  This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Use up-down-count mode to generate an asymmetric PWM:

- To achieve 50-0% asymmetric PWM use the following configuration: Load EPWM_CMPA / EPWM_CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50-0% PWM duty.

When using up-count mode to generate an asymmetric PWM:

- To achieve 0-100% asymmetric PWM use the following configuration: Load EPWM_CMPA / EPWM_CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

Copyright © 2026 Texas Instruments Incorporated

Figure 12-321 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCNT. In this mode 0-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When EPWM_CMPA = EPWM_TBPRD, the PWM signal is low for the entire period giving the 0% duty waveform. When CMPA = 0, the PWM signal is high achieving 100% duty.

When using this configuration in practice, if CMPA/CMPB is loaded on zero, then use CMPA/CMPB values greater than or equal to 1. If CMPA/CMPB is loaded on period, then use CMPA/CMPB values less than or equal to TBPRD - 1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

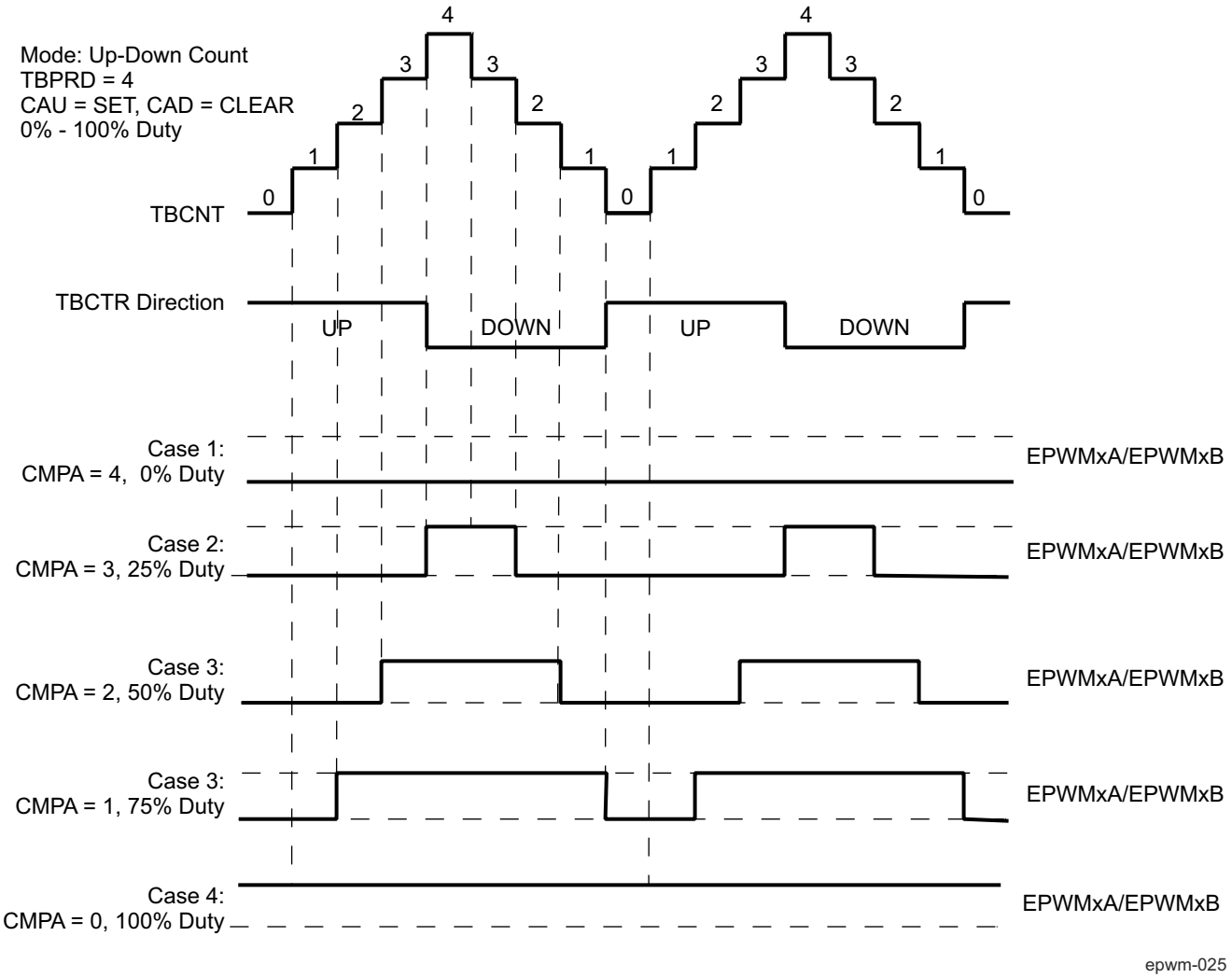


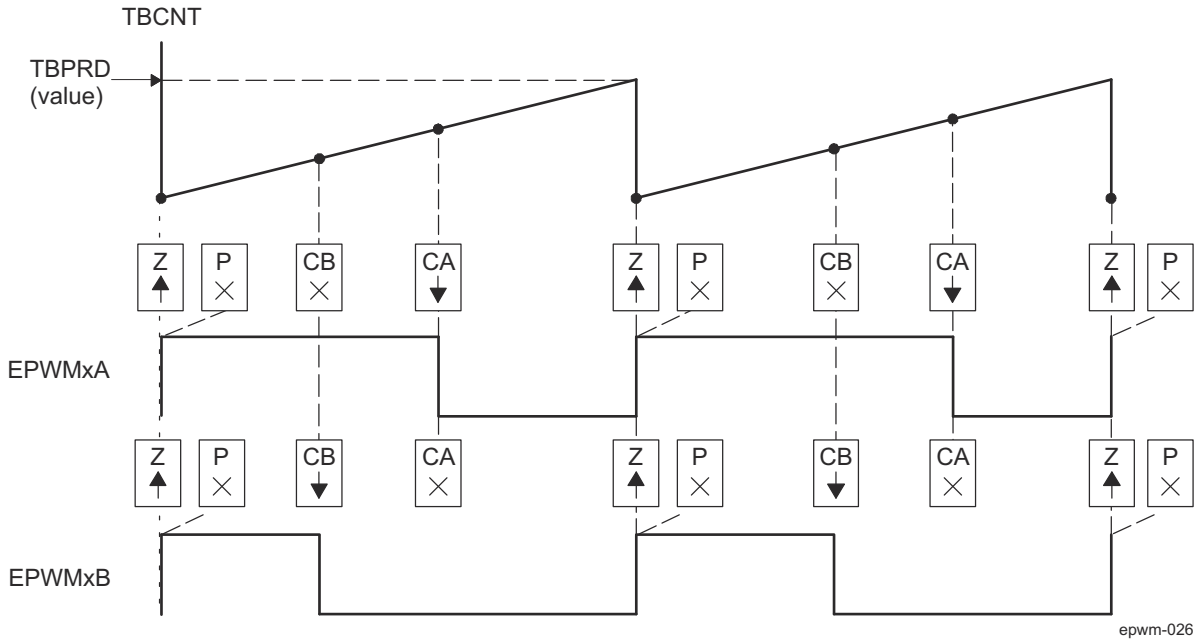
Figure 12-321. EPWM Up-Down-Count Mode Symmetrical Waveform

Copyright © 2026 Texas Instruments Incorporated

The PWM waveforms in Figure 12-322 through Figure 12-327 show some common action-qualifier configurations. Some conventions used in the figures are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers (EPWM_TBPRD, EPWM_CMPA, and EPWM_CMPB). The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from EPWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

Table 12-277 and Table 12-278 contain initialization and runtime register configurations for the waveforms in Figure 12-322.



A. PWM period = (TBPRD + 1) × T_{TBCLK}
B. Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
D. The "Do Nothing" actions (X) are shown for completeness, but will not be shown on subsequent diagrams.
E. Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

Figure 12-322. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active High

Copyright © 2026 Texas Instruments Incorporated

Table 12-277. EPWMx Initialization for Figure 12-322

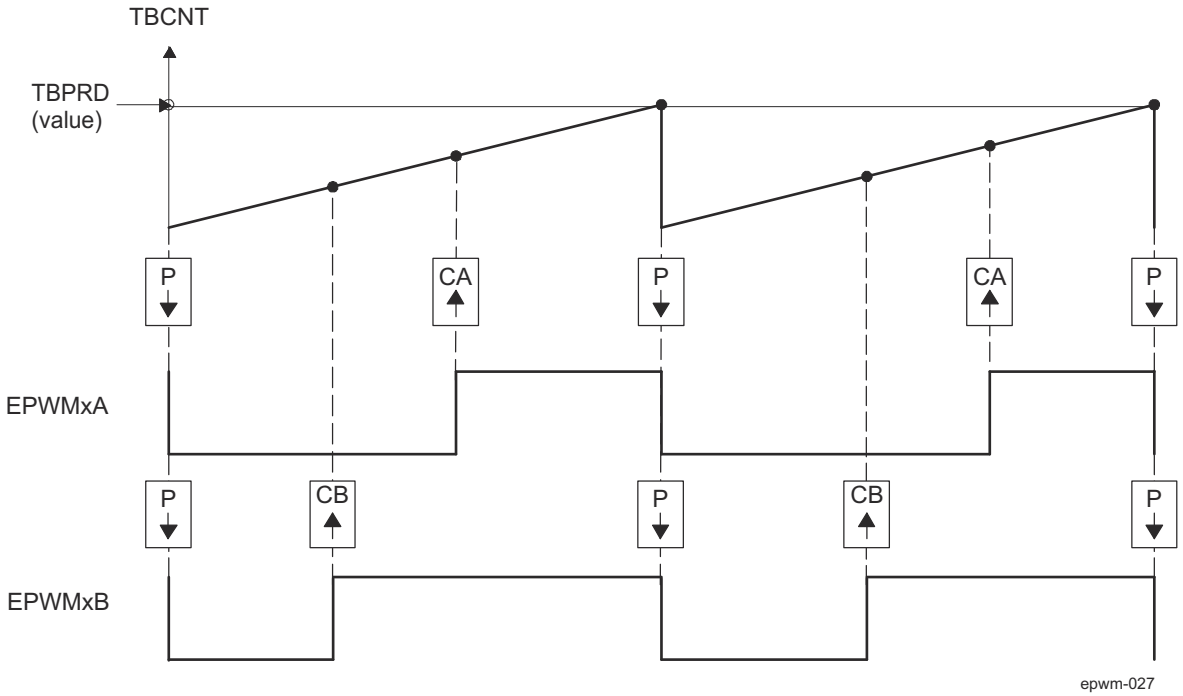
Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	ZRO	AQ_SET	
	CAU	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_SET	
	CBU	AQ_CLEAR	

Table 12-278. EPWMx Run Time Changes for Figure 12-322

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2026 Texas Instruments Incorporated

Table 12-279 and Table 12-280 contain initialization and runtime register configurations for the waveforms in Figure 12-323.



A. PWM period = (TBPRD + 1) × T_{TBCLK}
B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
D. The Do Nothing actions (X) are shown for completeness here, but will not be shown on subsequent diagrams.
E. Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

Figure 12-323. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB — Active Low

Copyright © 2026 Texas Instruments Incorporated

Table 12-279. EPWMx Initialization for Figure 12-323

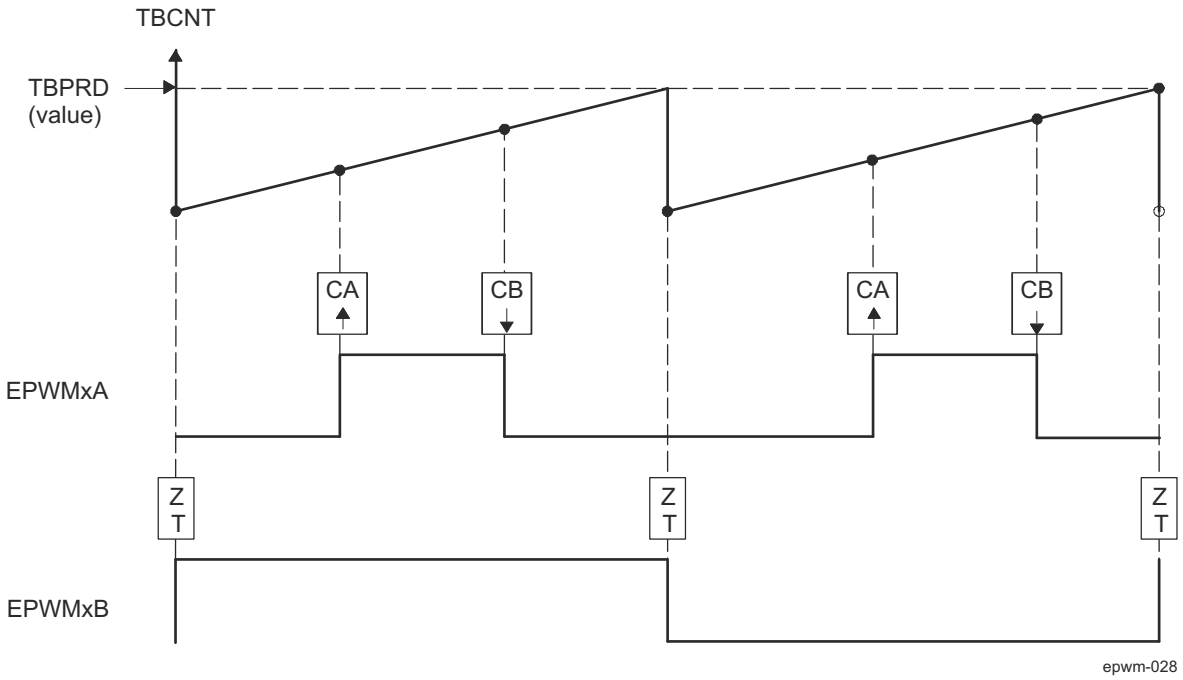
Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	
EPWM_AQCTLB	PRD	AQ_CLEAR	
	CBU	AQ_SET	

Table 12-280. EPWMx Run Time Changes for Figure 12-323

Register	Bit	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2026 Texas Instruments Incorporated

Table 12-281 and Table 12-282 contain initialization and runtime register configurations for the waveforms Figure 12-324. Use the code in Example 12-8 to define the headers.



A. PWM frequency = 1/((TBPRD + 1) × T_{TBCLK})
B. Pulse can be placed anywhere within the PWM cycle (0000h - TBPRD)
C. High time duty proportional to (CMPB - CMPA)
D. EPWMxB can be used to generate a 50% duty square wave with frequency = 1/2 × ((TBPRD + 1) × TBCLK)

Figure 12-324. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA

Copyright © 2026 Texas Instruments Incorporated

Table 12-281. EPWMx Initialization for Figure 12-324

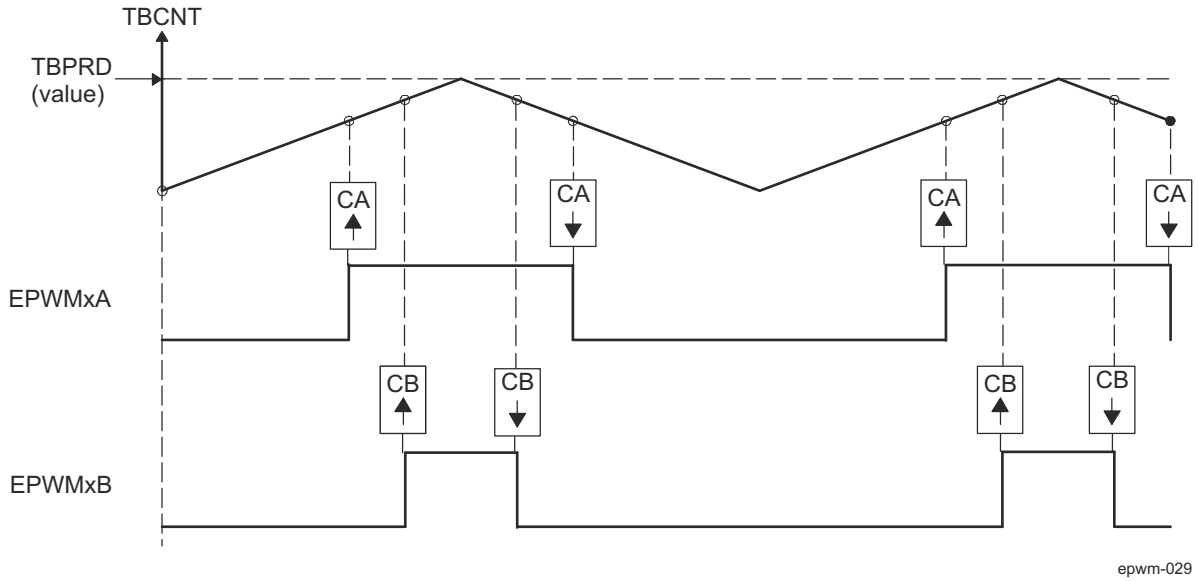
Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	200 (C8h)	Compare A = 200 TBCLK counts
EPWM_CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	CAU	AQ_SET	
	CBU	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_TOGGLE	

Table 12-282. EPWMx Run Time Changes for Figure 12-324

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	EdgePosB	

Copyright © 2026 Texas Instruments Incorporated

Table 12-283 and Table 12-284 contain initialization and runtime register configurations for the waveforms in Figure 12-325. Use the code in Example 12-8 to define the headers.



A. PWM period = 2 x TBPRD × T_{TBCLK}
B. Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
C. Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
D. Outputs EPWMxA and EPWMxB can drive independent power switches.

Figure 12-325. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low

Copyright © 2026 Texas Instruments Incorporated

Table 12-283. EPWMx Initialization for Figure 12-325

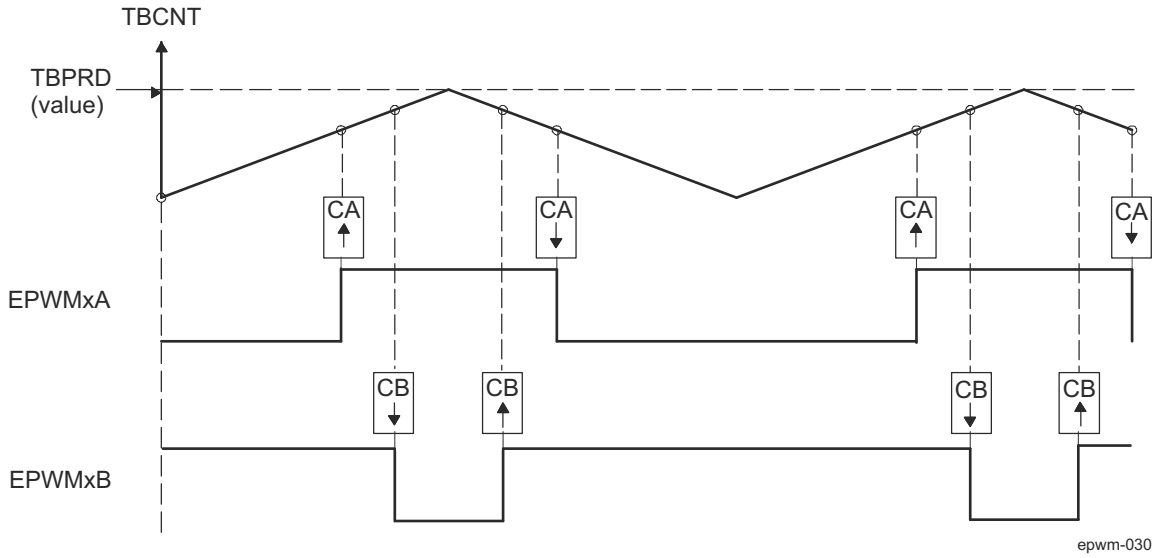
Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	400 (190h)	Compare A = 400 TBCLK counts
EPWM_CMPB	CMPB	500 (1F4h)	Compare B = 500 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
EPWM_AQCTLB	CBU	AQ_SET	
	CBD	AQ_CLEAR	

Table 12-284. EPWMx Run Time Changes for Figure 12-325

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2026 Texas Instruments Incorporated

Table 12-285 and Table 12-286 contain initialization and runtime register configurations for the waveforms in Figure 12-326. Use the code in Example 12-8 to define the headers.



A. PWM period = 2 × TBPRD × T_{TBCLK}
B. Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.
C. Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB.
D. Outputs EPWMx can drive upper/lower (complementary) power switches.
E. Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

Figure 12-326. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary

Copyright © 2026 Texas Instruments Incorporated

Table 12-285. EPWMx Initialization for Figure 12-326

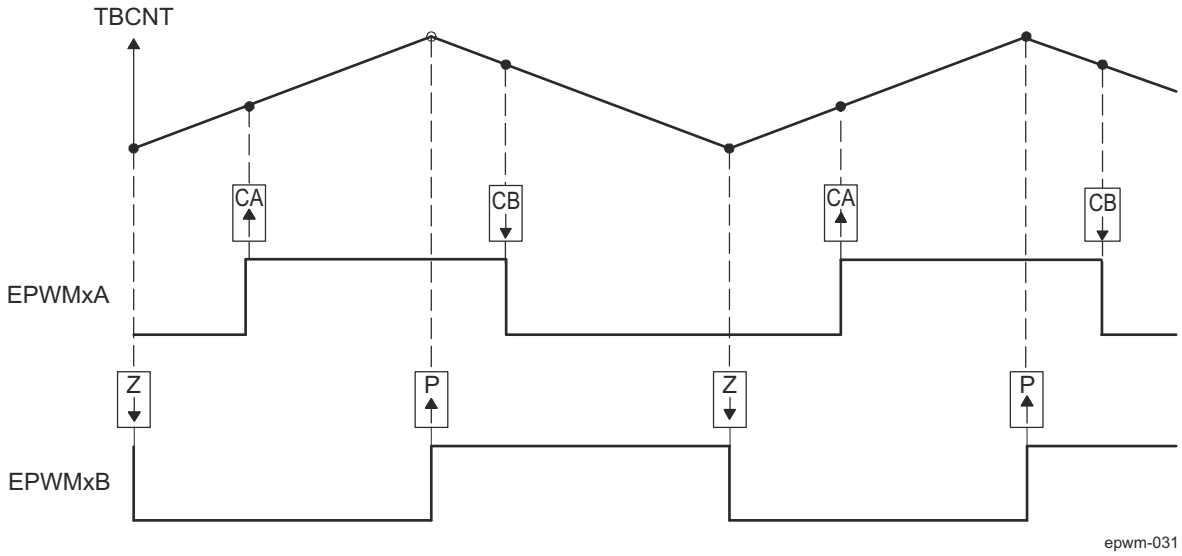
Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
EPWM_CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
EPWM_AQCTLB	CBU	AQ_CLEAR	
	CBD	AQ_SET	

Table 12-286. EPWMx Run Time Changes for Figure 12-326

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2026 Texas Instruments Incorporated

Table 12-287 and Table 12-288 contain initialization and runtime register configurations for the waveforms in Figure 12-327. Use the code in Example 12-8 to define the headers.



A. PWM period = 2 × TBPRD × TBCLK
B. Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
C. Duty modulation for EPWMxA is set by CMPA and CMPB.
D. Low time duty for EPWMxA is proportional to (CMPA + CMPB).
E. To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Set ! Clear and Clear Set).
F. Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB).

Figure 12-327. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA — Active Low

Copyright © 2026 Texas Instruments Incorporated

Table 12-287. EPWMx Initialization for Figure 12-327

Register	Bitfield	Value	Comments
EPWM_TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
EPWM_TBPHS	TBPHS	0	Clear Phase Register to 0
EPWM_TBCNT	TBCNT	0	Clear TB counter
EPWM_TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = FICLK
	CLKDIV	TB_DIV1	
EPWM_CMPA	CMPA	250 (FAh)	Compare A = 250 TBCLK counts
EPWM_CMPB	CMPB	450 (1C2h)	Compare B = 450 TBCLK counts
EPWM_CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on TBCNT = 0
	LOADBMODE	CC_CTR_ZERO	Load on TBCNT = 0
EPWM_AQCTLA	CAU	AQ_SET	
	CBD	AQ_CLEAR	
EPWM_AQCTLB	ZRO	AQ_CLEAR	
	PRD	AQ_SET	

Table 12-288. EPWMx Run Time Changes for Figure 12-327

Register	Bitfield	Value	Comments
EPWM_CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
EPWM_CMPB	CMPB	EdgePosB	Adjust duty for output EPWM1B

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.5 EPWM Dead-Band Generator (DB) Submodule

This section describes the Dead-Band Generator (DB) submodule in the PWM module.

12.4.3.4.5.1 Overview

Figure 12-328 illustrates the dead-band generator submodule within the EPWM module.

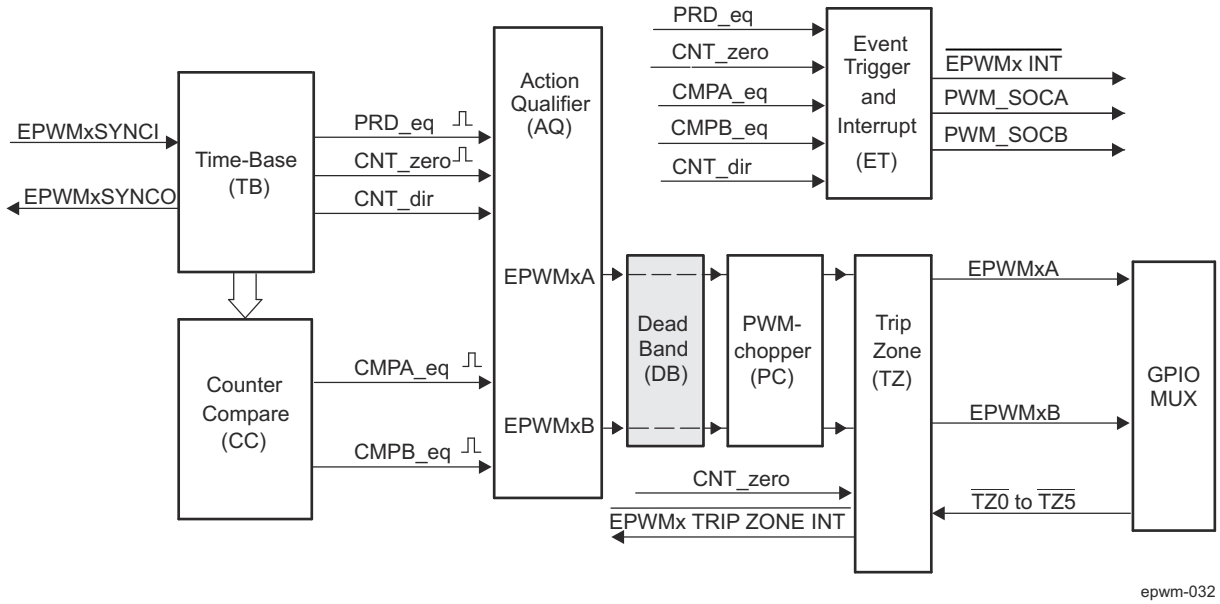


Figure 12-328. Dead-Band Generator Submodule

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the EPWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band generator submodule should be used.

The key functions of the dead-band generator submodule are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in Figure 12-328)

12.4.3.4.5.2 Controlling and Monitoring the EPWM Dead-Band Submodule

The dead-band generator submodule operation is controlled and monitored via the following registers:

Table 12-289. Dead-Band Generator Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_DBCTL	Dead-Band Control Register	1Eh	No
EPWM_DBRED	Dead-Band Rising Edge Delay Count Register	20h	No
EPWM_DBFED	Dead-Band Falling Edge Delay Count Register	22h	No

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.5.3 Operational Highlights for the EPWM Dead-Band Generator Submodule

The dead-band submodule has two groups of independent selection options as shown in Figure 12-329.

- **Input Source Selection:** The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the EPWM_DBCTL[5-4] IN_MODE control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:
  - EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  - EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  - EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  - EPWMxB In is the source for both falling-edge and rising-edge delay.
- **Output Mode Control:** The output mode is configured by way of the EPWM_DBCTL[1-0] OUT_MODE bit fields. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.
- **Polarity Control:** The polarity control (EPWM_DBCTL[3-2] POLSEL) allows to be specified whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

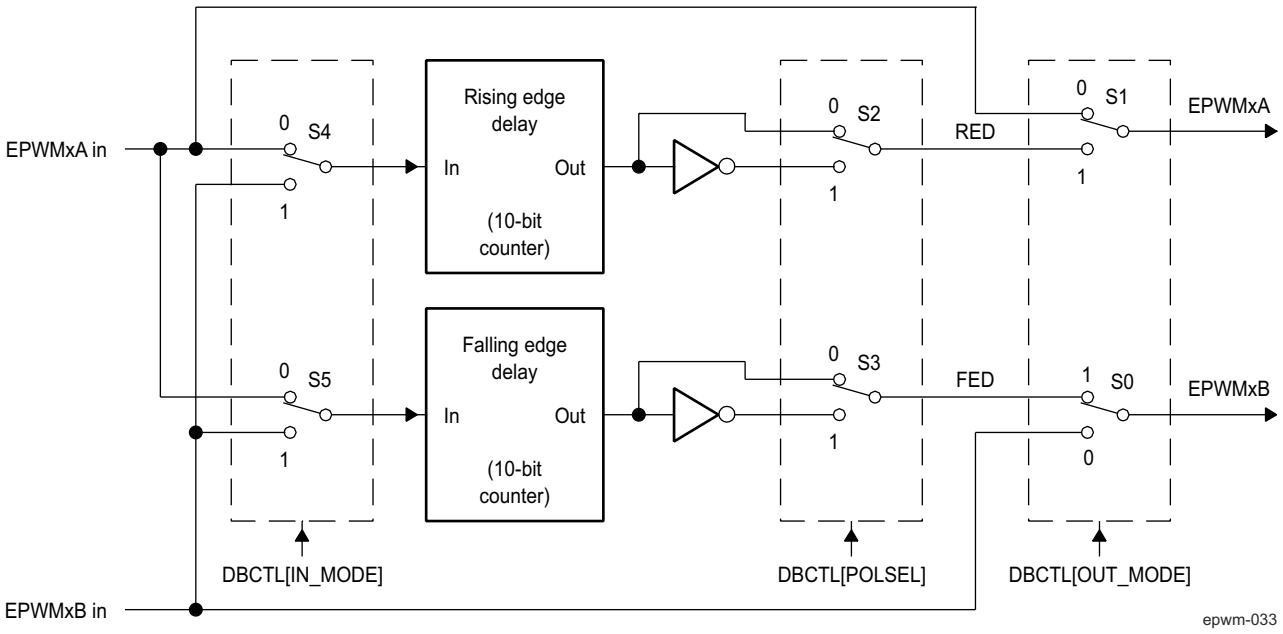


Figure 12-329. Configuration Options for the EPWM Dead-Band Generator Submodule

Copyright © 2026 Texas Instruments Incorporated

Although all combinations are supported, not all are typical usage modes. Table 12-290 lists some classical dead-band configurations. These modes assume that the EPWM_DBCTL[5-4] IN_MODE is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 12-290 fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)** Allows to be fully disabled the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings** These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in Figure 12-330. Note that to generate equivalent waveforms to Figure 12-330, configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay** Finally the last two entries in Table 12-290 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 12-290. Classical Dead-Band Operating Modes

Mode	Mode Description (1)	EPWM_DBCTL[3-2] POLSEL		EPWM_DBCTL[1-0] OUT_MODE	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	x	x	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxB In with No Delay	0 or 1	0 or 1	1	0

(1) These are classical dead-band modes and assume that EPWM_DBCTL[5-4] IN_MODE = 0h0. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN_MODE configuration.

Figure 12-330 shows waveforms for typical cases where 0% < duty < 100%.

Copyright © 2026 Texas Instruments Incorporated

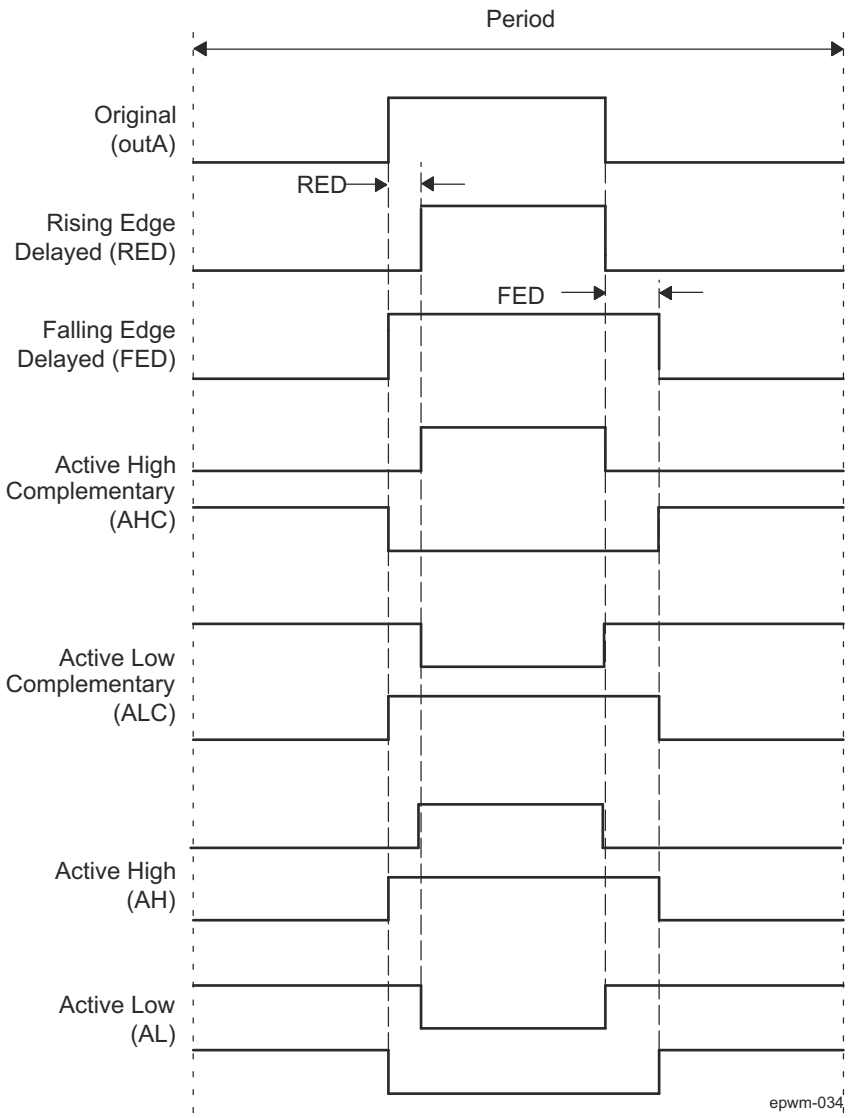


Figure 12-330. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)

The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the EPWM_DBRED and EPWM_DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formulas to calculate falling-edge-delay and rising-edge-delay are:

FED = EPWM_DBFED × T_{TBCLK}

RED = EPWM_DBRED × T_{TBCLK}

Where T_{TBCLK} is the period of TBCLK, the prescaled version of FICLK.

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.6 EPWM-Chopper (PC) Submodule

This section describes the PWM-Chopper (PC) submodule in the PWM module.

12.4.3.4.6.1 Overview

Figure 12-331 illustrates the PWM-chopper (PC) submodule within the EPWM module. The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if a pulse transformer-based gate drivers to control the power switching elements is needed.

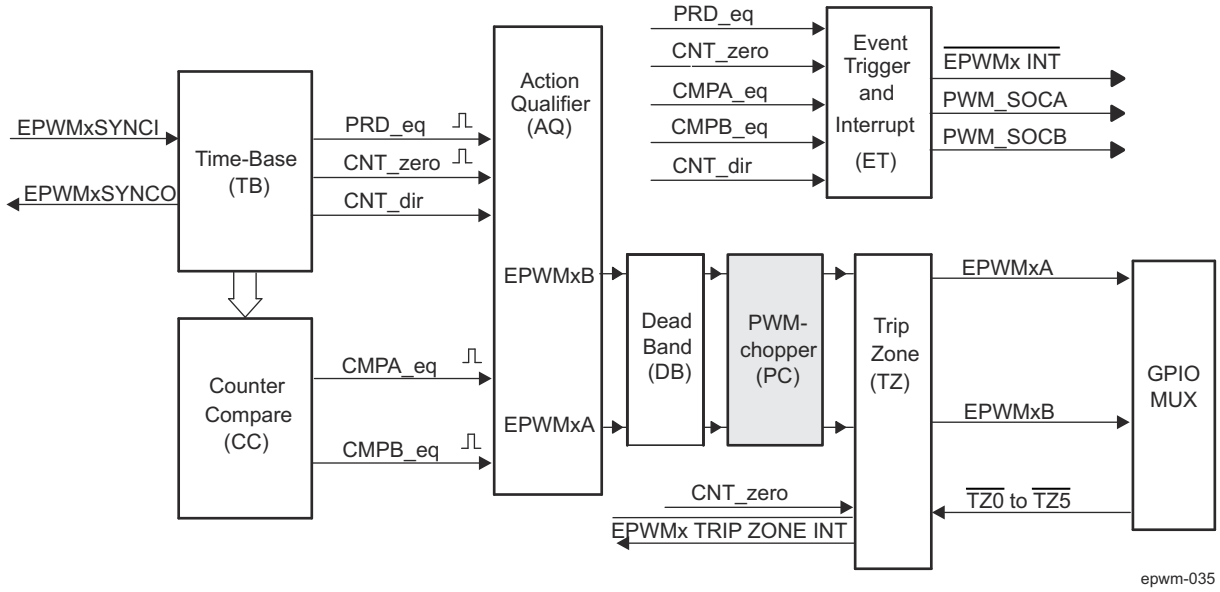


Figure 12-331. PWM-Chopper Submodule

12.4.3.4.6.2

PC module key features:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

12.4.3.4.6.3 Controlling the EPWM-Chopper Submodule

The EPWM-chopper submodule operation is controlled via the register in Table 12-291.

Table 12-291. EPWM-Chopper Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_PCCTL	PWM-chopper Control Register	3Ch	No

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.6.4 Operational Highlights for the EPWM-Chopper Submodule

Figure 12-332 shows the operational details of the EPWM-chopper submodule. The carrier clock is derived from FICLK. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the EPWM_PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

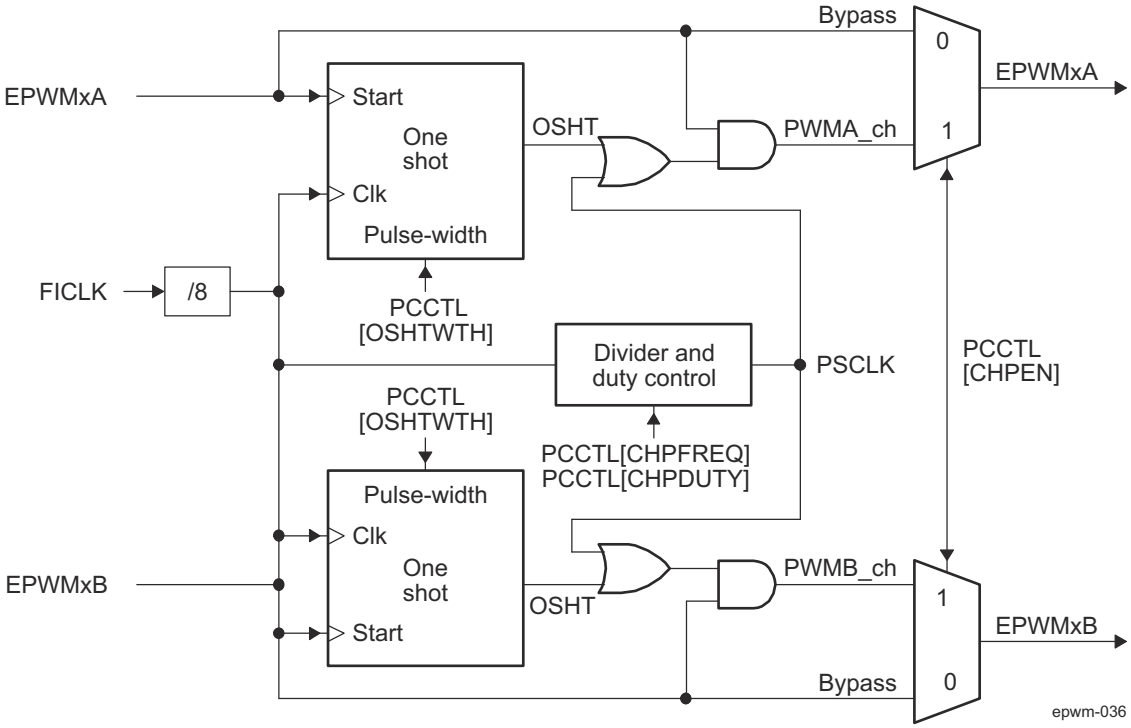


Figure 12-332. PWM-Chopper Submodule Signals and Registers

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.6.5 EPWM-Chopper Waveforms

Figure 12-333 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

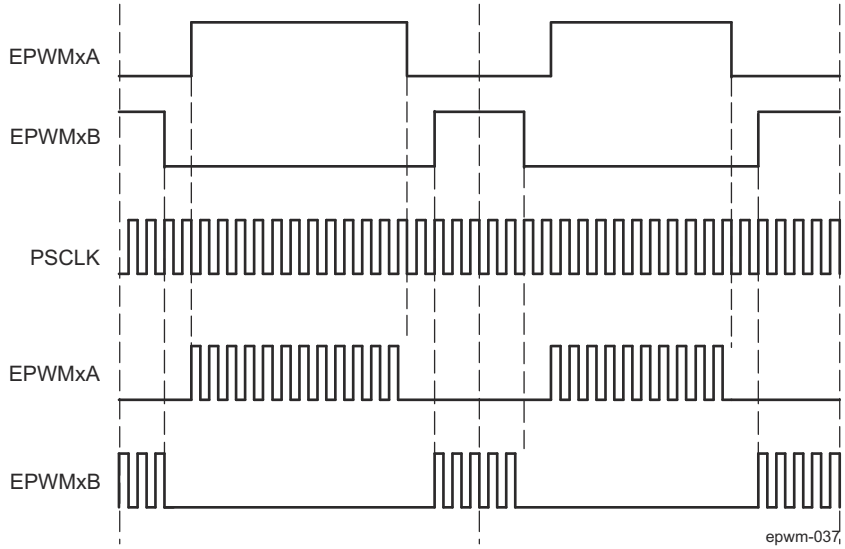


Figure 12-333. Simple EPWM-Chopper Submodule Waveforms Showing Chopping Action Only

12.4.3.4.6.5.1 EPWM-Chopper One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$T_{1stpulse} = T_{FICLK} \times 8 \times OSHTWTH$

Where T_{FICLK} is the period of the system clock (FICLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 12-334 shows the first and subsequent sustaining pulses.

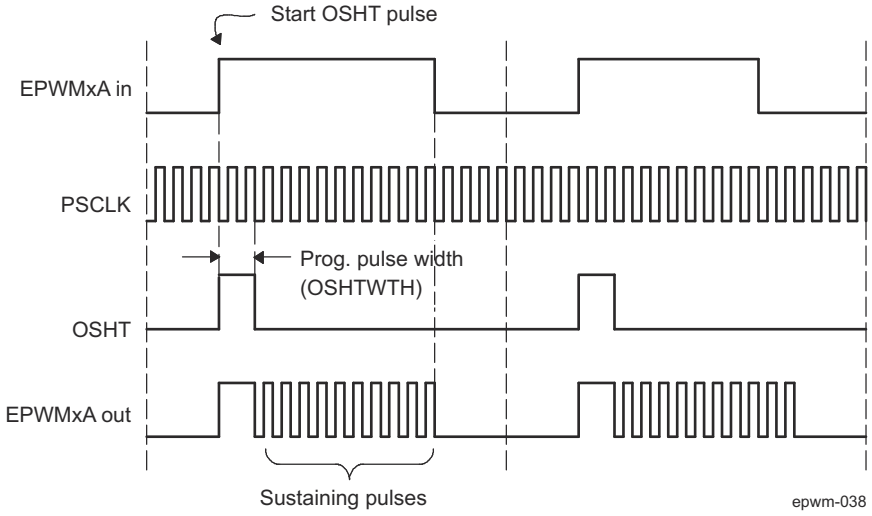


Figure 12-334. EPWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.6.5.2 EPWM-Chopper Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 12-335 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5 to 87.5%.

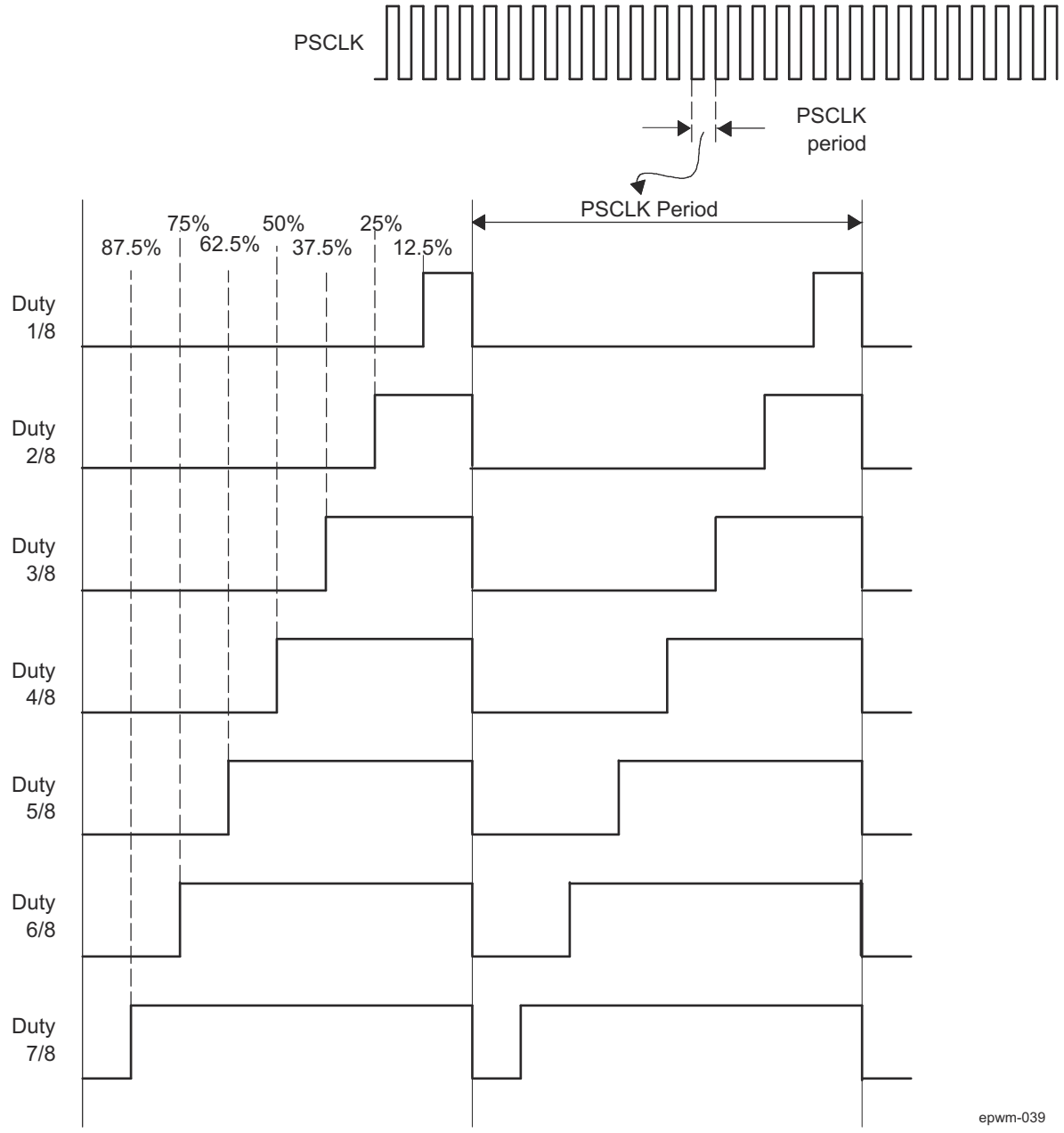


Figure 12-335. EPWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.7 EPWM Trip-Zone (TZ) Submodule

This section describes the Trip-Zone (TZ) submodule in the PWM module.

12.4.3.4.7.1 Overview

Figure 12-336 shows how the trip-zone (TZ) submodule fits within the EPWM module. Each EPWM module is connected to six $\overline{TZ}$ signals ($\overline{TZ0}$ to $\overline{TZ5}$) that are sourced from the GPIO MUX. These signals indicates external fault or trip conditions, and the EPWM outputs can be programmed to respond accordingly when faults occur. See *EPWM Integration* to determine the number of trip-zone pins available for the device.

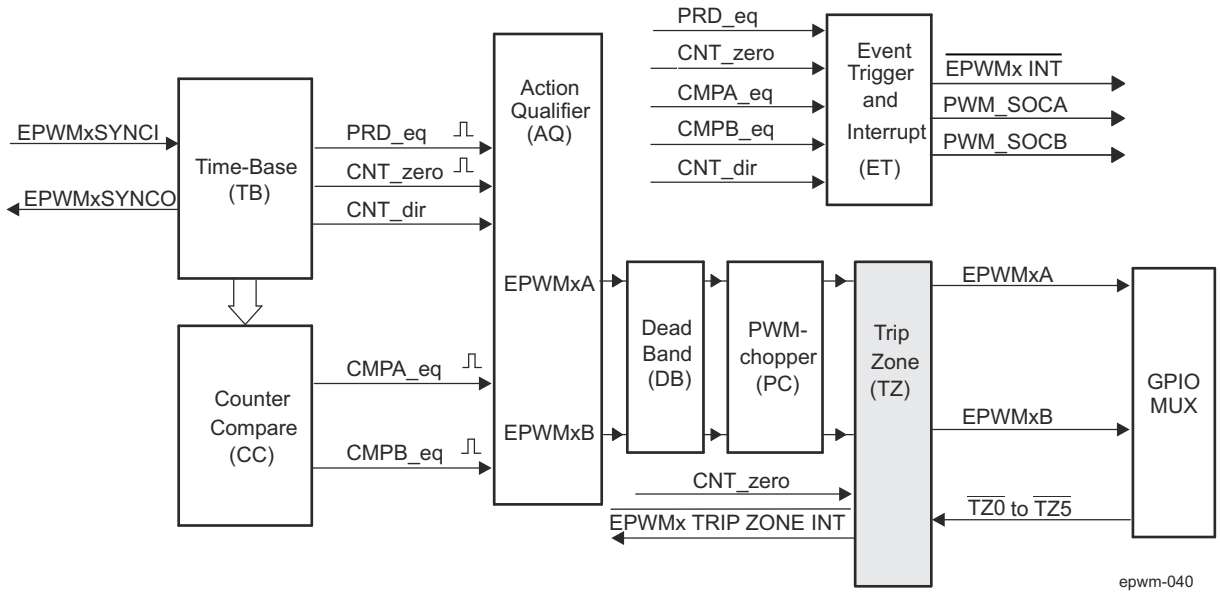


Figure 12-336. EPWM Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs $\overline{TZ0}$ to $\overline{TZ5}$ can be flexibly mapped to any EPWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

Note

For each EPWMx module, 6 tripzone input events are supported ($\overline{EPWMx_TRIP_TZ[5:0]}$). Each tripzone input for all EPWMx modules are tied to a common tripzone input pin, so that any EPWMx can be triggered by any of the six tripzone inputs.

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.7.2 Controlling and Monitoring the EPWM Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

Table 12-292. EPWM Trip-Zone Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_TZSEL	Trip-Zone Select Register (1)	24h	No
EPWM_TZCTL	Trip-Zone Control Register (1)	28h	No
EPWM_TZEINT	Trip-Zone Enable Interrupt Register (1)	2Ah	No
EPWM_TZFLG	Trip-Zone Flag Register (1)	2Ch	No
EPWM_TZCLR	Trip-Zone Clear Register (1)	2Eh	No
EPWM_TZFRC	Trip-Zone Force Register (1)	30h	No

(1) All trip-zone registers are EALLOW protected and can be modified only after executing the EALLOW instruction.

12.4.3.4.7.3 Operational Highlights for the EPWM Trip-Zone Submodule

The trip-zone signals at pin $\overline{TZ0}$ to $\overline{TZ5}$ is an active-low input signal. When the pin goes low, it indicates that a trip event has occurred. Each EPWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular EPWM module is determined by the EPWM_TZSEL register for that specific EPWM module. The trip-zone signal may or may not be synchronized to the system clock (FICLK). A minimum of 1 FICLK low pulse on the $\overline{TZ0}$ to $\overline{TZ5}$ inputs is sufficient to trigger a fault condition in the EPWM module. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on the $\overline{TZ0}$ to $\overline{TZ5}$ inputs.

The $\overline{TZ0}$ to $\overline{TZ5}$ inputs can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a EPWM module. The configuration is determined by the EPWM_TZSEL[5-0] CBCk and EPWM_TZSEL[13-8] OSHTk bit field (where k = 0 to 5 corresponds to the trip pin), respectively.

- **Cycle-by-Cycle (CBC):** When a cycle-by-cycle trip event occurs, the action specified in the EPWM_TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 12-293 lists the possible actions. In addition, the cycle-by-cycle trip event flag (EPWM_TZFLG[1] CBC) is set and a EPWMxTZINT interrupt is generated if it is enabled in the EPWM_TZEINT register.

  The specified condition on the pins is automatically cleared when the EPWM time-base counter reaches zero (EPWM_TBCNT[15-0] TBCNT = 0000h) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The EPWM_TZFLG[1] CBC flag bit will remain set until it is manually cleared by writing to the EPWM_TZCLR[1] CBC bit. If the cycle-by-cycle trip event is still present when the EPWM_TZFLG[1] CBC bit is cleared, then it will again be immediately set.
- **One-Shot (OSHT):** When a one-shot trip event occurs, the action specified in the EPWM_TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 12-293 lists the possible actions. In addition, the one-shot trip event flag (EPWM_TZFLG[2] OST) is set and a EPWMxTZINT interrupt is generated if it is enabled in the EPWM_TZEINT register. The one-shot trip condition must be cleared manually by writing to the EPWM_TZCLR[2] OST bit.

The action taken when a trip event occurs can be configured individually for each of the EPWM output pins by way of the EPWM_TZCTL[1-0] TZA and EPWM_TZCTL[3-2] TZB register bit field. One of four possible actions, shown in Table 12-293, can be taken on a trip event.

Copyright © 2026 Texas Instruments Incorporated

Table 12-293. Possible Actions On an EPWM Trip Event

EPWM_TZCTL[1-0] TZA and/or EPWM_TZCTL[3-2] TZB	EPWMxA and/or EPWMxB	Comment
0	High-Impedance	Tripped
1h	Force to High State	Tripped
2h	Force to Low State	Tripped
3h	No Change	Do Nothing. No change is made to the output.

Example of EPWM Trip-Zone Configurations

A one-shot trip event on $\overline{TZ0}$ pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the EPWM1 registers as follows:
  - EPWM_TZSEL[8] OSHT1 = 1: enables $\overline{TZ}$ as a one-shot event source for EPWM1
  - EPWM_TZCTL[1-0] TZA = 2: EPWM1A will be forced low on a trip event
  - EPWM_TZCTL[3-2] TZB = 2: EPWM1B will be forced low on a trip event
- Configure the EPWM2 registers as follows:
  - EPWM_TZSEL[8] OSHT1 = 1: enables $\overline{TZ}$ as a one-shot event source for EPWM2
  - EPWM_TZCTL[1-0] TZA = 1: EPWM2A will be forced high on a trip event
  - EPWM_TZCTL[3-2] TZB = 1: EPWM2B will be forced high on a trip event

12.4.3.4.7.4 Generating EPWM Trip-Event Interrupts

Figure 12-337 and Figure 12-338 illustrate the trip-zone submodule control and interrupt logic, respectively.

Copyright © 2026 Texas Instruments Incorporated

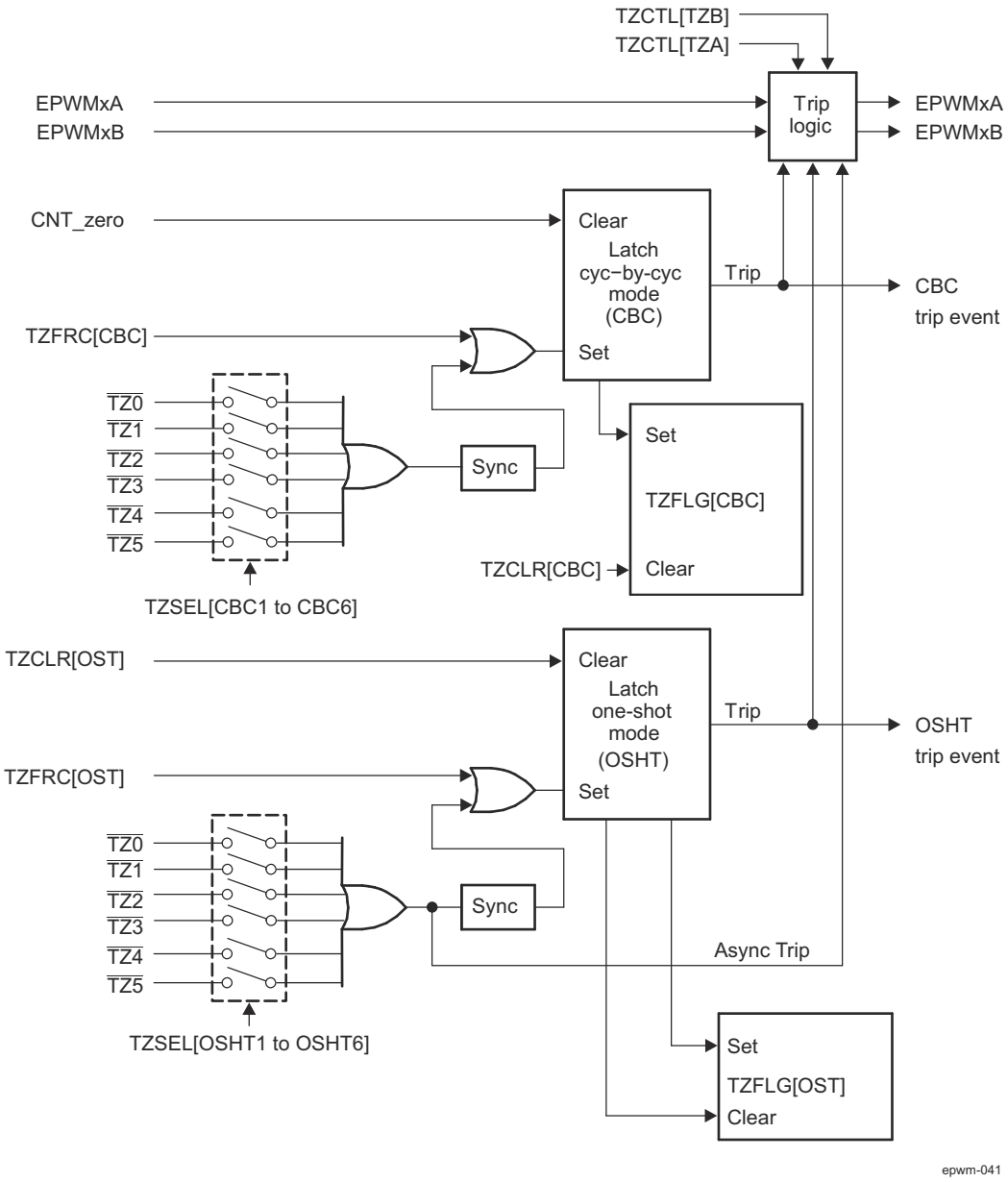


Figure 12-337. EPWM Trip-Zone Submodule Mode Control Logic

Copyright © 2026 Texas Instruments Incorporated

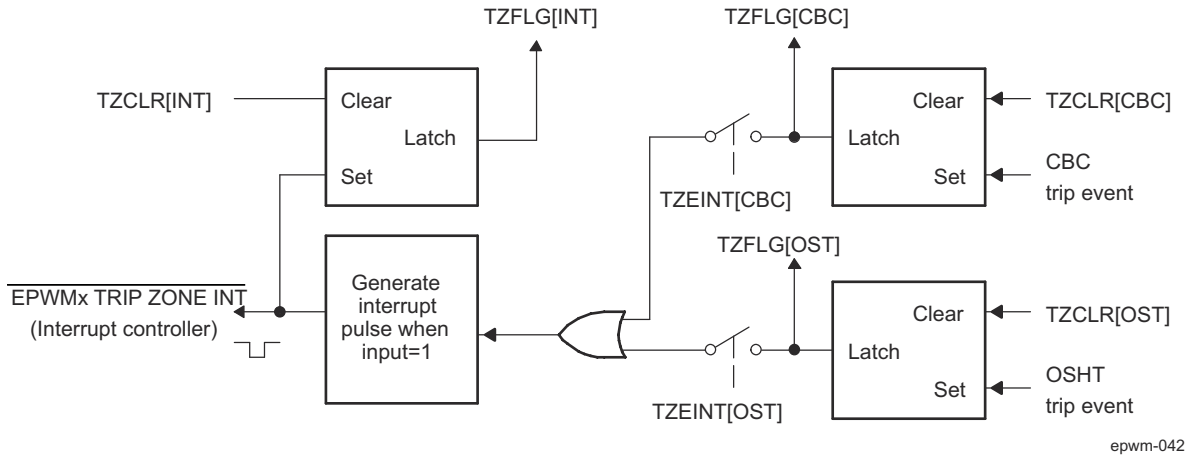


Figure 12-338. EPWM Trip-Zone Submodule Interrupt Logic

12.4.3.4.8 EPWM Event-Trigger (ET) Submodule

This section describes the Event-Trigger (ET) submodule in the PWM module.

12.4.3.4.8.1 Overview

Figure 12-339 shows the event-trigger (ET) submodule in the EPWM system. The event-trigger submodule manages the events generated by the time-base submodule and the counter-compare submodule to generate an aggregated interrupt request.

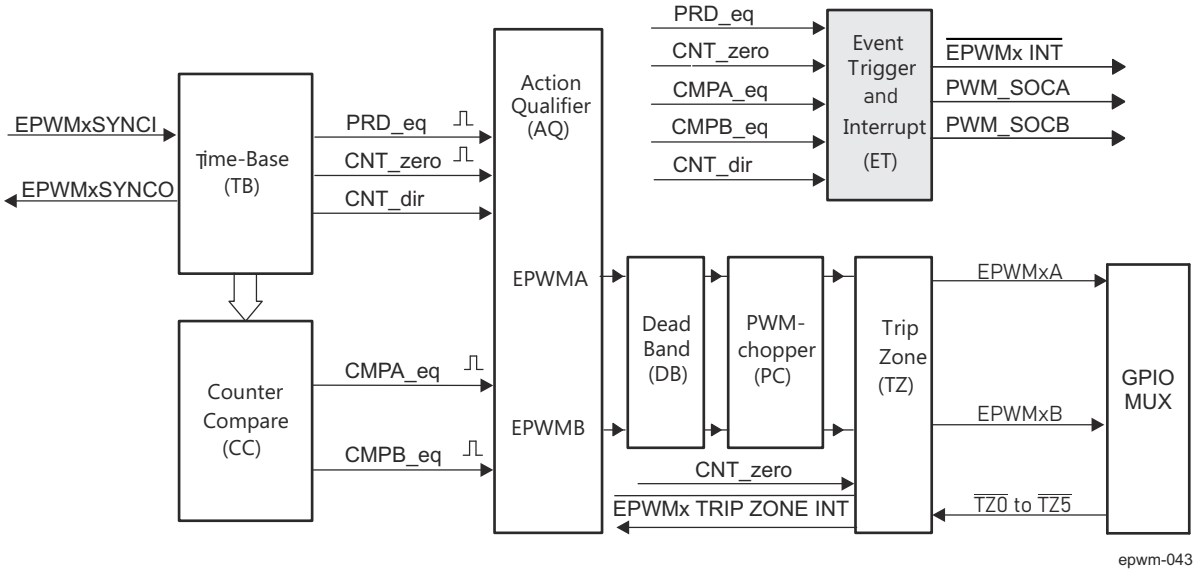


Figure 12-339. EPWM Event-Trigger Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base and counter-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags

Copyright © 2026 Texas Instruments Incorporated

12.4.3.4.8.2 Controlling and Monitoring the EPWM Event-Trigger Submodule

The key registers used to configure the event-trigger submodule are shown in Table 12-294:

Table 12-294. EPWM Event-Trigger Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
EPWM_ETSEL	Event-Trigger Selection Register	32h	No
EPWM_ETPS	Event-Trigger Prescale Register	34h	No
EPWM_ETFLG	Event-Trigger Flag Register	36h	No
EPWM_ETCLR	Event-Trigger Clear Register	38h	No
EPWM_ETFRC	Event-Trigger Force Register	3Ah	No

12.4.3.4.8.3 Operational Overview of the EPWM Event-Trigger Submodule

Each EPWM module has one interrupt request line as shown in Figure 12-340. Mapping interrupt lines to device host interrupt controllers is device specific and is covered in *EPWM Integration*.

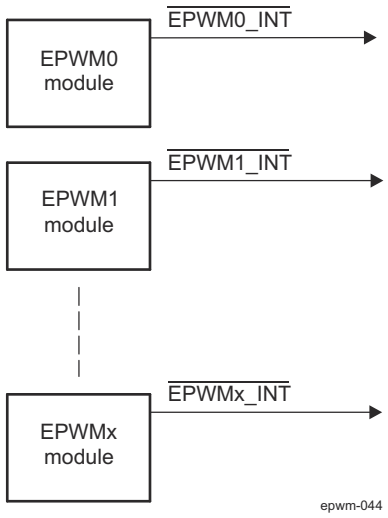


Figure 12-340. EPWM Event-Trigger Submodule Inter-Connectivity to Interrupt Controller

The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 12-341) and can be configured to prescale these events before issuing an Interrupt request. The event-trigger prescaling logic can issue Interrupt requests at:

- Every event
- Every second event
- Every third event

Copyright © 2026 Texas Instruments Incorporated

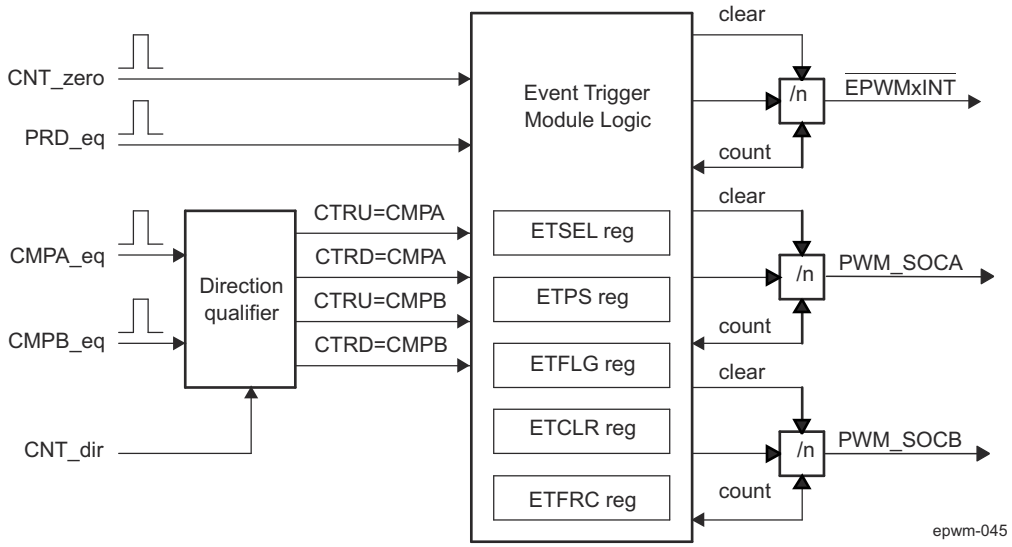


Figure 12-341. EPWM Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs

Copyright © 2026 Texas Instruments Incorporated

- ETSEL — This selects which of the possible events will trigger an interrupt.
- ETPS — This programs the event prescaling options previously mentioned.
- ETFLG — These are flag bits indicating status of the selected and prescaled events.
- ETCLR — These bits allow to clear the flag bits in the EPWM_ETFLG register via software.
- ETFRC — These bits allow software forcing of an event. Useful for debugging or software intervention.

A more detailed look at how the various register bits interact with the Interrupt is shown in Figure 12-342.

Figure 12-342 shows the event-trigger's interrupt generation logic. The interrupt-period (EPWM_ETPS[1-0] INTPRD) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

An interrupt cannot be generated on every fourth or more events.

Which event can cause an interrupt is configured by the interrupt selection (EPWM_ETSEL[2-0] INTSEL) bits. The event can be one of the following:

- Time-base counter equal to zero (EPWM_TBCNT[15-0] TBCNT = 0000h).
- Time-base counter equal to period (EPWM_TBCNT[15-0] TBCNT = EPWM_TBPRD[15-0] TBPRD).
- Time-base counter equal to the compare A register (EPWM_CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (EPWM_CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (EPWM_CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (EPWM_CMPB) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter (EPWM_ETPS[3-2] INTCNT) register bits. That is, when the specified event occurs the EPWM_ETPS[3-2] INTCNT bits are incremented until they reach the value specified by the EPWM_ETPS[1-0] INTPRD bit field. When EPWM_ETPS[3-2] INTCNT = EPWM_ETPS[1-0] INTPRD the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the interrupt controller.

When EPWM_ETPS[3-2] INTCNT reaches EPWM_ETPS[1-0] INTPRD, one of the following behaviors will occur:

- If interrupts are enabled, EPWM_ETSEL[3] INTEN = 1h and the interrupt flag is clear, EPWM_ETFLG[0] INT = 0h, then an interrupt pulse is generated and the interrupt flag is set, EPWM_ETFLG[0] INT = 1h, and the event counter is cleared EPWM_ETPS[3-2] INTCNT = 0h. The counter will begin counting events again.
- If interrupts are disabled, EPWM_ETSEL[3] INTEN = 0h, or the interrupt flag is set, EPWM_ETFLG[0] INT = 1h, the counter stops counting events when it reaches the period value EPWM_ETPS[3-2] INTCNT = EPWM_ETPS[1-0] INTPRD.
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[0] INT flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1h to the EPWM_ETFRC[0] INT bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0h, the counter is disabled and hence no events will be detected and the EPWM_ETFRC[0] INT bit is also ignored.

Copyright © 2026 Texas Instruments Incorporated

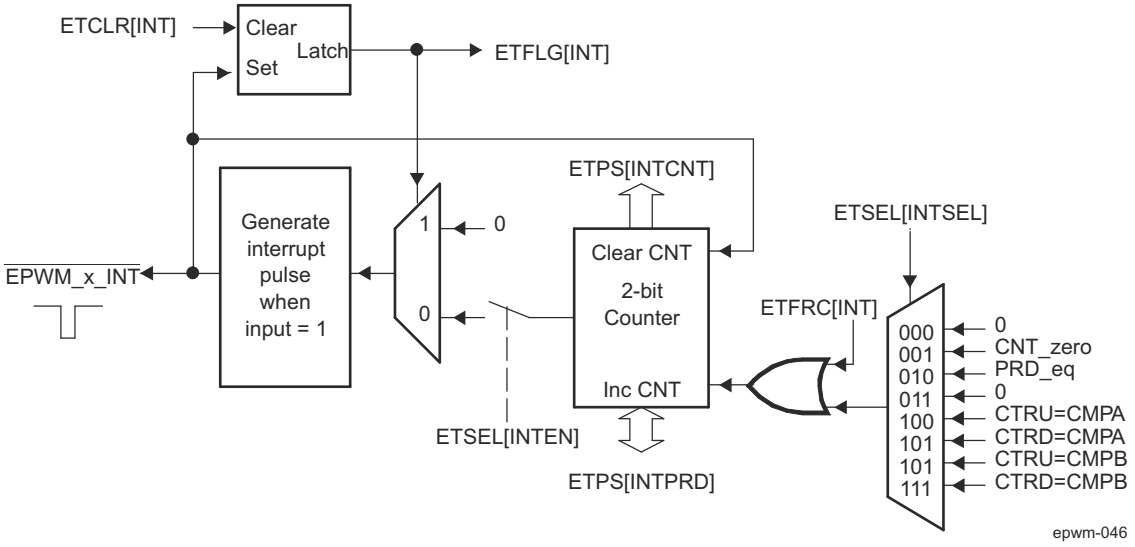


Figure 12-342. EPWM Event-Trigger Interrupt Generator

12.4.3.4.8.4 Operation Overview of the EPWM SOCx Pulse Generator

Figure 12-343 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The EPWM_ETPS[11-10] SOCACNT counter and EPWM_ETPS[9-8] SOCAPRD period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag EPWM_ETFLG[2] SOCA is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable EPWM_ETSEL[11] SOCAEN bit stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the EPWM_ETSEL[10-8] SOCASEL and EPWM_ETSEL[14-12] SOCBSEL bit field. The possible events are the same events that can be specified for the interrupt generation logic.

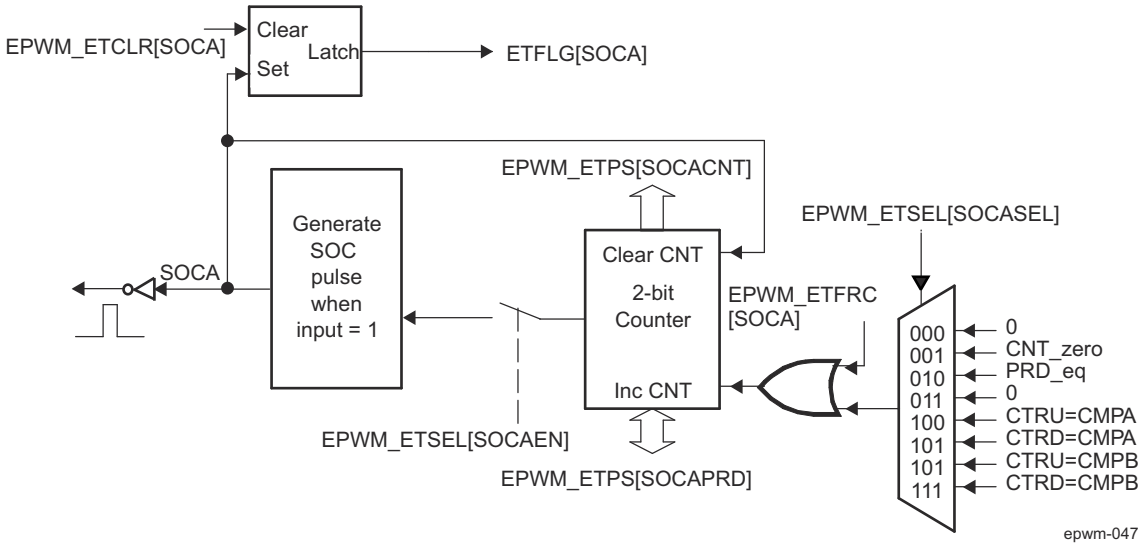


Figure 12-343. EPWM Event-Trigger SOCA Pulse Generator

Figure 12-344 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

Copyright © 2026 Texas Instruments Incorporated

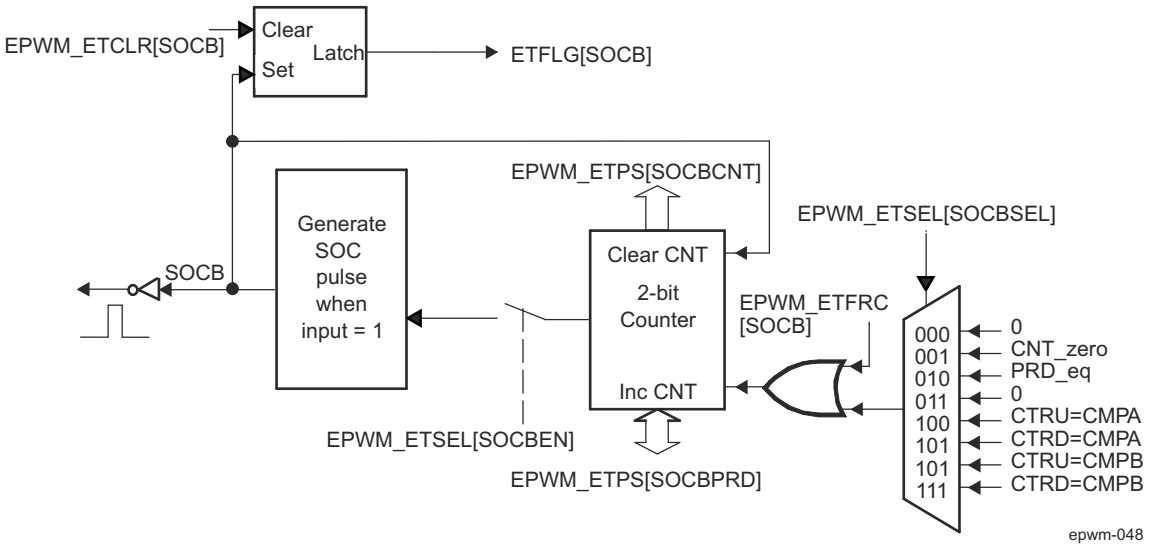


Figure 12-344. EPWM Event-Trigger SOCB Pulse Generator

12.4.3.4.9 EPWM Functional Register Groups

The Table 12-295 lists the groups of EPWM module registers according to their functionalities.

Table 12-295. EPWM Module Control and Status Registers Grouped by Submodule

Register Name	Offset	Size (×16)	Shadow	Register Description
Time-Base (TB) Submodule Registers				
EPWM_TBCTL	0h	1	No	Time-Base Control Register
EPWM_TBSTS	2h	1	No	Time-Base Status Register
EPWM_TBPHS	6h	1	No	Time-Base Phase Register
EPWM_TBCNT	8h	1	No	Time-Base Counter Register
EPWM_TBPRD	Ah	1	Yes	Time-Base Period Register
Counter-Compare (CC) Submodule Registers				
EPWM_CMPCTL	Eh	1	No	Counter-Compare Control Register
EPWM_CMPA	12h	1	Yes	Counter-Compare A Register
EPWM_CMPB	14h	1	Yes	Counter-Compare B Register
Action-Qualifier (AQ) Submodule Registers				
EPWM_AQCTLA	16h	1	No	Action-Qualifier Control Register for Output A (EPWMxA)
EPWM_AQCTLB	18h	1	No	Action-Qualifier Control Register for Output B (EPWMxB)
EPWM_AQSFRC	1Ah	1	No	Action-Qualifier Software Force Register
EPWM_AQCSFRC	1Ch	1	Yes	Action-Qualifier Continuous S/W Force Register Set
Dead-Band (DB) Generator Submodule Registers				
EPWM_DBCTL	1Eh	1	No	Dead-Band Generator Control Register
EPWM_DBRED	20h	1	No	Dead-Band Generator Rising Edge Delay Count Register
EPWM_DBFED	22h	1	No	Dead-Band Generator Falling Edge Delay Count Register
Trip-Zone (TZ) Submodule Registers				
EPWM_TZSEL	24h	1	No	Trip-Zone Select Register(1)
EPWM_TZCTL	28h	1	No	Trip-Zone Control Register (1)
EPWM_TZEINT	2Ah	1	No	Trip-Zone Enable Interrupt Register (1)

Copyright © 2026 Texas Instruments Incorporated

Table 12-295. EPWM Module Control and Status Registers Grouped by Submodule (continued)

Register Name	Offset	Size (×16)	Shadow	Register Description
EPWM_TZFLG	2Ch	1	No	Trip-Zone Flag Register (1)
EPWM_TZCLR	2Eh	1	No	Trip-Zone Clear Register (1)
EPWM_TZFRC	30h	1	No	Trip-Zone Force Register (1)
Event-Trigger (ET) Submodule Registers				
EPWM_ETSEL	32h	1	No	Event-Trigger Selection Register
EPWM_ETPS	34h	1	No	Event-Trigger Pre-Scale Register
EPWM_ETFLG	36h	1	No	Event-Trigger Flag Register
EPWM_ETCLR	38h	1	No	Event-Trigger Clear Register
EPWM_ETFRC	3Ah	1	No	Event-Trigger Force Register
PWM-Chopper Submodule Registers				
EPWM_PCCTL	3Ch	1	No	PWM-Chopper Control Register

(1) EALLOW protected registers.

12.4.3.4.10 Proper EPWM Interrupt Initialization Procedure

When the EPWM peripheral clock is enabled it is possible that interrupt flags may be set due to spurious events as the EPWM registers not being properly initialized. The proper procedure for initializing the EPWM peripheral is:

1. Disable global interrupts (CPU INTM flag)
2. Disable EPWM interrupts
3. Initialize peripheral registers
4. Clear any spurious EPWM flags
5. Enable EPWM interrupts
6. Enable global interrupts

Copyright © 2026 Texas Instruments Incorporated

12.4.4 Enhanced Quadrature Encoder Pulse (EQEP) Module

This section describes the Enhanced Quadrature Encoder Pulse (EQEP) module in the device.

12.4.4.1 EQEP Overview

The Enhanced Quadrature Encoder Pulse (EQEP) peripheral is used for direct interface with a linear or rotary incremental encoder to get position, direction and speed information from a rotating machine for use in high performance motion and position control system. The disk of an incremental encoder is patterned with a single track of slots patterns, as shown in Figure 12-345. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position and zero reference.

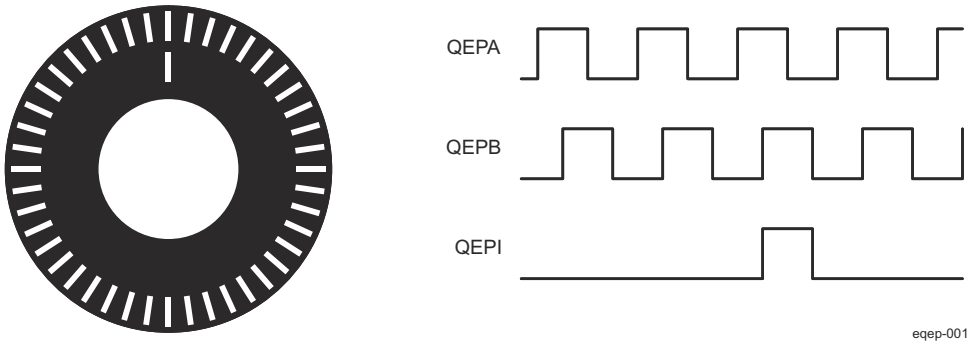


Figure 12-345. Optical Encoder Disk

To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90 degrees out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vise versa as shown in Figure 12-346.

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 kHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Copyright © 2026 Texas Instruments Incorporated

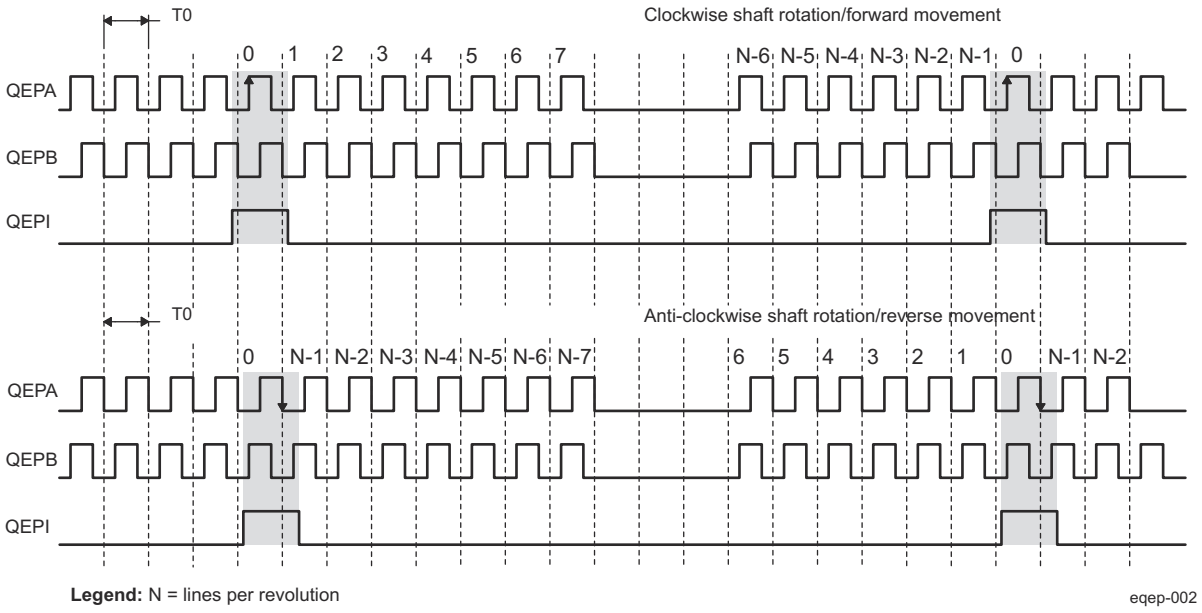


Figure 12-346. QEP Encoder Output Signal for Forward/Reverse Movement

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 12-347. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

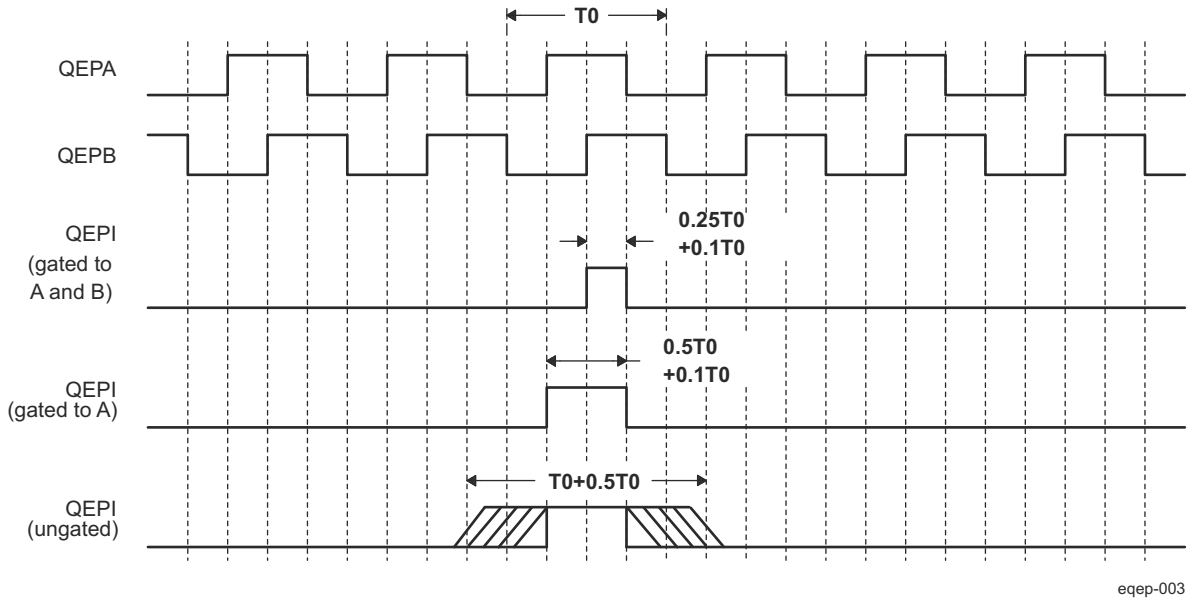


Figure 12-347. Index Pulse Example

Some typical applications for rotary encoders range from fax machines to motor controllers, and even robot localization. Rotary sensors are fundamental to the robotics and motion control systems found today. The optical shaft encoder can be used to improve a robot in various ways. The encoder can measure rotational distance traveled and speed, which can be used to monitor, for example, the angular position of a robot gripper arm or the speed of a robot.

Copyright © 2026 Texas Instruments Incorporated

General Issues: Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$V(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \tag{34}$$

$$V(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \tag{35}$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k - 1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k - 1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 34 is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity [x(k) - x(k-1)] is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant 1/T (where T is the constant time between unit time events and is known in advance).

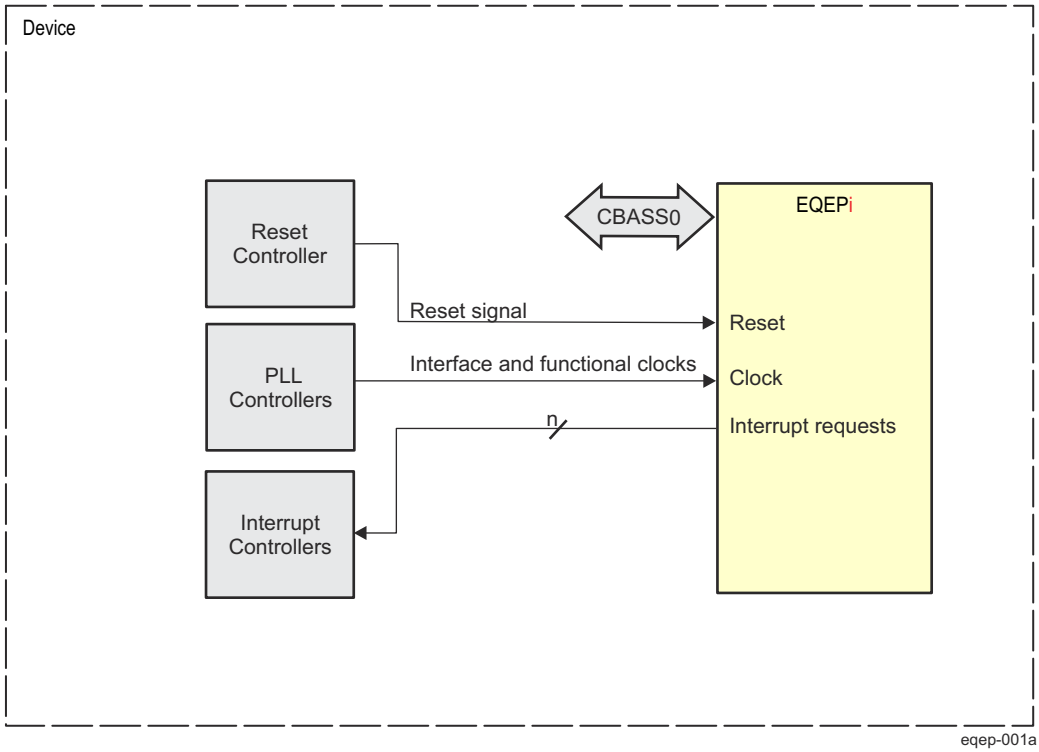
Estimation based on Equation 34 has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example, 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, Equation 35 provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. Equation 35 can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation of Equation 34. A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 35 at low speed and have the software switch over to Equation 34 when the motor speed rises above some specified threshold.

Figure 12-348 shows the EQEP module overview.

Copyright © 2026 Texas Instruments Incorporated



A. i = 0 to number of instances - 1.

Figure 12-348. EQEP Overview

12.4.4.1.1 EQEP Features

The EQEP module includes the following features:

- Input synchronization
- Three stage/six stage digital noise filter
- Quadrature decoder unit
- Position counter and control unit for position measurement
- Quadrature edge capture unit for low-speed measurement
- Unit time base for speed and frequency measurement
- Watchdog timer for detecting stalls
- EQEP inputs (A/B/INDEX and STROBE) are available at chip level
- EQEP phase error output is also available. The status of the phase error can be observed by software through the MAIN_CTRL_MMR_CFG0_EQEP_STAT register in the CTRL_MMR0 module.

Copyright © 2026 Texas Instruments Incorporated

- Counting modes:
  - Quadrature
  - Clockwise / Counter Clockwise
  - Count / Direction
- Start of Convert input for on-chip Strobe
- EQEP internal strobe (EQEP Strobe input is logically ORed with EQEP A and B inputs) may be used to:
  - Initialize the Position Counter with a non-zero value (for example, due to a limit switch input becoming active)
  - Snapshot the Position Counter into the EQEP_QPOSSLAT register
  - Gate the EQEP Index input preventing it from resetting the Position Counter

12.4.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.4.4.2 EQEP Environment

12.4.4.2.1 EQEP I/O Interface

The EQEPi (where i = 0 to 2) module is hereinafter referred to as EQEP module.

This section describes the EQEP external connections (environment).

Figure 12-349 shows the EQEP interface signals.

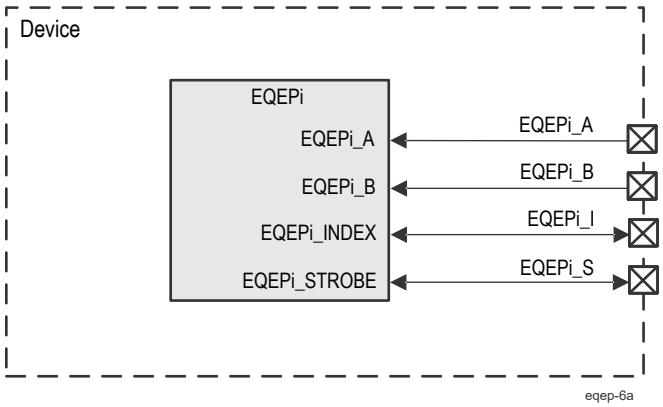


Figure 12-349. EQEP External Interface I/Os

Table 12-296 describes the EQEP I/O signals.

Table 12-296. EQEP I/O Signals

Module Pin	Device Level Signal Name	I/O Type[1]	Description	Module Pin Reset Value[2]
		EQEPi[4]		
EQEPi[4]_A	EQEPi[4]_A	I	EQEPi[4] Quadrature input A	HiZ
EQEPi[4]_B	EQEPi[4]_B	I	EQEPi[4] Quadrature input B	HiZ
EQEPi[4]_INDEX	EQEPi[4]_I	I/O[3]	EQEPi[4] Index input/output	HiZ
EQEPi[4]_STROBE	EQEPi[4]_S	I/O[3]	EQEPi[4] Strobe input/output	HiZ

(1) I = Input; O = Output
(2) HiZ = High Impedance
(3) Output functionality is not supported. Only inputs are pinned out. For more information see *Device Specific EQEP Features*.
(4) i represents an EQEP instance. See the device datasheet for available domains and EQEP instances

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables Pin Attributes and Pin Multiplexing in the device-specific Datasheet.

12.4.4.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4 EQEP Functional Description

This section provides the EQEP functional description and corresponding functional details about EQEPx inputs.

Note

Multiple identical EQEP modules can be contained in a system. For actual number of EQEP modules integrated in the device, refer to *EQEP Integration*. The letter x within a signal or module name is used to indicate a generic EQEP instance on a device. For example, output interrupt request, EQEP0_EQEP_INT_OUT_0 belongs to EQEP0, EQEP1_EQEP_INT_OUT_0 belongs to EQEP1, and so forth.

The EQEP peripheral contains the following major functional units (as shown in Figure 12-350):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Three stage/six stage digital noise filter
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG).

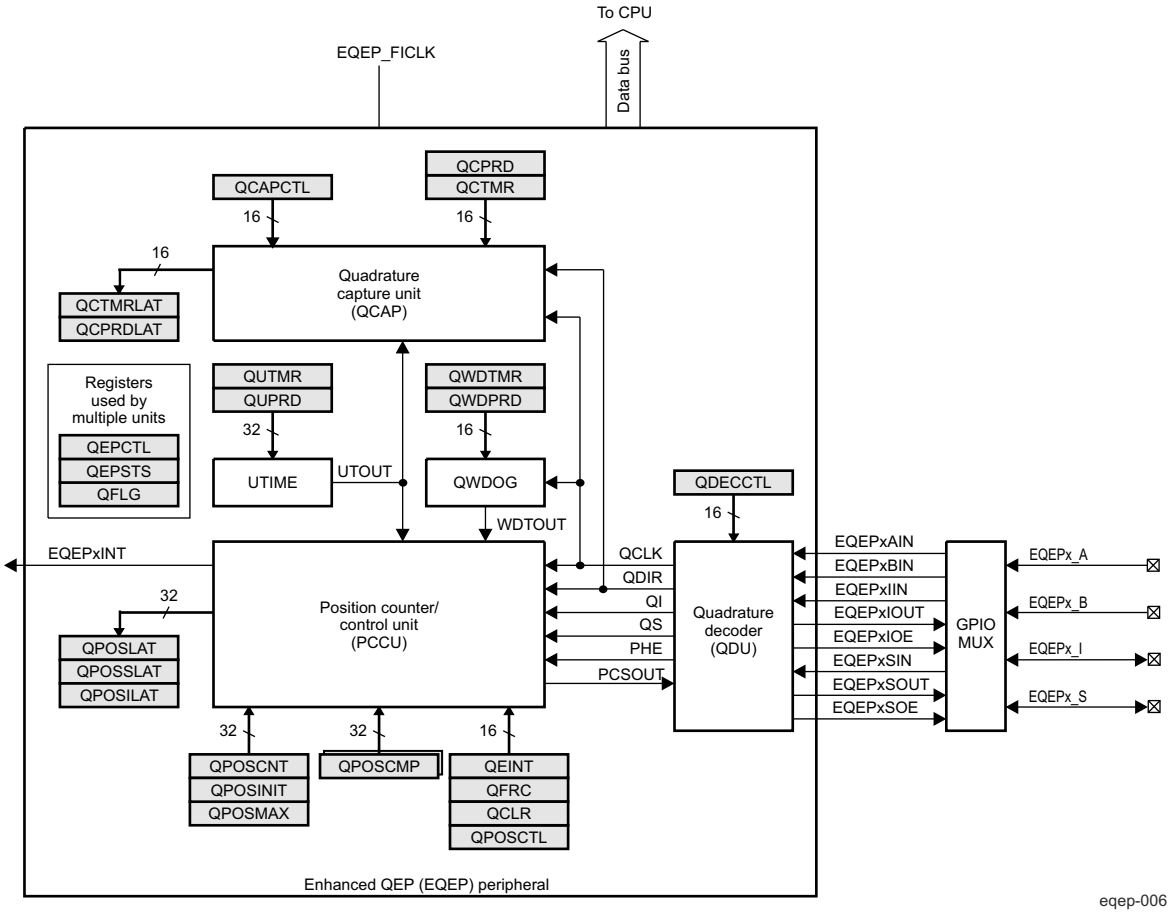


Figure 12-350. Functional Block Diagram of the EQEP Peripheral

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.1 EQEP Inputs

The EQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input.

- EQEPx_A and EQEPx_B: These two pins can be used in quadrature-clock mode or direction-count mode.
  - Quadrature-clock mode: The EQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of EQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.
  - Direction-count mode: In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The EQEPx_A pin provides the clock input and the EQEPx_B pin provides the direction input.
- EQEPx_I: Index or Zero Marker: The EQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the EQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.
- EQEPx_S: Strobe Input: This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.2 EQEP Quadrature Decoder Unit (QDU)

Figure 12-351 shows a functional block diagram of the QDU.

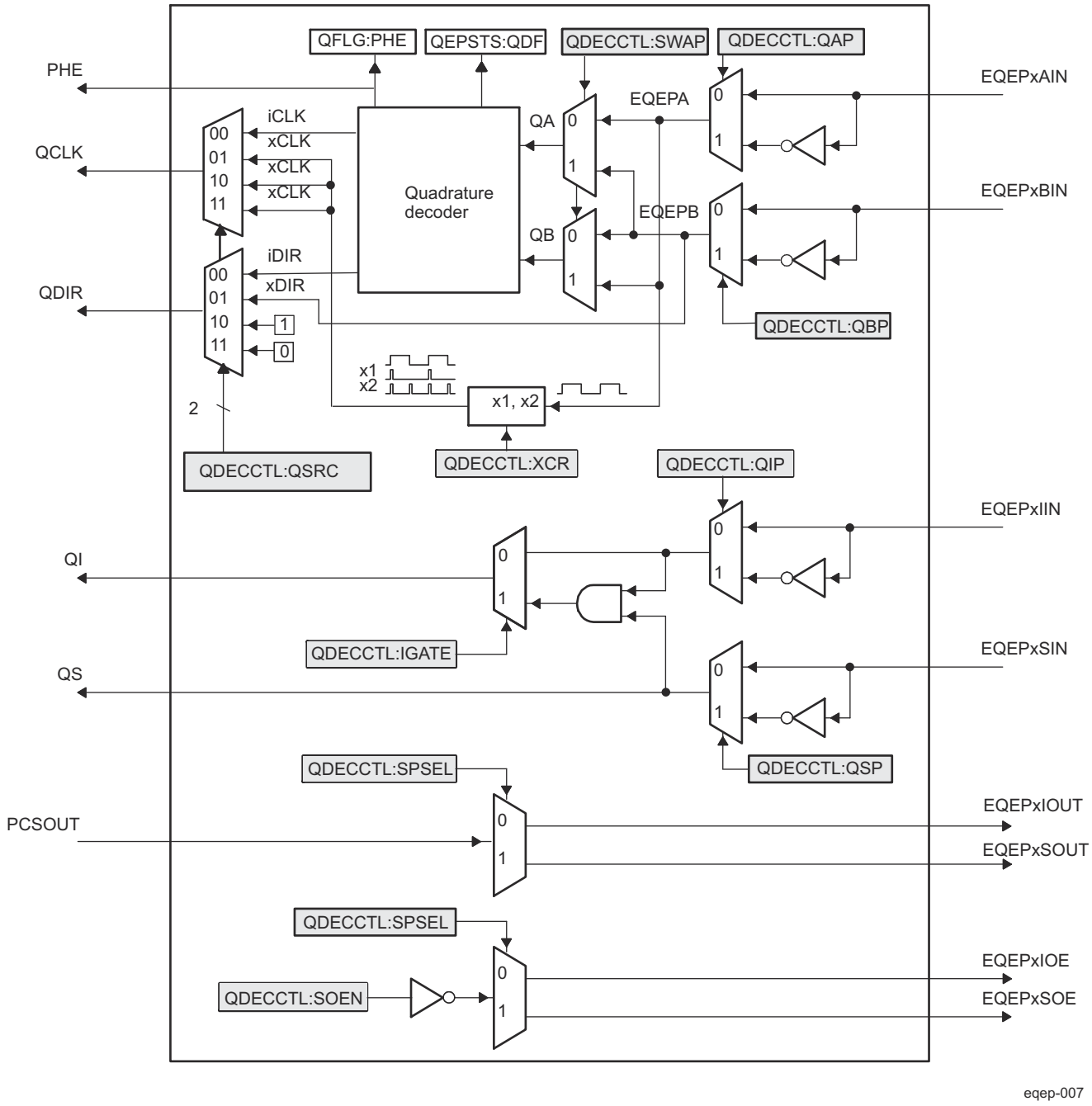


Figure 12-351. Functional Block Diagram of Decoder Unit

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.2.1 EQEP Position Counter Input Modes

Clock and direction input to position counter is selected using the QSRC bit in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL), based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode.

12.4.4.4.2.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

Direction Decoding

The direction decoding logic of the EQEP circuit determines which one of the sequences (EQEPA, EQEPB) is the leading sequence and accordingly updates the direction information in the QDF bit in the EQEP status register (EQEP_QEP_STS_CT). Table 12-297 and Figure 12-352 show the direction decoding logic in truth table and state machine form. Both edges of the EQEPA and EQEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the EQEP logic is four times that of each input sequence. Figure 12-353 shows the direction decoding and clock generation from the EQEP input signals.

Phase Error Flag

In normal operating conditions, quadrature inputs EQEPA and EQEPB will be 90 degrees out of phase. The phase error flag (QPEI_FLG) is set in the EQEP_QINT_EN_FLG register when edge transition is detected simultaneously on the EQEPA and EQEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 12-352 are invalid transitions that generate a phase error.

Count Multiplication

The EQEP position counter provides 4× times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both EQEP input clocks (EQEPA and EQEPB) as shown in Figure 12-353.

Reverse Count

In normal quadrature count operation, EQEPA input is fed to the QA input of the quadrature decoder and the EQEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL). This will swap the input to the quadrature decoder thereby reversing the counting direction.

Table 12-297. Quadrature Decoder Truth Table

Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment (+1)
	QB↓	DOWN	Decrement (-1)
	QA↓	TOGGLE	Increment (+1) or Decrement (-1)
QA↓	QB↓	UP	Increment (+1)
	QB↑	DOWN	Decrement(-1)
	QA↑	TOGGLE	Increment (+1) or Decrement (-1)
QB↑	QA↑	DOWN	Increment (+1)
	QA↓	UP	Decrement (-1)
	QB↓	TOGGLE	Increment (+1) or Decrement (-1)
QB↓	QA↓	DOWN	Increment (+1)
	QA↑	UP	Decrement (-1)
	QB↑	TOGGLE	Increment (+1) or Decrement (-1)

Copyright © 2026 Texas Instruments Incorporated

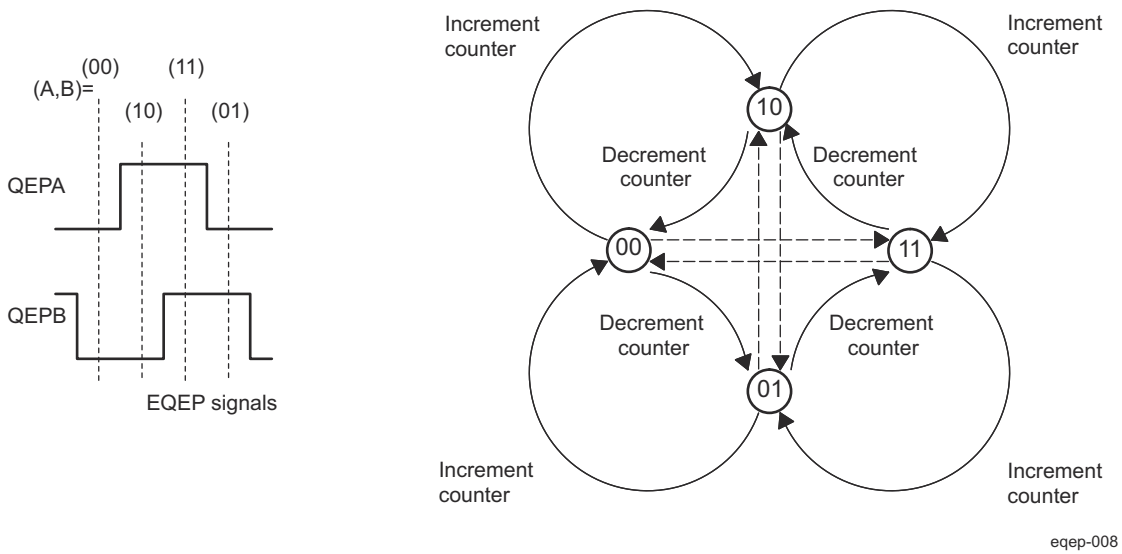


Figure 12-352. Quadrature Decoder State Machine

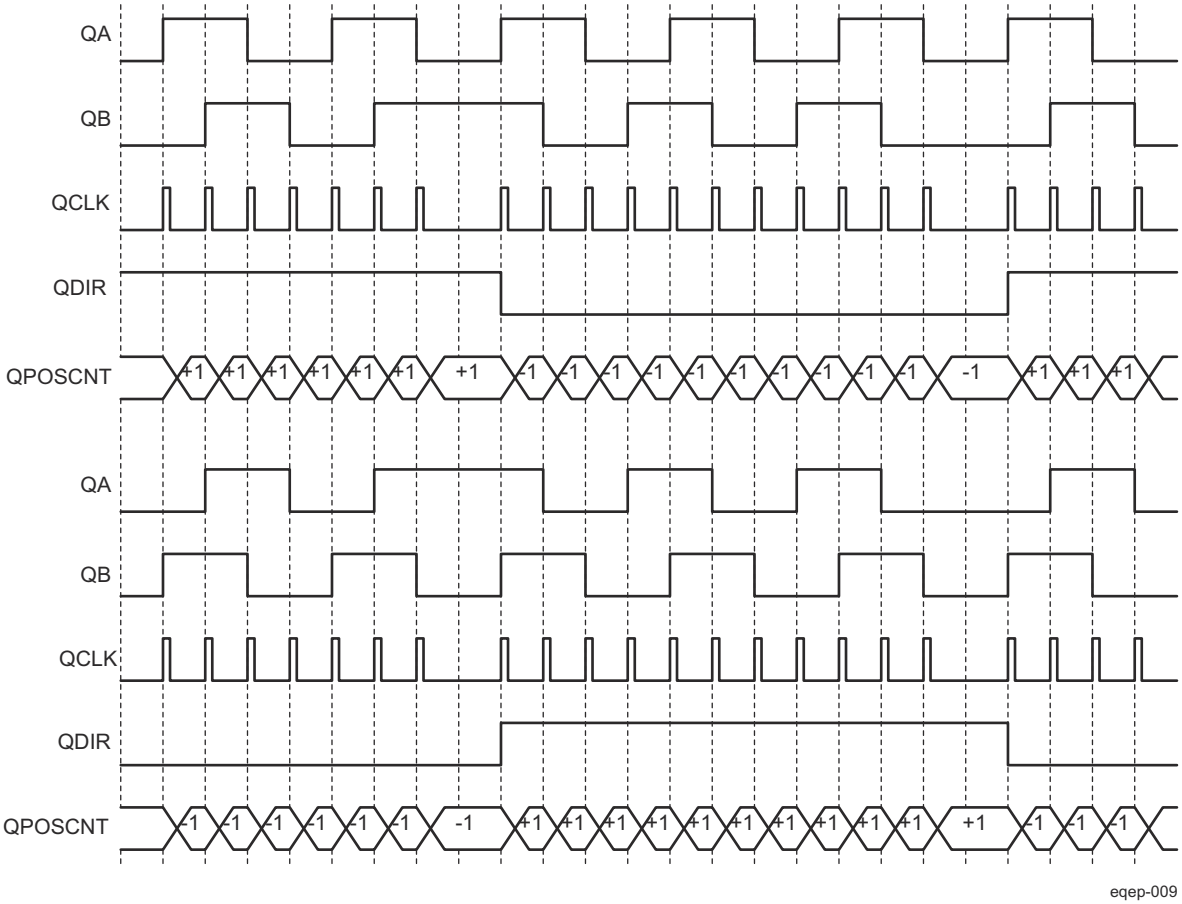


Figure 12-353. Quadrature-clock and Direction Decoding

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.2.1.2 EQEP Direction-count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. EQEPA input will provide the clock for position counter and the EQEPB input will have the direction information. The position counter is incremented on every rising edge of a EQEPA input when the direction input is high and decremented when the direction input is low.

12.4.4.4.2.1.3 EQEP Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the EQEPA input. Setting of the XCR bit in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL) enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2× factor.

12.4.4.4.2.1.4 EQEP Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the EQEPA input. Setting of the XCR bit in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL) enables clock generation to the position counter on both edges of a EQEPA input, thereby increasing the measurement resolution by 2× factor.

12.4.4.4.2.2 EQEP Input Polarity Selection

Each EQEP input can be inverted using the in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL[8-5]) control bits. As an example, setting of the QIP bit in EQEP_QDEC_QEP_CTL inverts the index input.

12.4.4.4.2.3 EQEP Position-Compare Sync Output

The EQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (EQEP_QPOSCNT) and the position-compare register (EQEP_QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the SOEN bit in the EQEP decoder and control register (EQEP_QDEC_QEP_CTL) enables the position-compare sync output and the SPSEL bit in EQEP_QDEC_QEP_CTL selects either an EQEP index pin or an EQEP strobe pin.

12.4.4.4.3 EQEP Position Counter and Control Unit (PCCU)

The position counter and control unit provides two configuration registers (EQEP_QDEC_QEP_CTL and EQEP_QCAP_QPOS_CTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

12.4.4.4.3.1 EQEP Position Counter Operating Modes

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement).

Copyright © 2026 Texas Instruments Incorporated

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX bifield value in EQEP_QPOSMAX register on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/ underflow in EQEP_QINT_EN_FLG register.

12.4.4.4.3.1.1 EQEP Position Counter Reset on Index Event (EQEP_QDEC_QEP_CTL[29-28] PCRM] = 0b00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next EQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the EQEP_QPOSMAX register on the next EQEP clock.

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP_QEP_STS_CT[1] FIMF) and direction on the first index event marker (EQEP_QEP_STS_CT[6] FIDF) in EQEP_QEP_STS_CT registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of EQEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of EQEPB for the forward rotation and on the rising edge of EQEPB for the reverse rotation as shown in Figure 12-354.

The position-counter value is latched to the EQEP_QPOSILAT register and direction information is recorded in the EQEP_QEP_STS_CT[4] QDLF bit on every index event marker. The position-counter error flag (EQEP_QEP_STS_CT[0] PCEF) and error interrupt flag (EQEP_QINT_EN_FLG[17] PCEI_FLG) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (EQEP_QEP_STS_CT[0] PCEF) is updated on every index event marker and an interrupt flag (EQEP_QINT_EN_FLG[17] PCEI_FLG) will be set on error that can be cleared only through software.

The index event latch configuration EQEP_QDEC_QEP_CTL[21-20] IEL bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

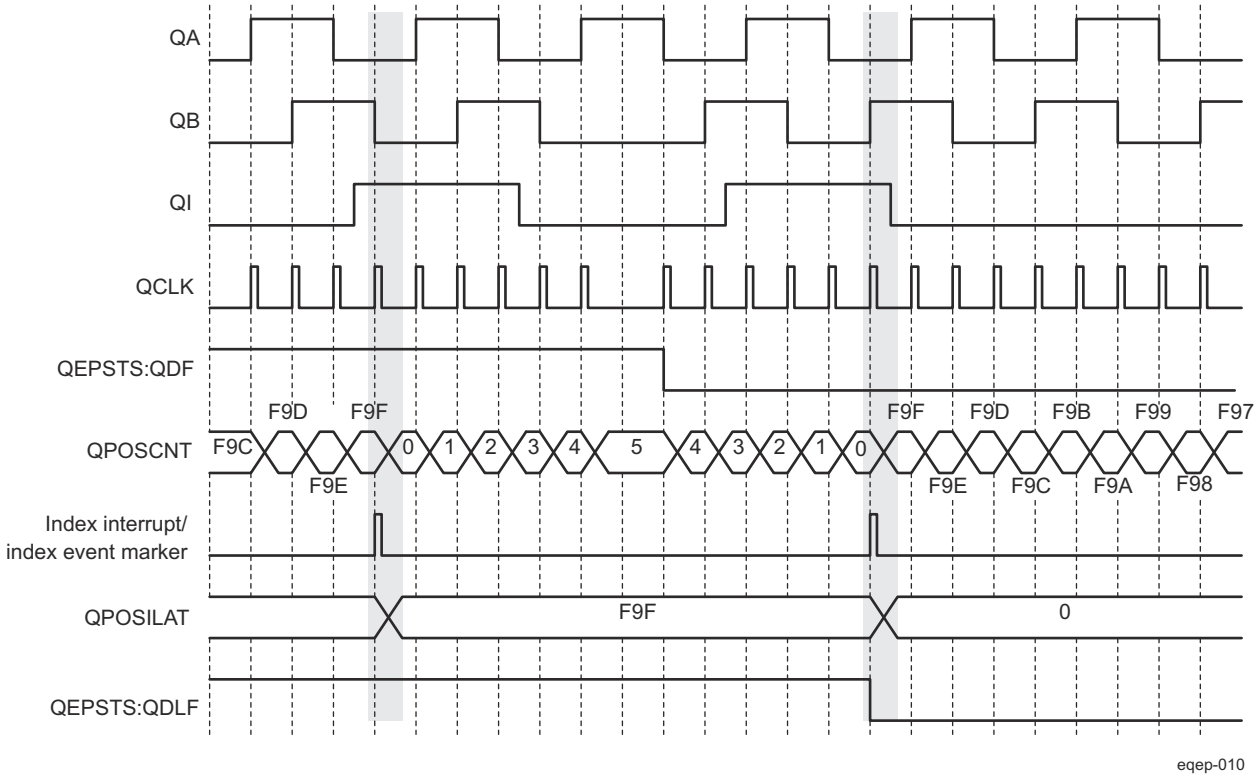


Figure 12-354. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or F9Fh)

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.3.1.2 EQEP Position Counter Reset on Maximum Position (EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b01)

If the position counter is equal to QPOSMAX (in EQEP_QPOSMAX register), then the position counter is reset to 0 on the next EQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to 0, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 12-355 shows the position counter reset operation in this mode.

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP_QEP_STS_CT[1] FIMF) and direction on the first index event marker (EQEP_QEP_STS_CT[6] FIDF); it also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for the software index marker (EQEP_QDEC_QEP_CTL[21-20] IEL= 0b11).

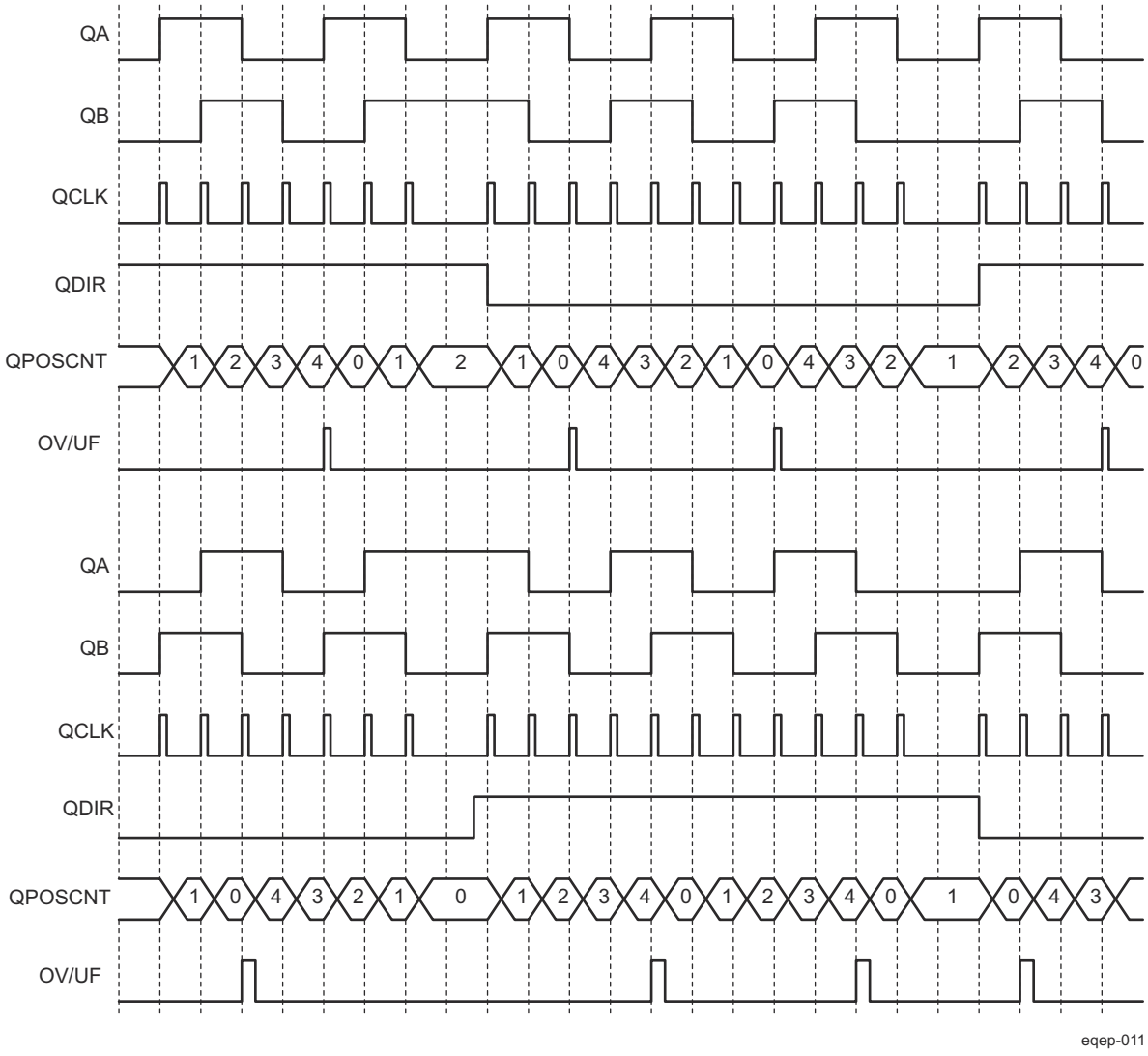


Figure 12-355. Position Counter Underflow/Overflow (QPOSMAX = 4)

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.3.1.3 Position Counter Reset on the First Index Event (EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next EQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the EQEP_QPOSMAX register on the next EQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in Section 12.4.4.4.3.1.2.

First index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP_QEP_STS_CT[1] FIMF) and direction on the first index event marker (EQEP_QEP_STS_CT[6] FIDF) in EQEP_QEP_STS_CT registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for software index marker (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b11).

12.4.4.4.3.1.4 Position Counter Reset on Unit Time out Event (EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b11)

In this mode, the QPOSCNT value is latched to the EQEP_QPOSLAT register and then the QPOSCNT field is reset (to 0 or the QPOSMAX value in the EQEP_QPOSMAX register, depending on the direction mode selected by EQEP_QDEC_QEP_CTL[15-14] QSRC bits on a unit time event). This is useful for frequency measurement.

12.4.4.4.3.2 EQEP Position Counter Latch

The EQEP index and strobe input can be configured to latch the position counter QPOSCNT (EQEP_QPOSCNT) into QPOSILAT (EQEP_QPOSILAT register) and QPOSSLAT (EQEP_QPOSSLAT register) bitfields, respectively, on occurrence of a definite event on these pins.

12.4.4.4.3.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b01 and EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b10 modes).

In such cases, the EQEP position counter can be configured to latch on the following events and direction information is recorded in the EQEP_QEP_STS_CT[4] QDLF bit on every index event marker.

- Latch on Rising edge (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b01)
- Latch on Falling edge (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b10)
- Latch on Index Event Marker (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (EQEP_QINT_EN_FLG[26] IELI_FLG) is set when the position counter is latched to the EQEP_QPOSILAT register. The index event latch configuration bits are ignored when EQEP_QDEC_QEP_CTL[29-28] PCRM = 0b00.

When Position counter reset on an index event mode is selected through EQEP_QDEC_QEP_CTL[29-28] PCRM bit field (value: 0h), EQEP_QDEC_QEP_CTL[21-20] IEL bit field must be configured to 0h and position counter value is latched into the EQEP_QPOSILAT[31-0] QPOSILAT register for every index marker.

Latch on Rising Edge (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b01)	The position counter value (QPOSCNT) is latched to the EQEP_QPOSILAT register on every rising edge of an index input.
Latch on Falling Edge (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b10)	The position counter value (QPOSCNT) is latched to the EQEP_QPOSILAT register on every falling edge of index input.
Latch on Index Event Marker/Software Index Marker (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b11)	The first index marker is defined as the quadrature edge following the first index edge. The EQEP peripheral records the occurrence of the first index marker (EQEP_QEP_STS_CT[1] FIMF) and

Copyright © 2026 Texas Instruments Incorporated

direction on the first index event marker (EQEP_QEP_STS_CT[6] FIDF) in the EQEP_QEP_STS_CT registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b11).

Figure 12-356 shows the position counter latch using an index event marker.

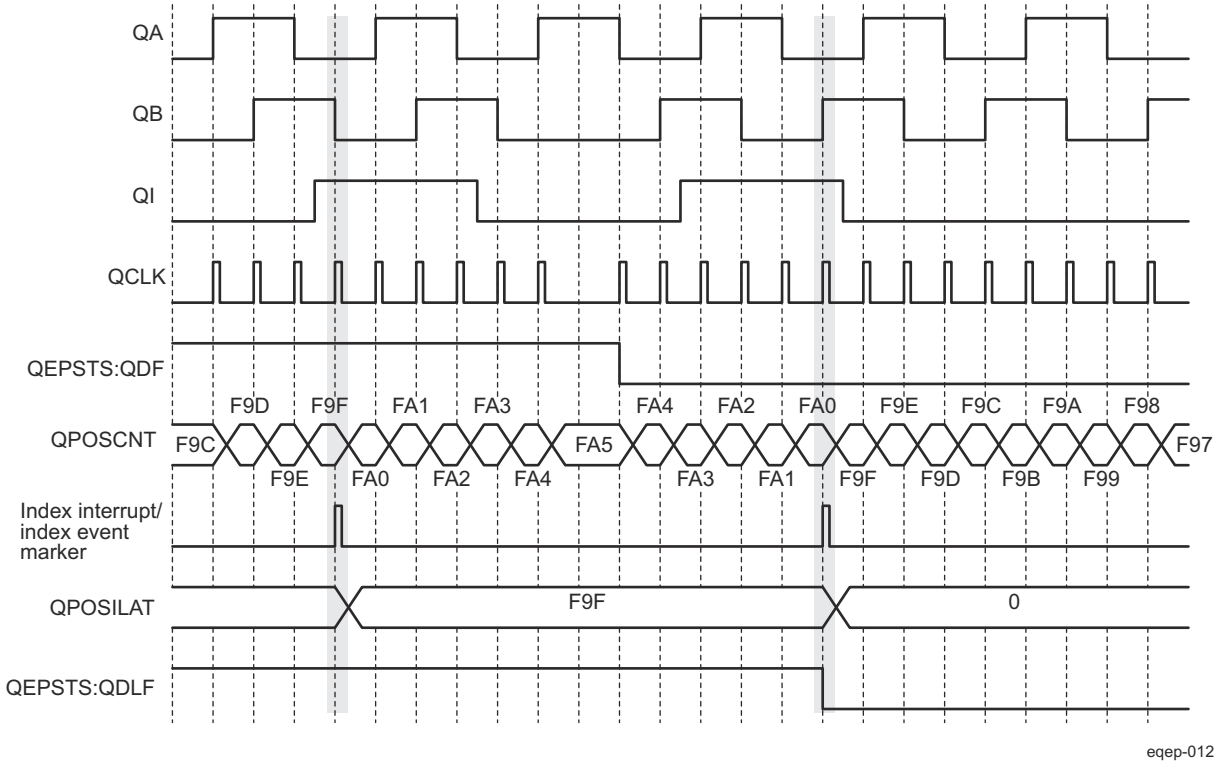


Figure 12-356. Software Index Marker for 1000-line Encoder (EQEP_QDEC_QEP_CTL[21-20] IEL = 0b01)

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.3.2.2 EQEP Strobe Event Latch

The position-counter value is latched to the EQEP_QPOSSLAT register on the rising edge of the strobe input (QCLK) by clearing the EQEP_QDEC_QEP_CTL[22] SEL bit. Latching on the falling edge of the strobe input (QCLK) can be done by inverting the strobe input using the EQEP_QDEC_QEP_CTL[5] QSP bit.

If the EQEP_QDEC_QEP_CTL[22] SEL bit is set, then the position counter value is latched to the EQEP_QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in Figure 12-357.

The strobe event latch interrupt flag (EQEP_QINT_EN_FLG[25] SELI_FLG) is set when the position counter is latched to the EQEP_QPOSSLAT register.

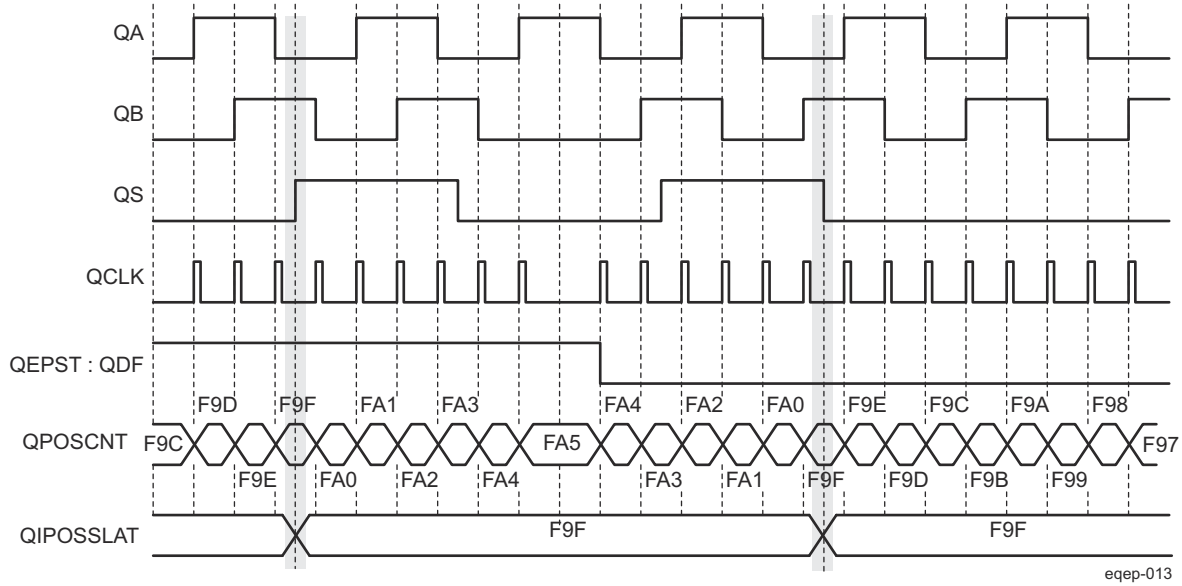


Figure 12-357. EQEP Strobe Event Latch (EQEP_QDEC_QEP_CTL[22] SEL = 0b1)

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.3.3 EQEP Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

Note

When all of the above events occur simultaneously, the sequence of priority is as follows: 1) Software Initialization, 2) strobe event initialization, 3) Index Event Initialization.

Index Event Initialization (IEI)

The EQEPx_I index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input.

If the EQEP_QDEC_QEP_CTL[25-24] IEI bits are 2h, then the position counter (EQEP_QPOSCNT) is initialized with a value in the EQEP_QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

The index event initialization interrupt flag (EQEP_QDEC_QEP_CTL[25-24] IEI) is set when the position counter is initialized with a value in the EQEP_QPOSINIT register.

If EQEP_QDEC_QEP_CTL[25-24] IEI bit field is configured with value 0h (default) or 1h, the index event initialization of position counter is disabled.

If EQEP_QDEC_QEP_CTL[25-24] IEI bit field is configured with value 3h, then the the position counter (EQEP_QPOSCNT) is initialized with a value in the EQEP_QPOSINIT register on the falling edge of strobe input signal.

Strobe Event Initialization (SEI)

If the EQEP_QDEC_QEP_CTL[27-26] SEI bits are 2h, then the position counter is initialized with a value in the EQEP_QPOSINIT register on the rising edge of strobe input.

If the EQEP_QDEC_QEP_CTL[27-26] SEI bits are 3h, then the position counter (EQEP_QPOSCNT) is initialized with a value in the EQEP_QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

The strobe event initialization interrupt flag (EQEP_QDEC_QEP_CTL[27-26] SEI) is set when the position counter is initialized with a value in the EQEP_QPOSINIT register.

If EQEP_QDEC_QEP_CTL[27-26] SEI bit field is configured with value 0h (default) or 1h, the strobe event initialization of position counter is disabled.

Software Initialization (SWI)

The position counter can be initialized in software by writing a 1h to the EQEP_QDEC_QEP_CTL[23] SWI bit, which will automatically be cleared after initialization.

12.4.4.4.3.4 EQEP Position-Compare Unit

The EQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 12-358 shows a diagram. The position-compare (EQEP_QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the EQEP_QCAP_QPOS_CTL[31] PCSHDW bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

Copyright © 2026 Texas Instruments Incorporated

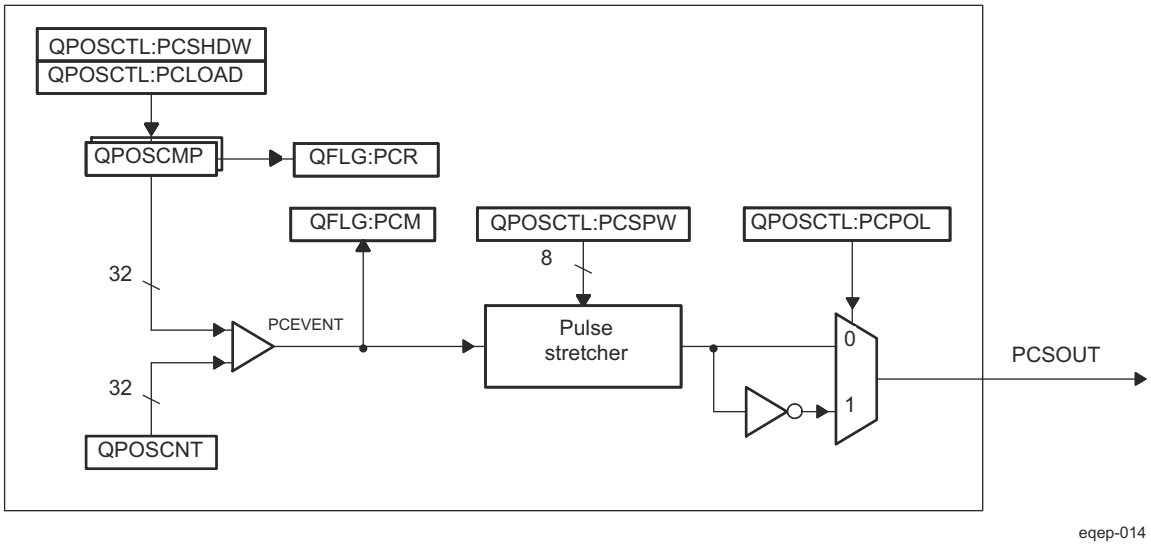


Figure 12-358. EQEP Position-compare Unit

In shadow mode, SW can configure the position-compare unit (EQEP_QCAP_QPOS_CTL[30] PCLOAD) to load the shadow register value into the active register on the following events and to generate the position-compare ready (EQEP_QINT_EN_FLG[23] PCRI_FLG) interrupt after loading.

- Load on compare match
- Load on position-counter zero event.

The position-compare match (EQEP_QINT_EN_FLG[24] PCMI_FLG) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (EQEP_QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

For example, if EQEP_QPOSCMP bitfield POSCMP = 0x2, the position-compare unit generates a position-compare event on the transition from 1 to 2 of the EQEP position counter for forward counting direction and on the transition from 3 to 2 of the EQEP position counter for reverse counting direction (see Figure 12-359).

Note

When EQEP_QCAP_QPOS_CTL[30] PCLOAD = 0h, the shadow register is loaded into the active register as soon as POSCNT becomes zero, and it should not generate another shadow load if the POSCNT continues to stay at zero

Copyright © 2026 Texas Instruments Incorporated

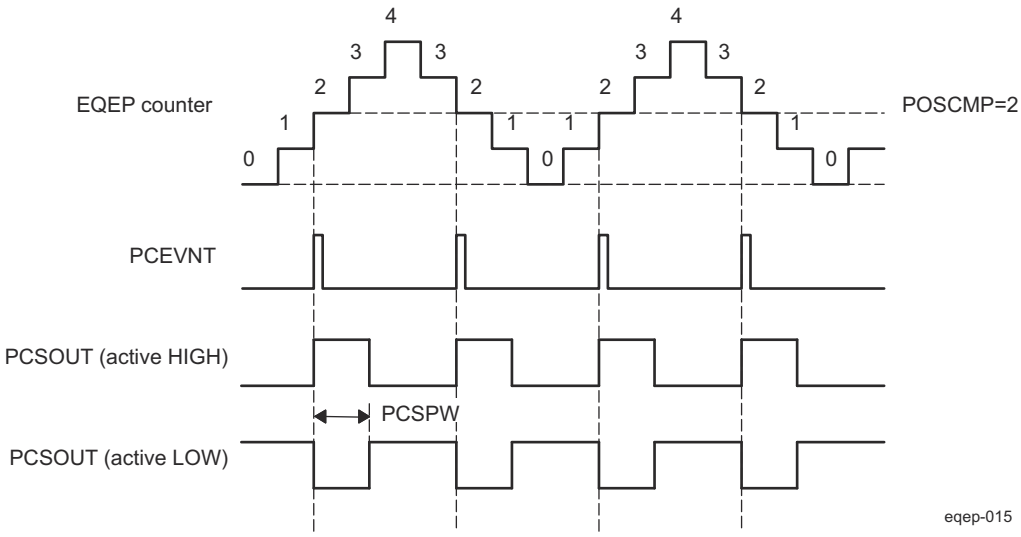


Figure 12-359. EQEP Position-compare Event Generation Points

The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 12-360.

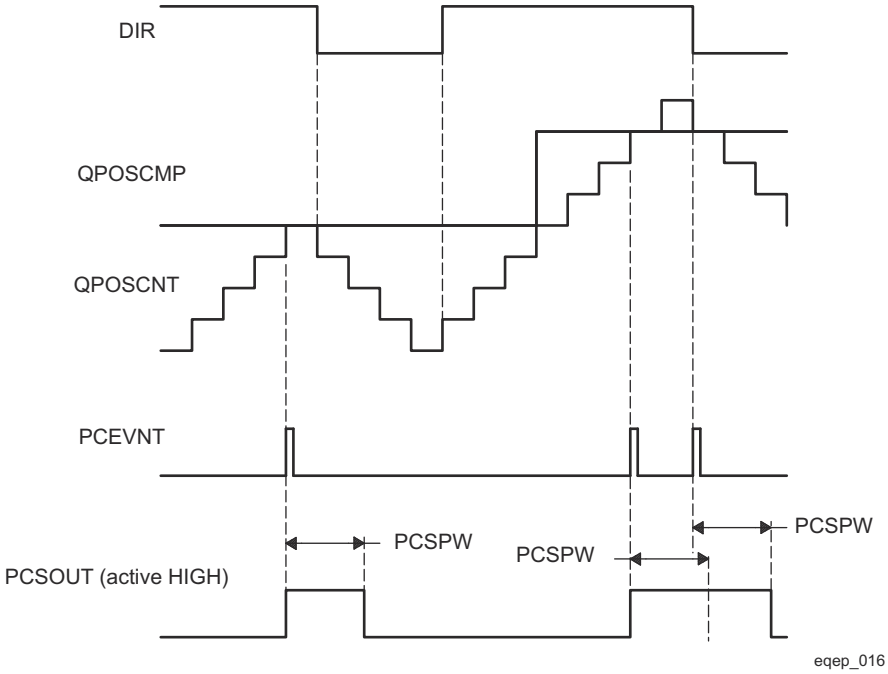


Figure 12-360. EQEP Position-compare Sync Output Pulse Stretcher

12.4.4.4.4 EQEP Edge Capture Unit

The EQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in Figure 12-361. This feature is typically used for low speed measurement using the following equation:

$$V(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (36)$$

Copyright © 2026 Texas Instruments Incorporated

where,

- X - Unit position is defined by integer multiple of quadrature edges (see Figure 12-362)
- ΔT - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The EQEP capture timer (QCTMR bitfield in EQEP_QCTMR register) runs from prescaled SYSCLKOUT and the prescaler is programmed by the EQEP_QCAP_QPOS_CTL[6-4] CCPS bits. The capture timer QCTMR value is latched into the capture period register (EQEP_QC_PRD_TLAT) on every unit position event and then the capture timer is reset.

Time measurement (ΔT) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the EQEP overflow error flag (EQEP_QEP_STS_CT[3] COEF) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (EQEP_QEP_STS_CT[2] CDEF).

Capture Timer (EQEP_QCTMR register) and Capture period register (EQEP_QC_PRD_TLAT) can be configured to latch on following events.

- CPU read of EQEP_QPOSCNT register
- Unit time-out event

If the EQEP_QDEC_QEP_CTL[18] QCLM bit is cleared, then the capture timer and capture period values are latched into the EQEP_QC_PRD_TLAT and EQEP_QCPRDLAT registers, respectively, when the CPU reads the position counter in EQEP_QPOSCNT.

Note

Capture timer and capture period values are not latched for an emulation read.

If the EQEP_QDEC_QEP_CTL[18] QCLM bit is set, then the position counter, capture timer, and capture period values are latched into the EQEP_QPOSLAT, EQEP_QC_PRD_TLAT and EQEP_QCPRDLAT registers, respectively, on unit time out.

Figure 12-363 shows the capture unit operation along with the position counter.

Note

The EQEP_QCAP_QPOS_CTL register should not be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLKOUT/4 to SYSCLKOUT/8, where SYSCLKOUT is equivalent to EQEPx_FICLK). The capture unit must be disabled before changing the prescaler.

Copyright © 2026 Texas Instruments Incorporated

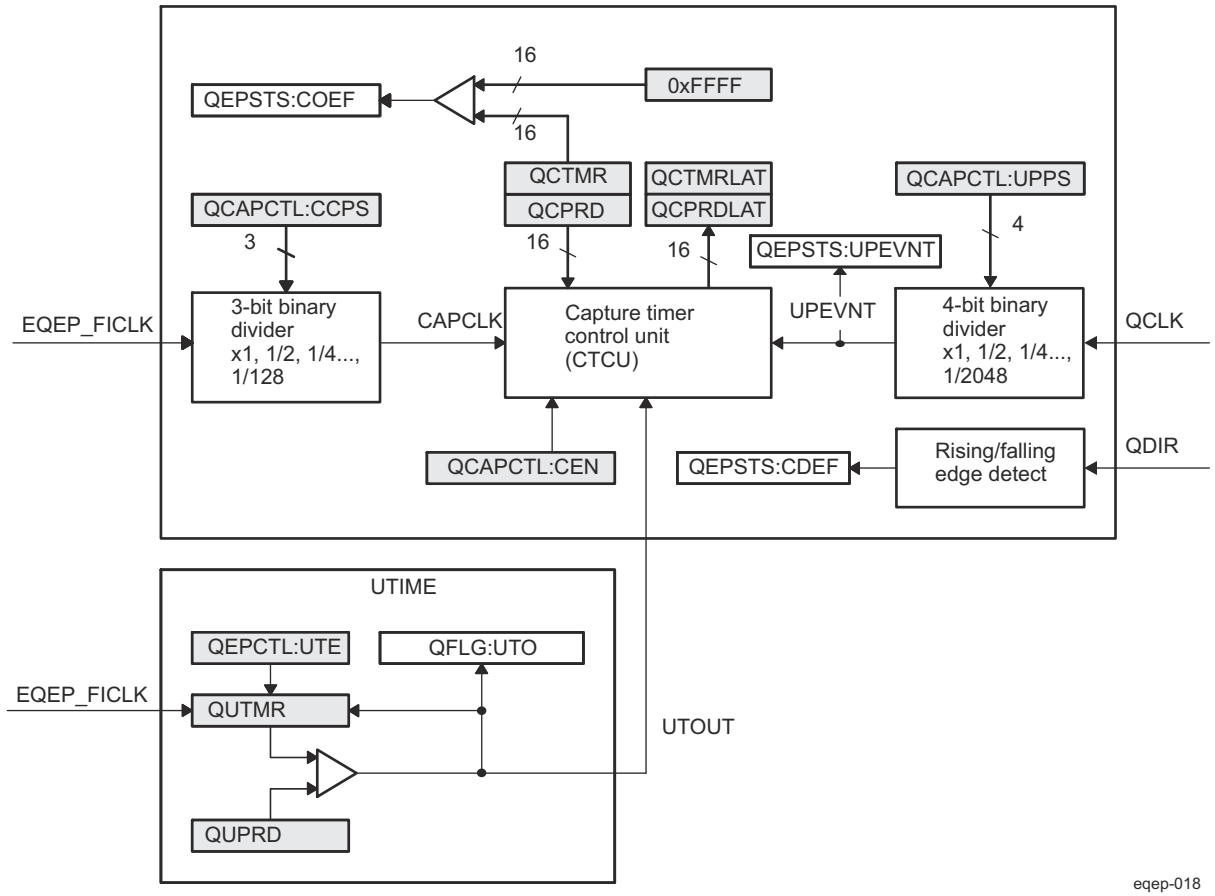
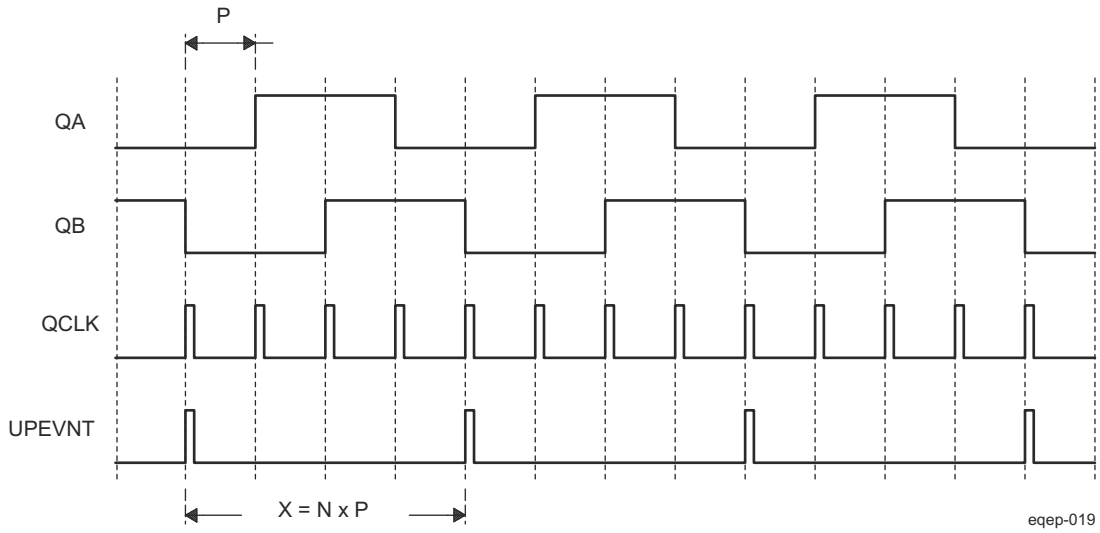


Figure 12-361. EQEP Edge Capture Unit



N - Number of quadrature periods selected using EQEP_QCAP_QPOS_CTL[3-0] UPPS bits

Figure 12-362. Unit Position Event for Low Speed Measurement (EQEP_QCAP_QPOS_CTL[UPPS] = 0010)

Copyright © 2026 Texas Instruments Incorporated

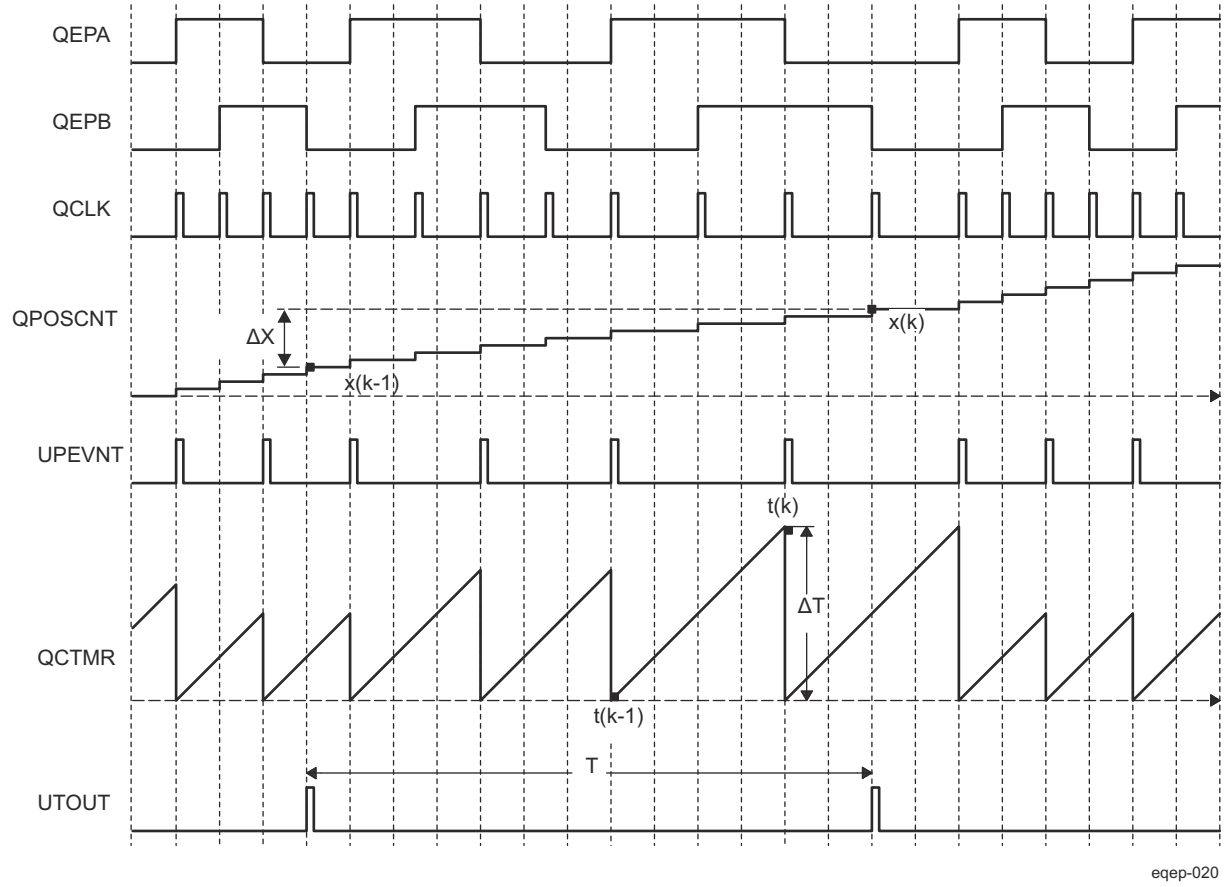


Figure 12-363. EQEP Edge Capture Unit - Timing Details

Velocity Calculation Equations:

$$V(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (37)$$

OR

$$V(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (38)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k - 1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement

t(k): Time instant "k"

t(k-1): Time instant "k - 1"

Copyright © 2026 Texas Instruments Incorporated

Unit time (T) and unit period (X) are configured using the EQEP_QUPRD and EQEP_QCAP_QPOS_CTL[3-0] UPPS registers. Incremental position output and incremental time output is available in the EQEP_QPOSLAT and EQEP_QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (EQEP_QUPRD)
ΔX	Incremental Position = QPOSLAT(k) - QPOSLAT(k - 1)
X	Fixed unit position defined by sensor resolution and QCAPCTL[3-0] UPPS bits
ΔT	Capture Period Latch (EQEP_QCPRDLAT)

12.4.4.4.5 EQEP Watchdog

The EQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The EQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (EQEP_QINT_EN_FLG[20] WTOI_FLG). The time-out value is programmable through the watchdog period bit field (EQEP_QWD_TMR_PRD[31-16] QWDPRD).

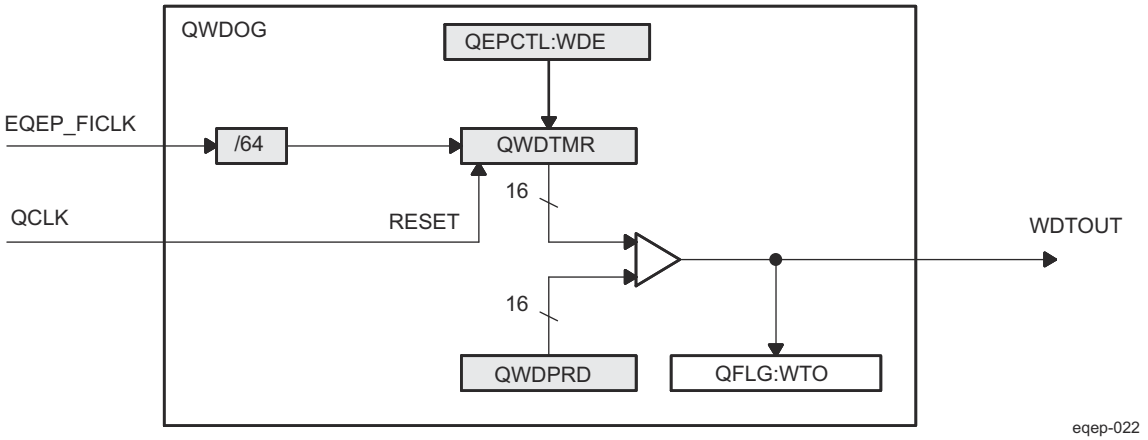


Figure 12-364. EQEP Watchdog Timer

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.6 Unit Timer Base

The EQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (EQEP_QINT_EN_FLG[27] UTOI_FLG) when the unit timer (QUTMR) matches the unit period register (EQEP_QUPRD).

The EQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event. The latched values are used for velocity calculation as described in Section Section 12.4.4.4.4.

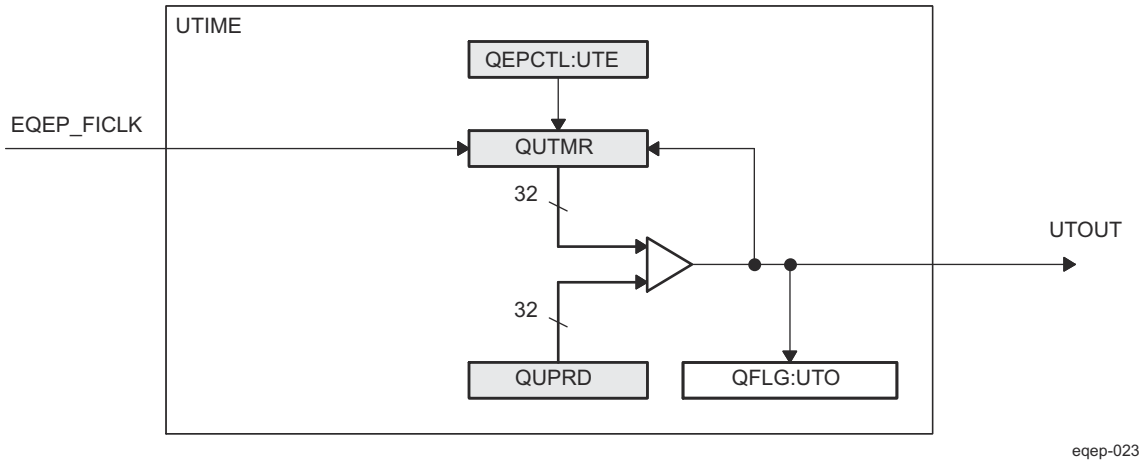


Figure 12-365. EQEP Unit Time Base

12.4.4.4.7 EQEP Interrupt Structure

Figure 12-366 shows how the interrupt mechanism works in the EQEP module.

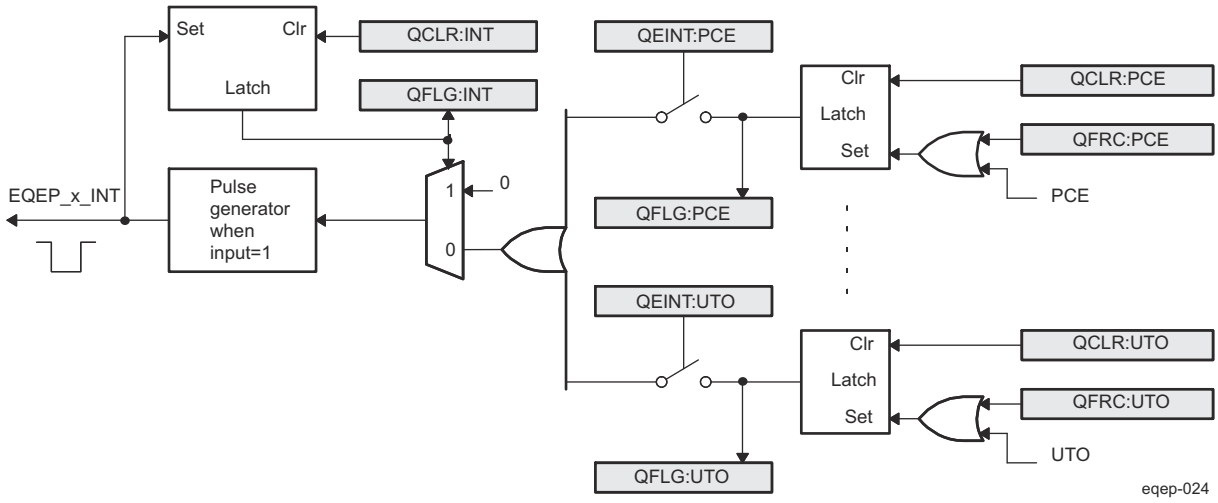


Figure 12-366. EQEP Interrupt Generation

Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL, and UTO) can be generated. The interrupt control register (EQEP_QINT_EN_FLG) is used to enable/disable individual interrupt event sources. The interrupt flag register (EQEP_QINT_EN_FLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is only generated to the interrupt controller if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (EQEP_QINT_CLR_FRC), before any other interrupt pulses are generated. SW can force an interrupt event by way of the interrupt force register (EQEP_QINT_CLR_FRC), which is useful for test purposes.

Copyright © 2026 Texas Instruments Incorporated

12.4.4.4.8 Summary of EQEP Functional Registers

Table 12-298 lists the registers with their memory locations, sizes, and reset values.

Table 12-298. EQEP Control and Status Functional Registers

Offset	Acronym	Register Description	Size (×16)/ #shadow
0h	EQEP_QPOSCNT	EQEP Position Counter Register	2/0
4h	EQEP_QPOSINIT	EQEP Position Counter Initialization Register	2/0
8h	EQEP_QPOSMAX	EQEP Maximum Position Count Register	2/0
Ch	EQEP_QPOSCMP	EQEP Position-Compare Register	2/1
10h	EQEP_QPOSILAT	EQEP Index Position Latch Register	2/0
14h	EQEP_QPOSSLAT	EQEP Strobe Position Latch Register	2/0
18h	EQEP_QPOSLAT	EQEP Position Counter Latch Register	2/0
1Ch	EQEP_QUTMR	EQEP Unit Timer Register	2/0
20h	EQEP_QUPRD	EQEP Unit Period Register	2/0
24h	EQEP_QWDTMR	EQEP Watchdog Timer Register	1/0
26h	EQEP_QWDPRD	EQEP Watchdog Period Register	1/0
28h	EQEP_QDECCTL	EQEP Decoder Control Register	1/0
2Ah	EQEP_QEPCTL	EQEP Control Register	1/0
2Ch	EQEP_QCAPCTL	EQEP Capture Control Register	1/0
2Eh	EQEP_QPOSCTL	EQEP Position Compare Control Register	1/0
30h	EQEP_QEINT	EQEP Interrupt Control Register	1/0
32h	EQEP_QFLG	EQEP Interrupt Flag Register	1/0
34h	EQEP_QCLR	EQEP Interrupt Clear Register	1/0
36h	EQEP_QFRC	EQEP Interrupt Force Register	1/0
38h	EQEP_QEPSTS	EQEP Status Register	1/0
3Ah	EQEP_QCTRM	EQEP Capture Timer Register	1/0
3Ch	EQEP_QCPRD	EQEP Capture Period Register	1/0
3Eh	EQEP_QCTMRLAT	EQEP Capture Timer Latch Register	1/0
40h	EQEP_QCPRDLAT	EQEP Capture Period Latch Register	1/0
5Ch	EQEP_PID	EQEP Peripheral ID Register	2/0

12.5 Audio Peripherals

Copyright © 2026 Texas Instruments Incorporated

12.5.1 Audio Tracking Logic (ATL)

This chapter describes the integration of the Audio Tracking Logic (ATL) subsystem in the device.

12.5.1.1 ATL Overview

The Audio Tracking Logic (ATL) is used by HD Radio™ applications to synchronize the digital audio output to the baseband clock. This same IP can also be used generically to track errors between two reference signals (such as frame syncs) and generate a modulated clock output (using software-controlled cycle stealing) which averages to some desired frequency. This process can be used as a hardware assist for asynchronous sample rate conversion algorithms.

12.5.1.1.1 ATL Features Overview

The ATL module supports the following features:

- One ATL module, containing four ATL instances, for HD Radio support and asynchronous sample rate conversion assistance
- Each instance tracks the time error between two syncs (local audio word select [AWS] and baseband word select [BWS])
- Each instance generates modulated ATCLK clock signals with software-initiated pulse stealing. The intention is to use these clocks to derive audio output bit clock on MCASP
- Selection between ATL_VCLK clock or functional ATL_PCLK to run error counting timers and to derive modulated ATCLK clock outputs.
- Clock and reset management: Receives clock and reset signals from the device PSC module. The ATL module receives hardware reset.
- Power management: The ATL belongs to the PD0 power domain.
- Local software reset

12.5.1.1.2 ATL Ports

Table 12-299. ATL Clocks and Resets

Clocks	
Module Clock Input	**Description**
ATL_VCLK	ATL interface clock
ATL_PCLK	ATL functional clock.
Resets	
Module Reset Input	**Description**
ATL_RST	ATL reset

Copyright © 2026 Texas Instruments Incorporated

12.5.2 Multichannel Audio Serial Port (MCASP)

This section describes the Multichannel Audio Serial Port (MCASP).

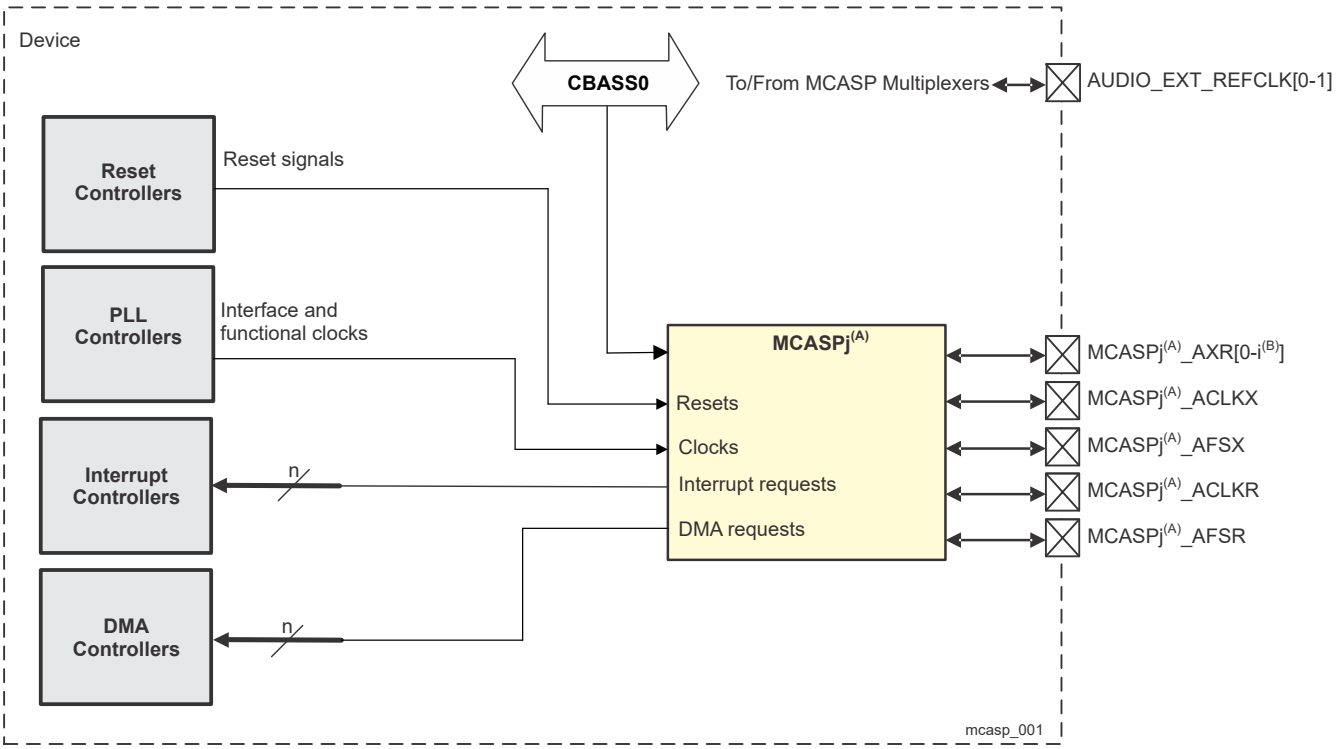
12.5.2.1 MCASP Overview

This section introduces the Multichannel Audio Serial Port (MCASP) module and describes its main functions and connections in the device.

The MCASP functions as a general-purpose audio serial port are optimized to the requirements of various audio applications. The MCASP module can operate in both transmit and receive modes. The MCASP is useful for time-division multiplexed (TDM) stream, Inter-IC Sound (I2S) protocols reception and transmission as well as for an inter-component digital audio interface transmission (DIT). The MCASP has the flexibility to gluelessly connect to a Sony/Philips digital interface (S/PDIF) transmit physical layer component.

Although inter-component digital audio interface reception (DIR) mode (this is, S/PDIF stream receiving) is not natively supported by the MCASP module, a specific TDM mode implementation for the MCASP receivers allows an easy connection to external DIR components (for example, S/PDIF to I2S format converters).

Figure 12-367 shows the MCASP modules overview.



A. j represents a valid instance of MCASP in a domain

B. i represents the maximum number of MCASPj_AXR signals - 1.

Figure 12-367. MCASP Modules Overview

12.5.2.1.1 MCASP Features

Each MCASP module includes the following main features:

- Independent serializer for each AXRx channel of each MCASP module
- Clock stop request/acknowledge protocol
- A single 32-bit buffer per serializer for transmit and receive operations
- Interconnect interface port for CBASS0

Copyright © 2026 Texas Instruments Incorporated

- Two independent clock generator modules for transmit and receive (clocking flexibility allows the MCASP to receive and transmit at different rates. For example, the MCASP can receive data at 48 kHz, but output up-sampled data at 96 kHz or 192 kHz)
- Each MCASP module functional clock can be generated:
  - Internally (controller mode)
  - Supplied over MCASP serial interface (target mode)
  - Has a controllable functional clock divide ratio
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator
  - TDM streams from 2 to 32, and 384 time slots
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits
  - Data formatter for bit manipulation
- Glueless connection to audio Analog-to-Digital Converters (ADC), Digital-to-Analog Converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components
- Wide variety of I2S and similar bit-stream format
- Integrated digital audio interface transmitter (DIT):
  - S/PDIF, IEC60958-1, AES-3 formats
  - Enhanced channel status/user data RAM
- 384-slot TDM with external digital audio interface receiver (DIR) device
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the MCASP receive section
- Support for 2 × DMA requests (one per direction):
  - 1 level-sensitive transmit direct memory access (DMA) request common for all of the MCASP serializers
  - 1 level-sensitive receive direct memory access (DMA) request common for all of the MCASP serializers
  - All transmit DMA requests are mapped to the device DMA controllers
- One transmit interrupt request common for all serializers
- One receive interrupt request common for all serializers
- Each of the Rx and Tx interrupts is propagated to different host processors via the device Interrupts

Note

Because a serializer receive and transmit channels data is shared on the same MCASP data pin, user can choose to have either Tx or Rx function from a serializer, not both at the same time.

Note

Some features may not be available. See *Module Integration* for more information.

12.5.2.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

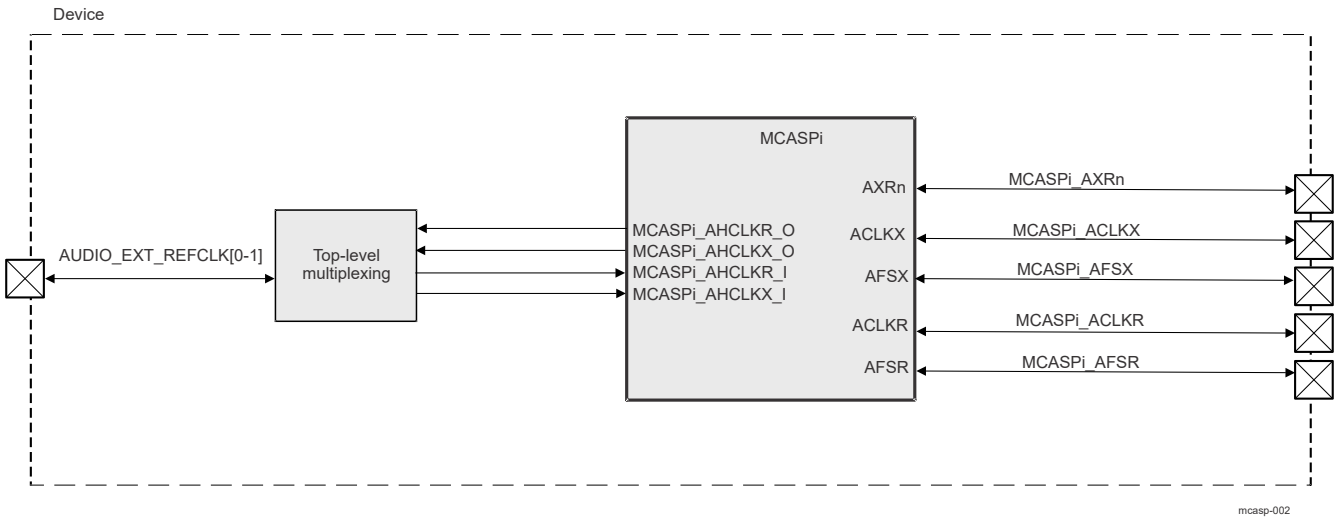
Copyright © 2026 Texas Instruments Incorporated

12.5.2.2 MCASP Environment

This section describes the MCASP application fields from an environment point of view (external connections). This section also lists the possible interfaces and describes the protocol and data format used in each case.

12.5.2.2.1 MCASP Signals

Figure 12-368 shows all of the MCASP interface signals.



Note

i represents a MCASP instance. See the device datasheet for available domains and MCASP instances.

n represents a AXR signal from a MCASP instance. See the device datasheet for available domains and MCASP instances.

Figure 12-368. MCASP Interface Signals

Table 12-300 describes the MCASP I/O signals.

Table 12-300. MCASP I/O Signals

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
		MCASPi[3] module		
AXR0	MCASPi[3]_AXR0	I/O	Audio transmit/receive data - channel 0	HiZ
AXR1	MCASPi[3]_AXR1	I/O	Audio transmit/receive data - channel 1	HiZ
AXR2	MCASPi[3]_AXR2	I/O	Audio transmit/receive data - channel 2	HiZ
AXR3	MCASPi[3]_AXR3	I/O	Audio transmit/receive data - channel 3	HiZ
AXR4	MCASPi[3]_AXR4	I/O	Audio transmit/receive data - channel 4	HiZ
AXR5	MCASPi[3]_AXR5	I/O	Audio transmit/receive data - channel 5	HiZ
AXR6	MCASPi[3]_AXR6	I/O	Audio transmit/receive data - channel 6	HiZ
AXR7	MCASPi[3]_AXR7	I/O	Audio transmit/receive data - channel 7	HiZ
AXR8	MCASPi[3]_AXR8	I/O	Audio transmit/receive data - channel 8	HiZ
AXR9	MCASPi[3]_AXR9	I/O	Audio transmit/receive data - channel 9	HiZ
AXR10	MCASPi[3]_AXR10	I/O	Audio transmit/receive data - channel 10	HiZ
AXR11	MCASPi[3]_AXR11	I/O	Audio transmit/receive data - channel 11	HiZ
AXR12	MCASPi[3]_AXR12	I/O	Audio transmit/receive data - channel 12	HiZ

Copyright © 2026 Texas Instruments Incorporated

Table 12-300. MCASP I/O Signals (continued)

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
AXR13	MCASPi[3]_AXR13	I/O	Audio transmit/receive data - channel 13	HiZ
AXR14	MCASPi[3]_AXR14	I/O	Audio transmit/receive data - channel 14	HiZ
AXR15	MCASPi[3]_AXR15	I/O	Audio transmit/receive data - channel 15	HiZ
ACLKX	MCASPi[3]_ACLKX	I/O	Transmit bit clock	HiZ
AFSX	MCASPi[3]_AFSX	I/O	Transmit frame synchronization	HiZ
ACLKR	MCASPi[3]_ACLKR	I/O	Receive bit clock	HiZ
AFSR	MCASPi[3]_AFSR	I/O	Receive frame synchronization	HiZ
MCASPi[3]_AHCLKX_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Transmit high-frequency controller clock. See *Module Integration*	HiZ
MCASPi[3]_AHCLKR_I/O	AUDIO_EXT_REFCLK[0-1]	I/O	Receive high-frequency controller clock. See *Module Integration*	HiZ

(1) I = Input; O = Output; I/O = Bidirectional
(2) HiZ = High Impedance
(3) i represents a MCASP instance. See the device datasheet for available domains and MCASP instances.

Note

MCASPi_AHCLKR_I/O and MCASPi_AHCLKX_I/O signals are multiplexed to AUDIO_EXT_REFCLK[0-1] device pins.

Note

For MCASPi_ACLKR_I/O, MCASPi_ACLKX_I/O, MCASPi_AHCLKR_I/O and MCASPi_AHCLKX_I/O signals to work properly, the RXACTIVE bit of the appropriate PADCFG_CTRL_MMR_CFG0_PADCONFIGy registers should be set to 0x1 because of retiming purposes.

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

Note

A serializer AXR data pin is shared between the transmit and receive logic of that serializer. The direction of data is determined in the *MCASP_PDIR* register and the function (Tx or Rx) is selected in the corresponding serializer control register MCASP_SRCTLn (n = 0 to 15).

12.5.2.2.2 MCASP Protocols and Data Formats

12.5.2.2.2.1 Protocols Supported

The MCASP supports a wide variety of protocols:

- Transmit section supports:
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - S/PDIF, IEC60958-1, AES-3 formats
- Receive section supports:
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots

Copyright © 2026 Texas Instruments Incorporated

  - TDM stream of 384 time slots specifically designed for easy interface to external digital interface receiver (DIR) device transmitting DIR frames to MCASP using the I2S protocol (one time slot for each DIR subframe)

The transmit and receive sections of the module may be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AFSR/X and AHCLKR/X_I/O
- Slot length (number of bits per time slot): 2, 4, 8, 12, 16, 20, 24, 28, 32 bits supported
- Word length (bits per word): 2, 4, 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length
- First-bit data delay: 0, 1, 2 bit clocks
- Left/right alignment of word inside slot
- Bit order: MSB first or LSB first
- Bit mask/pad/rotate function
  - Automatically aligns data internally in either Q31 or integer formats
  - Automatically masks nonsignificant bits (sets to 0, 1, or extends value of another bit)

Note

In I2S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

In DIT mode for MCASP, additional features of the transmitters are:

- Transmit-only mode 384 time slots (subframe) per frame
- Biphase encoded 3.3 V Output
- Channel status RAM (384 bits)
- User data RAM (384 bits)
- Support for consumer and professional applications
- Separate valid bit (V) for subframe A, B
- Stereo Support Only (Mono means send data 2 × via software)

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to all serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I2S mode, due to the need to generate Biphase Mark Encoded Data).

Note

The MCASP does NOT natively support DIR-mode reception (this is, receiving in the S/PDIF format). To allow this, the MCASP can use a DIR-input to I2S-output converter implemented by an external device (this is, external DIR component). To facilitate reception in this case, the TDM mode of MCASP receivers logic is extended to support a non-standard 384-slot TDM stream.

Note

An external transceiver must be connected to the MCASP port in the device to translate the electrical signals delivered by the MCASP (1.2 V or 1.8 V LVCMOS levels) to the electrical levels of the S/PDIF standard.

12.5.2.2.2.2 Definition of Terms

The serial bitstream transmitted or received by a MCASP serializer is a long sequence of 1s and 0s on an audio transmit/receive pins AXRn. However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

A basic synchronous serial interface consists of three important components: clock, frame sync, and data. Figure 12-369 shows two of the three basic components: the clock signal (ACLKX/ACLKR) and the data signals AXRn.

Copyright © 2026 Texas Instruments Incorporated

Figure 12-369 does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, each transmitter and receiver uses the signals ACLKX and ACLKR as serial clock, respectively. Optionally, a receiver can use ACLKX as the serial clock when a transmitter and receiver (not from the same serializer) of the MCASP are configured to operate synchronously.

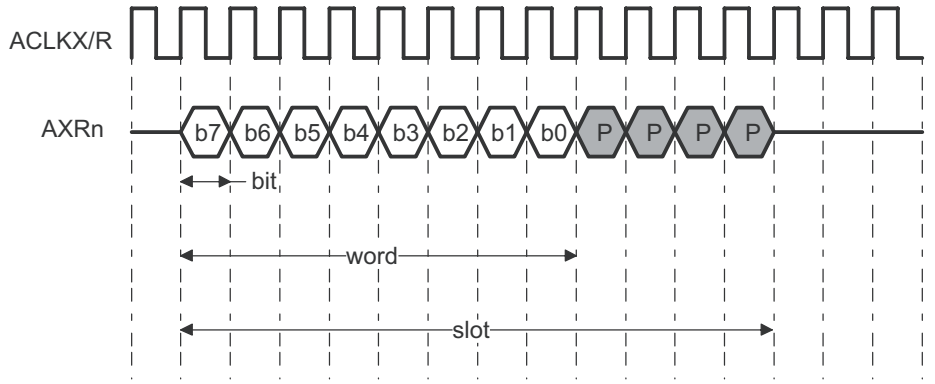
- Bit:

  A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A '1' is represented by a logic high on AXRn pins for the entire duration of the bit. A 0 is represented by a logic low on an AXRn pin for the entire duration of the bit.
- Word:

  A word is a group of bits that make up the data being transferred between the MCASP and the external device. Figure 12-369 shows an 8-bit word.
- Slot:

  A slot consists of the bits that make up the word and can consist of additional bits used to pad the word to a convenient number of bits for the interface between the MCASP and the external device. In Figure 12-369, the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with four 0s (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits can be shifted out of the MCASP on an AXRn pin with either MSB or LSB first.

  When the word size is smaller than the slot size, the word can be aligned to the left of the slot (beginning) or to the right of the slot (end). The additional bits in the slot not belonging to the word can be padded with 0, 1, or with one of the bits (typically, the MSB or LSB) from the data word, this is, left-aligned words within a slot are terminated with padding bits and right-aligned words within a slot are preceded by padding bits to fit in the slot size. Figure 12-370 shows these options.



(1) b7:b0 - bits. Bits b7 to b0 form a word.

(2) P - pad bits. Bits b7 to b0, together with the 4 pad bits, form a slot.

(3) In this example, the data is transmitted MSB first, left-aligned.

mcasp-003

Figure 12-369. Definition of Bit, Word, and Slot

Copyright © 2026 Texas Instruments Incorporated

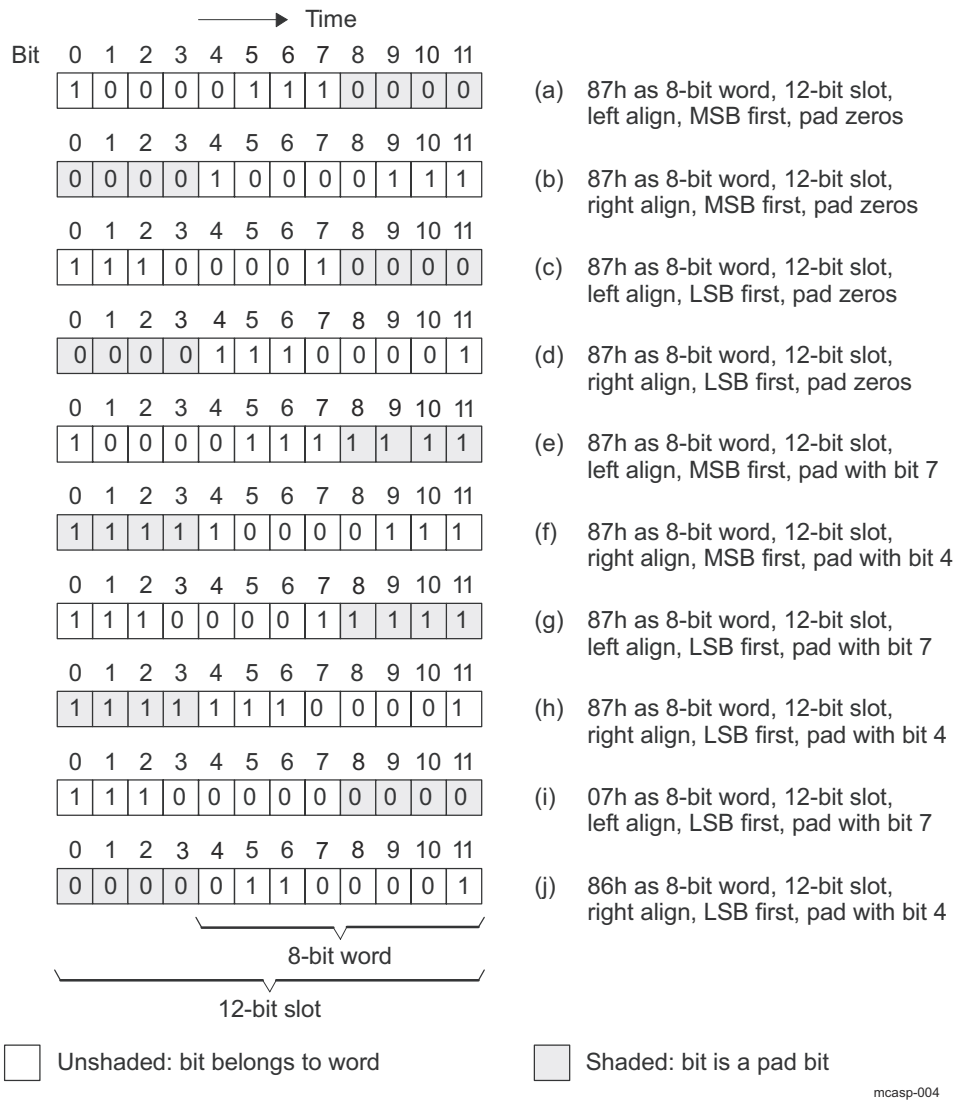


Figure 12-370. Bit Order and Word Alignment Within a Slot Examples

- Frame

  The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync in this chapter. A frame contains one or multiple slots, as determined by the desired protocol. Figure 12-371 shows an example frame of data and the frame definitions. In operation, the transmitter uses AFSX, and the receiver - AFSR signal. Figure 12-371 does NOT specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, each transmitter/receiver uses AFSX/AFSR as a frame synchronization signal, respectively. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the MCASP are configured to operate synchronously. This example shows two slots in a frame (I2S format) and a frame-sync (FS) duration of a slot length.

  This section shows only the generic definition of the frame sync. For more information about the frame-sync formats required for the transfer modes and protocols (TDM-mode and DIT-mode supported formats), see Section 12.5.2.2.2.3, *TDM Format* and Section 12.5.2.2.2.5, *S/PDIF-Coding Format*.

Copyright © 2026 Texas Instruments Incorporated

Note

All of the MCASP serializers share the same, device pad accessible, clock and frame signals, as follows:

– AHCLKX_I/O, ACLKX and AFSX for the transmitting section
– AHCLKR_I/O, ACLKR and AFSR for the receiving section

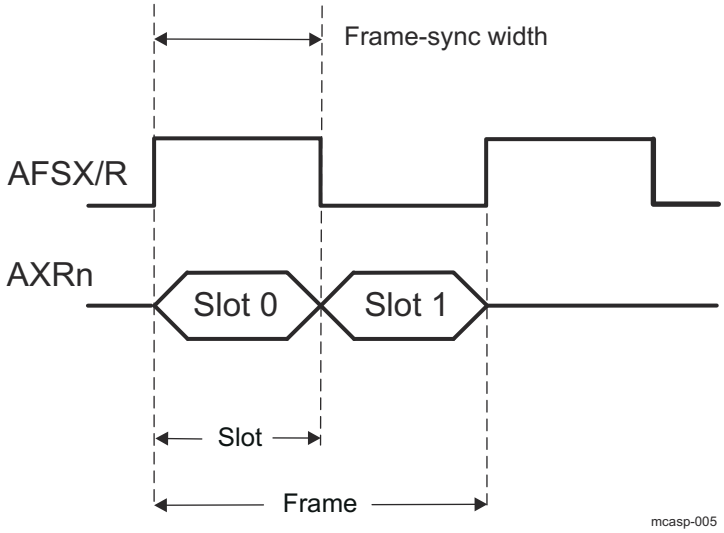


Figure 12-371. Definition of Frame and Frame-Sync Width

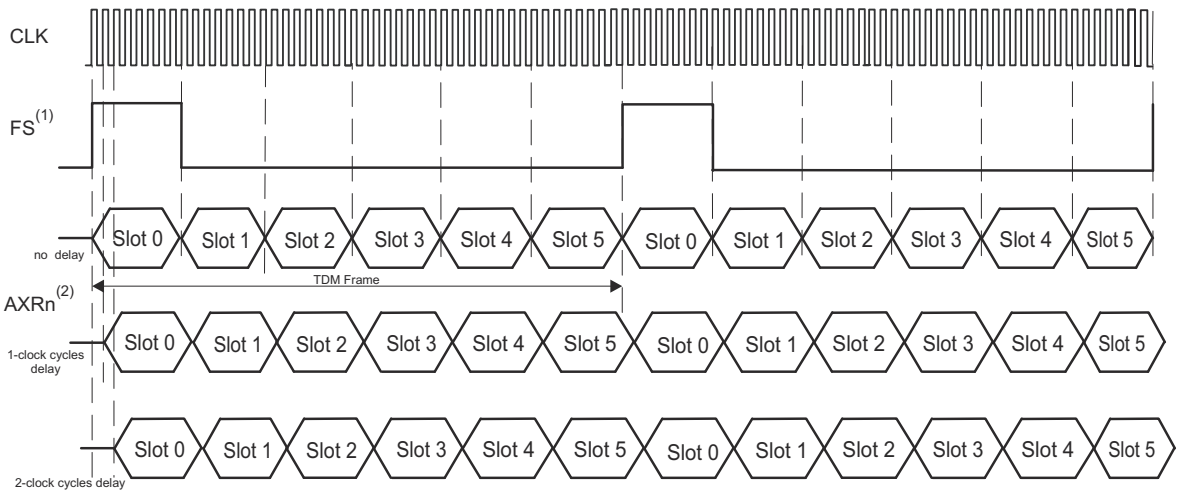
The following terms are used throughout this chapter:

- TDM: Time-division multiplexed. See Section 12.5.2.2.2.3, *TDM Format* for details on the TDM protocol.
- I2S: Inter-Integrated Sound protocol, commonly used on audio interfaces. The MCASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame).
- DIT: Digital audio interface transmit. The MCASP supports transmitting in S/PDIF format on each AXRn data pin.
- DIR: Digital audio interface receive. The MCASP does NOT natively support receiving in S/PDIF format on AXRn data pins and requires an external DIR-to-TDM or DIR-to-I2S converter chip for a DIR-frame reception.
- Slot or time slot: For DIT/DIR format, a MCASP time slot corresponds to a DIT/DIR subframe.

12.5.2.2.2.3 TDM Format

The TDM format is used to transfer data between the host CPU and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices. An example for a 6-slot (channel) TDM transmission on one MCASP data pin - AXRn is illustrated on Figure 12-372.

Copyright © 2026 Texas Instruments Incorporated



(1) - Frame sync duration of 1 slot - length is shown. A single bit - duration of FS is also supported

(2) - Slot 0 of AXRn stream is being offset with 0-, 1-, and 2- clock cycle delay from the frame sync, respectively.

mcasp-006

Figure 12-372. TDM Format - 6 channel example

The TDM format uses three signals in a basic synchronous serial interface: data (AXRn), clock (CLK) and frame sync (FS). The data signal present on AXRn pin is fully synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (see also Section 12.5.2.2.2.2), the latter being also refered to as the "time-slots" or "channels" in TDM terminology. A frame consists of multiple time-slots. Each TDM frame is marked by the frame sync signal (AFSX or AFSR). The TDM transfer is continuous and periodic, with no delays between slots.

Within a certain frame, the last bit of slot N is followed immediately on the next serial clock with the first bit of the next slot N+1. On the boundary between two adjacent TDM-frames, the last bit of the last slot from the frame M, is followed immediately on the next clock cycle with the first bit of the first slot from the next frame M+1. For MCASP, there is an option to offset the first bit of the first slot with a 0-,1- or 2-cycle delay from the frame sync signal.

The frame sync - AFSX/AFSR only marks the beginning of slot 0 and start of a new frame. Since it does not determine the boundaries of a slot, there is a requirement for a connected transmitter and receiver to agree on the number of transferred bits per slot.

In a typical audio system involving MCASP module, a single TDM data frame is transferred during each sample period Ts of a data converter. The user has following choices to implement multiple channels:

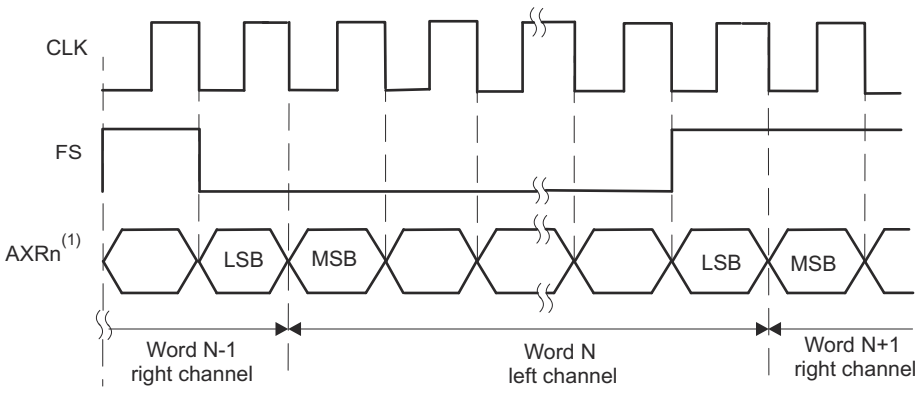
- Use more data slots (on a price of higher speed serial clock) per frame transmitted/received on just one of the available MCASP data pins AXRn.
- Use less number of slots per TDM frame (requires a slower serial clock), making them available on several of the MCASP pins AXRn.

12.5.2.2.2.4 I2S Format

The TDM transfer mode of the MCASP supports the I2S format when frame is configured to have 2 slots. The I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXRn. The "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format equals size of a slot. The frame signal is also referred to as "word select" in the I2S format.

The I2S protocol is illustrated on Figure 12-373.

Copyright © 2026 Texas Instruments Incorporated



(1) - The example shows I2S data MSB-first transmission with 1-clock cycle delay between FS and data MSB

mcasp-007

Figure 12-373. I2S Format Overview

12.5.2.2.2.5 S/PDIF Coding Format

The MCASP transmitter supports the S/PDIF format with 3.3 V biphase-mark encoded output. The S/PDIF format is supported by the DIT- transfer mode of the MCASP. This section briefly discusses the S/PDIF coding format.

Note

The DIR- reception of S/PDIF format frames is NOT natively supported from the device MCASP. For this purpose, an external DIR-to-TDM transfer mode adapter can be used between the remote device S/PDIF transmitter output and the MCASP TDM-only compatible receiver input.

12.5.2.2.2.5.1 Biphase-Mark Code

In S/PDIF format, the digital signal is coded using the biphase-mark code (BMC). For each serializer transmitter n, the clock, frame, and data are embedded in only one signal - the data signal AXRn. In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. Figure 12-374 and Table 12-301 show how data is encoded to the BMC format.

As shown in Figure 12-374, the clock frequency is twice the unencoded data bit rate. In addition, the clock is always programmed to 128 × f_s, where f_s is the sample rate (see Section 12.5.2.2.2.5.3, *Frame Format*, for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.

Copyright © 2026 Texas Instruments Incorporated

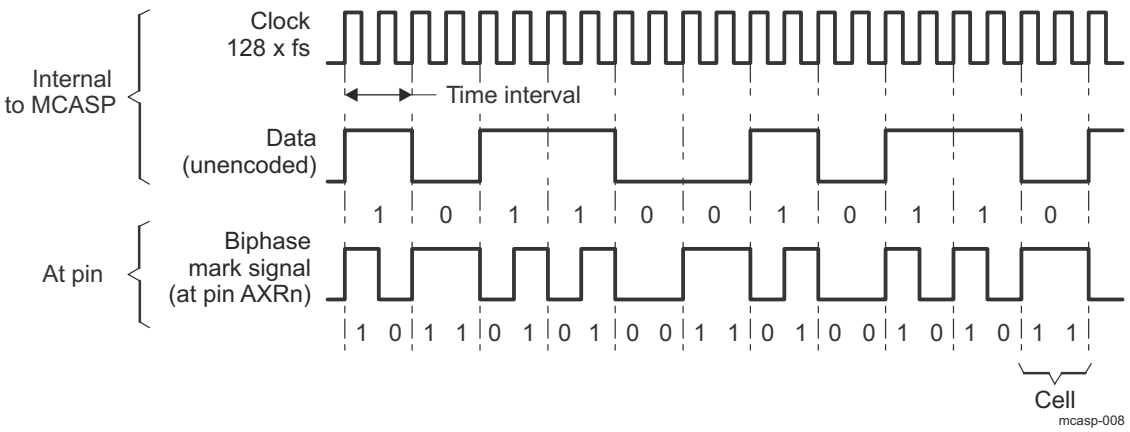


Figure 12-374. Biphase-Mark Code

Table 12-301. Biphase-Mark Encoder

Data (Unencoded)	Previous State at Pin AXRn	BMC-Encoded Cell Output at Pin AXRn
0	0	11
0	1	00
1	0	10
1	1	01

12.5.2.2.2.5.2 S/PDIF Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered 0 to 31. Figure 12-375 shows a subframe.

- Time intervals 0–3 carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is not encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See Table 12-302.
- Time intervals 4–27 carry the audio sample word in linear 2s-complement representation. The MSB is carried by time interval 27. When a 24-bit coding range is used, the LSB is in time interval 4. When a 20-bit coding range is used, time intervals 8-27 carry the audio sample word with the LSB in time interval 8. Time intervals 4–7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field can carry any other information.
- Time interval 28 carries the validity bit (V) associated with the main data field in the subframe.
- Time interval 29 carries the user data channel (U) associated with the main data field in the subframe.
- Time interval 30 carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- Time interval 31 carries a parity bit (P) such that time intervals 4–31 carry an even number of 1s and an even number of 0s (even parity). As listed in Table 12-302, the preambles (time intervals 0–3) are also defined with even parity.

Copyright © 2026 Texas Instruments Incorporated

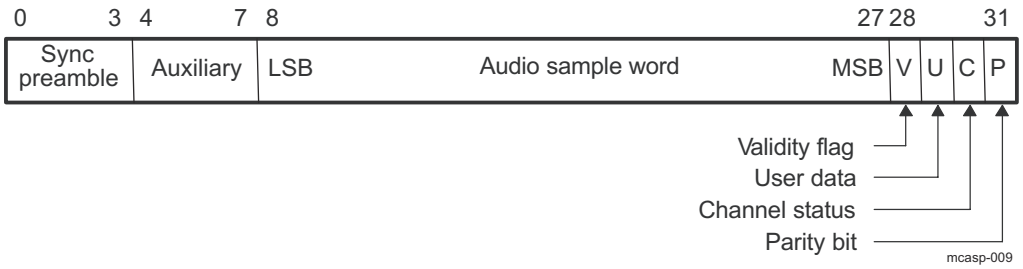


Figure 12-375. S/PDIF Subframe Format

As listed in Table 12-302, the MCASP DIT generates only one polarity of preambles, and it assumes the previous logical state is 0. This is because the MCASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXRn pin before continuing with the next transmission.

Table 12-302. Preamble Codes

Preamble Code(1)	Previous Logical State	Logical States on pin AXRn(2)	Description
B (or Z)	0	1110 1000	Start of a block and subframe 1
M (or X)	0	1110 0010	Subframe 1
W (or Y)	0	1110 0100	Subframe 2

(1) Historically, preamble codes are referred to as B, M, and W. For use in professional applications, preambles are referred to as Z, X, and Y, respectively.
(2) The preamble is not BMC-encoded. Each logical state is synchronized to the serial clock. These eight logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

12.5.2.2.2.5.3 Frame Format

An S/PDIF frame is composed of two subframes (see Figure 12-376). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency f_s. The S/PDIF format clock rate is therefore 128 × f_s (128 = 32 cells per subframe × 2 clocks per cell × 2 subframes per sample). For example, for an S/PDIF stream at a 192-kHz sampling frequency, the serial clock is 128 × 192 kHz = 24.58 MHZ.

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data (cell 28 validity bits for A- and B- channels, both set to '0'). The first subframe (left or A channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (right or B channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, cell 28 (validity bit) is set to logical 1.

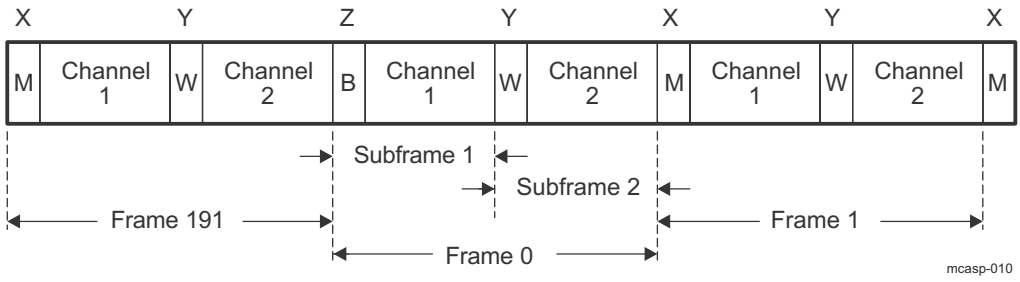


Figure 12-376. S/PDIF Frame Format

Copyright © 2026 Texas Instruments Incorporated

12.5.2.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

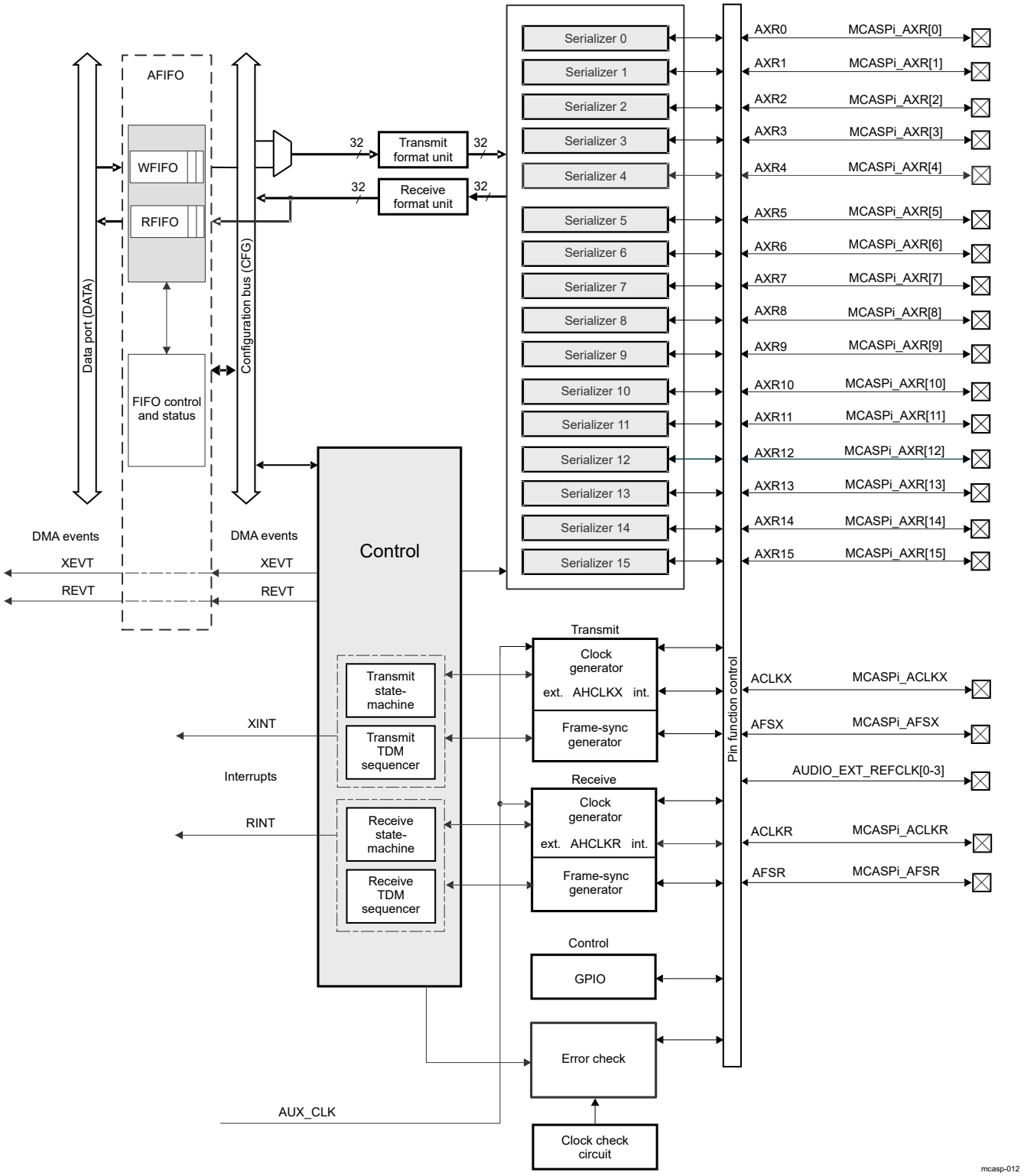
12.5.2.4 MCASP Functional Description

In the text throughout this section a single instance of MCASP is described assuming that all modules are functionally identical. For module availability and integration differences, see Section 12.5.2.2, *MCASP Environment*, and *Module Integration*.

12.5.2.4.1 MCASP Block Diagram

Figure 12-377 shows the major blocks of the MCASP module.

Copyright © 2026 Texas Instruments Incorporated



A. i represents a valid instance of MCASP in a domain. See the device datasheet for valid instances.

B. All signals might not be valid for an instance. See the device datasheet for specifics.

Figure 12-377. MCASP Module Block Diagram

Copyright © 2026 Texas Instruments Incorporated

Note

The internal and external clocks mentioned in this section are with respect to clock and frame-sync generator modules.

12.5.2.4.2 MCASP Clock and Frame-Sync Configurations

There are three scenarios to provide clock source signals for the Tx part and four scenarios for the Rx part of the MCASP serializers. The first three scenarios are identical between the Tx and Rx part of the MCASP. They feature an asynchronous operation between receiver and transmitter channels using independent Tx/Rx bit rate clock sources (either internal or external).

In the first scenario, the transmit - XCLK and receive - RCLK serial clocks (clock at the bit rate) are generated internally by passing through a couple of clock dividers off the internal functional clock source (AUXCLK). In this case, the bit rate clock is generated internally and is driven out on the pin ACLKX for the Tx part and pin ACLKR for the Rx part, respectively. An internally generated high-frequency clock can be optionally driven out onto the AHCLKX pin for the Tx part to serve as a reference clock for other components in the system.

In the second scenario, an external for the device clock, is passed on the ACLKX (for the TX part) and ACLKR (for the RX part) pins which are configured as inputs. In this case the Rx- /Tx- high-speed clock logic is bypassed for the XCLK/RCLK generation.

In the third (mixed) scenario, an externally driven (controller) high-frequency clock is applied on the AHCLKX (for the TX part) pin, which is configured as input. In this case the AHCLKX clock frequency can be divided down via programming the ACLKX associated divider to produce the necessary bit rate clock. The high-speed clock divider can NOT be used.

In the fourth clock generation scenario the bit rate clock for MCASP receivers - RCLK is derived from the bit rate clock of the MCASP transmitters - XCLK for a synchronous operation between transmitters and receivers. Hence, the whole receiver clock generator logic is bypassed.

A typical role of the MCASP frame sync signal is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

For an asynchronous operation, the AFSX (Tx part) and AFSR (Rx part) frame synchronization signals can be sourced internally or delivered externally independently for the Tx and Rx channels. During synchronous operation the receive frame sync - AFSR signal is derived from the transmit frame sync - AFSX signal. A synchronous and asynchronous mode applies to bit rate clock and frame sync signals at the same time.

12.5.2.4.2.1 MCASP Transmit Clock

The transmit high-speed and transmit clock configuration is controlled by the following registers:

- *MCASP_ACLKXCTL*
- *MCASP_AHCLKXCTL*

In case, the transmit bit clock, ACLKX, is generated internally, the *MCASP_ACLKXCTL*[5] CLKXM bit must be set to 1. Thus, the clock is divided down by a programmable bit clock divider (the *MCASP_ACLKXCTL*[4-0] CLKXDIV bit field) from the source signal.

If the transmit high-frequency controller clock, AHCLKX, is also sourced internally (that is first scenario described in Section 12.5.2.4.2, the *MCASP_AHCLKXCTL*[15] HCLKXM bit must be set to 1. Thus, the clock is divided down by a programmable high-clock divider (the *MCASP_AHCLKXCTL*[11-0] HCLKXDIV bit field) from the MCASP internal clock source AUXCLK.

Internally, the MCASP always shifts transmit data at the rising edge of the internal transmit clock - XCLK, (see Figure 12-378). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If *MCASP_ACLKXCTL*[7] CLKXP = 0, the CLKXP mux directly passes ACLKX signal to XCLK. As a result, the MCASP shifts transmit data at the rising edge of ACLKX. If *MCASP_ACLKXCTL*[7] CLKXP = 1, the CLKX mux passes the inverted version of ACLKX to XCLK. As a result, the MCASP shifts transmit data at the falling edge of ACLKX.

Copyright © 2026 Texas Instruments Incorporated

It can be seen in Figure 12-378 that XCLK is propagated to the Rx clock logic, to allow an internally synchronous operation between MCASP transmitters and receivers. This is used for example in the MCASP loopback mode.

Note

The polarity of ACLKX can be controlled in the *MCASP_ACLKXCTL*[7] CLKXP bit, regardless of ACLKX signal being internally or externally sourced.

In addition, there is an option to invert polarity of the AHCLKX controller high speed clock via writing the *MCASP_AHCLKXCTL*[14] HCLKXP bit.

Note

In a similar way, the polarity of AHCLKX clock can be controlled in the *MCASP_AHCLKXCTL*[14] HCLKXP bit, regardless of the AHCLKX signal being internally or externally sourced.

Figure 12-378 presents the block diagram of the transmit clock generator.

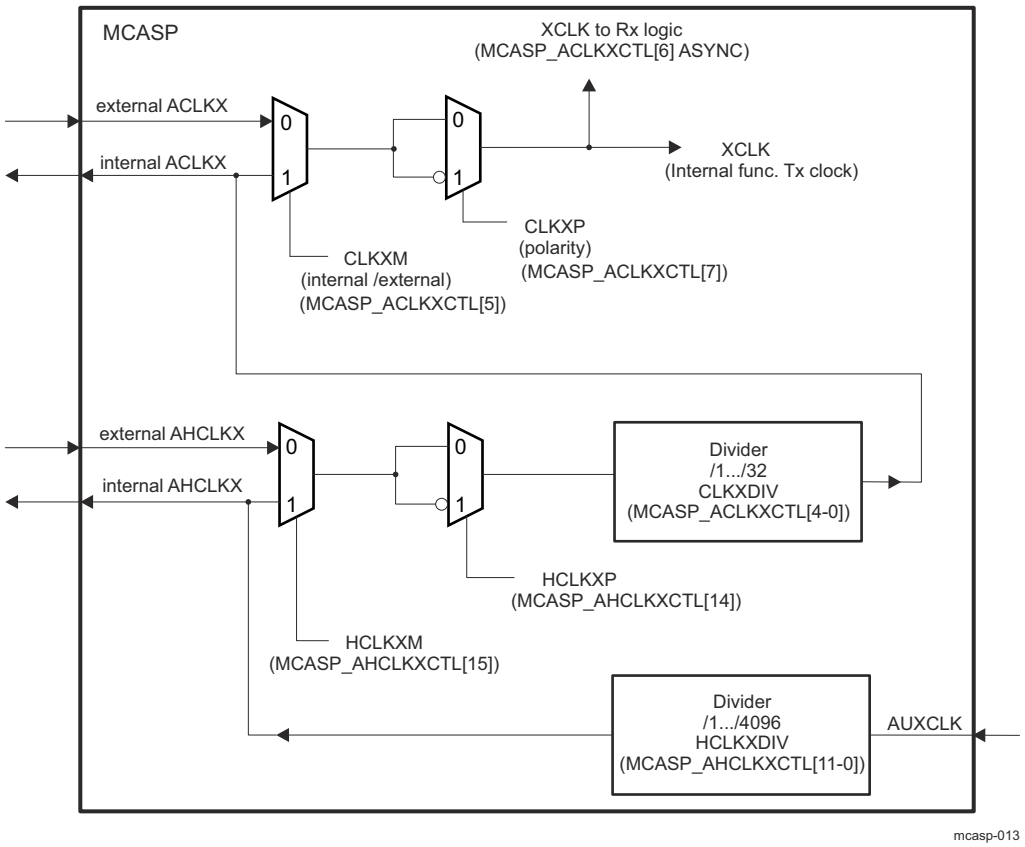


Figure 12-378. Transmit Clock Generator Block Diagram

Copyright © 2026 Texas Instruments Incorporated

Note

In this device:

- ACLKX is mapped on the device ball ACLKX
- internal AHCLKX is mapped on the device balls AUDIO_EXT_REFCLK[0-1]
- external AHCLKX is mapped on MCASPi_AHCLKX clock from the Device Configuration

For more information about MCASP integration, see Section 12.5.2.2, *MCASP Environment*, and *Module Integration*.

12.5.2.4.2.2 MCASP Receive Clock

The MCASP receive clock generator is built on a very similar to the transmit clock generator (but independent) circuit.

The receive clock configuration is controlled by the following registers:

- *MCASP_ACLKRCTL*
- *MCASP_AHCLKRCTL*

In case, the receive bit clock, ACLKR, is generated internally (but asynchronously to XCLK), the *MCASP_ACLKRCTL*[5] CLKRM bit must be set to 1. Thus, the clock is divided down by a programmable bit clock divider (the *MCASP_ACLKRCTL*[4-0] CLKRDIV bit field) from the source signal.

If the receive high-frequency controller clock, AHCLKR, is also sourced internally (that is, first scenario described in Section 12.5.2.4.2) and the *MCASP_AHCLKRCTL*[15] HCLKRM bit must be set to 1. Thus, the clock is divided down by a programmable high-clock divider (the *MCASP_AHCLKRCTL*[11-0] HCLKRDIV bit field) from the MCASP internal clock source AUXCLK.

Note

The polarity of ACLKR can be controlled in the *MCASP_ACLKRCTL*[7] CLKRP bit, regardless of ACLKR signal being internally or externally sourced.

In a similar way, the polarity of AHCLKR clock can be controlled in the *MCASP_AHCLKRCTL*[14] HCLKRP bit, regardless of the AHCLKR signal being internally or externally sourced.

There is an option for the MCASP receiver to be configured to operate synchronously to the ACLKX and AFSX signals. The XCLK output of the Tx Clock generator (see Figure 12-378 and Figure 12-379) becomes source of the receive clock (RCLK output), when the *MCASP_ACLKRCTL*[6] ASYNC bit in the transmit clock control register is set to '0b0'. For more information, refer to Section 12.5.2.4.2.4.

Figure 12-379 presents the block diagram of the receive clock generator.

Copyright © 2026 Texas Instruments Incorporated

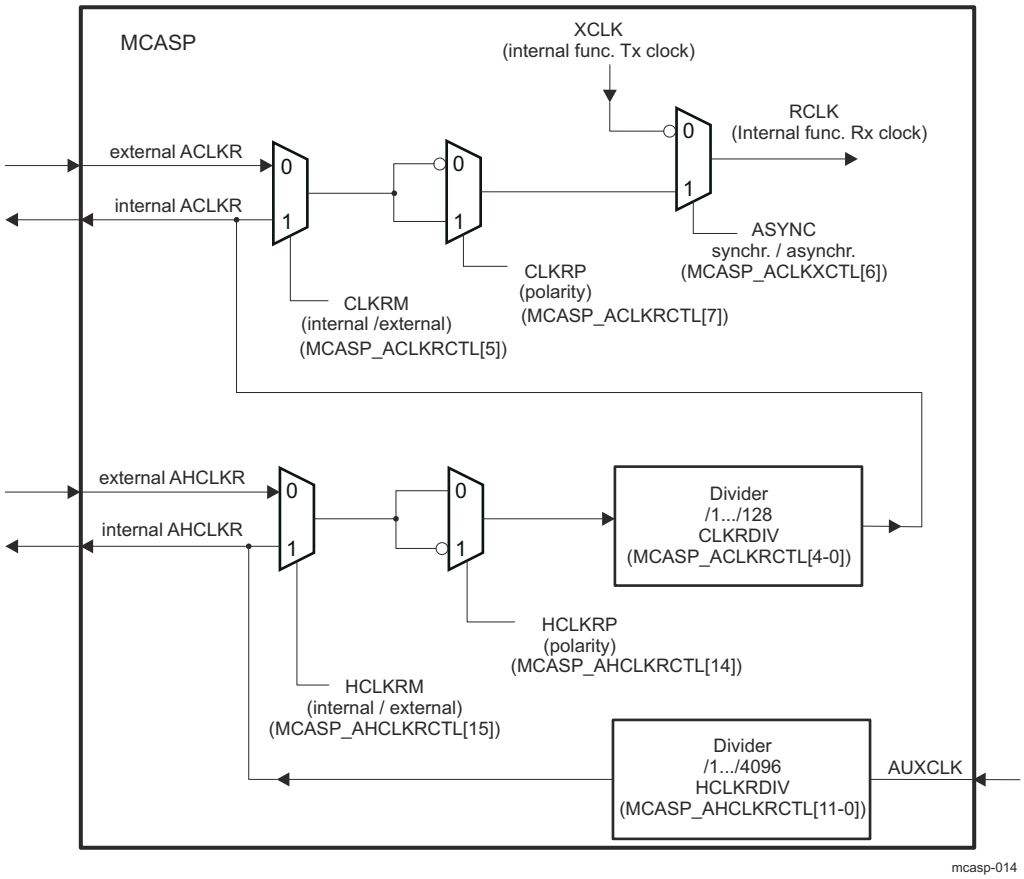


Figure 12-379. Receive Clock Generator Block Diagram

Note

In this device:

- ACLKR is mapped on the device ball ACLKR
- internal AHCLKX is mapped on the device balls AUDIO_EXT_REFCLK[0-1]
- external AHCLKR is mapped on Device Configuration MCASPi_AHCLKR from Device Configuration

For more information about MCASP integration, see Section 12.5.2.2, *MCASP Environment*, and , *Module Integration*.

12.5.2.4.2.3 Frame-Sync Generator

There are two different modes for frame sync: burst and TDM. The MCASP frame sync generator logic is illustrated in Figure 12-380.The I/O buffers are not part of the MCASP module, and are not shown in the figure.

Copyright © 2026 Texas Instruments Incorporated

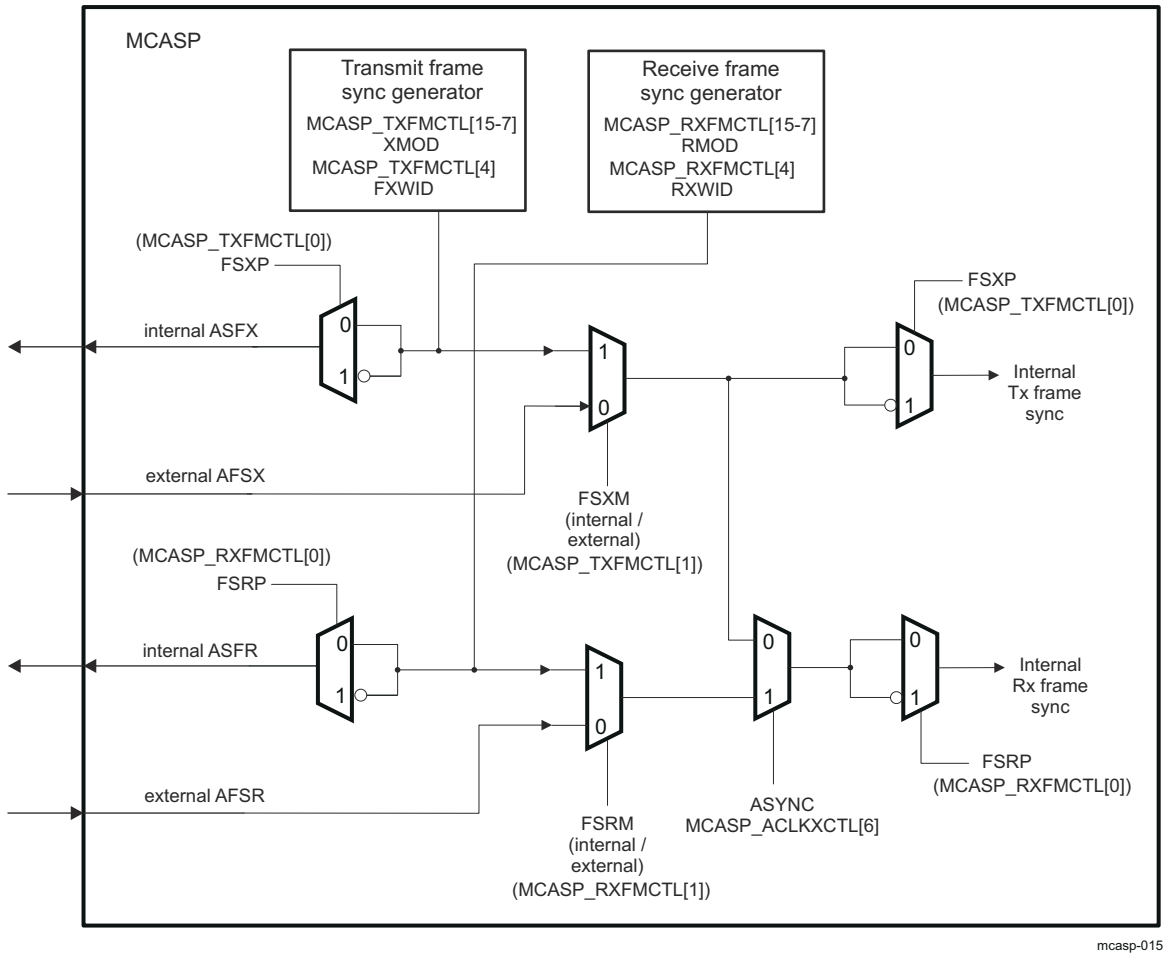


Figure 12-380. Frame Sync Generator Block Diagram

Note

For more on MCASP integration, see Section 12.5.2.2, *MCASP Environment*, and *Module Integration*.

For the transmit logic, following frame-sync generator configurations can be selected:

- Internally/externally generated frame-sync via configuring MCASP_AFSXCTL[1] FSXM bit
- Frame-sync polarity: Rising edge or falling edge via configuring MCASP_AFSXCTL[0] FSXP bit
- Frame-sync width: "single bit" or "single word" via configuring MCASP_AFSXCTL[4] FXWID bit
- Frame sync mode - the appropriate frame sync generation pattern for the selected transfer mode is defined in the MCASP_AFSXCTL[15-7] XMOD bit field, as follows:
  - For DIT mode (384 slots) - MCASP_AFSXCTL[15-7] XMOD = 0x180
  - For I2S mode (2 TDM slots) - MCASP_AFSXCTL[15-7] XMOD = 0x2
  - For TDM mode (from 3 to 32 TDM slots) - MCASP_AFSXCTL[15-7] XMOD bit field set in range 0x3 - 0x20
- Bit delay: 0, 1, or 2 cycles before the first data bit. This delay is defined in MCASP_XFMT[17-16] XDATDLY bit field

For the receive logic, following frame-sync generator configurations can be selected:

- Internally/externally generated frame-sync via configuring MCASP_AFSRCTL[1] FSRM bit
- Frame-sync polarity: Rising edge or falling edge via configuring MCASP_AFSRCTL[0] FSRP bit
- Frame-sync width: "single bit" or "single word" via configuring MCASP_AFSRCTL[4] FRWID bit

Copyright © 2026 Texas Instruments Incorporated

- Frame sync mode - the appropriate frame sync generation pattern for the selected transfer mode is defined in the MCASP_AFSRCTL[15-7] RMOD bit field, as follows:
  - For I2S mode (2 TDM slots) - MCASP_AFSRCTL[15-7] RMOD = 0x2
  - For TDM mode (from 3 to 32 TDM slots) - MCASP_AFSRCTL[15-7] RMOD set in range 0x3 - 0x20
  - For the special 384-slot TDM mode - MCASP_AFSRCTL[15-7] RMOD = 0x180
- Bit delay: 0, 1, or 2 cycles before the first data bit. This delay is defined in the MCASP_RFMT[17-16] RDATDLY bit field
- Selecting the source (AFSX or AFSR) of receiver internal frame synchronization. This is done in the same bit - *MCASP_ACLKXCTL*[6] ASYNC, used to define the receiver internal clock source. For more details, refer to Section 12.5.2.4.2.4, *Synchronous and Asynchronous Transmit and Receive Operations*.

Regardless of the AFSX/AFSR being internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSXP/FSRP = 1, the frame sync polarity is falling edge.

Note

Certain restrictions apply to the receive and transmit logic settings, when the *MCASP_ACLKXCTL*[6] ASYNC bit is set to 0b0. They are described in Section 12.5.2.4.2.4.

12.5.2.4.2.4 Synchronous and Asynchronous Transmit and Receive Operations

Synchronous Transmit and Receive Operations -

When the *MCASP_ACLKXCTL*[6] ASYNC bit is written to 0b0, the transmit and receive sections operate synchronously to the transmit section clock and transmit frame sync signals.

Though Rx section may have a different data format, it has to be configured to have the same slot size than the transmit section one. As shown on the Figure 12-379, with the ASYNC bit set to 0b0, the RCLK becomes an inverted version of the transmit clock generator XCLK output.

When the *MCASP_ACLKXCTL*[6] ASYNC = 0b0, both Rx and Tx sections use the same clock and frame sync signals. For this reason, they must be aligned on the following settings:

- MCASP_DITCTL[0] DITEN = 0 (that is, transmission in TDM mode is enabled)
- The total number of bits per frame must be the same (that is, RSSZ * RMOD product value must equal XSSZ * XMOD product value)
- The settings in the *MCASP_ACLKRCTL* register are NOT considered
- FSXM must match FSRM
- FXWID must match FRWID

For all other settings, the transmit and receive sections may be programmed independently.

Asynchronous Transmit and Receive Operations -

When the *MCASP_ACLKXCTL*[6] ASYNC = 0b1, Tx and Rx operate independently from each other with separate clock and frame sync signals.

Note

Synchronous transmit and receive operations are allowed only in the MCASP TDM (I2S) mode (this is, when MCASP_DITCTL[0] DITEN = 0b0).

12.5.2.4.3 MCASP Serializers

The MCASP serializers shift serial data in (Rx) and out (Tx) of the MCASP. A given serializer n consists of a shift register (XRSRn) with a single-entry data buffer XRBUFn used either for transmitting (write accessible in the MCASP_XBUFn register) or for receiving (read accessible in the MCASP_RBUFn register) data. In addition, each serializer has a dedicated control register (MCASP_SRCTLn) and a serial bidirectional data pin - AXRn. The register MCASP_SRCTLn allows n-th serializer to be configured as a transmitter, receiver, or as inactive.

Copyright © 2026 Texas Instruments Incorporated

There are transmit and receive data formatting units to support data alignment options of the MCASP which are shared between all Tx and Rx serializers, respectively.

A given serializer XRSRn shifter configured as a receiver in the MCASP_SRCTLn register, shifts in data through MCASP corresponding device level bidirectional data pad AXRn. A given serializer XRSRn shifter configured as a transmitter in the MCASP_SRCTLn register, shifts out data on MCASP corresponding device level bidirectional data pad AXRn (n = 0 to 15).

The serializer is clocked from the transmit section clock (ACLKX signal) if configured to transmit or clocked from the receive section clock (ACLKR signal) if configured to receive. A serializer configured to transmit and receive operates in lockstep, which means that for MCASP there are at most a couple of zones, one for transmit and one for receive.

Figure 12-381 shows the serializer block diagram.

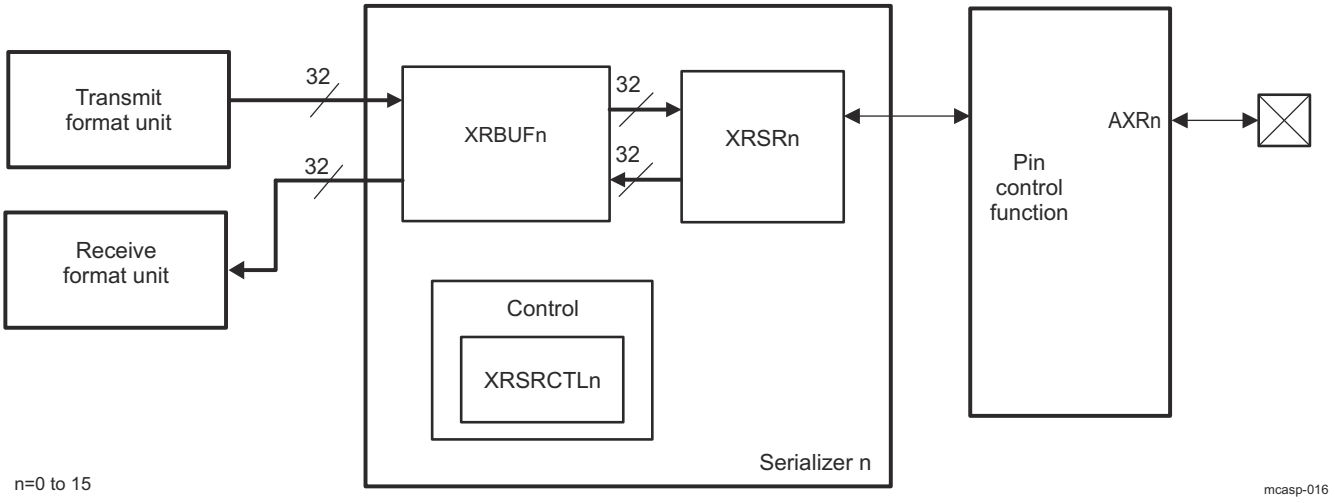


Figure 12-381. Individual Serializer and Connections Within MCASP

Transmission on the n-th serializer is performed as follows:

The MCASP is serviced by writing data into the MCASP_XBUFn register, which is an alias of the serializer data buffer - XRBUFn for transmit function. The data automatically passes through the transmit format unit before reaching the XRBUFn register in the serializer. The data is then copied from the XRBUFn to XRSRn and shifted out from AXRn synchronously to the serial clock.

Reception from the n-th serializer is performed as follows:

The data is shifted into the MCASP XRSRn serializer register through the AXRn pin, bit by bit. Once the entire slot becomes available within the XRSRn shift register, the data is copied into the serializer data buffer - XRBUFn, and can be accessed in the MCASP_RBUFn register, which is an alias of the serializer data buffer - XRBUFn for receive function. When software reads the data from this register, the MCASP passes the data through the receive format unit, hence it returns the formatted data.

Serializer controls:

A serializer n is configured as inactive via setting MCASP_SRCTLn[1-0] SRMOD bit field to 0x0.

For a transmitting serializer, the MCASP_SRCTLn[3-2] DISMOD bitfield, defines the AXRn pin output state, during inactive slots (HIGH, LOW or Hi-Z).

Transmit function for the n-th serializer is selected via setting MCASP_SRCTLn[1-0] SRMOD bit field to 0x1.

Receive function for the n-th serializer is selected via setting MCASP_SRCTLn[1-0] SRMOD bit field to 0x2 (n = 0 to 15).

Copyright © 2026 Texas Instruments Incorporated

In the DIT-transmission mode (that is S/PDIF format data transmission): in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.). For more information, see Section 12.5.2.2.2.5, *S/PDIF Coding Format*.

12.5.2.4.4 MCASP Format Units

The MCASP has one transmit data formatting unit and one receive data formatting unit, shared between the device MCASP serializers. These units automatically remap the data bits within the transmitted or received words between a natural format for the device processors (for example, a Q31 representation) and the required format for the external serial device (for example I2S format). During the remapping process, the format unit can also mask off certain bits.

Since all transmitters share the same data formatting unit, the MCASP only supports one transmit format at a time. For example, the MCASP does NOT transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the MCASP only supports one data format at a time, and this format applies to all receiving serializers.

Note

The MCASP can transmit in one format while receiving in a completely different format.

The bit mask and pad stage of each of Tx and Rx format units includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits. The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the MCASP_RFMT/MCASP_XFMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation. The bit order - reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit order reversal is not enabled, then the MCASP will naturally transmit and receive in an LSB first order. Finally, note that the RDATDLY/XDATDLY bits in the MCASP_RFMT/MCASP_XFMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, RDATDLY/XDATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay. The combination of all the options in the MCASP_RFMT/MCASP_XFMT register means that the MCASP supports a wide variety of data formats, both on the serial data lines, and in the device CPU data representation.

12.5.2.4.4.1 Transmit Format Unit

The MCASP transmit formatting unit consists of three stages :

- Bit mask (masks off bits)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB-first or LSB-first)

Figure 12-382 shows the transmit formatting unit.

The MCASP transmitter supports serial formats of:

- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Copyright © 2026 Texas Instruments Incorporated

Hardware support for these serial formats comes from the programmable options in the bitstream format register - MCASP_XFMT:

- XRVRS: bit reverse (1) or no bit reverse (0)
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits

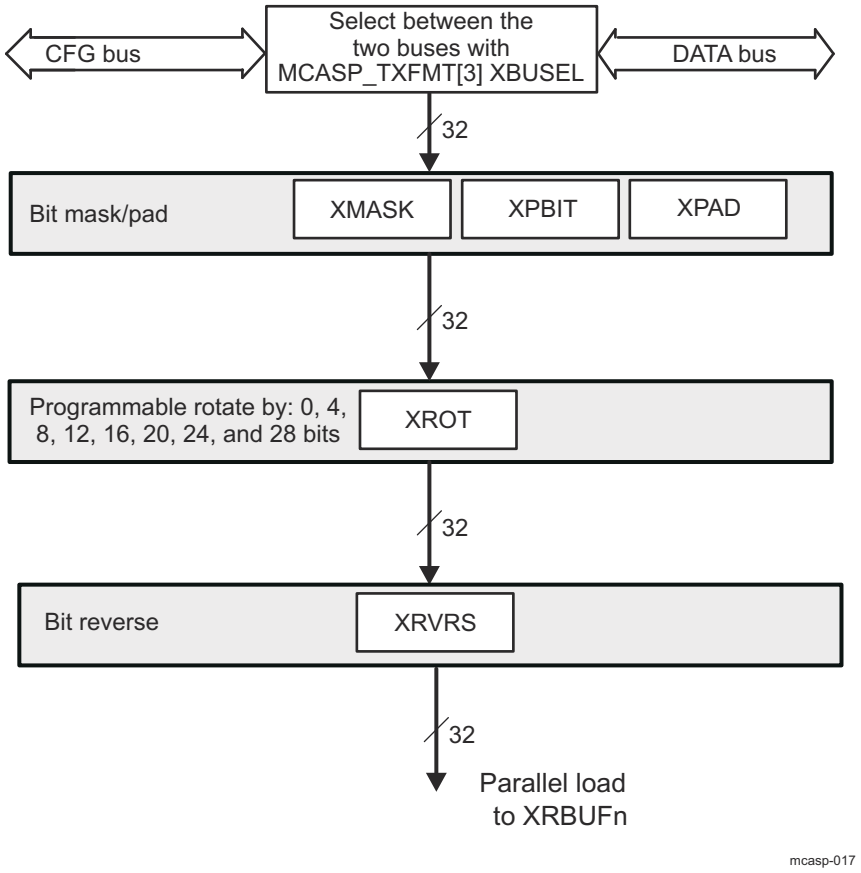


Figure 12-382. Transmit Format Unit

As shown in Figure 12-382, the data to the transmit format unit can come from the configuration port (CFG) or the data port (DATA). The selection is made through the MCASP_XFMT[3] XBUSEL bit. According to port type selected, data transfer has different behaviour. For more details, refer to the Section 12.5.2.4.10.1.3, *Transfers Through the DATA Port*, and Section 12.5.2.4.10.1.4, *Transfers Through the Configuration (CFG) Bus*.

In the transmit format unit (TFU), the input data bits are first masked-off with the MCASP_XMASK[31-0] XMASK contents. The masked data is then right-rotated to the MCASP_XFMT[2-0] XROT bit field positions, to produce the output word for a TDM- or DIT- transmission.

The bit mask stage includes a full 32-bit mask register, allowing selected individual bits to pass through the stage unchanged or be masked off.

12.5.2.4.4.1.1 TDM Mode Transmission Data Alignment Settings

The TDM-mode transmission settings are relevant for I2S-protocol and protocols using more than 2 TDM-slots.

XSSZ should always be programmed to match the slot size of the serial stream.

Copyright © 2026 Texas Instruments Incorporated

Note

Note that, TDM word size is not directly programmed into the MCASP, but rather is used to determine the rotation needed in the XROT field.

The Table 12-303 show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

The Table 12-303 assumes that all slot size (SLOT in Table 12-303) and word size (WORD in Table 12-303) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit, and the data is aligned in the same representation as it is written to the transmitter (typically Q31 or integer).

Table 12-303. MCASP TFU TDM Mode Settings

Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	MCASP_XFMT bits	
			XROT(1)	XRVRS
MSB first(2)	Left aligned	Q31 fraction	0	1
MSB first	Right aligned	Q31 fraction	SLOT - WORD	1
LSB first	Left aligned	Q31 fraction	32 - WORD	0
LSB first	Right aligned	Q31 fraction	32 - SLOT	0
MSB first(2)	Left aligned	Integer	WORD	1
MSB first	Right aligned	Integer	SLOT	1
LSB first	Left aligned	Integer	0	0
LSB first	Right aligned	Integer	(32 - (SLOT - WORD)) % 32	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator
(2) To transmit in I2S format, select MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

12.5.2.4.4.1.2 DIT Mode Transmission Data Alignment Settings

In case of a DIT-mode (S/PDIF protocol) transmission, while left-aligned Q31 data should be right-rotated to a multiple by 4 positions, no right-rotation is required for a right-aligned Q31 data. Because this is a rotation process, not a shifting process, bit 0 gets shifted back into bit 31 during the process.

The MCASP_XFMT[17-16] XDATDLY bit field must be set to a 0-bit delay (0x0 value).

For left-aligned Q31 data, the following transmit format unit settings process the data into right-aligned data, ready for transmission:

- MCASP_XFMT[2-0] XROT =
  - 0x2 (rotate right by 8 bits) - for a 24-bit output audio data
  - 0x3 (rotate right by 12 bits) - for a 20-bit output audio data
  - 0x4 (rotate right by 16 bits) - for a 16-bit output audio data
- MCASP_XFMT[15] XRVRS = 0x0 – Bit reversal is not enabled; the MCASP naturally transmits and receives in a LSB-first order.
- MCASP_XMASK[32] XMASK = 0xFFFFFF00 – 0xFFFF0000
- MCASP_XFMT[14-13] XPAD = 0x0 (Pad extra bits with 0s.)

For right-aligned data, the following transmit format unit settings process the data into right-aligned audio data ready for transmission:

- MCASP_XFMT[2-0] XROT = 0x0 (rotate right by 0 bits regardless of the audio word length)

Copyright © 2026 Texas Instruments Incorporated

- MCASP_XFMT[15] XRVRS = 0x0 – Bit reversal is not enabled; the MCASP naturally transmits and receives in a LSB-first order.
- MCASP_XMASK[32] XMASK = 0x00FFFFFF – 0x0000FFFF
- MCASP_XFMT[14-13] XPAD = 0x0 (Pad extra bits with 0s.)

The example settings provided in Table 12-304 should be applied to MCASP in cases of DIT-transmitting a Q31 data as a 24-bit, 20-bit and 16-bit left- or right- aligned audio word, respectively. Note that the listed settings let the MCASP TFU preserve the most significant bits and cut only the LSBs of the original Q31 CPU data:

Table 12-304. MCASP TFU DIT-Mode Example Settings

Output Audio Word Alignment	Audio Word Length	Right-rotation (multiple of 4-bit positions)	XMASK	XROT
LEFT	16	16	0xFFFF0000	0x4
LEFT	20	12	0xFFFFF000	0x3
LEFT	24	8	0xFFFFFF00	0x2
RIGHT	16	0	0x0000FFFF	0x0
RIGHT	20	0	0x000FFFFF	0x0
RIGHT	24	0	0x00FFFFFF	0x0

Assuming that a Q31 data word 0xFA5AFxxx (where x-marked nibbles of the data are applied as padding bits of the word) is generated on the MCASP CFG (peripheral) port. To transmit a left-aligned 20-bit version of same word, preserving the MSBs, according to the Table 12-304, the user must set XMASK = 0xFFFFF000, and to select a right-rotation to 12 positions (XROT = 0x3).

- After applying 0-s (XPAD = 0) as masking-off bits at the first TFU stage, word is transformed to the word 0xFA5AF000.
- After a rotation by 12 positions to the right is performed in TFU, the 20-bit output word obtained is: 0x000FA5AF. Thus the word gets ready for transmission being mapped with its LS-bit as bit 8 and its MS-bit as bit 27 within a S/PDIF bitstream. This word is shifted in a LSB-to-MSB order (XRVRS = 0x0) out of the XRSR register during a DIT-transmission.

Assuming that a right-aligned Q31 data word - 0x yyyyE4B4 is generated by software on the MCASP CFG (peripheral) port (with the presumption that y-marked nibbles of the input data are applied as padding bits). To transmit a right-aligned 16-bit version of same word, preserving the MSBs, according to the table MCASP TFU Example Settings, user is supposed to set XMASK = 0x0000FFFF, and to select right-rotation to 0 positions (XROT = 0x0).

- After masking-off with 0s at first TFU stage, word is transformed to 0x0000E4B4.
- Since no rotation is applied, the 16-bit output word obtained is actually the one obtained in the masking stage – 0x0000E4B4.

The above examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

12.5.2.4.4.2 Receive Format Unit

The MCASP receive formatting unit consists of three stages:

- Bit mask (masks off bits)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

Figure 12-383 shows the receive format unit (RFU).

Copyright © 2026 Texas Instruments Incorporated

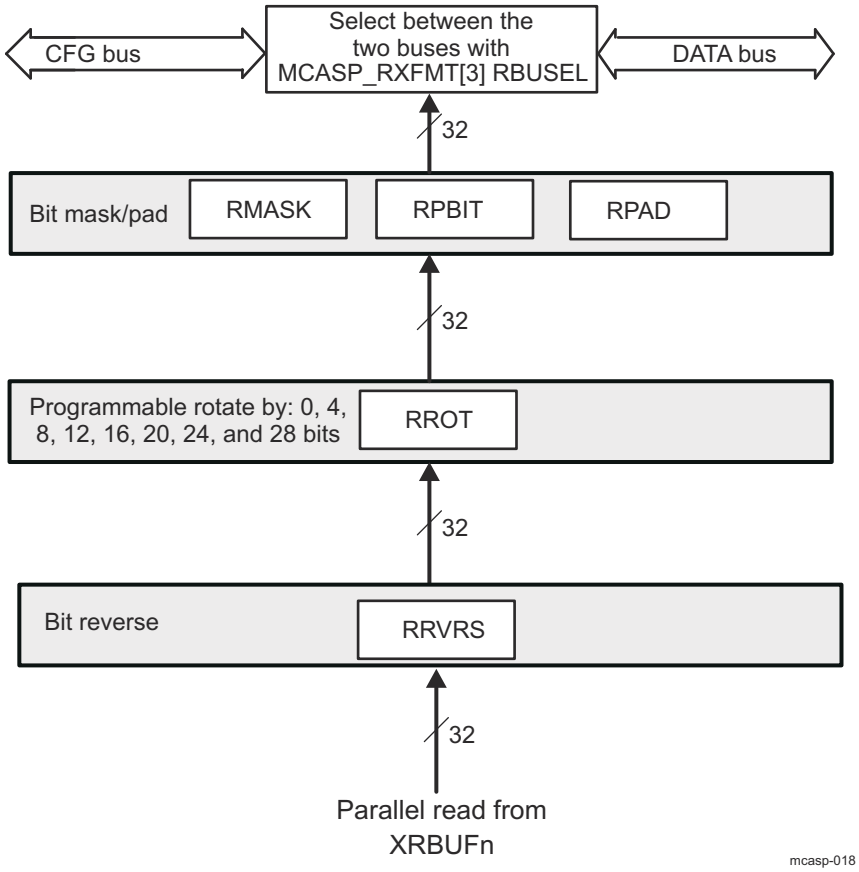


Figure 12-383. Receive Format Unit

The MCASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment when more bits are available per slot than bits per word within the slot, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register - MCASP_RFMT:

- RRVRS: bit reverse (1) or no bit reverse (0)
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits

As shown on Figure 12-383, the data processed in the RFU can be output to host CPU through the configuration port (CFG) or the data port (DATA). The selection is made through the MCASP_RFMT[3] RBUSEL bit. According to port type selected, data transfer has different behaviour. For more details, refer to the Section 12.5.2.4.10.1.3, *Transfers Through the DATA Port*, and Section 12.5.2.4.10.1.4, *Transfers Through the Configuration (CFG) Bus*.

12.5.2.4.4.2.1 TDM Mode Reception Data Alignment Settings

RSSZ should always be programmed to match the slot size of the serial stream.

Copyright © 2026 Texas Instruments Incorporated

Note

Note that the word size is not directly programmed into the MCASP, but rather is used to determine the rotation needed in the RROT field.

Table 12-305 shows the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

Table 12-305. MCASP RFU Settings

Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	MCASP_RFMT bits	
			RROT(1)	RRVRS
MSB first(2)	Left aligned	Q31 fraction	SLOT	1
MSB first	Right aligned	Q31 fraction	WORD	1
LSB first	Left aligned	Q31 fraction	(32 - (SLOT - WORD)) % 32	0
LSB first	Right aligned	Q31 fraction	0	0
MSB first(2)	Left aligned	Integer	SLOT - WORD	1
MSB first	Right aligned	Integer	0	1
LSB first	Left aligned	Integer	32 - SLOT	0
LSB first	Right aligned	Integer	32 - WORD	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator
(2) To receive in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)

The Table 12-305 assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be received in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1. The receive bit mask/pad unit operates on data as the final step of the receive format unit (see Figure 12-383), and the data is aligned in the same representation as it is read from the receiver (typically Q31 or integer).

12.5.2.4.5 MCASP State-Machines

The receive and transmit sections have independent state machines.

Each state-machine controls the interactions between the various units in the MCASP Rx and Tx sections, respectively. In addition, each state-machine keeps track of error conditions and serial port status. No serial transfers can occur until the RX/TX state-machine is released from reset.

The transmit state-machine is controlled by the transmit bitstream format register (MCASP_XFMT) and it reports the MCASP status and error conditions in the transmitter status register (MCASP_XSTAT).

Similarly, the receive state-machine is controlled by the receive bitstream format register (MCASP_RFMT) and it reports the MCASP status and error conditions in the receiver status register (MCASP_RSTAT).

12.5.2.4.6 MCASP TDM Sequencers

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of the MCASP_RTDM/ MCASP_XTDM register and determines if the MCASP should receive/transmit in that time slot.

There are two possibilities for a slot: The MCASP either performs Rx/Tx operations during the time slot (transmit/ receive bit is active), or the MCASP skips Rx/Tx operations during the time slot (transmit/receive bit is inactive). In the latter case, no transfers between the XRBUF and XRSR registers in the serializer would occur during that time slot.

Copyright © 2026 Texas Instruments Incorporated

In addition, during time of inactive slots, the serializers programmed as transmitters place their data output pins - AXRn in a predetermined state - logic low, high, or high impedance (tri-stated) as programmed in each serializer control MCASP_SRCTLn[3-2] DISMOD register. Refer also to Section 12.5.2.4.9.2.1, *TDM Time Slots Generation and Processing*, for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

In case of a DIT-transmission (S/PDIF transfers): the time division multiplexing (TDM) sequencer is used to count the 384 subframes (slots) in the DIT block. If currently transmitting slot 1, slot 2 (next value of the TDM slot counter) should be used during the encode phase to select the appropriate C, V, and U bit, because the data encoded and written to a MCASP_XBUFn register during the current time slot (slot 1) is actually shifted out on the next time slot (n = 0 to 15).

The transmit TDM sequencer is controlled by the MCASP_XTDM register and reports the current transmit slot to the MCASP_XTDMSLOT[9-0] XSLOTCNT bit field.

12.5.2.4.7 MCASP Software Reset

The MCASP can be put into reset through the global transmit and receive control register (*MCASP_GBLCTL*). A valid serial clock must be supplied to the MCASP to assert the software reset bits in the *MCASP_GBLCTL* register.

12.5.2.4.8 MCASP Power Management

Table 12-306 describes power-management features available to the MCASP.

Table 12-306. Local Power-Management Features

Feature	Registers	Description
Target clock stop modes	*MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE	Force-clock stop, no-clock stop, and smart-clock stop modes are available.

CAUTION

No wakeup schema is supported for the MCASP. To ensure a correct behavior after enabling MCASP at device Device Configuration level, the user software is strongly recommended to choose *No Idle* mode, setting *MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE bit field to 0x1. Before disabling MCASP at device Device Configuration level, user software is strongly recommended to choose a *Smart-Idle* mode, setting *MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE bit field to 0x2.

12.5.2.4.9 MCASP Transfer Modes

12.5.2.4.9.1 Burst Transfer Mode

The MCASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two processors. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

When operating in burst frame sync mode (see Figure 12-384), as specified for transmit (MCASP_AFSXCTL[15-7] = 0) and receive (MCASP_AFSRCTL[15-7] RMOD = 0), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration (MCASP_AFSRCTL[4] FRWID = 0, MCASP_AFSXCTL[4] FXWID = 0).

Copyright © 2026 Texas Instruments Incorporated

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the XBUFn (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the RBUFn (for every serializer set to operate as a receiver) has been read.

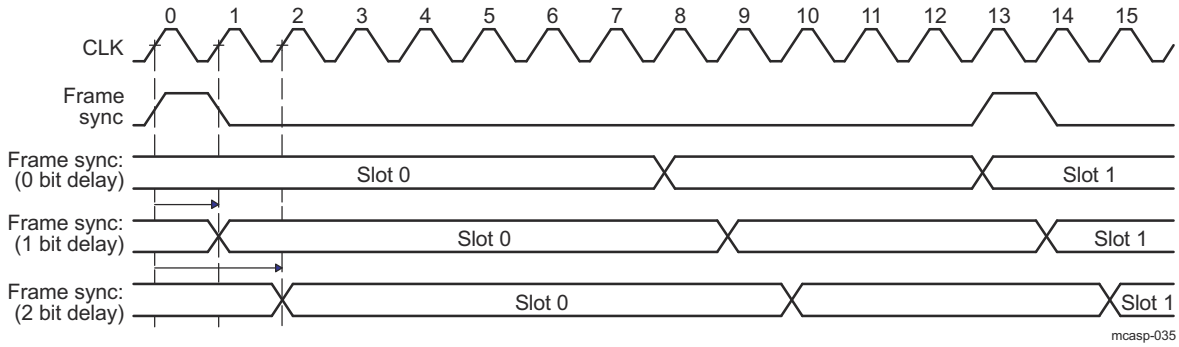


Figure 12-384. Burst Frame Sync Mode

The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- *MCASP_PFUNC*: The clock, frame, data pins must be configured for MCASP function.
- *MCASP_PDIR*: The clock, frame, data pins must be configured to the direction desired.
- *MCASP_PDOUT*, *MCASP_PDIN*, *MCASP_PDCLR*: Not applicable. Leave at default.
- *MCASP_GBLCTL*: Follow the initialization sequence in Section 12.5.2.5.1.2, *MCASP Global Initialization*, to configure this register.
- *MCASP_AMUTE*: Not applicable. Leave at default.
- MCASP_DLBCTL: If loopback mode is desired, configure this register according to Section 12.5.2.4.14, *MCASP Loopback Modes*, otherwise leave this register at default.
- MCASP_DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- MCASP_RMASK/MCASP_XMASK: Mask desired bits according to Section 12.5.2.4.4, *MCASP Format Units*.
- MCASP_RFMT/MCASP_XFMT: Program all fields according to data format desired. See Section 12.5.2.4.4, *MCASP Format Units*.
- MCASP_RFMT/MCASP_XFMT: Clear RMOD/XMOD bits to 0 to indicate burst mode. Clear FRWID/FXWID bits to 0 for single bit frame sync duration. Configure other fields as desired.
- *MCASP_ACLKRCTL*/*MCASP_ACLKXCTL*: Program all fields according to bit clock desired. See Section 12.5.2.4.2, *MCASP Clock and Frame-Sync Configurations*.
- *MCASP_AHCLKRCTL*/*MCASP_AHCLKXCTL*: Program all fields according to high-frequency clock desired. See Section 12.5.2.4.2, *MCASP Clock and Frame-Sync Configurations*.
- MCASP_RTDM/MCASP_XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- MCASP_RINTCTL/MCASP_XINTCTL: Program all fields according to interrupts desired.
- MCASP_RCLKCHK/MCASP_XCLKCHK: Not applicable. Leave at default.
- MCASP_SRCTLn: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default (n = 0 to 15).
- MCASP_DITCSRAi, MCASP_DITCSRBi, MCASP_DITUDRAi, MCASP_DITUDRBi: Not applicable. Leave at default (i = 0 to 5).

12.5.2.4.9.2 Time-Division Multiplexed (TDM) Transfer Mode

The MCASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in Section 12.5.2.2.2.3, *TDM Format*.

Copyright © 2026 Texas Instruments Incorporated

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock
- AFSX - transmit frame sync (or commonly called left/right clock)
- One or more serial data pins, AXRn, whose serializers are configured to transmit

For more details on MCASP transmitting serializers clock and frame sync options, refer to the Section 12.5.2.4.2.1, *Transmit Clock*, and Section 12.5.2.4.2.3, *Frame-Sync Generator*.

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- One or more serial data pins, AXRn, whose serializers are configured to receive

For more details on MCASP receiving serializers clock and frame sync options, refer to Section 12.5.2.4.2.2, *Receive Clock*, and Section 12.5.2.4.2.3, *Frame-Sync Generator*.

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are highlighted in bold:

- *MCASP_PFUNC*: The clock, frame, data pins must be configured for MCASP function.
- *MCASP_PDIR*: The clock, frame, data pins must be configured to the direction desired.
- *MCASP_PDOUT*, *MCASP_PDIN*, *MCASP_PDCLR*: Not applicable. Leave at default.
- *MCASP_GBLCTL*: Follow the initialization sequence is described in the Section 12.5.2.5.2, *MCASP Operational Modes Configuration*.
- *MCASP_AMUTE*: Leave this register at default state.
- MCASP_DLBCTL: If loopback mode is desired, configure this register according to Section 12.5.2.4.14, otherwise leave this register at default.
- MCASP_DITCTL: DITEN must be left at default 0 to select TDM mode (transmitters only).
- MCASP_RMASK/MCASP_XMASK: Mask desired bits according to Section 12.5.2.4.4, *MCASP Format Units*.
- MCASP_RFMT/MCASP_XFMT: Program all fields according to data format desired. See the Section 12.5.2.4.4, *MCASP Format Units*.
- MCASP_AFSRCTL/MCASP_AFSXCTL: Set RMOD/XMOD bits to (0x2 - 0x20) for Rx/Tx (2- 32 slots) TDM mode. In addition, set RMOD to 0x180 if 384-slot TDM stream has to be received by MCASP. Configure other fields as desired.
- *MCASP_ACLKRCTL*/*MCASP_ACLKXCTL*: Program all fields according to bit clock desired. For more inforamtion, refer to Section 12.5.2.4.2, *MCASP Clock and Frame-Sync Configurations*.
- *MCASP_AHCLKRCTL*/*MCASP_AHCLKXCTL*: Program all fields according to high-frequency clock desired. For more details, refer to Section 12.5.2.4.2, *MCASP Clock and Frame-Sync Configurations*.
- MCASP_RTDM/MCASP_XTDM: Program all fields according to the time slot characteristics desired.
- MCASP_XINTCTL: Program all fields according to transmit interrupts desired.
- MCASP_RCLKCHK/MCASP_XCLKCHK: Program all fields according to clock checking desired.
- MCASP_SRCTLn: Program all fields according to serializer operation desired (n = 0 to 15).

Note

The MCASP_DITCSRAi, MCASP_DITCSRBi, MCASP_DITUDRAi, MCASP_DITUDRBi (i = 0 to 5) settings are NOT applicable in TDM transfer modes. They have to be kept at their default values.

12.5.2.4.9.2.1 TDM Time Slots Generation and Processing

TDM mode on the MCASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the MCASP may be configured to participate or to be inactive by configuring MCASP_XTDM and/or MCASP_RTDM registers.

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either MCASP_XTDM or MCASP_RTDM register to determine if the MCASP transmits/receives in that time slot.

Copyright © 2026 Texas Instruments Incorporated

Note

If a MCASP_XTDM/MCASP_RTDM bit defines an active slot (number of slot matches the bit position), the MCASP functions normally during that time slot; otherwise, the MCASP is inactive during that time slot; no update to the buffer occurs, and no event is generated. MCASP (transmit only) data pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bitfield MCASP_SRCTLn[3-2] DISMOD (n = 0 to 15).

Figure 12-385 shows when the transmit DMA event - XINT is generated. See Section 12.5.2.4.10.1, *Data Ready Status and Event/Interrupt Generation* for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless of the previous time slot (slot N - 1) being active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUFn to XRSRn generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.

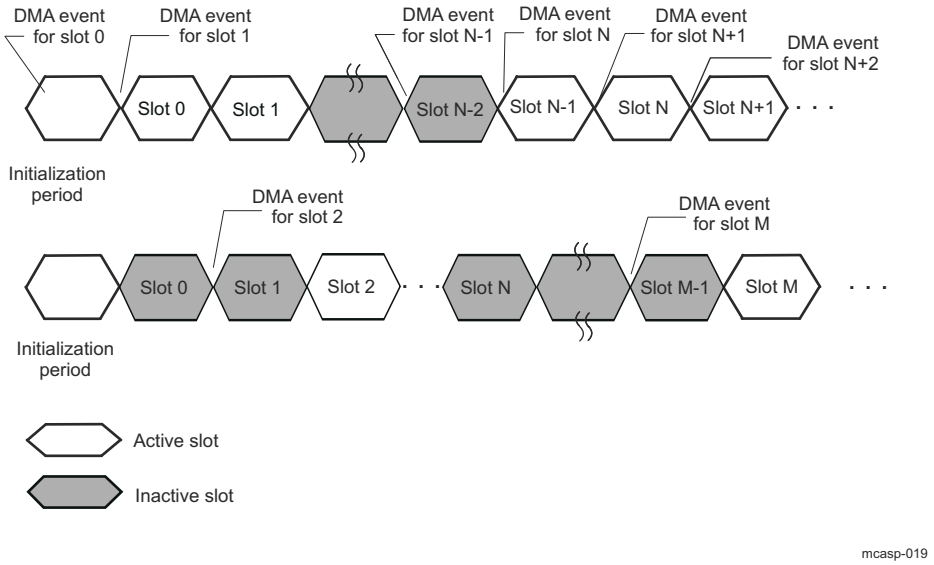


Figure 12-385. Transmit DMA Event (XINT) Generation in TDM Time Slots

12.5.2.4.9.2.2 Special 384-Slot TDM Mode for Connection to External DIR

The MCASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF receiver ICs whose natural block (block corresponds to MCASP frame) size is 384 samples. The receive TDM time slot register (MCASP_RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block). To receive data in DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK
- One or more serial data pins, AXRn, whose serializers have been configured to receive
- For this special DIR mode, the control registers can be configured just as for TDM mode, except set MCASP_AFSRCTL[15-7] RMOD bit field to 384 (0x180) to receive 384 time slots

Copyright © 2026 Texas Instruments Incorporated

12.5.2.4.9.3 DIT Transfer Mode

The DIT transfer mode of the MCASP also supports transmission of audio data in S/PDIF, AES-3, and IEC-60958 formats. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode applies only to a serializer configured as transmitter, not as receiver. For a description of the S/PDIF format, see Section 12.5.2.2.2.5, *S/PDIF Coding Format*.

12.5.2.4.9.3.1 Transmit DIT Encoding

When the MCASP operates in DIT mode, the data transmitted is output as a biphase-mark encoded bitstream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bitstream by the MCASP. The MCASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

Note

The transmit TDM time slot register (MCASP_XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in DIT mode, the following pins are typically required:

- AHCLKX – transmit high-frequency controller clock (The internal clock source can be used instead.)
- One serial data pin (AXRn) of a serializer n configured to transmit.

For DIT Mode Transmission Data Alignment Settings see Section 12.5.2.4.4.1.2, *DIT Mode Transmission Data Alignment Settings*.

If the MCASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23–0 after passing through the first three stages of the transmit format unit.

12.5.2.4.9.3.2 Transmit DIT Clock and Frame-Sync Generation

The DIT transmitter works only in the following configuration:

- In the transmit frame control register (MCASP_AFSXCTL):
  - Internally generated transmit frame sync, FSXM = 1
  - Rising-edge frame sync, FSXP = 0
  - Bit-width frame sync, FXWID = 0
  - 384-slot TDM, XMOD = 1 1000 0000b
- In the transmit clock control register (*MCASP_ACLKXCTL*), ASYNC = 1
- In the transmit bitstream format register (MCASP_XFMT), XSSZ = 1111 (32-bit slot size)

All combinations of AHCLKX and ACLKX are supported.

The following summarizes the register configurations required for DIT mode. DIT mode-specific bit fields are in bold face:

- *MCASP_PFUNC*: The data pin - AXRn must be configured for MCASP function. If AHCLKX is used, it must also be configured for MCASP function. Other pins can be configured to function as GPIOs, if desired.
- *MCASP_PDIR*: The data pin must be configured as output. If internal clock source AUXCLK is used as the reference clock, it may be output as the AHCLKX device level signal by configuring AHCLKX pin as an output.
- *MCASP_GBLCTL*: Global initialization
- *MCASP_AMUTE*: Leave this register at default state.
- MCASP_DITCTL: The DITEN bit must be set to 0b1 to enable DIT mode. Configure other bits as desired.
- MCASP_XMASK: Mask the desired bits, depending upon left-aligned or right-aligned internal data.

Copyright © 2026 Texas Instruments Incorporated

- MCASP_XFMT: XDATDLY = 0. XRVRS = 0. XPAD = 0. XSSZ = Fh (32-bit slot). XBUSEL = configured as desired. The XROT bit is configured, as described in the Section 12.5.2.4.4.1.2 .
- MCASP_AFSXCTL: Configure the bits according to former discussions.
- *MCASP_ACLKXCTL*: ASYNC = 1. Program the CLKXDIV bits to obtain the bit clock rate desired. CLKXM = 1.
- *MCASP_AHCLKXCTL*: Program the HCLKXDIV bits to obtain the high-frequency bit clock rate desired.
- MCASP_XTDM: Set to FFFF FFFFh for all active slots for DIT transfers.
- MCASP_XINTCTL: Program all fields according to the interrupts desired.
- MCASP_XCLKCHK: Program all fields according to the clock checking desired.
- MCASP_SRCTLn: Set SRMOD = 1 (transmitter) for the DIT pins (n = 0 to 15).
- MCASP_DITCSRAi and MCASP_DITCSRBi: Program the channel status bits as desired (i = 0 to 5).
- MCASP_DITUDRAi and MCASP_DITUDRBi: Program the user data bits as desired (i = 0 to 5).

Note

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to 2 serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for TDM (I2S) mode, due to the need to generate Biphase Mark Encoded Data - see Section 12.5.2.2.2.5.1, *Biphase-Mark Code*).

12.5.2.4.9.3.3 DIT Channel Status and User Data Register Files

The channel status registers (MCASP_DITCSRAi and MCASP_DITCSRBi) and user data registers (MCASP_DITUDRAi and MCASP_DITUDRBi) are not double-buffered. Typically, programmers use one of the synchronizing interrupts, such as the last slot, to create an event at a safe time so the register may be updated. In addition, the software reads the transmit TDM slot counter to determine which word of the register is being used (i = 0 to 5).

It is a software requirement to avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it is undetermined whether old or new data is used to encode the bitstream.

The DIT subframe format is defined in Section 12.5.2.2.2.5.2, *S/PDIF Subframe Format*. The channel status information (C) and user data (U) are defined in the following DIT control registers:

- *MCASP_DITCSRA0* to *MCASP_DITCSRA5*: The 192 bits in these six registers contain the channel status information for the left channel within each frame.
- *MCASP_DITCSRA0* to *MCASP_DITCSRA5*: The 192 bits in these six registers contain the channel status information for the right channel within each frame.
- *MCASP_DITCSRA0* to *MCASP_DITCSRA5*: The 192 bits in these six registers contain the user data information for the left channel within each frame.
- *MCASP_DITCSRA0* to *MCASP_DITCSRA5*: The 192 bits in these six registers contain the user data information for the right channel within each frame.
- The S/PDIF block format is shown in Figure 12-376. There are 192 frames within a block (frame 0 to frame 191). There are two subframes within each frame (subframes 1 and 2 for the left and right channels, respectively).

The channel status and user data information sent on each subframe is summarized in Table 12-307.

Table 12-307. Channel Status and User Data for Each DIT Block

Frame	Subframe	Preamble	Channel Status Defined in:	User Data Defined in:
Defined by DITCSRA0, DITCSRB0, DITUDRA0, DITUDRB0				
0	1 (L)	B	DITCSRA0[0]	DITUDRA0[0]
0	2 (R)	W	DITCSRB0[0]	DITUDRB0[0]
1	1 (L)	M	DITCSRA0[1]	DITUDRA0[1]
1	2 (R)	W	DITCSRB0[1]	DITUDRB0[1]
2	1 (L)	M	DITCSRA0[2]	DITUDRA0[2]

Copyright © 2026 Texas Instruments Incorporated

Table 12-307. Channel Status and User Data for Each DIT Block (continued)

Frame	Subframe	Preamble	Channel Status Defined in:	User Data Defined in:
2	2 (R)	W	DITCSRB0[2]	DITUDRB0[2]
…	…	…	…	…
31	1 (L)	M	DITCSRA0[31]	DITUDRA0[31]
31	2 (R)	W	DITCSRB0[31]	DITUDRB0[31]
Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1				
32	1 (L)	M	DITCSRA1[0]	DITUDRA1[0]
32	2 (R)	W	DITCSRB1[0]	DITUDRB1[0]
…	…	…	…	…
63	1 (L)	M	DITCSRA1[31]	DITUDRA1[31]
63	2 (R)	W	DITCSRB1[31]	DITUDRB1[31]
Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2				
64	1 (L)	M	DITCSRA2[0]	DITUDRA2[0]
64	2 (R)	W	DITCSRB2[0]	DITUDRB2[0]
…	…	…	…	…
95	1 (L)	M	DITCSRA2[31]	DITUDRA2[31]
95	2 (R)	W	DITCSRB2[31]	DITUDRB2[31]
Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3				
96	1 (L)	M	DITCSRA3[0]	DITUDRA3[0]
96	2 (R)	W	DITCSRB3[0]	DITUDRB3[0]
…	…	…	…	…
127	1 (L)	M	DITCSRA3[31]	DITUDRA3[31]
127	2 (R)	W	DITCSRB3[31]	DITUDRB3[31]
Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4				
128	1 (L)	M	DITCSRA4[0]	DITUDRA4[0]
128	2 (R)	W	DITCSRB4[0]	DITUDRB4[0]
…	…	…	…	…
159	1 (L)	M	DITCSRA4[31]	DITUDRA4[31]
159	2 (R)	W	DITCSRB4[31]	DITUDRB4[31]
Defined by DITCSRA5, DITCSRB5, DITUDRA5, DITUDRB5				
160	1 (L)	M	DITCSRA5[0]	DITUDRA5[0]
160	2 (R)	W	DITCSRB5[0]	DITUDRB5[0]
…	…	…	…	…
191	1 (L)	M	DITCSRA5[31]	DITUDRA5[31]
191	2 (R)	W	DITCSRB5[31]	DITUDRB5[31]

12.5.2.4.10 MCASP Data Transmission and Reception

The MCASP is serviced by writing data to the MCASP_XBUFn registers for transmit operations, and by reading data from the MCASP_RBUFn registers for receive operations. The MCASP sets status flags and notifies the software whenever data is ready to be serviced. The Section 12.5.2.4.10.1, *Data Ready Status and Event/ Interrupt Generation*, discusses data-ready status in details (n = 0 to 15).

The MCASP transmit/receive XRBUFn buffer can be accessed through one of the two peripheral ports of the device:

- DATA port: This port is dedicated to DMA initiated data transfers on the device for MCASP transmit (Tx) purposes.
- Configuration bus (CFG): The configuration bus- CFG port is used for peripheral configuration control and receive/transmit data transfers initiated by the host CPU in the device.

Copyright © 2026 Texas Instruments Incorporated

Section 12.5.2.4.10.1.3, *Transfers Through the Data Port (DATA)*, and Section 12.5.2.4.10.1.4, *Transfers Through the Configuration Bus (CFG),* discuss how to perform transfers through the data port (DATA) and the configuration port (CFG), respectively.

A device CPU and DMA usages are discussed in Section 12.5.2.4.10.1.5, *Using the device CPUs for MCASP Servicing*, and Section 12.5.2.4.10.1.6, *Using the DMA for MCASP Servicing*, respectively.

MCASP DATA port allows DMAs to access the MCASP transmit buffer more efficiently on the CBASS0, using burst transfers. The physical addresses to access these registers are listed in *DMA Registers*.

12.5.2.4.10.1 Data Ready Status and Event/Interrupt Generation

12.5.2.4.10.1.1 Transmit Data Ready

The transmit data ready flag - XDATA in the MCASP_XSTAT register reflects the data ready status of XRBUFn buffers for all of the active slot transmitting serializers. The XDATA flag is set whenever data is transferred from a transmitting serializer buffer - XRBUFn to its corresponding XRSRn shift register. Thus, the XDATA bit indicates the global event that some of the serializers data buffer - XRBUFn is emptied and ready to accept new data from the host (CPU or DMA). The transmit data ready event is individually indicated per serializer in its corresponding control register MCASP_SRCTLn[4] XRDY status bit. When this bit is set to 0b1, it notifies to host that this serializer Tx buffer must be serviced (written). When MCASP_XBUFn is written to by the host, the MCASP_SRCTLn[4] XRDY is deasserted to 0b0. As XDATA global flag is an OR-event of all active serializers XRDY flags, it indicates to software the moment, when write service operation has to be initiated by the MCASP host (XDATA=0b1). The XRDY flags have to be sequentially scanned by user software to determine which serializer MCASP_XBUFn register has to be currently written. Once all requested MCASP_XBUFn are written, the serializers control XRDY flags are cleared to 0b0. As a consequence, XDATA flag is deasserted to 0b0, to indicate to SW that write operation is completed for all serializers.

The global XDATA flag can be cleared when the MCASP_XSTAT[5] XDATA bit is written to 0b1, or once MCASP_XBUFn registers of all the serializers, that have previously raised their XRDY flags, are written with corresponding active slot data by the host.

Whenever XDATA is set, the XINT event is automatically generated on MCASP[0-2]_XMIT_DMA_EVT line (if enabled in the *MCASP_XEVTCTL* register) to notify the DMA of the MCASP_XBUFn empty status. An interrupt - MCASP[0-2]_XMIT_INTR_PEND can be also generated if the XDATA interrupt is enabled in the MCASP_XINTCTL register (for details, see Section 12.5.2.4.12.1, *Transmit Data Ready Interrupt*).

For DMA requests, the MCASP does not require that MCASP_XSTAT register be read between DMA events. This means that, even if MCASP_XSTAT register already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Because the serializer acts in lockstep, only one DMA event is generated to indicate that the transmit serializer is ready to be written to with new data.

Figure 12-386 shows the timing details of when XINT is generated at the MCASP boundary. In this example, as soon as the last bit (A0) of word A is transmitted, the MCASP sets the XDATA flag and generates an XINT event. However, it takes up to five MCASP interface clocks (XINT latency) before XINT is active at the MCASP boundary. Upon XINT, the CPU can begin servicing the MCASP by writing word C into the MCASP_XBUFn (service time). The CPU must write word C into the MCASP_XBUFn within the setup time required by the MCASP (setup time) (n = 0 to 15).

The maximum service time (see Figure 12-386) can be calculated as:

Service Time = Time Slot – XINT Latency –Setup Time

Copyright © 2026 Texas Instruments Incorporated

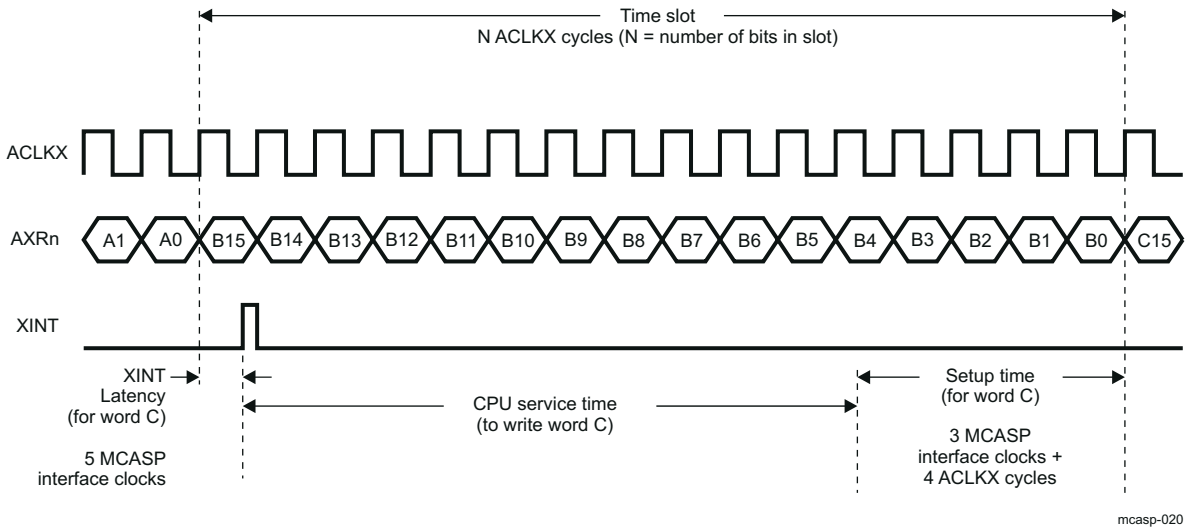


Figure 12-386. Service Time Upon Transmit DMA Event (XINT)

12.5.2.4.10.1.2 Receive Data Ready

Similarly, the receive data ready flag - RDATA in the MCASP_RSTAT register reflects the data ready status of XRBUFn buffers for all of the active slot receiving serializers. The RDATA flag is set whenever data is transferred from a receiving serializer shift register XRSRn to its corresponding XRBUFn data buffer. Thus, the RDATA bit indicates the global event that some of the receivers data buffer - RXBUFn already contains received data (this is, a buffer is full) and is ready to transfer it to the host. The receive data ready event is individually indicated per serializer in its corresponding control register MCASP_SRCTLn [5] RRDY status bit. When this bit is set to 0b1, it notifies to host that this serializer Rx buffer must be serviced (read). When MCASP_RBUFn register is read from the host, the MCASP_SRCTLn [5] RRDY bit is deasserted to 0b0. As RDATA global flag is an OR-event of all active serializers RRDY flags, it indicates to software the moment, when read service operation has to be initiated by the MCASP host (RDATA = 0b1). The RRDY flags have to be sequentially scanned by user software to determine which serializer MCASP_RBUFn register has to be currently read. Once all requested MCASP_RBUFn registers are read, the serializers control RRDY flags are cleared to 0b0. As a consequence, RDATA flag is deasserted to 0b0, to indicate to SW that read operation is completed for all serializers.

The global RDATA flag can be cleared when the MCASP_RSTAT[5] RDATA bit is written to 0b1, or once MCASP_RBUFn registers of all the serializers, that have previously raised their RRDY flags, are read by the host.

Whenever RDATA flag is set, the RINT event is automatically generated on MCASP[0-2]_REC_DMA_EVT line (if enabled in the MCASP_PIDTCTL register) to notify the DMA of the MCASP_RBUFn full status. An interrupt - MCASP[0-2]_REC_INTR_PEND can be also generated if the RDATA interrupt is enabled in the MCASP_RINTCTL register (for details, see Section 12.5.2.4.12.1, *Receive Data Ready Interrupt*).

Figure 12-387 shows the timing details of when RINT event is generated at the MCASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the MCASP sets the RDATA flag and generates an RINT event. However, it takes up to five MCASP interface clocks (RINT Latency) before RINT is active at the MCASP boundary. Upon RINT, the CPU can begin servicing the MCASP by reading Word A from the MCASP_RBUFn (service time). The CPU must read Word A from the MCASP_RBUFn register no later than the setup time required by the MCASP (Setup Time) (n = 0 to 15).

The maximum service time (see Figure 12-387) can be calculated as:
Service Time = Time Slot - RINT Latency - Setup Time

Copyright © 2026 Texas Instruments Incorporated

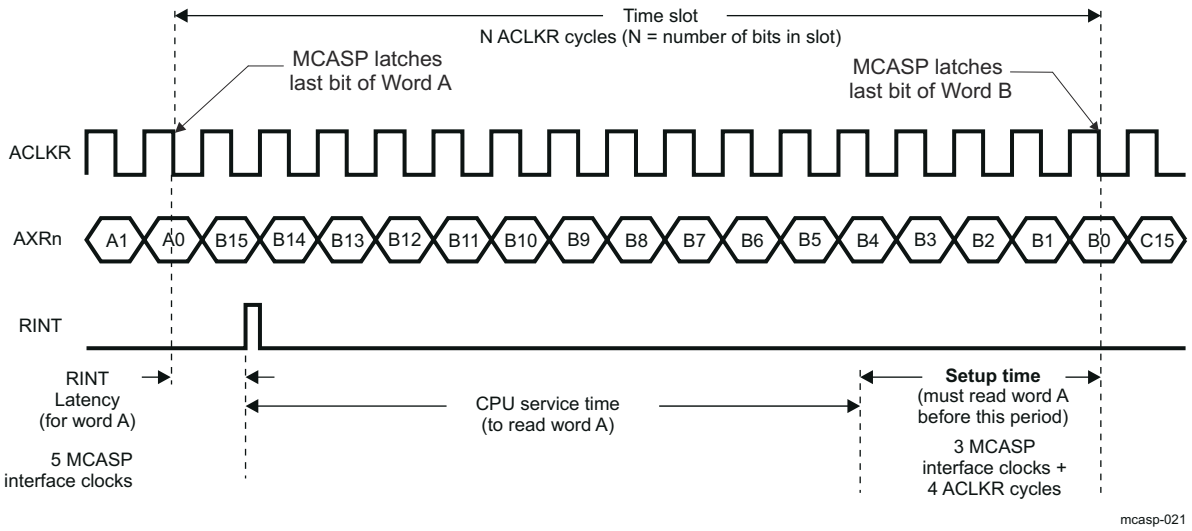


Figure 12-387. CPU Service Time Upon Receive Event (RINT)

12.5.2.4.10.1.3 Transfers Through the Data Port (DATA)

CAUTION

To perform internal transfers through the DATA port, clear the XBUSEL/RBUSEL bit to 0b0 in the MCASP_XFMT/MCASP_RFMT register, respectively. Failure to do so may result in software malfunction.

In a typical MCASP transfer scenario, the DMA Controller write accesses the XRBUFn transmit buffer through the MCASP data port (DATA) on CBASS0 Interconect. CPU hosts can access both XRBUFn transmit and receive data buffers on their corresponding DATA port address via DATA port corresponding address. To perform transfers through the DATA port, simply have the DMA Controller write the MCASP Tx buffer through Interconect DATA port location. Refer to *DMA Registers*. Although the transfer is passed through an integrated AFIFO transmit/receive buffer, the host (DMA or CPU) must follow the described below procedure to access the data buffers of each serializer, regardless the AFIFO is enabled or disabled. The AFIFO operation is described in Section 12.5.2.4.11.

For accesses through the DATA port, the DMA/CPU services all the serializers through accessing only a single address. In addition, as can be seen in *DMA Registers*, the same physical DATA port address is used regardless of a read or write access is performed. The MCASP automatically cycles through the active slot transmitting/receiving serializers, internally generating the appropriate offsets.

Note

DATA port allows the DMA/CPU to automatically access only the data buffers. There is no way for DMA/CPU to access the MCASP configuration registers addressing their corresponding MCASP DATA port.

For transmit operations through the DATA port, the host must always write to the same transmit buffer DATA port address (which is same than the receive buffer DATA port adress) to service all of the active slot transmitting serializers. Regardless of MCASP serializer 0 being configured inactive or active, the user software must always configure the destination address to match the DATA port location of TXBUF buffer (See *DMA Registers*).

In addition, the DMA/CPU must write the buffers of all transmitting serializers in incremental (although not necessarily consecutive) order. For example, if only serializers 1 and 3 are set up as active transmitters, to the same transmit buffer DATA port address twice - first data for serializer 1 and second data for serializer 3

Copyright © 2026 Texas Instruments Incorporated

upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

Note

For write transfers through MCASP DATA port it is preferable to use DMA on corresponding Interconect. This is because DMAs initiated traffic gets better advantage of the burst transfers supported by DATA port.

For receive operations through the DATA port, the DMA/CPU must always read from the same receive buffer DATA port address (which is same than the transmit buffer DATA port adress) to service all of the active slot receiving serializers. Regardless of MCASP serializer 0 being configured inactive or active, the user software must always configure the DMA/CPU source address to match the DATA port location of RXBUF buffer (See *DMA Registers*).

In addition, reads from the receive buffer for all active slot receiving serializers through the Rx DATA port return data in incremental (although not necessarily consecutive) order. For example, if serializers 0, 1 and 3 are set up as active receivers, the CPU should read from the same receive buffer DATA port address three times to obtain data for serializers 0,1 and 3 in this exact order, upon each receive data ready event.

Note

To service a serializer for a transmit or receive operation through the MCASP DATA port, the initiator always writes (preferably DMA) and reads from the same address (refer to *DMA Registers*), respectively.

Note

When transmitting through the DATA port, the DMA/CPU must write data (at the same address) to each serializer configured as *active* (active slot selected in MCASP_XTDM) and *transmit* (Tx enabled in MCASP_SRCTLn) within each time slot. Failure to do so results in a buffer underrun condition (see Section 12.5.2.4.15.1, *Buffer Underrun Error - Transmitter*). Similarly, when DMA/CPU receives, data must be read from each serializer configured as *active* (active slot selected in MCASP_RTDM) and *receive* (Rx enabled in MCASP_SRCTLn) within each time slot. Failure to do results in a buffer overrun condition (see Section 12.5.2.4.15.2, *Buffer Overrun Error - Receiver*) (n = 0 to 15).

12.5.2.4.10.1.4 Transfers Through the Configuration Bus (CFG)

CAUTION

To perform internal transfers through the configuration bus, set the XBUSEL/RBUSEL bit to 1 in the MCASP_XFMT/MCASP_RFMT registers, respectively. Failure to do so may result in software malfunction.

Note

MCASP[0-2] whose data ports are accessible directly via CBASS0 do not support FIFO/constant addressing modes. Incrementing transfers must be used instead.

In this method, the CPU accesses the XRBUFn transmit or receive buffer through corresponding configuration bus (CFG) address.

The exact XRBUFn transmit/receive buffer physical address for any particular serializer is determined by adding the transmit/receive buffer alias register offset for that particular serializer to the base address of MCASP CFG port actual for CBASS0 accesses. The XRBUFn buffer of the n-th serializer configured as a transmitter is aliased

Copyright © 2026 Texas Instruments Incorporated

- MCASP_XBUFn in the CFG port address space. For example, the XRBUF2 transmit buffer is mapped as the MCASP_XBUF2 register. Similarly, the XRBUFn buffer of the n-th serializer configured as a receiver is aliased
- MCASP_RBUFn in the CFG port address space. For example, the XRBUF3 receive buffer is mapped as the MCASP_RBUF3 register.

Accessing the XRBUF through the DATA port (see Section 12.5.2.4.10.1.3) is different than CFG port accesses because the DATA port access demands the same physical address, regardless of transfer direction or current channel index , while accessing through the peripheral configuration port - CFG, the CPU must provide the exact MCASP_XBUFn or MCASP_RBUFn address upon accessing n-th serializer TX or RX buffer, respectively. For more details about MCASP_XBUFn and MCASP_RBUFn addresses corresponding to MCASP CFG port, see *CFG Registers* (n = 0 to 15).

12.5.2.4.10.1.5 Using a Device CPU for MCASP Servicing

The device CPUs can be used to service the MCASP transmit channels through interrupts (upon MCASP[0-2]_XMIT_INTR_PEND and MCASP[0-2]_REC_INTR_PEND interrupts). Because these interrupt events are connected to device COMPUTE_CLUSTER0, PRUSS, MAIN2MCU_LVL_INTRTR0, R5FSS0/1_INTRTR0, C66SS0/1_INTRTR0, R5FSS0/1 modules, they could be software mapped to input interrupt lines of any device CPU. Another way to service the transmit and receive channels, a polling of the XDATA bit in the MCASP_XSTAT register and RDATA bit in the MCASP_RSTAT register can be performed by device CPUs, respectively. As discussed in Section 12.5.2.4.10.1.3, *Transfers Through the Data Port (DATA)*, and Section 12.5.2.4.10.1.4, *Transfers Through the Configuration Bus (CFG)*, the device CPUs can access MCASP XRBUF serializer buffer through their corresponding DATA and CFG port locations.

To use the device CPUs to service the MCASP through interrupts, the XDATA/RDATA bit must be enabled in the respective MCASP_XINTCTL/MCASP_RINTCTL registers, to generate interrupts MCASP[0-2]_XMIT_INTR_PEND/MCASP[0-2]_REC_INTR_PEND to the device CPUs upon data ready

12.5.2.4.10.1.6 Using the DMA for MCASP Servicing

Note

The associated static Transfer Request (TR) operations of PDMA0, located in front of MCASP, must be configured to match the MCASP configuration. For more information, refer to chapter *DMA Controllers*.

The typical scenario is to use the DMA to service the MCASP transmit and receive logic through the DATA port. The transfer passes through integrated AFIFO transmit/receive buffer. If AFIFO is enabled, DMA requests are collected and fed to a device DMA controller (see Figure 12-377). The data transfer is managed by the AFIFO according to generated transmit and receive events in the MCASP and data is fed to transmit buffers and fetched from receive buffers as described in Section 12.5.2.4.11, *MCASP Audio FIFO (AFIFO)*. The generation of transmit and receive request is described below. After generation of transmit/receive DMA events from MCASP module, these events are collected in AFIFO and on specific AFIFO conditions described in Section 12.5.2.4.11, *MCASP Audio FIFO (AFIFO)* the requests (transmit or receive) are forwarded to a DMA controller via MCASP[0-2]_XMIT_DMA_EVT and MCASP[0-2]_REC_DMA_EVT outputs. If the AFIFO is disabled (default state) it is transperrant for the MCASP module and all request are directly sent to the DMA controller.

Copyright © 2026 Texas Instruments Incorporated

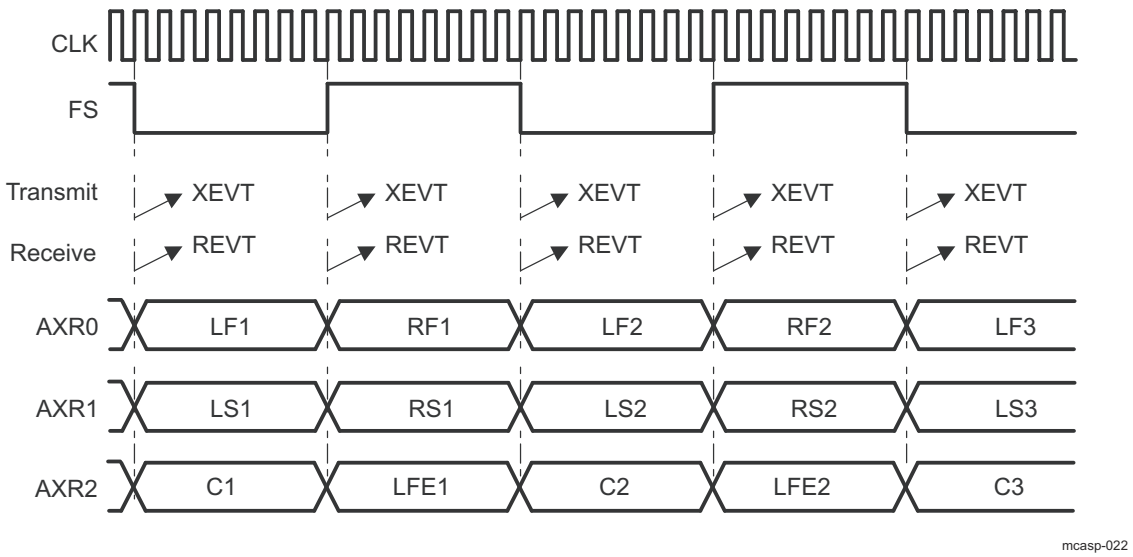


Figure 12-388. DMA Transmit and Receive Event in an Audio Example – One Event

In transmit mode, the DMA event - XINT (MCASP[0-2]_XMIT_DMA_EVT output), which is triggered upon each XDATA transition from 0 to 1, is used to service the MCASP TXBUFn transmit buffers. In receive mode, the DMA event RINT (MCASP[0-2]_REC_DMA_EVT output) which is triggered upon each RDATA transition from 0 to 1, is used to service the MCASP RXBUFn receive buffers.

Figure 12-388 is an example of an audio system with six audio channels (LF, RF, LS, RS, C and LFE) transmitted or received through the MCASP signals - AXR0, AXR1 and AXR2. It shows the points at which events XINT/RINT are triggered.

In Figure 12-388, a Tx DMA event XINT is triggered on each time slot. In the example, XINT is triggered for each of the transmit audio channel time slot (time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Transmit DMA events are generated automatically upon transmit data ready, provided that DMA TX requests generation is enabled in the *MCASP_XEVTCTL* register. Similarly, Rx DMA event RINT is triggered for each of the receive audio channel time slot. Receive DMA events are generated automatically upon receive data ready, provided that DMA RX requests generation is enabled in the MCASP_PIDTCTL register.

12.5.2.4.11 MCASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). The RFIFO and the WFIFO are the same size: 64 32-bit Words. To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See Figure 12-389 for a high-level block diagram of the AFIFO. The AFIFO may be enabled/disabled and configured via the MCASP_WFIFOCTL and MCASP_RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the MCASP.

Copyright © 2026 Texas Instruments Incorporated

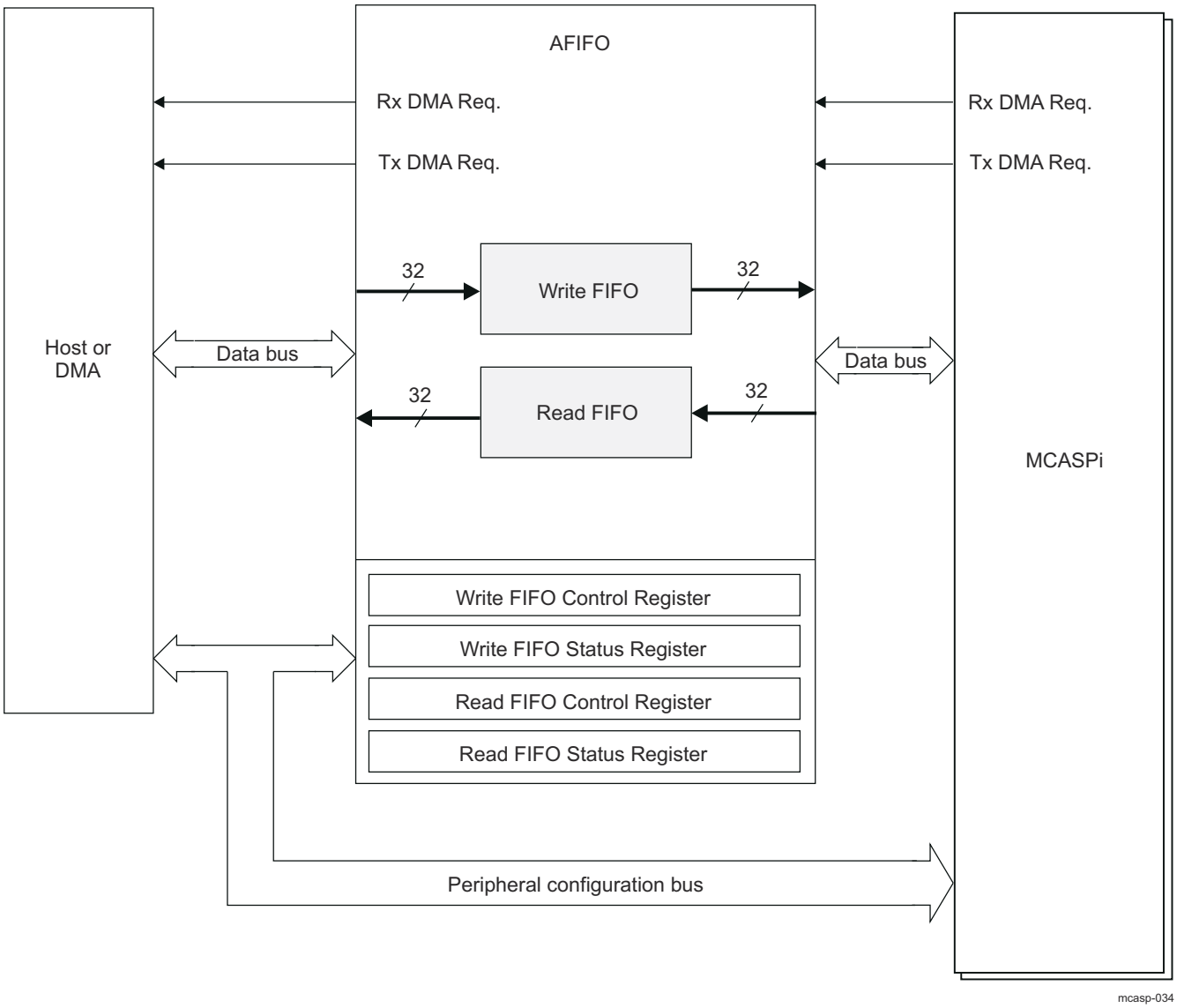


Figure 12-389. MCASP Audio FIFO (AFIFO) Block Diagram

12.5.2.4.11.1 AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the MCASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the MCASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the MCASP. When the Write FIFO is enabled, transmit DMA requests from the MCASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller. If the Write FIFO is enabled, upon a transmit DMA request from the MCASP, the WFIFO writes WNUMDMA 32-bit words to the MCASP if and when there are at least WNUMDMA words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes WNUMDMA words to the MCASP (see description for the MCASP_WFIFOCTL[7-0] WNUMDMA). If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded. Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

12.5.2.4.11.1.1 Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the

Copyright © 2026 Texas Instruments Incorporated

host or DMA controller is reduced. If the Write FIFO has space to accept WNUMEVT 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once WNUMEVT words have been written to the FIFO, it checks again to see if there is space for WNUMEVT 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled. Note that if transmit DMA event pacing is desired, MCASP_WFIFOCTL[15-8] WNUMEVT bit field should be set to a non-zero integer multiple of the value in MCASP_WFIFOCTL[7-0] WNUMDMA bit field. If transmit DMA event pacing is not desired, then the value in MCASP_WFIFOCTL[15-8] WNUMEVT bit field should be set equal to the value in MCASP_WFIFOCTL[7-0] WNUMDMA bit field.

12.5.2.4.11.2 AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from MCASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the MCASP generates receive DMA requests as needed; the AFIFO is “invisible” to the MCASP. When the Read FIFO is enabled, receive DMA requests from the MCASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller. If the Read FIFO is enabled and the MCASP makes a receive DMA request, the RFIFO reads RNUMDMA 32-bit words from the MCASP, if and when the RFIFO has space for RNUMDMA words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads RNUMDMA words from the MCASP (see description for theMCASP_RFIFOCTL[7-0] RNUMDMA). If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than RNUMEVT words, those words will be read correctly, emptying the FIFO.

12.5.2.4.11.2.1 Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced. If the Read FIFO contains at least RNUMEVT 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once RNUMEVT 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another RNUMEVT words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty. Note that if receive DMA event pacing is desired, MCASP_RFIFOCTL[15-8] RNUMEVT bit field should be set to a non-zero integer multiple of the value in MCASP_RFIFOCTL[7-0] RNUMDMA bit field. If receive DMA event pacing is not desired, then the value in MCASP_RFIFOCTL[15-8] RNUMEVT bit field should be set equal to the value in MCASP_RFIFOCTL[7-0] RNUMDMA bit field.

12.5.2.4.11.3 Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete. If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete. If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

12.5.2.4.12 MCASP Events and Interrupt Requests

Table 12-308 lists all the transmit event flags. Table 12-309 lists all the Receive event flags. Source of each of these TX/RX events can be a TX/RX channel from any MCASP serializer configured as transmitter or receiver respectively.

Table 12-308. TX Events

Event Mask[1]	Event Flag	Map to [1]	Description
MCASP_XINTCTL[0] XUNDRN	MCASP_XSTAT[0] XUNDRN	MCASP[0-2]_XMIT_INTR_PEND	Transmit buffer underrun
MCASP_XINTCTL[1] XSYNCERR	MCASP_XSTAT[1] XSYNCERR	MCASP[0-2]_XMIT_INTR_PEND	Unexpected transmit frame sync
MCASP_XINTCTL[2] XCKFAIL	MCASP_XSTAT[2] XCKFAIL	MCASP[0-2]_XMIT_INTR_PEND	Transmit clock failure

Copyright © 2026 Texas Instruments Incorporated

Table 12-308. TX Events (continued)

Event Mask(1)	Event Flag	Map to (1)	Description
MCASP_XINTCTL[3] XDMAERR	MCASP_XSTAT[7] XDMAERR	MCASP[0-2]_XMIT_INTR_PEND	DATA port transmit error
MCASP_XINTCTL[4] XLAST	MCASP_XSTAT[4] XLAST	MCASP[0-2]_XMIT_INTR_PEND	Transmit last slot interrupt
MCASP_XINTCTL[5] XDATA	MCASP_XSTAT[5] XDATA	MCASP[0-2]_XMIT_INTR_PEND	Transmit data-ready interrupt
MCASP_XINTCTL[7] XSTAFRM	MCASP_XSTAT[6] XSTAFRM	MCASP[0-2]_XMIT_INTR_PEND	Transmit start of frame interrupt
n.a.	MCASP_XSTAT[8] XERR	n.a.	OR-event of all Tx-error events: (XDMAERR | XCKFAIL | XUNDRN | XSYNCERR). It is cleared ONLY when all error flags are cleared
n.a.	MCASP_XSTAT[3] XTDMSLOT	n.a.	Qualifies the current TDM slot as an odd or an even slot.

(1) Every MCASP module generates separate interrupt event.

Table 12-309. RX Events

Event Mask(1)	Event Flag	Map to(1)	Description
MCASP_RINTCTL[0] ROVRN	MCASP_RSTAT[0] ROVRN	MCASP[0-2]_REC_INTR_PEND	Receive buffer overrun
MCASP_RINTCTL[1] RSYNCERR	MCASP_RSTAT[1] RSYNCERR	MCASP[0-2]_REC_INTR_PEND	Unexpected receive frame sync
MCASP_RINTCTL[2] RCKFAIL	MCASP_RSTAT[2] RCKFAIL	MCASP[0-2]_REC_INTR_PEND	Receive clock failure
MCASP_RINTCTL[3] RDMAERR	MCASP_RSTAT[7] RDMAERR	MCASP[0-2]_REC_INTR_PEND	DATA port receive error
MCASP_RINTCTL[4] RLAST	MCASP_RSTAT[4] RLAST	MCASP[0-2]_REC_INTR_PEND	Receive last slot
MCASP_RINTCTL[5] RDATA	MCASP_RSTAT[5] RDATA	MCASP[0-2]_REC_INTR_PEND	Receive data-ready
MCASP_RINTCTL[7] RSTAFRM	MCASP_RSTAT[6] RSTAFRM	MCASP[0-2]_REC_INTR_PEND	Receive start of frame
n.a.	MCASP_RSTAT[8] RERR	n.a.	OR-event of all Rx-error events: (RDMAERR | RCKFAIL | ROVRN | RSYNCERR). RERR event is cleared once all error flags are cleared.
n.a.	MCASP_RSTAT[3] RTDMSLOT	n.a.	Qualifies the current TDM slot as an odd or an even slot.

(1) Every MCASP module generates separate interrupt event.

Software has to read the MCASP_XSTAT/MCASP_RSTAT register to determine which event occurs at a global level for MCASP Tx/Rx logic. In addition user software has to scan the XRDY/RRDY read-only flags in the MCASP_SRCTLn registers to determine which active serializer is the actual source of the event.

A Tx interrupt line (MCASP[0-2]_XMIT_INTR_PEND) is asserted (active high) when one of the MCASP_XSTAT notified events occurs, provided that it is enabled in its corresponding MCASP_XINTCTL bit. Similarly, a Rx interrupt line (MCASP[0-2]_REC_INTR_PEND) is asserted (active high) when one of MCASP_RSTAT notified events occurs, provided that it is enabled in its corresponding MCASP_RINTCTL bit. See also Section 12.5.2.4.12.4, *Multiple Interrupts* and the Section 12.5.2.4.10.1, *Data Ready Status and Event/Interrupt Generation* (n = 0 to 15).

Copyright © 2026 Texas Instruments Incorporated

12.5.2.4.12.1 Transmit Data Ready Event and Interrupt

The transmit data-ready interrupt (XDATA) is generated if the MCASP_XSTAT[5] XDATA bit is 1 and MCASP_XINTCTL[5] XDATA bit is enabled. The Section 12.5.2.4.10.1, *Data Ready Status and Event/Interrupt Generation*, provides details on when XDATA is set in the MCASP_XSTAT register.

A transmit-start-of-frame interrupt (XSTAFRM) is triggered by the recognition of a transmit frame sync.

A transmit-last-slot interrupt (XLAST) is a qualified version of the data-ready interrupt (XDATA). It has the same behavior than the data-ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended is the next-to-last TDM slot, the current slot is the last slot).

12.5.2.4.12.2 Receive Data Ready Event and Interrupt

The receive data-ready interrupt (RDATA) is generated if the MCASP_RSTAT[5] RDATA bit is 1 and MCASP_RINTCTL[5] RDATA bit is enabled. The Section 12.5.2.4.10.1, *Data Ready Status and Event/Interrupt Generation*, provides details on when the MCASP_RSTAT[5] RDATA bit is set.

A receiver start of frame (RSTAFRM) interrupt is triggered by the recognition of a receiver frame sync.

A receiver last slot (RLAST) interrupt is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

12.5.2.4.12.3 Error Interrupt

Upon detection, the following error conditions generate interrupt flags:

In the transmit status register (MCASP_XSTAT):

- Transmit underrun (XUNDRN)
- Unexpected transmit frame sync (XSYNCERR)
- Transmit clock failure (XCKFAIL)
- Transmit DATA port error (XDMAERR)

Each interrupt source also has a corresponding enable bit in the transmit interrupt control register (MCASP_XINTCTL). If the enable bit is set, an interrupt is requested when the interrupt flag is set in MCASP_XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

In the receive status register (MCASP_RSTAT) :

- Receiver overrun (ROVRN)
- Unexpected receive frame sync (RSYNCERR)
- Receive clock failure (RCKFAIL)
- Receive DATA port error (RDMAERR)

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (MCASP_RINTCTL). If the enable bit is set, an interrupt is requested when the interrupt flag is set in MCASP_RSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

12.5.2.4.12.4 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in MCASP_XSTAT is set and the interrupt is enabled in MCASP_XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the MCASP transmit interrupt port, but that request has not yet been serviced.
- **Serviced:** The CPUs write to MCASP_XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the serializer with the interrupt flag set in MCASP_XSTAT/ MCASP_RSTAT and the interrupt enabled in MCASP_XINTCTL/MCASP_RINTCTL generates a request on the MCASP transmit or receive interrupt port.

Copyright © 2026 Texas Instruments Incorporated

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the MCASP transmit or receive interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the MCASP transmit or receive interrupt port.

The interrupt is serviced with the CPU writing to MCASP_XSTAT/MCASP_RSTAT. If any interrupt requests are active after the write, a new request is generated on the MCASP transmit or receive interrupt port.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

12.5.2.4.13 MCASP DMA Requests

The MCASP can generate one DMA request to the DMA controller to transmit (MCASP[0-2]_XMIT_DMA_EVT) or receive (MCASP[0-2]_REC_DMA_EVT) data. A DMA request to transmit data is generated if the *MCASP_XEVTCTL*[0] XDATDMA bit is cleared. A DMA request to receive data is generated if the MCASP_PIDTCTL[0] RDATDMA bit is cleared.

12.5.2.4.14 MCASP Loopback Modes

The MCASP features a digital loopback mode (DLB) that allows loopback test transfers in TDM mode between MCASP transmitters and receivers within the same device. In loopback mode, the output of a transmit serializer is connected internally to the input of a receive serializer. Therefore, a receiver data can be checked against a transmitter data to ensure that the MCASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode (XMOD = 0x180) or burst mode (XMOD = 0).

Figure 12-390 shows the basic logical connection of the serializers in loopback mode.

Copyright © 2026 Texas Instruments Incorporated

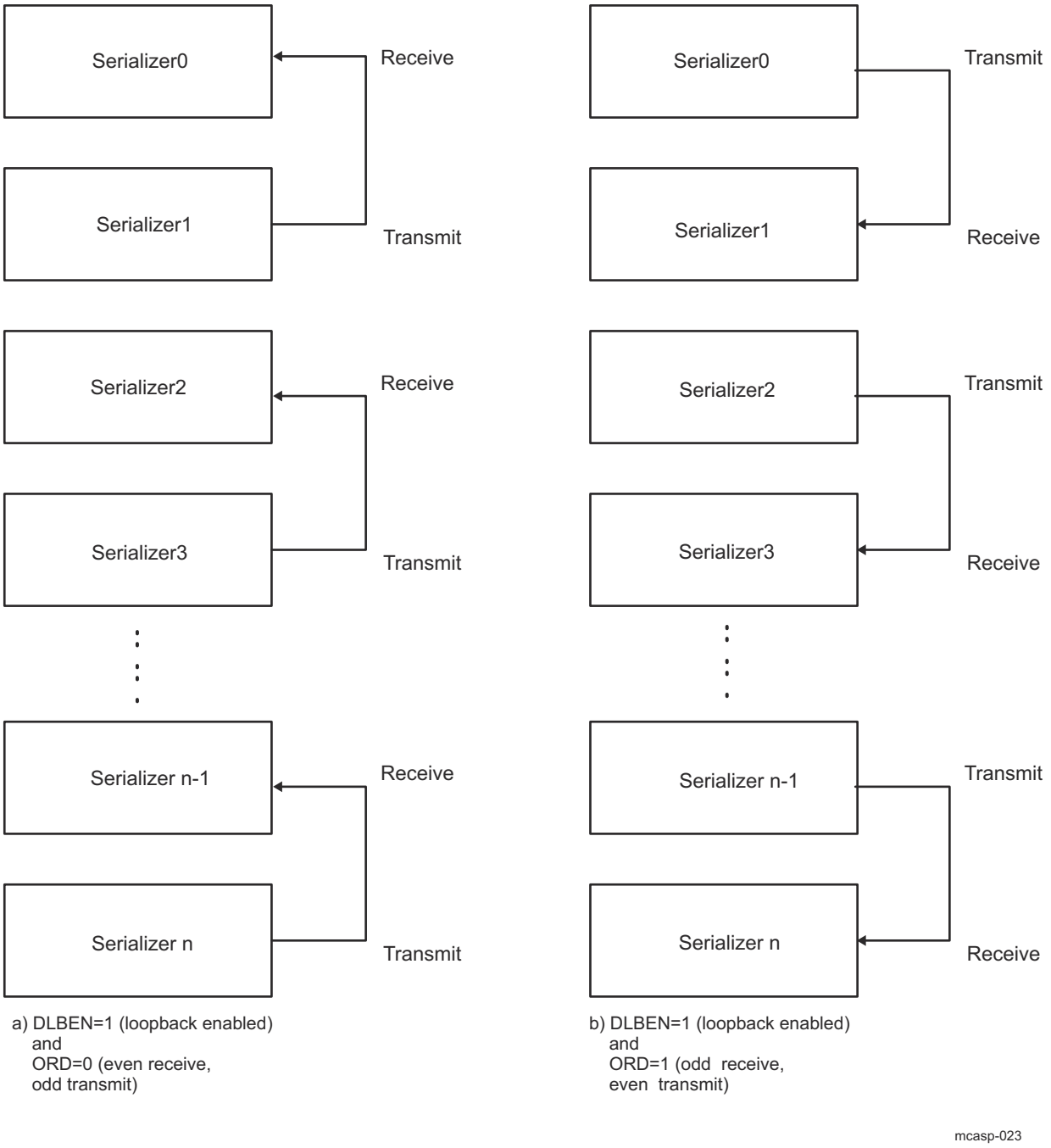


Figure 12-390. MCASP Serializers Operation in Loopback Mode

Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register - MCASP_DLBCTL as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, the odd serializers must be configured as transmitters and even serializers as receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, the even serializers must be configured as transmitters and odd serializers as receivers.

User can choose in software (the MCASP_DLBCTL[4] IOLBEN bit) between a MCASP module internal loopback and a device I/O level loopback.

Copyright © 2026 Texas Instruments Incorporated

When a **MCASP internal loopback** is selected (MCASP_DLBCTL[4] IOLBEN = 0b0), it is NOT necessary to configure *MCASP_PFUNC* and *MCASP_PDIR* registers for MCASP pin settings. Nevertheless, data can be optionally made externally visible at the I/O pin of the transmit serializer, if the pin is configured as a MCASP output pin by setting the corresponding *MCASP_PFUNC* bit to 0 (this is, to function as MCASP, not GPIO) and *MCASP_PDIR* bit to 1 (output).

When a **device I/O level loopback** is selected (MCASP_DLBCTL[4] IOLBEN = 0b1), the *MCASP_PFUNC* and *MCASP_PDIR* registers must be configured with the appropriate settings for all AXRn pins, according to ORD bit configuration.

In case of device I/O loopback, the connectivity is externally applied between device pads (this is, reaching device I/O buffers).

When in loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the MCASP. The transmit and receive sections operate synchronously. This is achieved by setting the MCASP_DLBCTL[3-2] MODE bit field to 0x1 and the *MCASP_ACLKXCTL*[6] ASYNC bit to 0b0.

12.5.2.4.14.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format :

- The MCASP_DLBCTL[0] DLBEN bit must be set to 0b1 to enable a loopback mode. It must be kept at 0b0 during normal MCASP operation.
- The MCASP_DLBCTL[4] IOLBEN bit must be set to select between internal (MCASP local) loopback mode or device I/O level loopback mode.
- The MCASP_DLBCTL[3-2] MODE bit field must be set to 0x1 for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The MCASP_DLBCTL[1] ORD bit must be programmed appropriately to select odd or even serializers to be transmitters or receivers.
- The corresponding serializers must be configured accordingly.
- The *MCASP_ACLKXCTL*[6] ASYNC bit must be cleared to 0b0 to ensure synchronous transmit and receive operations.
- The MCASP_AFSRCTL[15-7] RMOD bit field and MCASP_AFSXCTL[15-7] XMOD bit field must be set within range (0x2 - 0x20) to indicate TDM mode.

Note

Loopback mode does not apply to DIT or burst mode, because MCASP receivers do NOT natively support DIR - reception.

12.5.2.4.15 MCASP Error Reporting

The MCASP includes error-checking capability for the serial protocol and data underrun. In addition, the MCASP includes a timer that continually measures the high-frequency controller clock every 32 AHCLKX clock cycles. The value of the timer can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the controller clock goes out of a specified range.

When one or more errors (software selectable) are detected, an interrupt can be generated if desired, based on one or more error sources.

12.5.2.4.15.1 Buffer Underrun Error -Transmitter

A buffer underrun occurs when a serializer is instructed by the transmit state-machine to transfer data from XRBUFn buffer to XRSRn shift register, but the corresponding (MCASP_XBUFn) register has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state-machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The MCASP_XSTAT[0] XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the host explicitly writes 1 to the XUNDRN bit to clear it (n = 0 to 15).

Copyright © 2026 Texas Instruments Incorporated

In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at 128 bfs). By shifting out a pair of zeros, a clock can be recovered on the receiver. To recover, reset the MCASP and restart with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the MCASP and start again with the proper initialization.

12.5.2.4.15.2 Buffer Overrun Error-Receiver

A buffer overrun occurs when a serializer is instructed to transfer data from XRSRn shift register to XRBUFn receiver buffer, but the corresponding MCASP_RBUFn register has not yet been read since the last time the transfer occured. When this occurs, the receiver state machine sets the overrun flag - ROVRN. However, the individual serializer writes over the data in the XRBUFn buffer register (destroying the previous sample) and continues shifting (n = 0 to 15).

An overrun is checked only once per time slot. The MCASP_RSTAT[0] ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the host catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the host explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

12.5.2.4.15.3 DATA Port Error - Transmitter

A transmit DATA port error, as indicated by the MCASP_XSTAT[7] XDMAERR bit, occurs when the DMA or device CPU writes more words to the DATA port of the MCASP than it should.

The MCASP_XSTAT[7] XDMAERR = 0b1 indicates that the DMA or device CPU wrote too many words to the MCASP DATA port for a given transmit DMA event. Writing too few words results in a transmit underrun error setting the MCASP_XSTAT[0] XUNDRN bit.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the MCASP and the DMA or device CPU. The MCASP transmitter and the DMA must be reinitialized to resynchronize them.

12.5.2.4.15.4 DATA Port Error - Receiver

A receive DATA port error, as indicated by the MCASP_RSTAT[7] RDMAERR bit, occurs when the DMA or device CPU reads more words from the DATA port of the MCASP than it should.

The MCASP_RSTAT[7] RDMAERR bit indicates that the DMA or device CPU read too many words from the MCASP DATA port for a given receive RINT event. Reading too few words results in a receiver overrun error setting the MCASP_RSTAT[0] ROVRN bit.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the MCASP and the DMA or device CPU. The MCASP receiver and the DMA must be reinitialized to resynchronize them.

12.5.2.4.15.5 Unexpected Frame Sync Error

An unexpected frame sync occurs in when:

- in burst mode and TDM mode, the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- in TDM mode, an unexpected frame sync occurs also if the frame sync does NOT occur exactly during the correct bit clock (not a cycle earlier or later) and before slot 0.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. **Early:** An early unexpected frame sync occurs when the MCASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
   - Error event flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).

Copyright © 2026 Texas Instruments Incorporated

- Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.

2. **Late:** A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
   - Error event flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
   - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in burst mode and TDM mode. However, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

12.5.2.4.15.6 Clock Failure Detection

12.5.2.4.15.6.1 Clock Failure Check Startup

It is initially expected of the clock failure circuits to generate an error until at least one measurement is taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not be enabled immediately, but only after a specific startup procedure.

To start the transmit clock failure check procedure:

1. Configure the transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (MCASP_XCLKCHK).
2. Clear the transmit clock failure flag (XCKFAIL) in the transmit status register (MCASP_XSTAT).
3. Wait until the first measurement is taken (> 32 AHCLKX clock periods).
4. Verify that no clock failure is detected.
5. Repeat Step 2 through Step 4 until the clock is running and is no longer issuing clock failure errors.
6. After the transmit clock is measured and falls within the acceptable range, the following can be enabled:
   a. The transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (MCASP_XINTCTL)

To start the receive clock failure check procedure:

1. Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (MCASP_RCLKCHK).
2. Clear receive clock failure flag (RCKFAIL) in the receive status register (MCASP_RSTAT).
3. Wait until first measurement is taken (> 32 AHCLKR clock periods).
4. Verify no clock failure is detected.
5. Repeat steps 2–4 until clock is running and is no longer issuing clock failure errors.
6. After the receive clock is measured and falls within the acceptable range, the following may be enabled:
   a. the receive clock failure (RCKFAIL) interrupt enable bit in the receive interrupt control register (MCASP_RINTCTL)

12.5.2.4.15.6.2 Transmit Clock Failure Check and Recovery

The transmit clock failure check circuit (see Figure 12-391) works off the internal MCASP interface clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of interface clocks for every 32 high-rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (MCASP_XCLKCHK) every 32 high-rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (the MCASP_XSTAT[2] XCKFAIL bit) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is less than XMIN. The logic continually compares the current count (from the running interface clock counter) to the maximum allowable boundary (XMAX). This is so that if the external clock completely stops, the counter value is not copied to XCNT. An out-of-range maximum condition

Copyright © 2026 Texas Instruments Incorporated

occurs when the count is greater than XMAX. The XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

For the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset.

If a clock failure is detected, the MCASP_XSTAT[2] XCKFAIL transmit clock failure flag is set. This causes an interrupt if the MCASP_XINTCTL[2] XCKFAIL transmit clock failure interrupt enable bit is set.

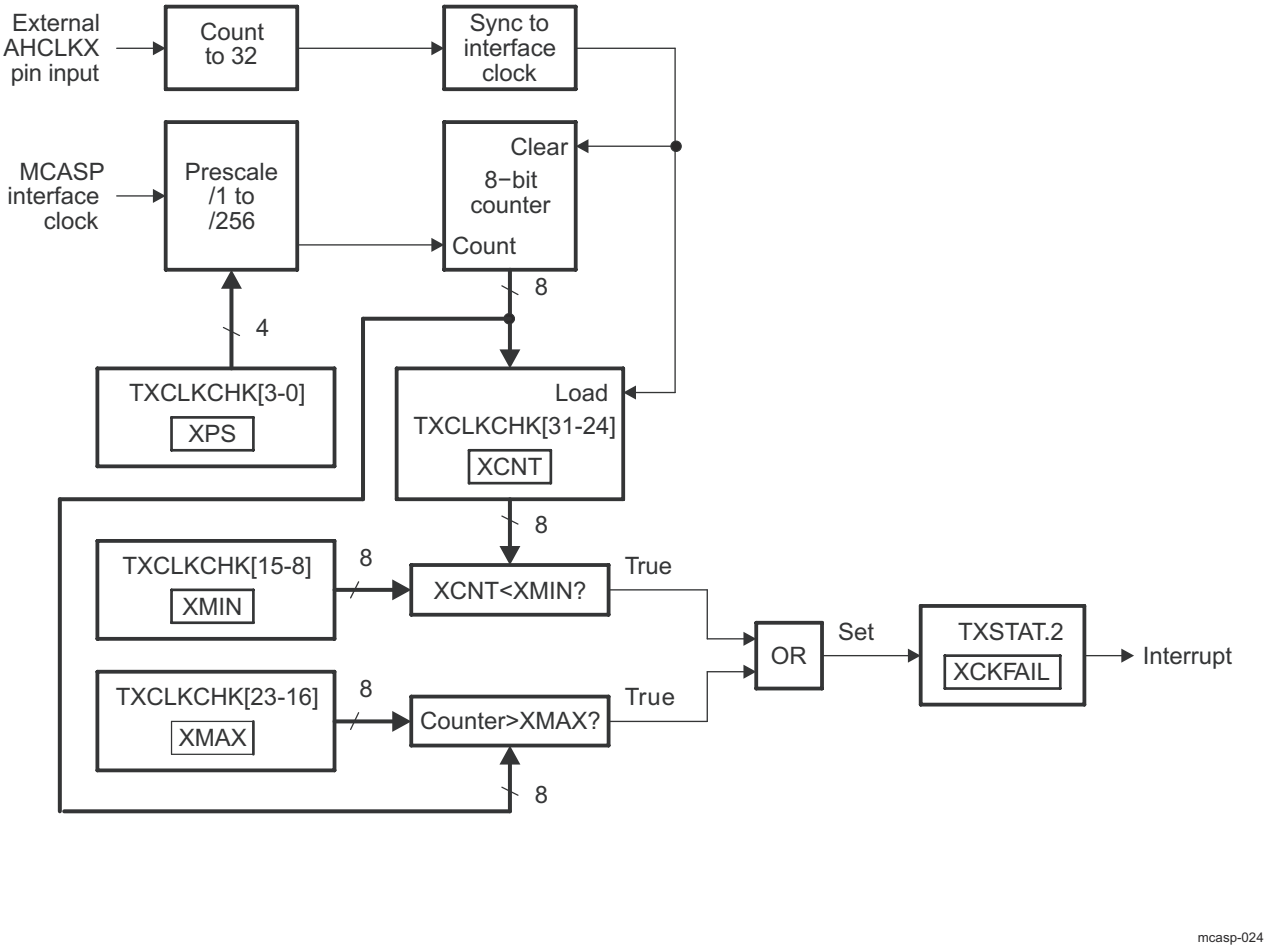


Figure 12-391. Transmit Clock Failure Detection Circuit Block Diagram

12.5.2.4.15.6.3 Receive Clock Failure Check and Recovery

The receive clock failure check circuit (see Figure 12-392) works off both the internal MCASP interface clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of interface clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (MCASP_RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an event (the MCASP_RSTAT[2] RCKFAIL bit) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running interface clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range

Copyright © 2026 Texas Instruments Incorporated

maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset.

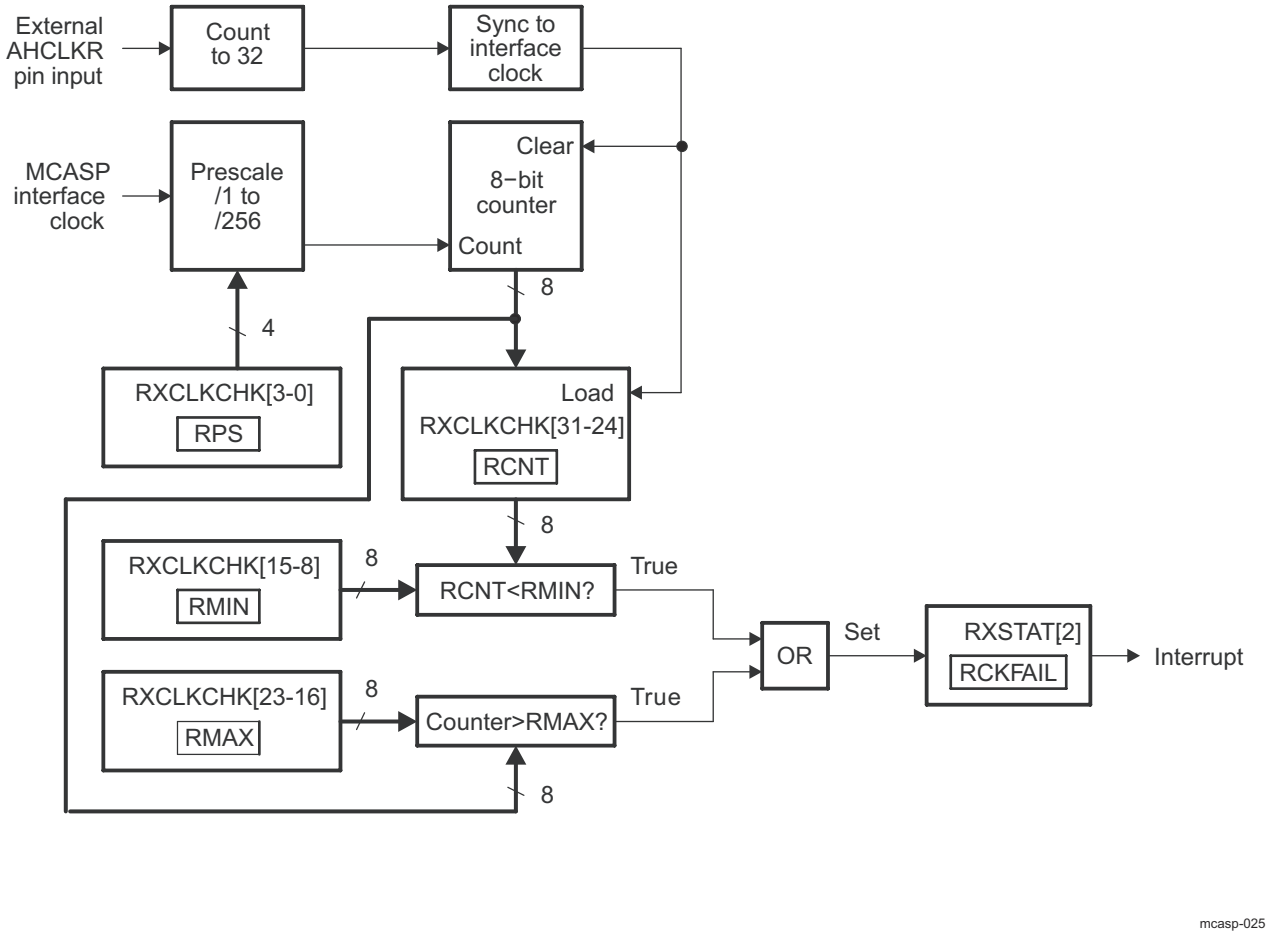


Figure 12-392. Receive Clock Failure Detection Circuit Block Diagram

Copyright © 2026 Texas Instruments Incorporated

12.5.2.5 MCASP Programming Guide

This section describes the low-level hardware programming sequences for the configuration and use of the MCASP module.

12.5.2.5.1 MCASP Global Initialization

12.5.2.5.1.1 Surrounding Modules Global Initialization

This section identifies the requirements for initializing the surrounding modules when the MCASP module is used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the MCASP (for more information, see *Module Integration*, and *MCASP Environment*).

Table 12-310 describes the global initialization of surrounding modules.

Table 12-310. Global Initialization of Surrounding Modules

Surrounding Modules	Comments
LPSC3	Module reset must be enabled. For more information about the module configuration, see *Reset*.
PLLCTRL0	PLLCTRL0 configuration must be done to enable the clocks to the MCASP modules, see *Clocking*.
PLL4	PLL4 configuration must be done to enable the clocks to the MCASP modules, see *Clocking*.
PLL2	PLL2 configuration must be done to enable the clocks to the MCASP modules, see *Clocking*.
ATL	ATL configuration must be done to enable the clocks to the MCASP modules, see *Audio Tracking Logic (ATL)*.
COMPUTE_CLUSTER0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling COMPUTE_CLUSTER0 interrupts, see *Interrupts*.
MAIN2MCU_LVL_INTRTR0	Device INTCs must be configured to enable the interrupt request generation. For information about enabling MAIN2MCU_LVL_INTRTR0 interrupts, see *Interrupts*.
R5FSS0_CORE0/1	Device INTCs must be configured to enable the interrupt request generation. For information about enabling R5FSS0_CORE0/1 interrupts, see *Interrupts*.
PDMA0_MCASP_G0	PDMA0_MCASP_G0 controllers configuration must be done to enable the module PDMA0_MCASP_G0 channel request, see *PDMA Controller*.
Interconnects	For information about the CBASS0 interconnects configuration, see *System Interconnect*.

Note

NAVSS configuration (UMDA channel set-up) must be done in order to move data from/to MCASP via the PDMA Controller. For more information, see *Navigator Subsystem (NAVSS)*.

Note

The COMPUTE_CLUSTER0, MAIN2MCU_LVL_INTRTR0, R5FSS0_CORE0/1, and the PDMA0_MCASP_G0 configurations are required when the interrupt and DMA-based communication modes are used. Further initialization of the selected interrupt and DMA controllers of the host CPU must be done for full functionality of the MCASP DMA and interrupt lines.

12.5.2.5.1.2 MCASP Global Initialization

####### *12.5.2.5.1.2.1 Main Sequence – MCASP Global Initialization for DIT-Transmission*

The procedure in Table 12-311 initializes the MCASP serializers transmitters to operate in DIT-mode (S/PDIF-transmission protocol) after a power-on reset (POR).

CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the *MCASP_GBLCTL* register.

Copyright © 2026 Texas Instruments Incorporated

Table 12-311. MCASP Transmitters Global Initialization for DIT-Mode Operation

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP components.	*MCASP_GBLCTL*[12-8]	0x00
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	*MCASP_GBLCTL*[12-8]	=0x00
3. Configure the local power management.	*MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE	0x1
4. Configure the transmit format unit.	See Section 12.5.2.5.1.2.1.1.	
5. Configure the transmit frame sync generator.	See Section 12.5.2.5.1.2.1.2.	
6. Configure the transmit clock generator.	See Section 12.5.2.5.1.2.1.3.	
7. Configure the TDM sequencer—set all slots active.	MCASP_XTDM[31-0] XTDMS	0xFFFF FFFF
8. Configure the desired n-th serializer (n=0 to 3) for transmit mode operation.(3)	MCASP_SRCTLn [1-0] SRMOD; n = 0 to 15	0x1
9. Configure the MCASP pins functionality.	See Section 12.5.2.5.1.2.1.4.	
10. Enable the MCASP DIT - transmission mode.	MCASP_DITCTL[0] DITEN	0x1(2)
11. Configure DIT-specific subframe fields.	See Table 12-316.	
12. Release from reset state the divider that outputs the AHCLKX clock.(1)	*MCASP_GBLCTL*[9] XHCLKRST	0x1
13. Poll the bit to ensure that it is successfully latched in the register.	*MCASP_GBLCTL*[9] XHCLKRST	=0x1
14. Release from reset state the divider that outputs the ACLKX clock.(1)	*MCASP_GBLCTL*[8] XCLKRST	0x1
15. Poll the bit to ensure that it is successfully latched in the register.	*MCASP_GBLCTL*[8] XCLKRST	=0x1

(1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the *MCASP_AHCLKXCTL* and *MCASP_ACLKXCTL* registers are ignored; hence, the transmission clock does not stop during the reset state of the dividers.
(2) This globally configures all active transmitters to operate in DIT-mode.
(3) For an unused serializer n, write MCASP_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).

12.5.2.5.1.2.1.1 Subsequence – Transmit Format Unit Configuration for DIT-Transmission

The procedure in Table 12-312 configures the transmit frame format unit of the MCASP module for a DIT-transmission.

Note

- The first transmit data bit always has a 0-bit delay.
- The bitstream is always transmitted in least-significant-bit (LSB)-first order.
- Pad value for extra bits in a certain slot is always 0.

Table 12-312. Transmit Format Unit Configuration for DIT-Transmission

Step	Register/Bit Field/Programming Model	Value
Configure the slot size to 32 bits.	MCASP_XFMT[7-4] XSSZ	0xF
IF: the data to transmit is left- aligned	Software test condition	
Set data mask in the range 0xFFFF FF00 – 0xFFFF 0000.	MCASP_XMASK[31-0] XMASK	0x-(1)
Rotate data right by a multiple-of-4- bit positions.	MCASP_XFMT[2-0] XROT	0x-(1)
ELSE		
Set data mask in the range 0x00FF FFFF– 0x0000 FFFF.	MCASP_XMASK[31-0] XMASK	0x-(1)
Rotate data right by 0-bit positions.	MCASP_XFMT[2-0] XROT	0x0
ENDIF		

Copyright © 2026 Texas Instruments Incorporated

Table 12-312. Transmit Format Unit Configuration for DIT-Transmission (continued)

Step	Register/Bit Field/Programming Model	Value
Select to write data to active serializers transmit buffers using peripheral (CFG) or DATA port	MCASP_XFMT[3] XBUSEL	0x-

(1) Refer to Section 12.5.2.4.4.1, *Transmit Fromat Unit* and Section 12.5.2.4.4.1.2, *DIT-Mode Transmission Data Alignment Settings*.

12.5.2.5.1.2.1.2 Subsequence – Transmit Frame Synchronization Generator Configuration for DIT-Transmission

The procedure in Table 12-313 configures the transmit frame synchronization generator of the MCASP module.

Note

The frame synchronization signal is always rising-edge active and always has a single-bit width.

Table 12-313. Transmit Frame-Synchronization Generator Configuration for DIT-Transmission

Step	Register/Bit Field/Programming Model	Value
Select 384-slot size block.	MCASP_AFSXCTL[15-7] XMOD	0x180
Select internally-generated transmit frame sync.	MCASP_AFSXCTL[1] FSXM	0x1

12.5.2.5.1.2.1.3 Subsequence – Transmit Clock Generator Configuration for DIT-Transmission

Note

By default, the ACLKX and AHCLKX clocks are generated only from the MCASP internal clock source.

The procedure in Table 12-314 configures the transmit clock generator of the MCASP module.

Table 12-314. Transmit Clock Generator Configuration in DIT-Mode

Step	Register/Bit Field/Programming Model	Value
Set the divisor for the internally generated high frequency clock– AHCLKX.	*MCASP_AHCLKXCTL*[11-0] HCLKXDIV	0x-
Set he divisor for the internally generated transmission clock– ACLKX.	*MCASP_ACLKXCTL*[4-0] CLKXDIV	0x-
Configure the transmit clock failure detect logic.	See Section 12.5.2.4.15.6.1, *Clock Failure Check Startup*.	

12.5.2.5.1.2.1.4 Subsequence - MCASP Pins Functional Configuration

The procedure in Table 12-315 configures the MCASP pins for MCASP functionality.

Table 12-315. MCASP Pins Functional Configuration

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	*MCASP_PFUNC*[31-0]	0x0
Configure the MCASP pins as outputs: AFSX AHCLKX ACLKX Desired i-th MCASP data pin AXRi is configured as an output for DIT-transmission.	*MCASP_PDIR*[28] AFSX; *MCASP_PDIR*[27] AHCLKX; *MCASP_PDIR*[26] ACLKX; *MCASP_PDIR*[i] AXRi	0x1 0x1 0x1 0x1

12.5.2.5.1.2.1.5 Subsequence – DIT-specific Subframe Fields Configuration

The procedure in Table 12-316 configures the DIT-specific subframe fields as part of the S/PDIF format data.

Copyright © 2026 Texas Instruments Incorporated

Table 12-316. DIT-Specific Subframe Fields Configuration

Step	Register/Bit Field/Programming Model	Value
Configure the valid bit value for odd time slots.	MCASP_DITCTL[3] VB	0x-
Configure the valid bit value for even time slots.	MCASP_DITCTL[2] VA	0x-
Configure the user data bit for each subframe A and B in a 384-slot S/PDIF block.	MCASP_DITUDRAi[31-0] DITUDRAi, where i = 0 to 5 MCASP_DITUDRBi[31-0] DITUDRBi, where i = 0 to 5	0x- 0x-
Configure the channel status bit for each subframe A and B in a 384-slot S/PDIF block.	MCASP_DITCSRAi[31-0], where i = 0 to 5 MCASP_DITCSRBi[31-0], where i = 0 to 5	0x- 0x-

12.5.2.5.1.2.2 Main Sequence – MCASP Global Initialization for TDM-Reception

The procedure in Table 12-317 initializes a MCASP serializer n receiver(s) to operate in TDM-mode (the only mode supported by MCASP receivers) after a power-on reset (POR). This is used for I2S (2-slot TDM) and other TDM-based audio protocols reception.

CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the *MCASP_GBLCTL* register.

Note

The MCASP receivers support only TDM-frames (including 384-TDM frames) reception. DIT-frames reception (this is, S/PDIF stream) can be implemented indirectly via an external DIR-chip converter with DIT-input and TDM (I2S)-compatible output connected to device MCASP receiver input (TDM-only compatible).

Table 12-317. MCASP Receivers Global Initialization for TDM-Mode Operation

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP receive components.	*MCASP_GBLCTL*[4-0]	0x00
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	*MCASP_GBLCTL*[4-0]	=0x00
3. Configure the local power management.	*MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE	0x1
4. Configure the receive format unit.	See Section 12.5.2.5.1.2.2.1.	
5. Configure the receive frame sync generator.	See Section 12.5.2.5.1.2.2.2.	
6. Configure the receive clock generator.	See Section 12.5.2.5.1.2.2.3.	
7. Program all bits - RTDMSk (where k = 0 to 31) according to the time slot characteristics desired (positions of active versus inactive slots within a frame).	MCASP_RTDM[k] RTDMSk , where k = 0 to 31	0x-
8. Configure the desired n-th serializer for receive mode operation.(4)	MCASP_SRCTLn [1-0] SRMOD; n = 0 to 15	0x2
9. Configure the MCASP pins functionality.	See Section 12.5.2.5.1.2.2.4.	
10. Optional: Configure a MCASP Rx channel for a loopback operation (TDM mode only) in MCASP_DLBCTL [31-0].	See Section 12.5.2.4.14.1, *Loopback Mode Configurations*.	0x-(5)
11. Release from reset state the divider that outputs the AHCLKR clock.(1) See also(2).	*MCASP_GBLCTL*[1] RHCLKRST	0x1
12. Poll the bit to ensure that it is successfully latched in the register. See also(2).	*MCASP_GBLCTL*[1] RHCLKRST	=0x1

Copyright © 2026 Texas Instruments Incorporated

Table 12-317. MCASP Receivers Global Initialization for TDM-Mode Operation (continued)

Step	Register/Bit Field/Programming Model	Value
13. Release from reset state the divider that outputs the ACLKR clock. (1) See also(3).	*MCASP_GBLCTL*[0] RCLKRST	0x1
14. Poll the bit to ensure that it is successfully latched in the register. See also(3).	*MCASP_GBLCTL*[0] RCLKRST	=0x1

(1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the *MCASP_AHCLKRCTL* and *MCASP_ACLKRCTL* registers are ignored; hence, the reception clock does not stop during the reset state of the dividers.
(2) This step is necessary even if external high-frequency serial clocks are used.
(3) This step can be skipped if external serial clocks are used and they are running.
(4) For an unused serializer n, write MCASP_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).
(5) In this case the receiver clock and frame sync are derived from the MCASP transmitter logic, so *MCASP_ACLKXCTL*[6] ASYNC must be set to 0b0. Neither MCASP internal receiver clock and frame sync generators, nor external clock and frame sync source are used.

12.5.2.5.1.2.2.1 Subsequence – Receive Format Unit Configuration in TDM Mode

The procedure in Table 12-318 configures the receive frame format unit of the MCASP module for TDM slots reception.

Table 12-318. Receive Format Unit Configuration for TDM-Reception

Step	Register/Bit Field/Programming Model	Value
Configure the desired TDM-slot size	MCASP_RFMT[7-4] RSSZ	0x-(1)
Set data mask out value (0x0000 0000 - 0xFFFF FFFF).	MCASP_RMASK[31-0] RMASK	0x-(2)
Select a padding value for masked-out bits.	MCASP_RFMT[14-13] RPAD	0x-
Specify position (0x0-0x1F) of the bit in corresponding register MCASP_RBUFn which value to be used as a pad value in case MCASP_RFMT[14-13] RPAD = 0x2. (n = 0 to 15)	MCASP_RFMT[12-8] RPBIT	0x-
Rotate data right by a multiple of 4- bit positions.	MCASP_RFMT[2-0] RROT	0x-(3)
Received stream bit order (LSB- or MSB-first). Must be set to 0x1 for an I2S stream reception (MSB-first).	MCASP_RFMT[15] RRVRS	0x-(3)
Specify a delay between frame sync and first bit of data in number of bits. Must be set to 0x1 for an I2S stream reception.	MCASP_RFMT[17-16] RDATDLY	0x-
Select to read data from active serializers receive buffers using peripheral (CFG) or DATA port	MCASP_RFMT[3] RBUSEL	0x-

(1) Refer to Section 12.5.2.4.4.2, *Receive Format Unit*, regarding options for received TDM-slot sizes.
(2) For more details on Rx masking value, refer to Section 12.5.2.4.4.2.1, *TDM - Mode Reception Data Alignment Settings*
(3) For more details on rotation and received TDM stream bit order, refer to Section 12.5.2.4.4.2.1, *TDM - Mode Reception Data Alignment Settings* and Table 12-305, *MCASP RFU Settings*.

12.5.2.5.1.2.2.2 Subsequence – Receive Frame Synchronization Generator Configuration in TDM Mode

The procedure in Table 12-319 configures the transmit frame synchronization generator of the MCASP module.

Note

The same bit - *MCASP_ACLKXCTL*[6] ASYNC which is used to determine if MCASP receivers and transmitters work synchronously on the same clock, is also used to define if receiver frame sync is derived from the transmit frame sync generator, or generated independently in the receiver (either inernally or externally sourced). Hence, the settings in below table Table 12-319 have no effect, if *MCASP_ACLKXCTL*[6] ASYNC = 0.

Copyright © 2026 Texas Instruments Incorporated

Table 12-319. Receive Frame-Synchronization Generator Configuration for TDM-Reception

Step	Register/Bit Field/Programming Model	Value
Select number of TDM slots per frame. Must be set to 0x2, in case of an I2S-reception. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3.	MCASP_AFSRCTL[15-7] RMOD	0x-(1)
Choose the receive frame sync width -single bit/single word. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3.	MCASP_AFSRCTL[4] FRWID	0x-
Select start of received frame sync polarity - rising / falling edge. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3.	MCASP_AFSRCTL[0] FSRP	0x-
IFreceive frame sync - FS is internally generated	Software test condition	
Select internally- generated receive frame sync. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3.	MCASP_AFSRCTL[1] FSRM	0b1
If MCASP receiver is required to output internally generated frame, AFSR pin must be set as an output in step 9 of the sequence documented in the Table 12-317. This must not be done in current step because the frame control register - MCASP_AFSXCTL must be appropriately configured prior to AFSR pin outputting a frame to an external device.	*MCASP_PDIR*[31] AFSR	0b1
ELSE		
Select externally- generated receive frame sync. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3.	MCASP_AFSRCTL[1] FSRM	0b0
Setup the AFSR pin as input (device level: MCASPi_AFSR)	*MCASP_PDIR*[31] AFSR	0b0
ENDIF		
To generate MCASP receive frame sync in receiver logic, select an asynchronous frame sync.	*MCASP_ACLKXCTL*[6] ASYNC	0b1

(1) Must be set to 0x180 in case of 384-TDM slot frame reception from a DIR component I2S-output. For more details on TDM-frame settings, refer to *Module Integration*.

12.5.2.5.1.2.2.3 Subsequence – Receive Clock Generator Configuration

The procedure in Table 12-320 configures the receive clock generator of the MCASP module.

Note

The settings in below table Table 12-320 have no effect, if *MCASP_ACLKXCTL*[6] ASYNC = 0 (this is, receive clock is sourced from the inverted version of the transmit clock). For example, such is the case when MCASP loopback mode is used.

Table 12-320. Receive Clock Generator Configuration

Step	Register/Bit Field/Programming Model	Value
To use the MCASP receive clock generator, select an asynchronous receiver clock schema (ASYNC = 1). Otherwise an inverted version of transmit clock XCLK is used (receiver synchronized with transmitter).	*MCASP_ACLKXCTL*[6] ASYNC	0b1
IF receive clock - RCLK is internally generated	Software test condition	
The high-speed receive clock - AHCLKR is internally generated based on AUXCLK		
Select an internally-generated high-frequency clock.	*MCASP_AHCLKXCTL*[15] HCLKRM	0b1

Copyright © 2026 Texas Instruments Incorporated

Table 12-320. Receive Clock Generator Configuration (continued)

Step	Register/Bit Field/Programming Model	Value
Select the internal high-speed clock source polarity: non-inverted or inverted.	*MCASP_AHCLKXCTL*[14] HCLKRP	0x-
Set the divisor for the internally generated high-frequency clock – AHCLKR in range (1 - 4096).	*MCASP_AHCLKXCTL*[11-0] HCLKRDIV	0x-
Select an internally-generated receive clock.	*MCASP_ACLKXCTL*[5] CLKRM	0b1
Receiver samples on rising/falling edge. Select Rx sampling on the rising edge if transmitter shifts out on falling edge, and vice versa.	*MCASP_ACLKXCTL*[7] CLKRP	0x-
Set the divisor for the internally generated receive clock– ACLKR in range (1 - 32).	*MCASP_ACLKXCTL*[4-0] CLKRDIV	0x-
Optional: If MCASP receiver is required to output internally generated clock, ACLKR pin must be set as an output in step 9 of the sequence documented in the Table 12-317. This must not be done in current step because the clock control register - *MCASP_ACLKRCTL* must be appropriately configured prior to ACLKR pin outputting a receive clock to an external device.	*MCASP_PDIR*[29] ACLKR	0b1
ELSE		
Select an externally-generated receive clock. Note that in this case the AHCLKR signal path and the CLKRDIV divider are NOT used.	*MCASP_ACLKXCTL*[5] CLKRM	0b0
Receiver samples on rising/falling edge. Select Rx sampling on the rising edge if transmitter shifts out on falling edge, and vice versa.	*MCASP_ACLKXCTL*[7] CLKRP	0x-
Setup an input direction for the ACLKR pin	*MCASP_PDIR*[29] ACLKR	0b0
ENDIF		
Configure the transmit clock failure detect logic.	See Section 12.5.2.4.15.6.1, *Clock Failure Check Startup*.	

12.5.2.5.1.2.2.4 Subsequence—MCASP Receiver Pins Functional Configuration

The procedure in Table 12-321 configures the MCASP pins for MCASP functionality.

Table 12-321. MCASP Receiver Pins Functional Configuration

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	*MCASP_PFUNC*[31-0]	0x0
Configure the MCASP pins direction: AFSR ACLKR Desired n-th MCASP data pin AXRn is configured as an input for receiving.	*MCASP_PDIR*[31] AFSR; *MCASP_PDIR*[29] ACLKR; *MCASP_PDIR*[n] AXRn;	0x-(1) 0x-(2) 0x0

(1) See Table 12-319.
(2) For more details on MCASP clock configurations, refer to Table 12-320.

12.5.2.5.1.2.3 Main Sequence – MCASP Global Initialization for TDM -Transmission

The procedure in Table 12-322 initializes a MCASP serializer n transmitter(s) to operate in TDM-mode after a power-on reset (POR). This is used for I2S (2-slot TDM) and other TDM-based audio protocols transmission.

CAUTION

Before performing MCASP global initialization, If external clock ACLKR is used, it must be running already for proper synchronization of the *MCASP_GBLCTL* register.

Copyright © 2026 Texas Instruments Incorporated

Table 12-322. MCASP Transmitters Global Initialization for TDM-Mode Operation

Step	Register/Bit Field/Programming Model	Value
1. Apply software reset to different MCASP transmit components.	*MCASP_GBLCTL*[12-8]	0x00
2. Poll the bits to ensure the active reset value (0x00) is successfully latched into the register.	*MCASP_GBLCTL*[12-8]	=0x00
3. Configure the local power management.	*MCASP_PWRIDLESYSCONFIG*[1-0] IDLE_MODE	0x1
4. Configure the transmit format unit.	See Section 12.5.2.5.1.2.3.1.	
5. Configure the transmit frame sync generator.	See Section 12.5.2.5.1.2.3.2.	
6. Configure the transmit clock generator.	See Section 12.5.2.5.1.2.3.3.	
7. Program all bits - XTDMSk, where k = 0 to 31, according to the time slot characteristics desired (positions of active versus inactive slots within a frame).	MCASP_XTDM[k] XTDMSk, where k = 0 to 31(4)	0x-
8. Configure the desired n-th serializer for transmit mode operation.(3)	MCASP_SRCTLn[1-0] SRMOD; n = 0 to 15	0x1
9. Setup all active transmitters to operate in TDM mode.	MCASP_DITCTL[0] DITEN	0x0(2)
10. Configure the MCASP pins functionality.	See Section 12.5.2.5.1.2.3.4.	
11. Optional: Configure a MCASP Tx channel for loopback operation (TDM mode only) in MCASP_DLBCTL [31-0].	See Section 12.5.2.4.14.1, *Loopback Mode Configurations*.	0x-
12. Release from reset state the divider that outputs the AHCLKR clock. See(1)	*MCASP_GBLCTL*[9] XHCLKRST	0x1
13. Poll the bit to ensure that it is successfully latched in the register.	*MCASP_GBLCTL*[9] XHCLKRST	=0x1
14. Release from reset state the divider that outputs the ACLKR clock. See(1)	*MCASP_GBLCTL*[8] XCLKRST	0x1
15. Poll the bit to ensure that it is successfully latched in the register.	*MCASP_GBLCTL*[8] XCLKRST	=0x1

(1) During reset state the local MCASP internal clock dividers maintain a 1:1 ratio at their outputs. The values stored in the *MCASP_AHCLKXCTL* and *MCASP_ACLKXCTL* registers are ignored; hence, the transmission clock does not stop during the reset state of the dividers.
(2) All active transmit channels operate either in TDM mode or in DIT mode depending on DITEN value. There is no option to choose Tx Mode between DIT and TDM separately per serializer transmitter.
(3) For an unused serializer n, write MCASP_SRCTLn [1-0] SRMOD = 0x0 to disable it (n = 0 to 15).
(4) Appropriately program in the MCASP_SRCTLn[3-2] DISMOD bit field, the desired level (high-impedance state, 0, or 1) at AXRn output, during time of inactive slots. Note, that this setting does NOT apply when all slots are programmed to be active within a frame (in particular DIT-mode) (n = 0 to 15).

12.5.2.5.1.2.3.1 Subsequence – Transmit Format Unit Configuration in TDM Mode

The procedure in Table 12-323 configures the transmit frame format unit of the MCASP module for TDM slots transmission.

Table 12-323. Transmit Format Unit Configuration for TDM-Transmission

Step	Register/Bit Field/Programming Model	Value
Configure the desired TDM-slot size	MCASP_XFMT[7-4] XSSZ	0x-(1)
Set data mask out value (0x0000 0000 - 0xFFFF FFFF).	MCASP_XMASK[31-0] XMASK	0x-(2)
Select a padding value for masked-out bits.	MCASP_XFMT[14-13] XPAD	0x-
Specify position (0x0-0x1F) of the bit in corresponding register MCASP_XBUFn which value to be used as a pad value in case MCASP_XFMT[14-13] XPAD = 0x2 (n = 0 to 15).	MCASP_XFMT[12-8] XPBIT	0x-
Rotate data right by a multiple of 4- bit positions.	MCASP_XFMT[2-0] XROT	0x-(3)
transmitted stream bit order (LSB- or MSB-first). Must be set to 0x1 for an I2S stream transmission (MSB-first).	MCASP_XFMT[15] XRVRS	0x-(3)

Copyright © 2026 Texas Instruments Incorporated

Table 12-323. Transmit Format Unit Configuration for TDM-Transmission (continued)

Step	Register/Bit Field/Programming Model	Value
Specify a delay between frame sync and first bit of data in number of bits. Must be set to 0x1 for an I2S stream transmission.	MCASP_XFMT[17-16] XDATDLY	0x-
Select to write data to active serializers transmit buffers using peripheral (CFG) or DATA port	MCASP_XFMT[3] XBUSEL	0x-

(1) Refer to Section 12.5.2.4.4.1, *Transmit Format Unit* , regarding options for transmitted TDM-slot sizes.
(2) For more details on Tx masking value, refer to Section 12.5.2.4.4.1.1, *TDM - Mode Transmission Data Alignment Settings*
(3) For more details on rotation and transmitd TDM stream bit order, refer to Section 12.5.2.4.4.1.1, *TDM - Mode Transmission Data Alignment Settings* and Table 12-303, *MCASP TFU TDM Mode Settings*.

12.5.2.5.1.2.3.2 Subsequence – Transmit Frame Synchronization Generator Configuration in TDM Mode

The procedure in Table 12-324 configures the transmit frame synchronization generator of the MCASP module.

Table 12-324. Transmit Frame-Synchronization Generator Configuration for TDM-Transmission

Step	Register/Bit Field/Programming Model	Value
Select number of TDM slots per frame (2 - 32). Must be set to 0x2, in case of an I2S-transmission. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3, *Frame-Sync Generator*.	MCASP_AFSXCTL[15-7] XMOD	0x-
Choose the transmit frame sync width -single bit/single word. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3, *Frame-Sync Generator*.	MCASP_AFSXCTL[4] FXWID	0x-
Select start of transmit frame sync polarity - rising / falling edge. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3, *Frame-Sync Generator*.	MCASP_AFSXCTL[0] FSXP	0x-
IF transmit frame sync - FS is internally generated	Software test condition	
Select internally- generated transmit frame sync. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3, *Frame-Sync Generator*.	MCASP_AFSXCTL[1] FSXM	0b1
If MCASP transmitter is required to output internally generated frame, AFSX pin must be set as an output in step 10 of the sequence documented in the Table 12-322. This must NOT be done in current step because the frame control register - MCASP_AFSXCTL must be appropriately configured prior to AFSX pin outputting a frame sync to an external device.	*MCASP_PDIR*[28] AFSX	0b1
ELSE		
Select externally- generated transmit frame sync. For more details on frame-sync generator, refer to Section 12.5.2.4.2.3, *Frame-Sync Generator*.	MCASP_AFSXCTL[1] FSXM	0b0
Setup the AFSX pin as input	*MCASP_PDIR*[28] AFSX	0b0

12.5.2.5.1.2.3.3 Subsequence – Transmit Clock Generator Configuration for TDM Cases

The procedure in Table 12-325 configures the transmit clock generator of the MCASP module.

Table 12-325. Transmit Clock Generator Configuration for TDM Cases

Step	Register/Bit Field/Programming Model	Value
IF transmit clock - XCLK is internally generated	Software test condition	
IF high-speed transmit clock - AHCLKX is internally generated based on AUXCLK	Software test condition	
Select an internally-generated high-frequency clock.	*MCASP_AHCLKXCTL*[15] HCLKXM	0b1

Copyright © 2026 Texas Instruments Incorporated

Table 12-325. Transmit Clock Generator Configuration for TDM Cases (continued)

Step	Register/Bit Field/Programming Model	Value
Select the high-frequency clock source polarity: non-inverted or inverted.	*MCASP_AHCLKXCTL*[14] HCLKXP	0x-
Set the divisor for the internally generated high-frequency clock – AHCLKX in range (1 - 4096).	*MCASP_AHCLKXCTL*[11-0] HCLKXDIV	0x-
Optional: If MCASP transmitter is required to output internally generated high-frequency clock, AHCLKX pin must be set as an output in step 10 of the sequence documented in the Table 12-322. This must NOT be done in current step because the clock control register - *MCASP_AHCLKXCTL* must be appropriately configured prior to AHCLKX pin outputting a high-speed clock to an external device.	*MCASP_PDIR*[27] AHCLKX	0b1
ELSE		
Select an externally-generated high frequency clock (HCLKXDIV divider can not be used).	*MCASP_AHCLKXCTL*[15] HCLKXM	0b0
Select the high-speed transmit clock source polarity: non-inverted or inverted.	*MCASP_AHCLKXCTL*[14] HCLKXP	0x-
Setup an input directon for the AHCLKX pin	*MCASP_PDIR*[27] AHCLKX	0b0
ENDIF		
Select an internally-generated transmit clock.	*MCASP_ACLKXCTL*[5] CLKXM	0b1
Transmitter samples on rising/falling edge. Select Tx shifting out data on the rising edge if receiver samples on falling edge, and vice versa.	*MCASP_ACLKXCTL*[7] CLKXP	0x-
Set the divisor for the internally generated transmit clock– ACLKX in range (1 - 32).	*MCASP_ACLKXCTL*[4-0] CLKXDIV	0x-
Optional: If MCASP transmitter is required to output internally generated clock, ACLKX pin) must be set as an output in step 10 of the sequence documented in the Table 12-322. This must NOT be done in current step because the clock control register - *MCASP_ACLKXCTL* must be appropriately configured prior to ACLKX pin outputting a transmit clock to an external device.	*MCASP_PDIR*[26] ACLKX	0b1
ELSE		
Select an externally-generated transmit clock. Note that in this case the AHCLKX signal path and the CLKXDIV divider are NOT used.	*MCASP_ACLKXCTL*[5] CLKXM	0b0
Transmitter samples on rising/falling edge. Select Tx shifting out data on the rising edge if receiver samples on falling edge, and vice versa.	*MCASP_ACLKXCTL*[7] CLKXP	0x-
Setup an input directon for the ACLKX pin	*MCASP_PDIR*[26] ACLKX	0b0
ENDIF		
Configure the transmit clock failure detect logic.	See Section 12.5.2.4.15.6.1, *Clock Failure Check Startup*.	

12.5.2.5.1.2.3.4 Subsequence—MCASP Transmit Pins Functional Configuration

The procedure in Table 12-326 configures the MCASP pins for MCASP functionality.

Table 12-326. MCASP Transmit Pins Functional Configuration

Step	Register/Bit Field/Programming Model	Value
Configure module different pins to have MCASP functionality.	*MCASP_PFUNC*[31-0]	0x0

Copyright © 2026 Texas Instruments Incorporated

Table 12-326. MCASP Transmit Pins Functional Configuration (continued)

Step	Register/Bit Field/Programming Model	Value
Configure the MCASP pins direction: AFSX AHCLKX ACLKX Desired n-th MCASP data pin AXRn is configured as an ouput for transmission.	*MCASP_PDIR*[28] AFSR; *MCASP_PDIR*[27] AHCLKR; *MCASP_PDIR*[26] ACLKR; *MCASP_PDIR*[n] AXRn	0x-(1) 0x-(2) 0x-(2) 0x1

(1) See Table 12-324.

12.5.2.5.2 MCASP Operational Modes Configuration

12.5.2.5.2.1 MCASP Transmission Modes

12.5.2.5.2.1.1 Main Sequence – MCASP DIT- /TDM- Polling Transmission Method

Figure 12-393 shows the MCASP DIT-/TDM- polling method.

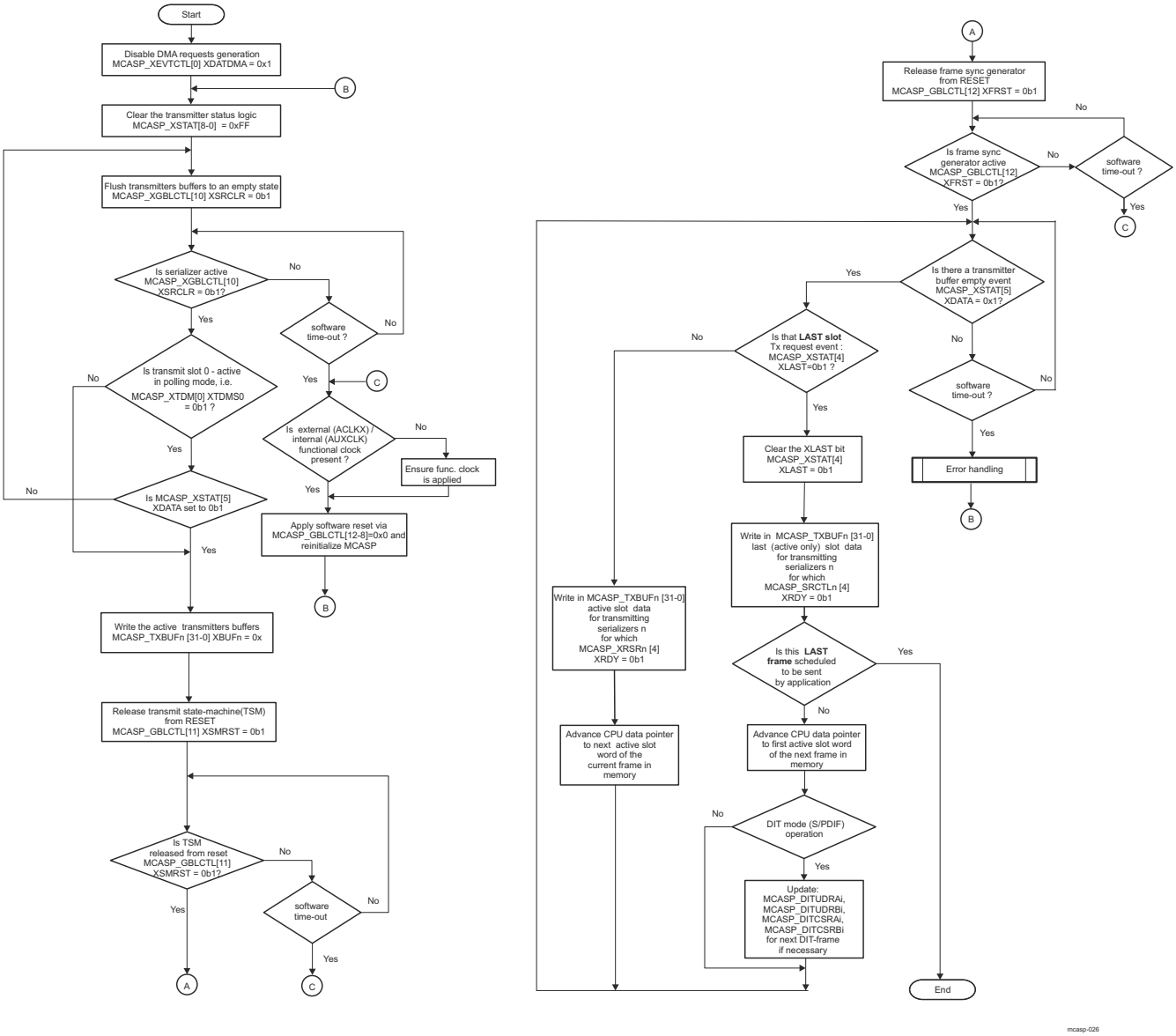


Figure 12-393. MCASP DIT- /TDM- Transmission Polling Method

Copyright © 2026 Texas Instruments Incorporated

These registers are for MCASP DIT-/TDM- transmission polling method: *MCASP_XEVTCTL*, MCASP_XSTAT, *MCASP_GBLCTL*, MCASP_XTDM, MCASP_XBUFn, MCASP_SRCTLn, MCASP_DITUDRAi, MCASP_DITUDRBi, MCASP_DITCSRAi, MCASP_DITCSRBi (n = 0 to 15 and i = 0 to 5).

Table 12-327 summarizes the subprocess call for the DIT-/TDM- transmission polling mode.

Table 12-327. Subprocess Call Summary for Main Sequence – MCASP DIT-/TDM- Transmission Polling Method

Subprocess Name	Cross-Reference
Error handling	Figure 12-399

12.5.2.5.2.1.2 Main Sequence – MCASP DIT- /TDM - Interrupt Transmission Method

Figure 12-394 shows the initial setup for interrupt-based transmission.

Copyright © 2026 Texas Instruments Incorporated

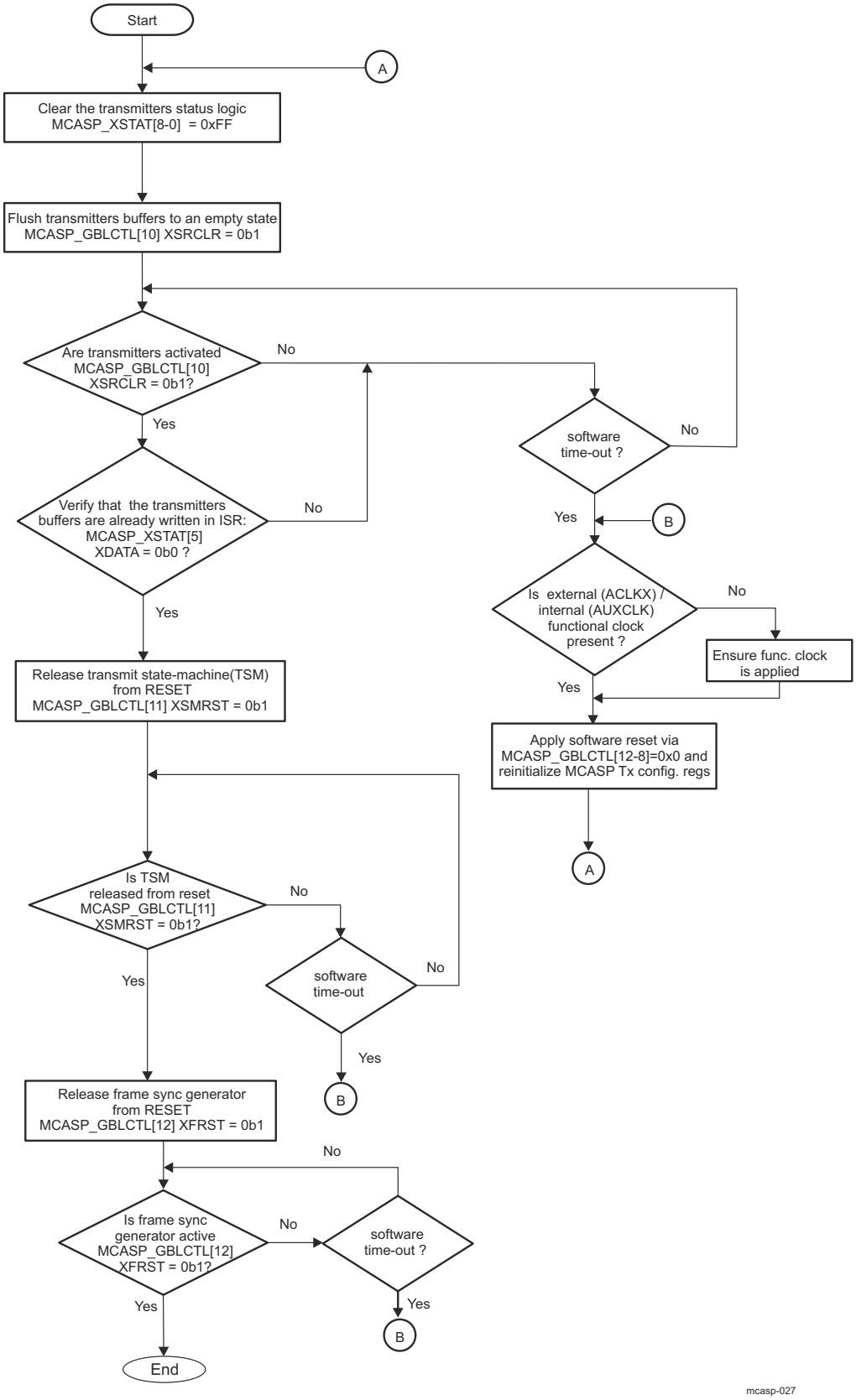


Figure 12-394. Subsequence – DIT-/TDM- Transmission Startup Procedure

Copyright © 2026 Texas Instruments Incorporated

Table 12-328 shows the configuration of the MCASP using an interrupt method for DIT-/TDM- transmission.

Table 12-328. MCASP DIT-/TDM- Interrupt Transmission Model

Step	Register/Bit Field/Programming Model	Value
Disable Tx DMA requests generation.	*MCASP_XEVTCTL*[0] XDATDMA	0x1
Enable the data ready event transmit interrupt.	MCASP_XINTCTL[5] XDATA	0x1
Optional: Enable the transmit error event interrupts.	MCASP_XINTCTL[2] XCKFAIL MCASP_XINTCTL[1] XSYNCERR MCASP_XINTCTL[0] XUNDRN	0x1 0x1 0x1
Optional: Enable the start of frame interrupt. Optional: Enable the last slot data interrupt (useful for DIT user data/ channel status next S/PDIF frame info update).	MCASP_XINTCTL[7] XSTAFRM MCASP_XINTCTL[4] XLAST	0x1 0x1
IFwrite transfer is through the MCASP DATA port (MCASP_XFMT[3] XBUSEL is set to 0b0).	Software test condition (setting is done in step4 of the *MCASP Transmitters Global Initialization* - see Table 12-311)	
Enable the DATA port error based interrupt.	MCASP_XINTCTL[3] XDMAERR	0x1
ELSE		
Disable the DATA port error based interrupt.	MCASP_XINTCTL[3] XDMAERR	0x0
ENDIF		
DIT/TDM - Transmission Startup Procedure	See Figure 12-394.	

These registers are for MCASP DIT-/TDM- transmission startup procedure: *MCASP_GBLCTL*, *MCASP_XSTAT*.

12.5.2.5.2.1.3 Main Sequence –MCASP DIT- /TDM - Mode DMA Transmission Method

Table 12-329 shows the configuration of the ↓ using the DMA method for transmission. Possible interrupt error event servicing is also considered. shows the initial setup for DMA - based transmission.

Note

Because of the DATA port burst access capability with the DMA method, it is strongly recommended that DMA transfers are initiated through the MCASP DATA port.

Table 12-329. MCASP DMA Transmission Model with Interrupt Events Servicing

Step	Register/Bit Field/Programming Model	Value
Recommended: Select DATA port to access the transmit buffers.	MCASP_XFMT[3] XBUSEL	0x0
Enable the Tx DMA requests generation.	*MCASP_XEVTCTL*[0] XDATDMA	0x0
Enable the Tx DMA error event, because of MCASP DATA port usage.	MCASP_XINTCTL[3] XDMAERR	0x1
Optional: Enable the transmit error event interrupts.	MCASP_XINTCTL[2] XCKFAIL MCASP_XINTCTL[1] XSYNCERR MCASP_XINTCTL[0] XUNDRN	0x1 0x1 0x1
Optional: Enable the start of frame interrupt. Optional: Enable the last slot data interrupt.	MCASP_XINTCTL[7] XSTAFRM MCASP_XINTCTL[4] XLAST	0x1 0x1
Disable the data ready event transmit interrupt, as DMA is used to service this request.	MCASP_XINTCTL[5] XDATA	0x0
DMA startup transmission procedure. This procedure is identical than the one shown in Figure 12-394. The only difference is that DMA automatically services all the XINT events raised by the MCASP, and no CPU data processing intervention is required. The CPU is involved only in error handling shown in Figure 12-397.	See Figure 12-394.	

Copyright © 2026 Texas Instruments Incorporated

12.5.2.5.2.2 MCASP Reception Modes

12.5.2.5.2.2.1 Main Sequence – MCASP Polling Reception Method

Figure 12-395 shows the MCASP polling reception method.

Note

The MCASP polling reception model considers thedevice CPUs as the accessor of audio data from the MCASP receive buffers.

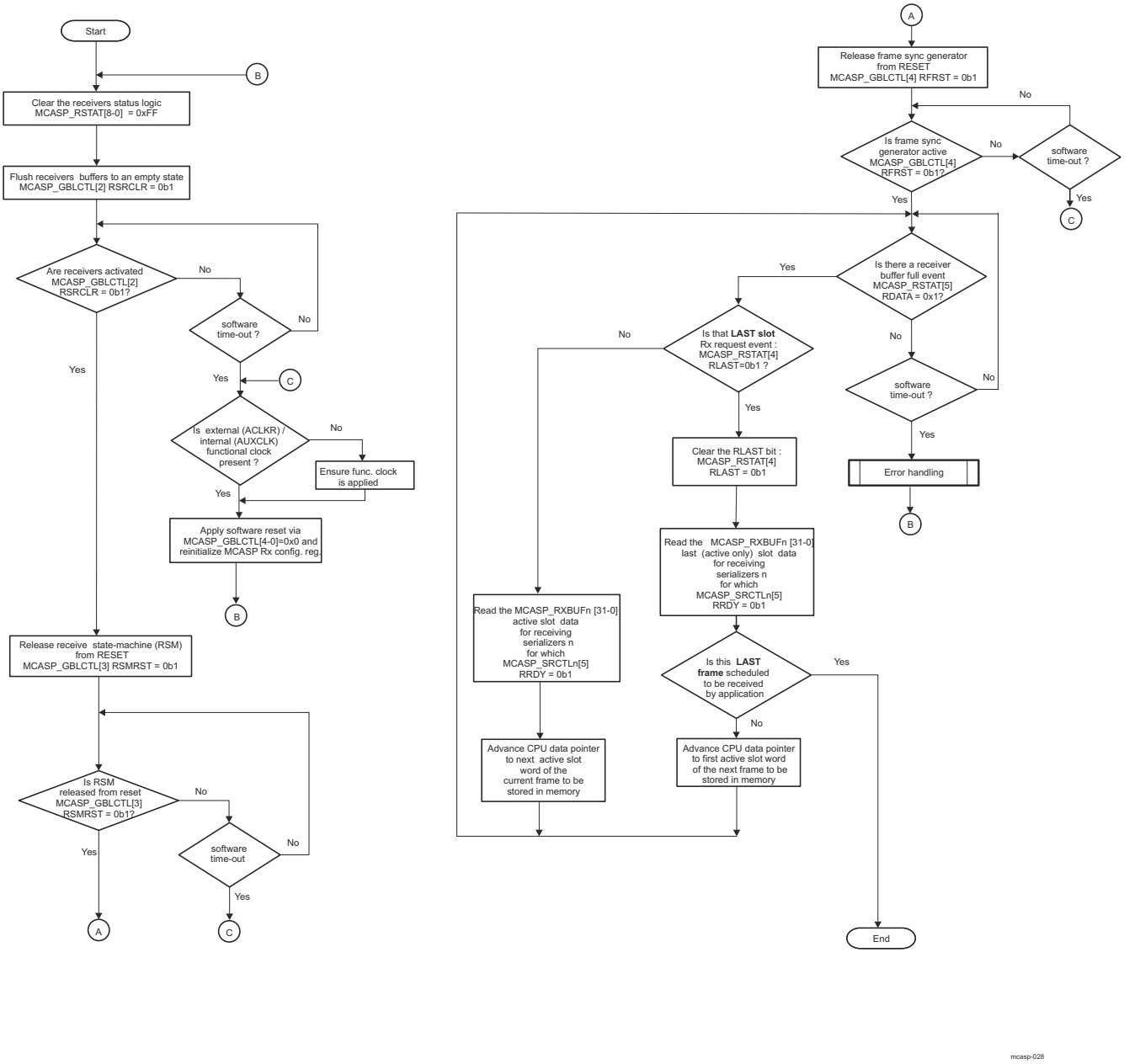


Figure 12-395. MCASP Polling Reception Method

These registers are for MCASP reception polling method: MCASP_RSTAT, MCASP_XGBLCTL, MCASP_RBUFn, MCASP_SRCTLn (n = 0 to 15).

Copyright © 2026 Texas Instruments Incorporated

Table 12-330 summarizes the subprocess call for the polling mode.

Table 12-330. Subprocess Call Summary for Main Sequence – MCASP Reception Polling Method

Subprocess Name	Cross-Reference
Error handling	Figure 12-400

12.5.2.5.2.2.2 Main Sequence – MCASP TDM - Interrupt Reception Method

Figure 12-396 shows the initial setup for interrupt-based reception.

Copyright © 2026 Texas Instruments Incorporated

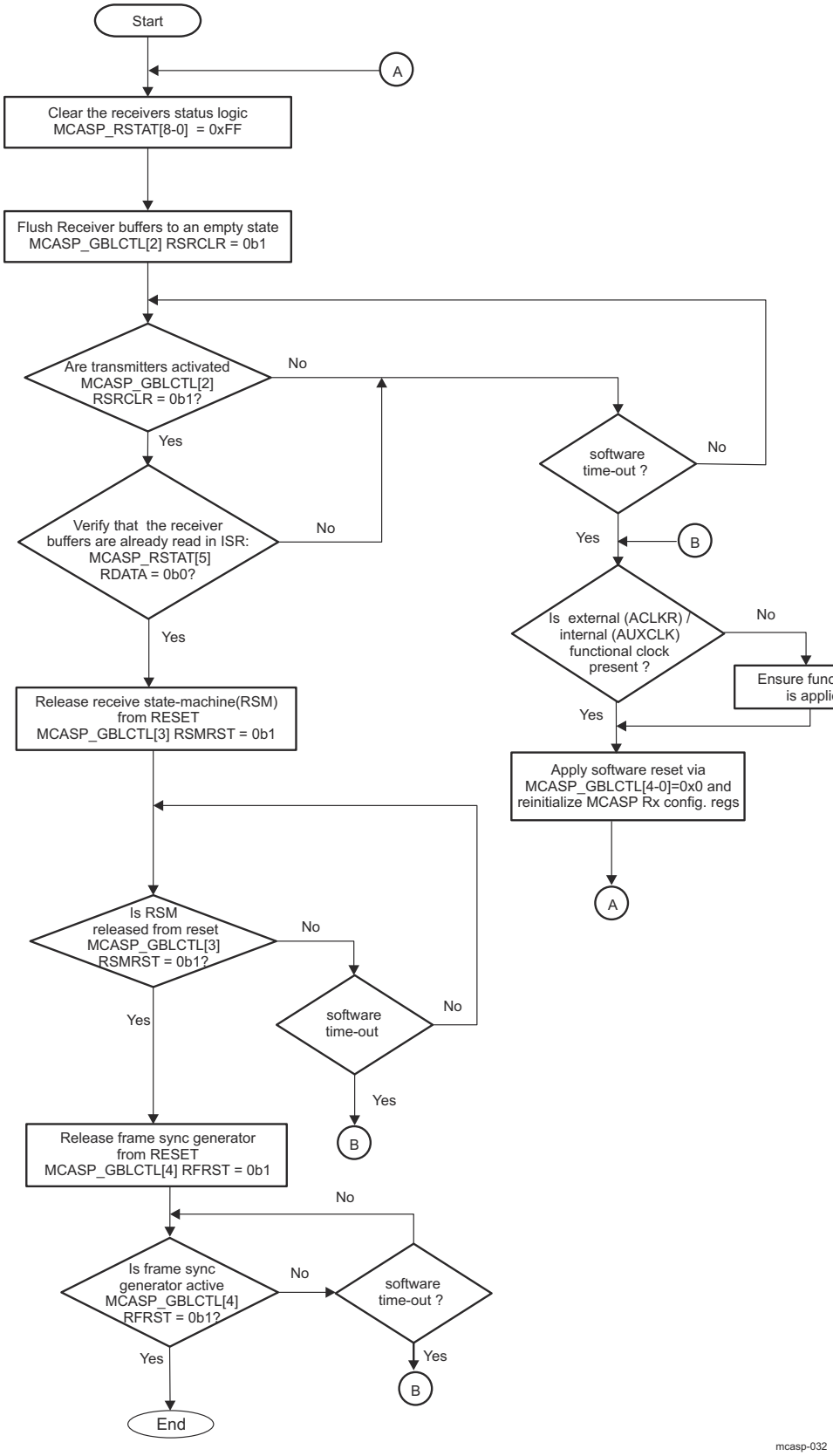


Figure 12-396. Subsequence – TDM - Reception Startup Procedure

Copyright © 2026 Texas Instruments Incorporated

Table 12-331 shows the configuration of the MCASP using an interrupt method for TDM- reception.

Table 12-331. MCASP TDM- Interrupt Reception Model

Step	Register/Bit Field/Programming Model	Value
Disable Rx DMA requests generation.	MCASP_PIDTCTL[0] RDATDMA	0x1
Enable the data ready event receive interrupt.	MCASP_RINTCTL[5] RDATA	0x1
Optional: Enable the receive error event interrupts.	MCASP_RINTCTL[2] RCKFAIL MCASP_RINTCTL[1] RSYNCERR MCASP_RINTCTL[0] ROVRN	0x1 0x1 0x1
Optional: Enable the start of frame interrupt. Optional: Enable the last slot data interrupt	MCASP_RINTCTL[7] RSTAFRM MCASP_RINTCTL[4] RLAST	0x1 0x1
IF read transfer is through the MCASP DATA port (MCASP_RFMT[3] RBUSEL is set to 0b0).	Software test condition (setting is done in step4 of the *MCASP Receivers Global Initialization for TDM-Mode Operation* - see Table 12-317)	
Enable the DATA port error based interrupt.	MCASP_RINTCTL[3] RDMAERR	0x1
ELSE		
Disable the DATA port error based interrupt.	MCASP_RINTCTL[3] RDMAERR	0x0
ENDIF		
TDM - Transmission Startup Procedure	See Figure 12-396.	

These registers are for MCASP TDM- interrupt reception model: MCASP_XGBLCTL, MCASP_RSTAT.

12.5.2.5.2.2.3 Main Sequence – MCASP TDM - Mode DMA Reception Method

Table 12-332 shows the configuration of the MCASP using the DMA method for reception. Possible interrupt error event servicing is also considered. shows the initial setup for DMA - based transmission.

Note

Because of the DATA port burst access capability with the DMA method, it is strongly recommended that DMA transfers are initiated through the MCASP DATA port.

Table 12-332. MCASP DMA Reception Model with Interrupt Events Servicing

Step	Register/Bit Field/Programming Model	Value
Recommended: Select DATA port to access the transmit buffers.	MCASP_RFMT[3] RBUSEL	0x0
Enable the Rx DMA requests generation.	MCASP_PIDTCTL[0] RDATDMA	0x0
Enable the Rx DMA error event, because of MCASP DATA port usage.	MCASP_RINTCTL[3] RDMAERR	0x1
Optional: Enable the receive error event interrupts.	MCASP_RINTCTL[2] RCKFAIL MCASP_RINTCTL[1] RSYNCERR MCASP_RINTCTL[0] ROVRN	0x1 0x1 0x1
Optional: Enable the start of frame interrupt. Optional: Enable the last slot data interrupt.	MCASP_RINTCTL[7] RSTAFRM MCASP_RINTCTL[4] RLAST	0x1 0x1
Disable the data ready event receive interrupt, as DMA is used to service this request.	MCASP_RINTCTL[5] RDATA	0x0
DMA startup reception procedure. This procedure is identical than the one shown in Figure 12-396. The only difference is that DMA automatically services all the RINT events raised by the MCASP, and no CPU data processing intervention is required. The CPU is involved only in error handling shown in Figure 12-398.	See Figure 12-396.	

Copyright © 2026 Texas Instruments Incorporated

12.5.2.5.2.3 MCASP Event Servicing

12.5.2.5.2.3.1 MCASP DIT-/TDM- Transmit Interrupt Events Servicing

Figure 12-397 shows the flow of DIT-/TDM- mode transmit interrupt events servicing for the MCASP module.

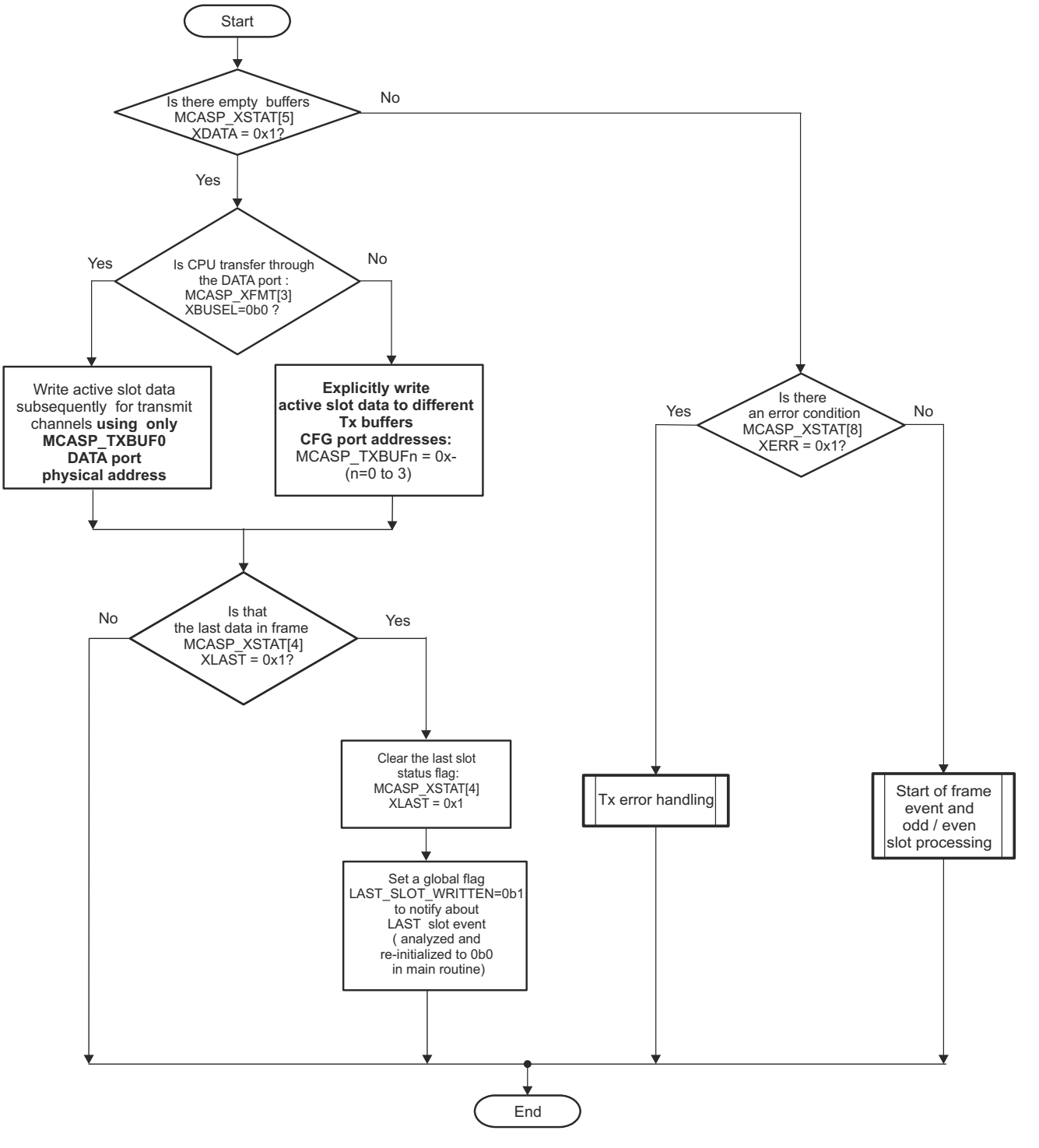


Figure 12-397. MCASP Transmit Interrupt Events Servicing

Copyright © 2026 Texas Instruments Incorporated

12.5.2.5.2.3.2 MCASP TDM- Receive Interrupt Events Servicing

Figure 12-398 shows the flow of DIT-/TDM- mode transmit interrupt events servicing for the MCASP module.

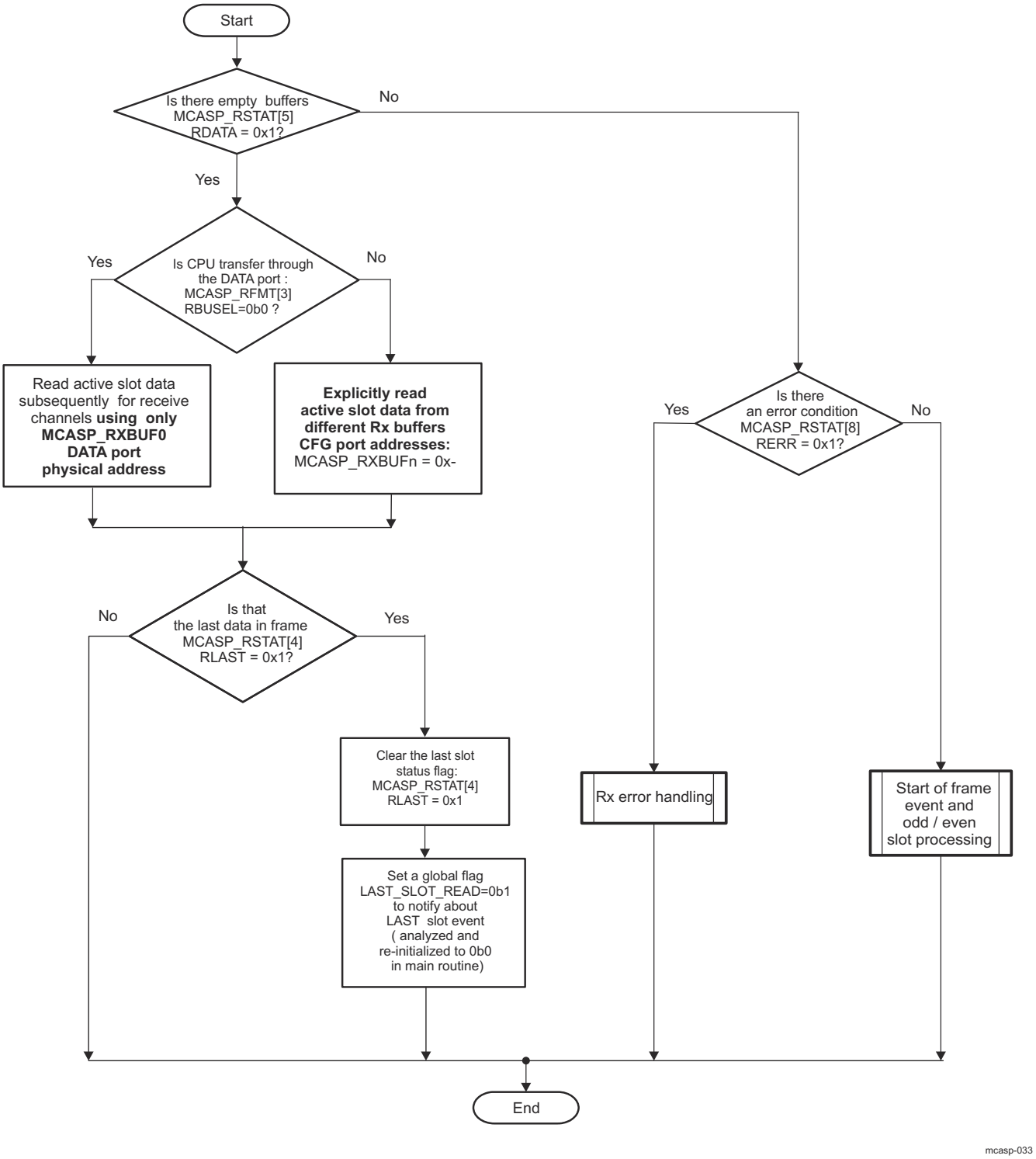


Figure 12-398. MCASP Receive Interrupt Events Servicing

These registers are for MCASP receive interrupt events servicing: MCASP_RSTAT, MCASP_RBUFn, MCASP_RFMT (n = 0 to 15).

Copyright © 2026 Texas Instruments Incorporated

Table 12-333 lists the subprocess call summary for receive interrupt events servicing.

Table 12-333. Subprocess Call Summary for Receive Interrupt Events Servicing

Subprocess Name	Cross-Reference
MCASP receive error handling	Figure 12-400
Start of frame handling	Section 12.5.2.4.12.2, *Receive Data Ready Event and Interrupt*

12.5.2.5.2.3.3 Subsequence – MCASP DIT-/TDM -Modes Transmit Error Handling

Figure 12-399 shows the transmit error handling schema for the MCASP, which can be implemented as part of the Tx interrupt service routine or as part of the Tx polling sequence.

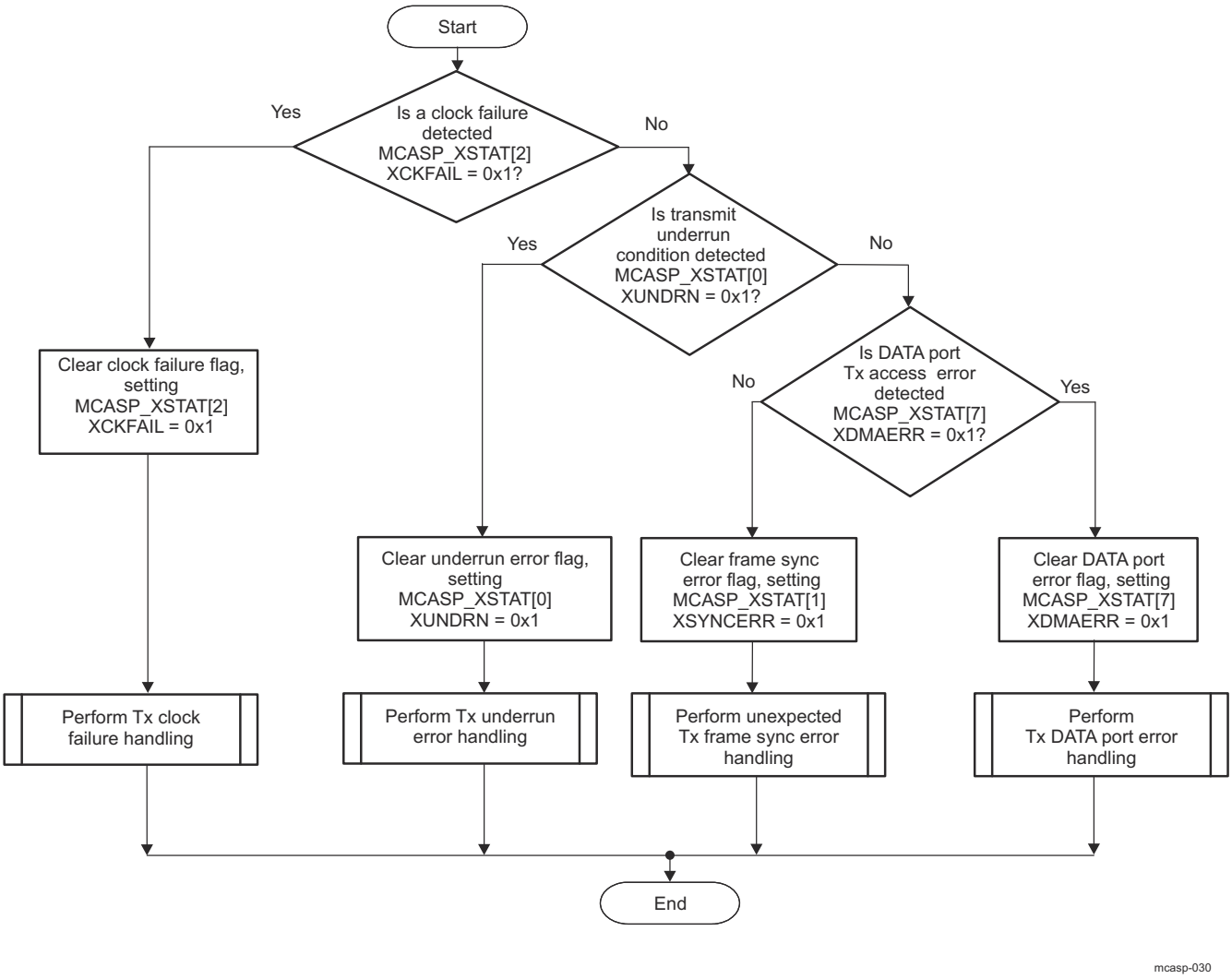


Figure 12-399. MCASP Transmit Error Handling

Copyright © 2026 Texas Instruments Incorporated

Note

- For more information about transmit clock failure handling, see Section 12.5.2.4.15.6.2, *Transmit Clock Failure Check and Recovery*.
- For more information about transmit buffer underrun handling, see Section 12.5.2.4.15.1, *Buffer Underrun Error -Transmitter*.
- For more information about DATA port Tx error handling, see Section 12.5.2.4.15.3, *DATA Port Error - Transmitter*.
- For more information about unexpected Tx frame sync error handling, see Section 12.5.2.4.15.5, *Unexpected Frame Sync Error*.

12.5.2.5.2.3.4 Subsequence – MCASP Receive Error Handling

Figure 12-400 shows the receive error handling schema for the MCASP, which can ONLY be implemented as part of the Rx polling sequence.

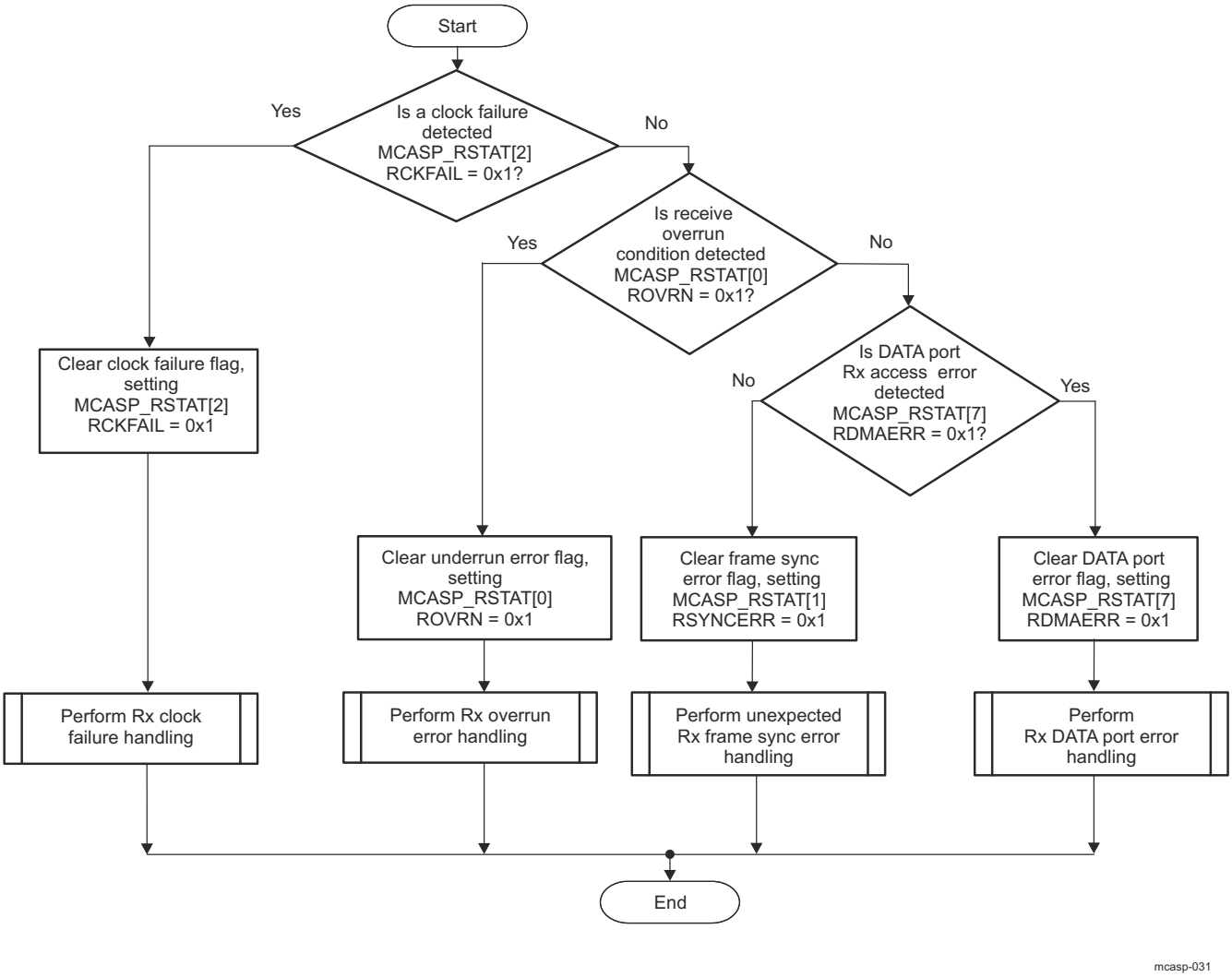


Figure 12-400. MCASP Receive Error Handling

This register is for MCASP receive error handling: MCASP_RSTAT.

Copyright © 2026 Texas Instruments Incorporated

Note

- For more information about receive clock failure handling, see Section 12.5.2.4.15.6.3, *Receive Clock Failure Check and Recovery*.
- For more information about receive buffer overrun handling, see Section 12.5.2.4.15.2, *Buffer Overrun Error - Receiver*.
- For more information about DATA port Rx error handling, see Section 12.5.2.4.15.4, *DATA Port Error - Receiver*.
- For more information about unexpected Rx frame sync error handling, see Section 12.5.2.4.15.5, *Unexpected Frame Sync Error*.

12.6 Camera Peripherals

Copyright © 2026 Texas Instruments Incorporated

12.6.1 Camera Serial Interface Receiver (CSI_RX_IF)

The following sections describe the Camera Serial Interface Receiver (CSI_RX_IF) modules in the device.

12.6.1.1 CSI_RX_IF Overview

The integration of the CSI_RX_IF module allows the device to stream video inputs from multiple cameras to internal memory.

Figure 12-401 shows the CSI_RX_IF module overview.

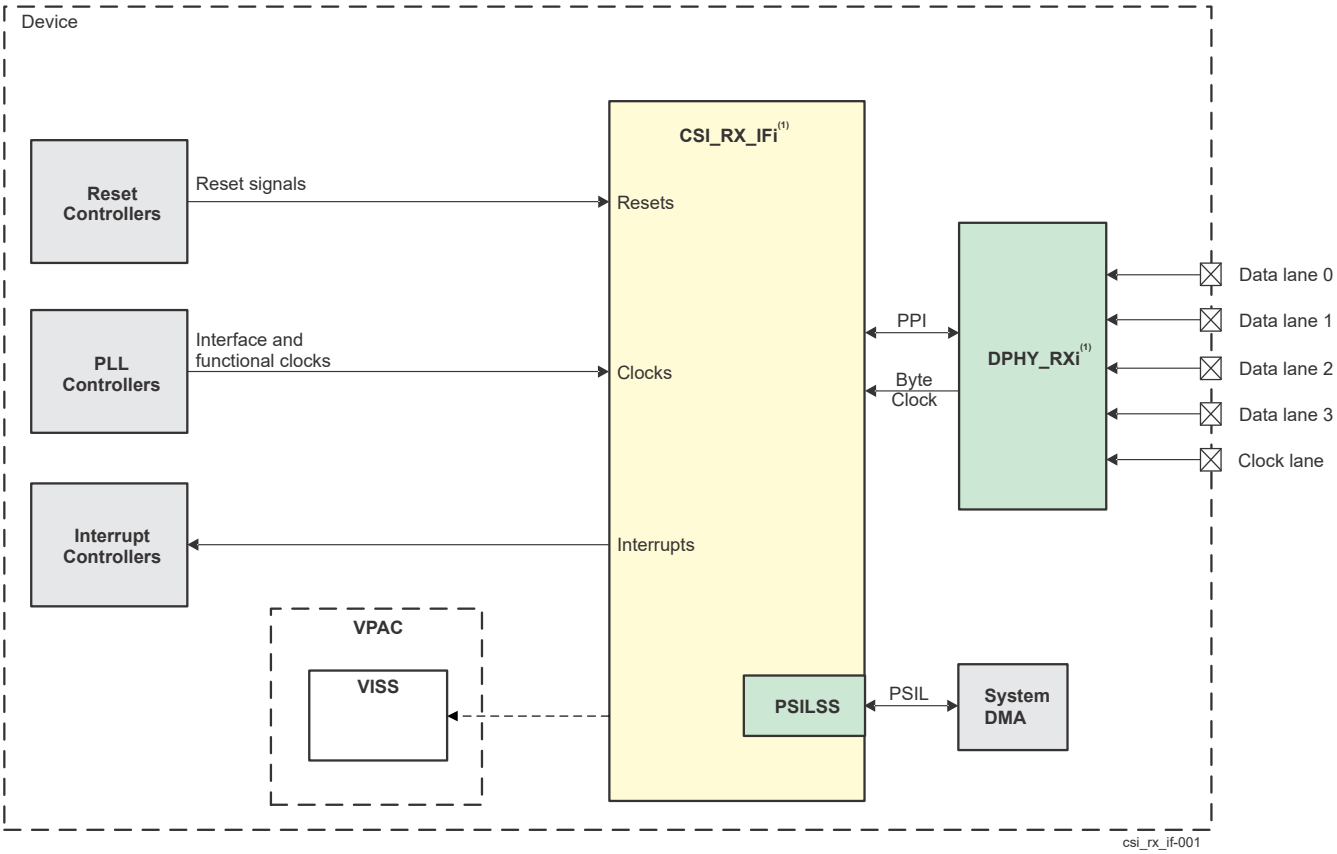


Figure 12-401. CSI_RX_IF Module Overview

12.6.1.1.1 CSI_RX_IF Features

The CSI_RX_IF module supports the following features:

- Compliant to MIPI CSI v1.3
- Supports up to 16 virtual channels per input (partial MIPI CSI v2.0 feature)
- Data rate up to 2.5 Gbps per lane (wire rate)
- Supports 1, 2, 3, or 4 Data Lane connection to DPHY_RX
- Programmable formats including YUV420, YUV422, RGB, Raw, and User Defined (over 25 different formats supported)
- One independent (simultaneous) output stream:
  - One (up to 32 Channels) DMA interface through a 128-bit PSI_L connection to DMSS for transfers to memory:
    - Byte packed (32x4) format, elastic buffer mode
    - Max rate 1 data cycle every 4 main clocks
    - ByteValid per byte in Last Data Phase (LDP)
    - 32 thread ID's supported (virtual channel & data type combinations); Flexible number of threads (32 Max)

Copyright © 2026 Texas Instruments Incorporated

    - Virtual channels and data types mapped via mmr to PSI_L thread ID's
    - Internal FF based FIFO; RAM based buffer (2kx128)
- Functional and data path error interrupts
- ECC support

12.6.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.6.1.2 CSI_RX_IF Environment

The CSI_RX_IF has no dedicated pins. At the device level, video inputs come from the DPHY_RX, see Section 12.6.2.

12.6.1.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.6.1.4 CSI_RX_IF Functional Description

12.6.1.4.1 CSI_RX_IF Block Diagram

Figure 12-402 depicts a simplified internal block diagram of the CSI_RX_IF and its surroundings.

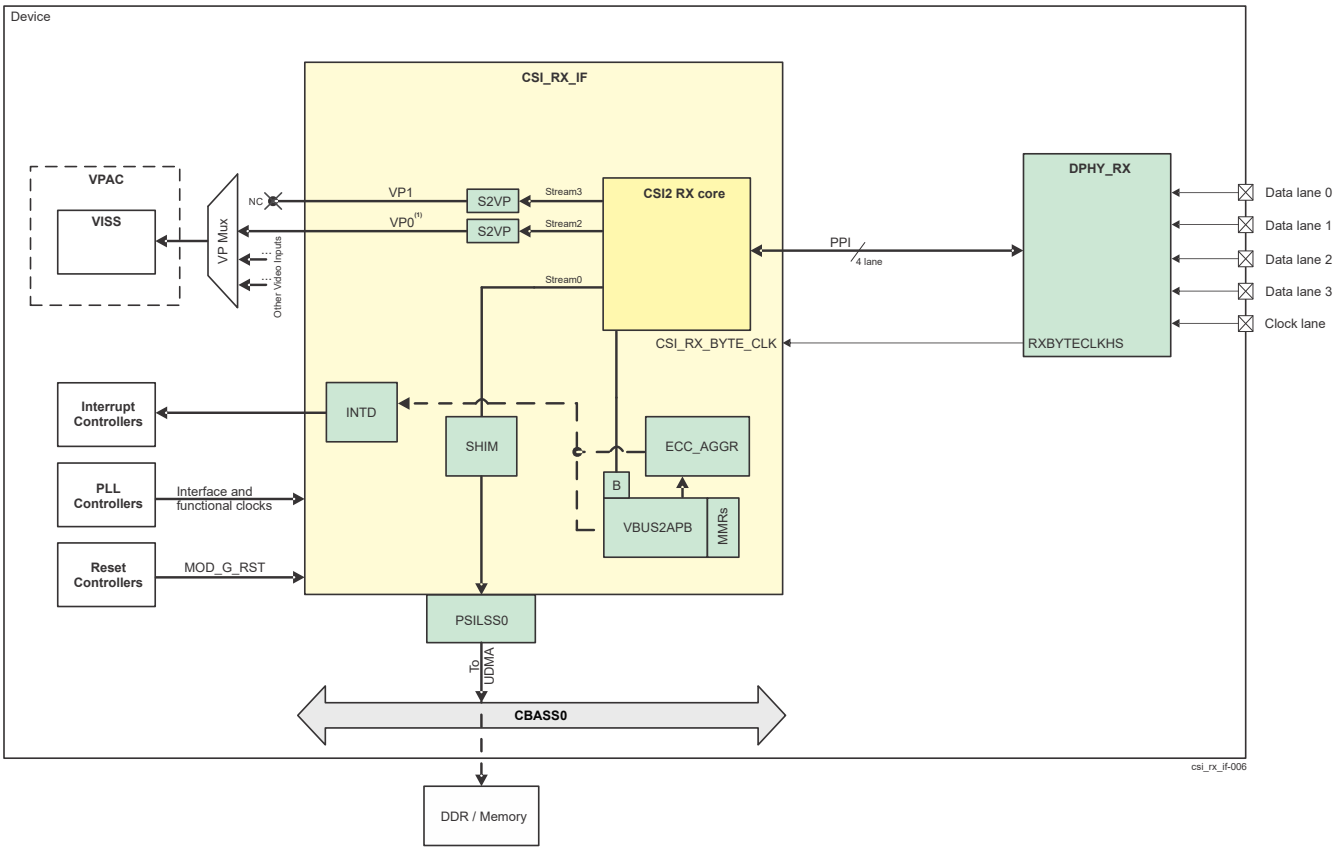


Figure 12-402. CSI_RX_IF Block Diagram

The CSI2 core streams 4 channels of data to the export path.

Stream0 (high BW DMA up to 32 channel contexts) streams 32-bits worth of data, that is re-formatted and sent as pixel format into 128 bits as PSIL data. There are 32 sets of MMRs/Registers that map virtual channels and data type to PSIL threads. Data is sent out PSIL in FIFO order as it is received. There is no priority. Data re-formatting is done to end up with correct data organization in memory so that other ip can use the data. The data type must be configured in MMRs to ensure proper reformatting. See Section 12.6.1.4.5. The large buffer of PSIL data (2048x128) has ECC protection, see Section 12.6.1.4.7

12.6.1.4.2 CSI_RX_IF Hardware and Software Reset

An active low asynchronous hardware reset is provided to CSI_RX_IF by device LPSC. It is internally resynchronized to the functional clock domain.

A software reset is triggered by configuring the CSIRX_SOFT_RESET bit-fields for protocol reset and/or module reset.

12.6.1.4.3 CSI_RX_IF Clock Configuration

There are four clock domain in the CSI_RX_IF.

1. The CSI_RX_MAIN_CLK clock runs the CSI2 core PSIL DMA. The minimum clock rate for the CSI_RX_MAIN_CLK is 312.5MHz when DPHY is at max data rate (slower clock permissible when DPHY is at lower data rates). When CSI_RX_MAIN_CLK is operating lower than 312.5MHz, then the clock is

Copyright © 2026 Texas Instruments Incorporated

essentially limiting the clock rate of the DPHY_RX. Said another way, CSI_RX_MAIN_CLK must be at least the CSI_RX_BYTE_CLK rate, else FIFOs will overflow.
The maximum clock rate is 500MHz 16ffc, 650 Max
f(CSI_RX_BYTE_CLK) × nb_lanes / 4; where f(CSI_RX_BYTE_CLK) is the minimum frequency of CSI_RX_BYTE_CLK.

2. The CSI_RX_VBUS_CLK is the interface configuration clock that runs at half the speed of the CSI_RX_MAIN_CLK.
3. The CSI_RX_BYTE_CLK is the clock supplied by the DPHY_RX PLL and is divided down to byte clock. The DPHY_RX is designed for max of 10gbps. This translates to a max byte clock of 312.5MHz. The clock is inactive when DPHY_RX is not in HS operation.

Table 12-334 shows the CSI_RX_IF and DPHY_RX inter-clock dependencies.

Table 12-334. CSI_RX_IF Inter-clock Dependencies

	CSI_RX_MAIN_CLK	CSI_RX_VBUS_CLK	CSI_RX_VP_CLK	CSI_RX_BYTE_CLK
Min freq	CSI_RX_BYTE_CLK freq	CSI_RX_MAIN_CLK / 2 freq	CSI_RX_BYTE_CLK freq	N/A
Max freq	500MHz	CSI_RX_MAIN_CLK / 2 freq	720MHz	312.5MHz

12.6.1.4.4 CSI_RX_IF Interrupt Events

This section describes the register configuration of the interrupt events that can trigger the several CSI_RX_IF interrupt signals. For detailed description and mapping of the interrupt ot the device interrupt processors see *CSI_RX_IF Integration*.

The interrupts are generally handled wihtin the INTD module of the CSI_RX_IF, although there are several interrupt registers in the ECC_AGGR for ECC errors and in the VBUS2APB for stream monitoring errors/flags.

Table 12-335 lists the event generation and corresponding registers of the CSI_RX_IF controller.

Table 12-335. CSI_RX_IF Interrupt Events Cross Table

Event	Mask Regsiter	Status Register	Description
CSI_RX_CSI_ERR_IRQ	CSIRX_ERROR_IRQS_MASK_CFG	CSIRX_ERROR_IRQS	Stream error detected. The CSI_RX_IF0 will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSI_RX_CSI_ERR_IRQ. Read the status regsiter bitfields to trace the source of the event.
CSI_RX_CSI_IRQ	INTD_CFG_ENABLE_REG_LEVEL_0 CSIRX_ERROR_IRQS_MASK_CFG	INTD_CFG_STATUS_REG_LEVEL_0 CSIRX_ERROR_IRQS	Global functional interrupt that various resynchronized sources converge into interrupt generation.
CSI_RX_CSI_LEVEL	CSIRX_MONITOR_IRQS_MASK_CFG	CSIRX_STREAM0_FIFO_FILL_LVL CSIRX_STREAM1_FIFO_FILL_LVL CSIRX_STREAM2_FIFO_FILL_LVL CSIRX_STREAM3_FIFO_FILL_LVL CSIRX_STREAM0_FIFO_FILL_LVL	PSI_L fifo overflow or VP0/VP1 frame/ line mismatch (at the minimum an error interrupt will occur at the end of frame but may be issued within the frame as well). Read the status regsiter bitfields to trace the source of the event.
CSI_RX_CORR_LEVEL	CSIRX_ASF_INT_MASK	CSIRX_ASF_INT_STATUS	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.
CSI_RX_UNCORR_LEVEL	CSIRX_ASF_INT_MASK	CSIRX_ASF_INT_STATUS	This interrupt is for checking the interface signals of the CSI_RX_IF0 controller for parity.

Copyright © 2026 Texas Instruments Incorporated

Table 12-335. CSI_RX_IF Interrupt Events Cross Table (continued)

Event	Mask Regsiter	Status Register	Description
CSI_RX_CSI_FATAL	CSIRX_ASF_INT_MASK	CSIRX_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSIRX_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSIRX_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.
CSI_RX_CSI_NONFATAL	CSIRX_ASF_INT_MASK	CSIRX_ASF_INT_STATUS	ASF port non-fatal interrupt. Level sensitive. Set CSIRX_ASF_FATAL_NONFATAL_SELECT to whether fatal or non-fatal ASF interrupt is triggered. If any of the CSIRX_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.

12.6.1.4.5 CSI_RX_IF Data Memory Organization Details

Figure 12-403 shows the YUV422-8 data organization in memory.

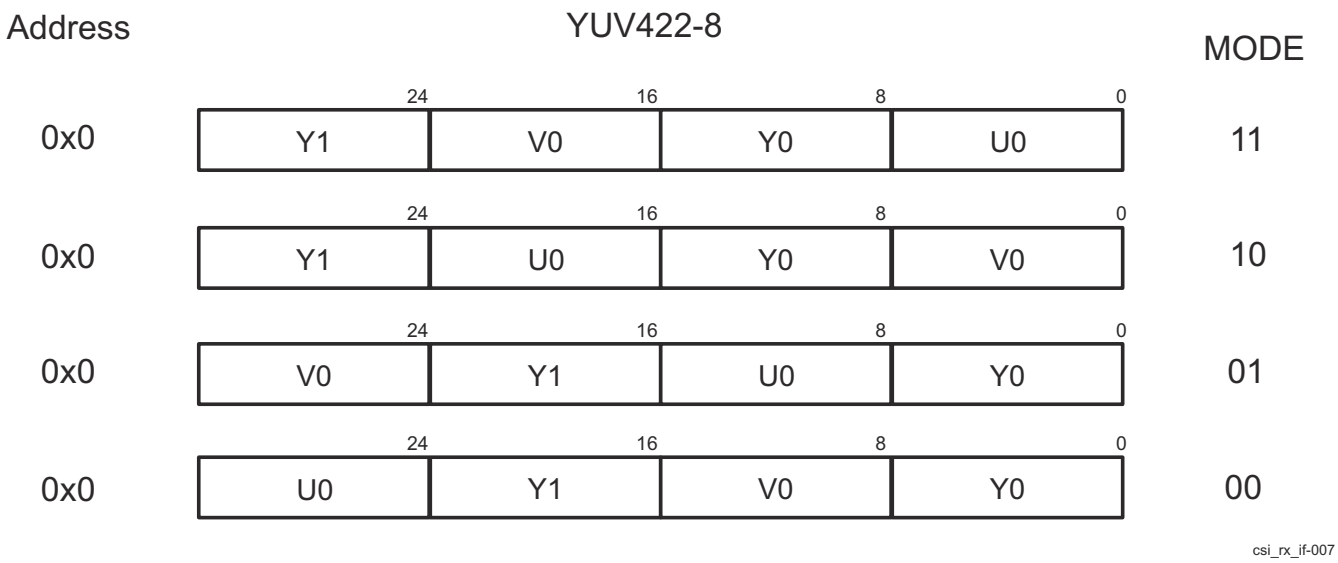


Figure 12-403. CSI_RX_IF YUV422-8 Memory Data Organization

Figure 12-404 shows the YUV422-10 data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

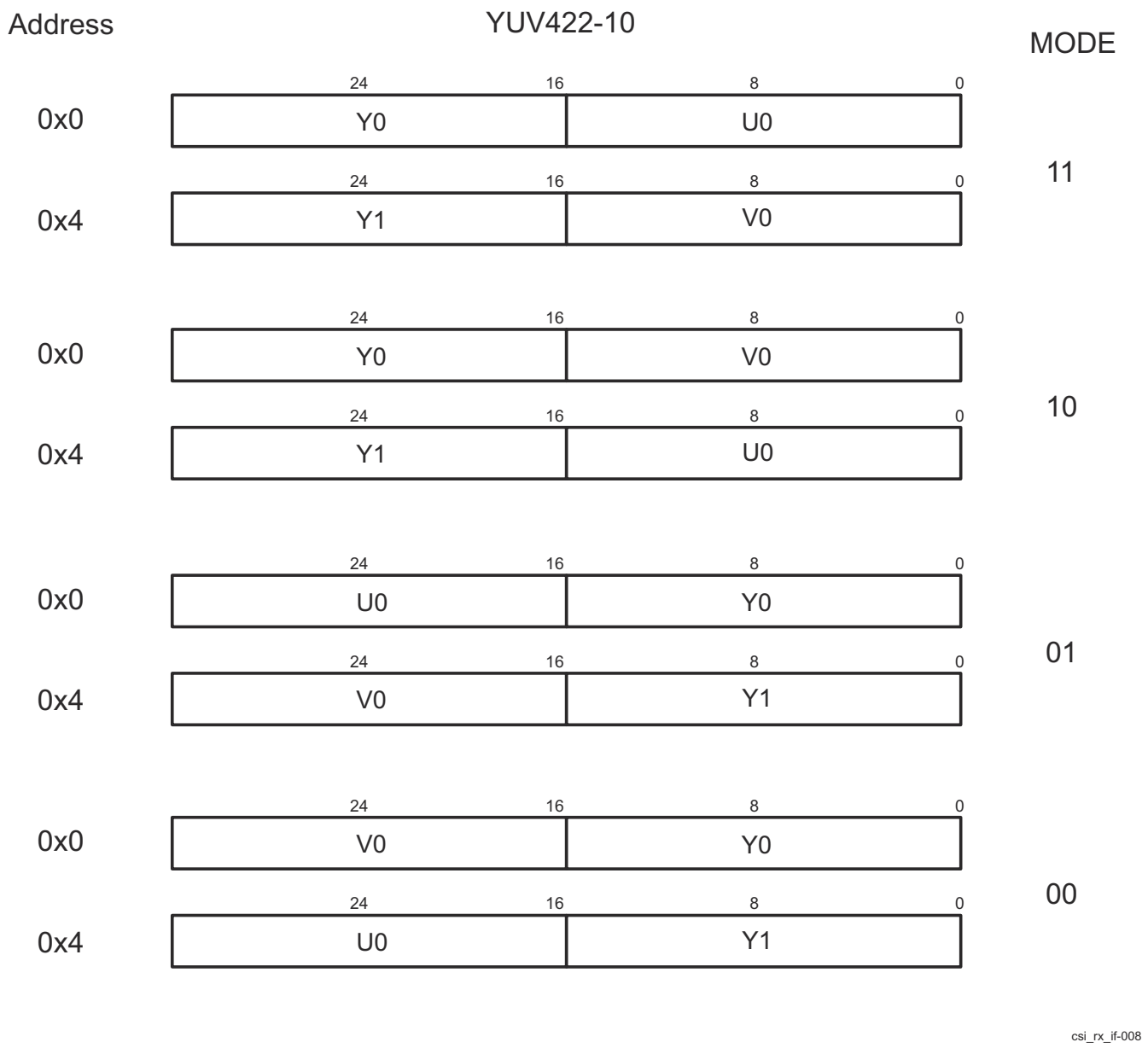


Figure 12-404. CSI_RX_IF YUV422-10 memory data organization

Figure 12-405 shows the YUV420-8 data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

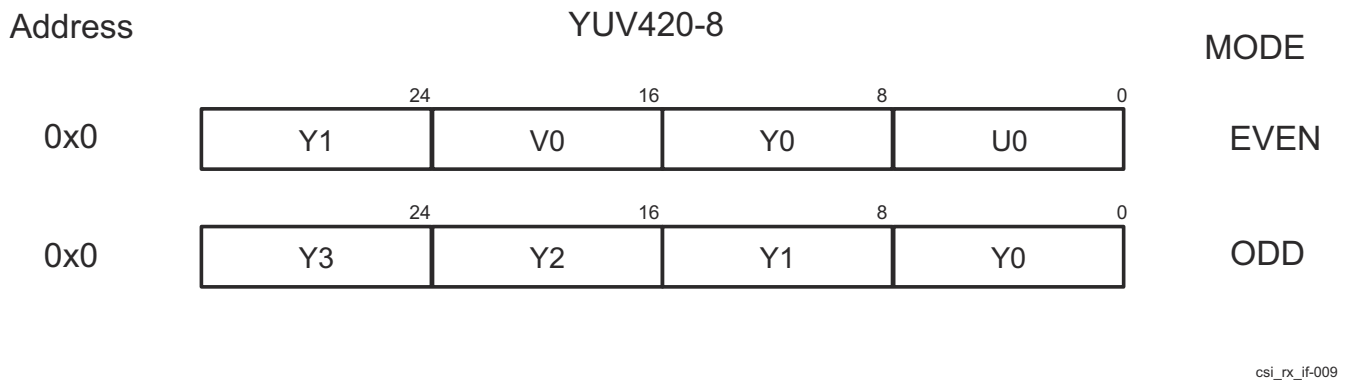


Figure 12-405. CSI_RX_IF YUV420-8 memory data organization

Figure 12-406 shows the YUV420-10 data organization in memory.

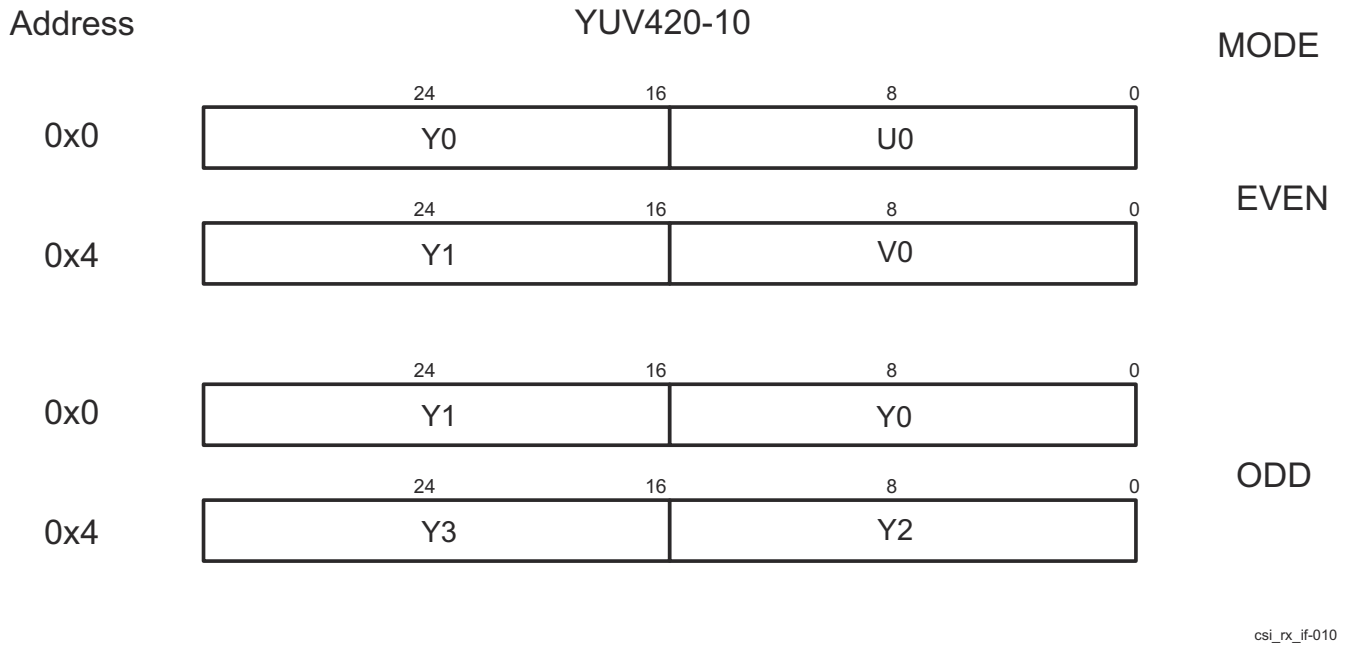


Figure 12-406. CSI_RX_IF YUV420-10 memory data organization

Figure 12-407 shows the RAW data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

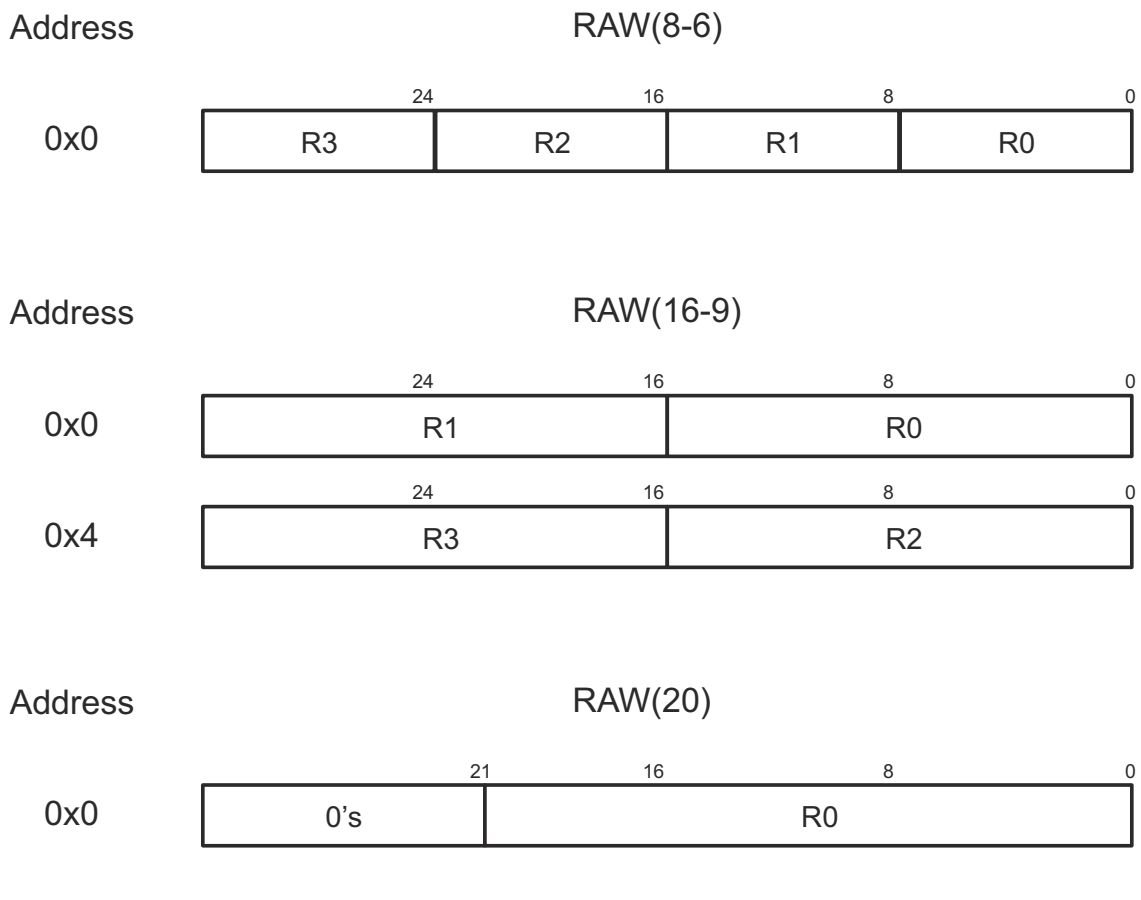


Figure 12-407. CSI_RX_IF RAW (UNPACKED) memory data organization

Figure 12-408 shows the RGB data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

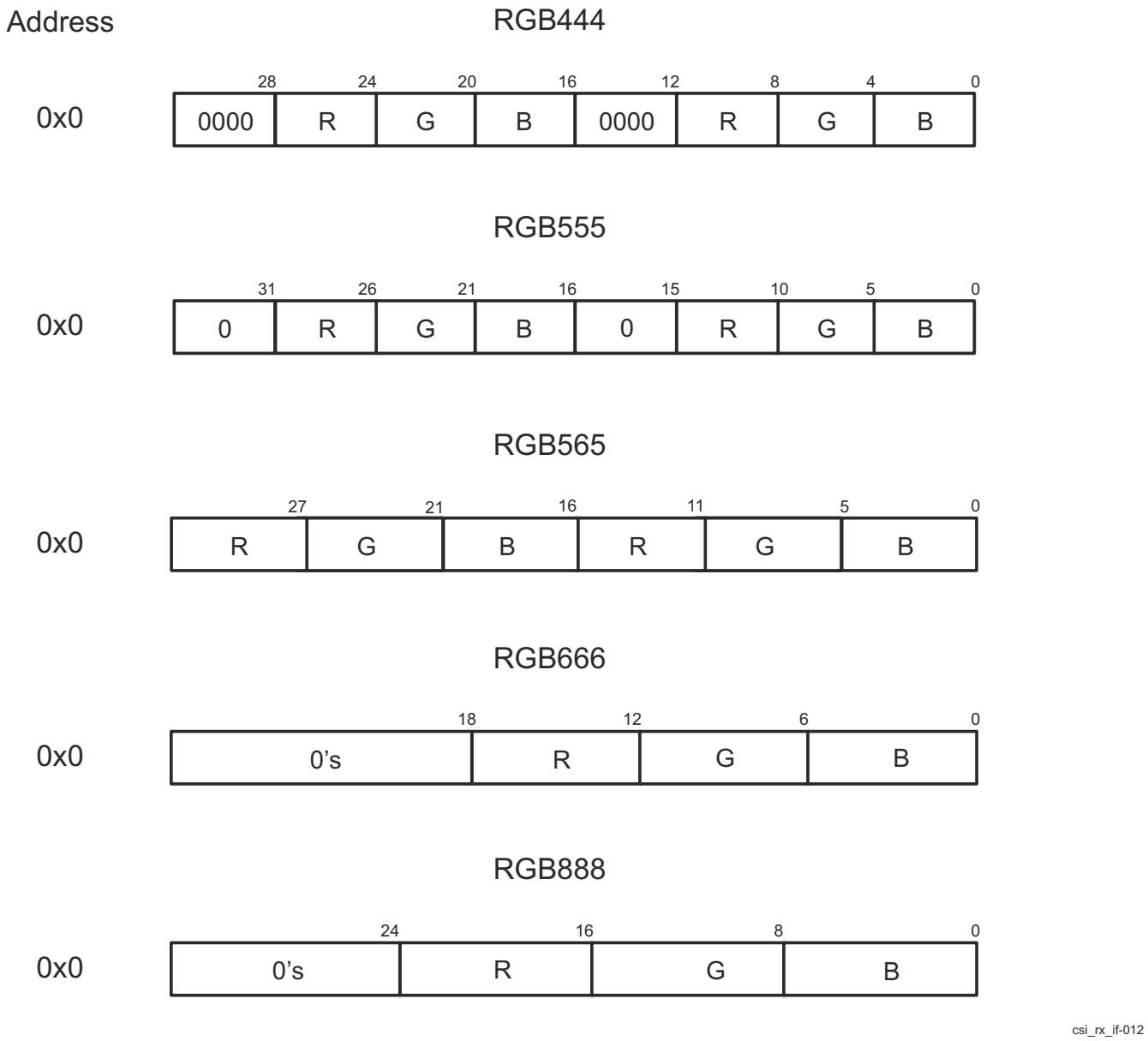
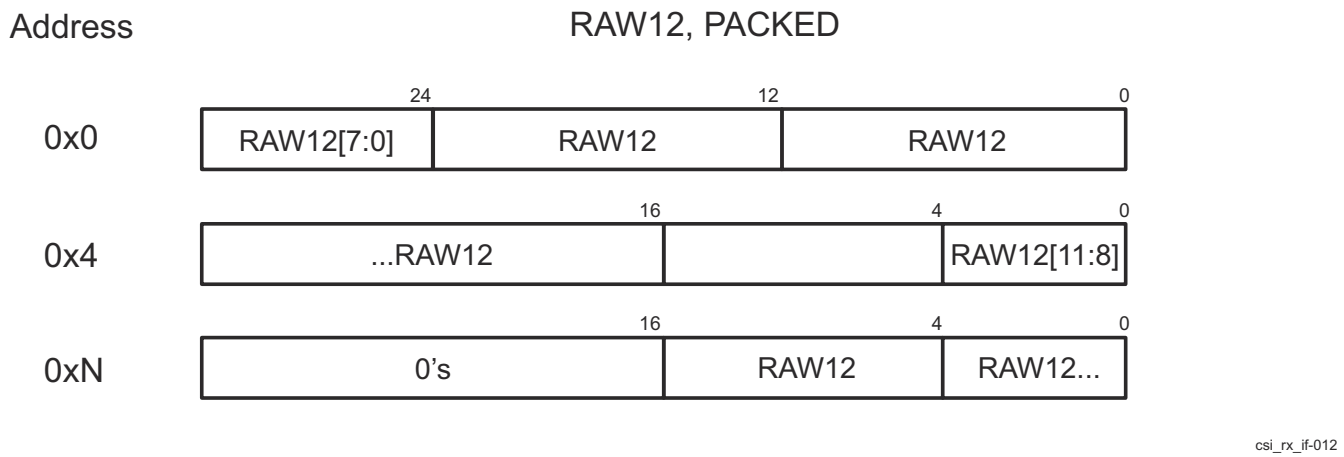


Figure 12-408. CSI_RX_IF RGB memory data organization

Figure 12-409 shows the RAW12 (PACKED) data organization in memory.

Copyright © 2026 Texas Instruments Incorporated



A. Where data does not fill out to a byte boundary it is zero extended.

Figure 12-409. CSI_RX_IF RAW12 (PACKED) memory data organization

12.6.1.4.6 CSI_RX_IF PSI_L (DMA) Interface

Note

This section describes CSI_RX_IF PSI_L specific functionality related to the camera interface. The general PSI_L functional description and registers are described in the PSI_L specific chapter (see *PSIL Subsystem (PSILSS)*).

12.6.1.4.6.1 PSI_L DMA framing

The CSI_RX_IF stream interface provides the usual VSYNC/HSYNC signals. The signals are per virtual channel where the transition from 1-to-0 or 0-to-1 represents SOF/EOF or SOL/EOL. VSYNC/HSYNC transition is usually not coincident with a valid data phase, though there are use cases where it can be. Because of the nature of CSI protocol, the VSYNC and HSYNC transactions require at least a single clock cycle and can only transition sequentially (important fact in handling of VSYNC events).

There is a rarely used mode of CSI where an entire frame is passed in a single packets without line breaks. In this mode, there are no HSYNC transitions at line starts/ends.

Lastly, there is redundant information provided with each CSI packet where the packet length is supplied in bytes.

To overcome these framing restrictions, the CSI_RX_IF PSILSS0 passes metadata through the DMA FIFO. Anytime metadata is inserted into the FIFO, the CSI_RX_IF stream is stalled for a clock cycle. The forms of supported meta tags are:

- SOF for a given virtChan
- EOF for a given virtChan
- Packet length in bytes for a given dmaCntx

The PSILSS0 will disregard the HSYNC signal entirely. The long packet beginning is used as SOL and the EOL will be "counted" using the packet length provided. In the special case where packets are of FRAME length, the SOL/EOL handling is identical.

The PSILSS0 solution for VSYNC handling is to create the concept of metadata FIFO entry which is indicated by a metadata bit. The SOF or EOF VSYNC information plus the virtualChan index is fed into the FIFO. On the output of the FIFO, the SOF is saved for each context of that virtualChan type. The very next data phase of that channel context will be marked as SOF. Additionally a state bit per context is maintained indicating the channel is MOP (middle of packet). Once the EOF arrives at the FIFO output, all contexts of that type virtual channel (and currently in MOP state) will receive a PSI_EOP with zero active bytes of data.

Copyright © 2026 Texas Instruments Incorporated

CAUTION

The line and frame size is not known outside the CSI core (i.e. not passed to the DMA interface). Therefore any mismatches at system level programming will be unknown. If the DMA line/frame size does not match the CSI_RX_IF line/frame size, it is assumed the DMA will not result in any adverse side affects as a result of not enough data or too much data.

12.6.1.4.6.2 PSI_L DMA error handling due to FIFO overflow

The DMA error handling is also called a PSI_L protocol enforcer. It is intended to prevent hang of the PSILSS0. Unnatural packet size should not cause hang so only SOP/SOL/EOL/EOP framing is enforced. The following list highlights the error handling mechanism:

- For context cleanup the protocol enforcer:
  - cycle through each context index checking if context is in MOL or MOP
  - close out MOP with EOP and MOL&MOP with EOL&EOP
- dropOnFloor SOP if currently in MOPstate
- dropOnFloor EOP if not currently in MOPstate
- dropOnFloor FIFO data
- EOL context if EOP and MOLstate
- After closing out all open contexts the PSILSS0 logic will then wait till end of frame per virtual channel. Once a new frame starts it will then start sending out data from that new frame

12.6.1.4.7 CSI_RX_IF ECC Protection Support

ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

The ECC protection on the CSI_RX_IF RAM provides Single Error Correction and Double Error Detection (SEC/DED). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC protection also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

The ECC aggregator in the CSI_RX_IF subsystem level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. It provides a single EOI-handshake based interrupt to the interrupt processors (for both single and double error detections) and a standard 32-bit VBUSP interface for configuring and querying the ECC register set, see *CSI_RX_IF_ECC_AGGR_0 Registers*. For complete details on the features and functions of the ECC aggregator, see chapter ,*ECC Aggregator*.

12.6.1.4.8 CSI_RX_IF Programming Guide

12.6.1.4.8.1 Overview

This section details specific steps on how to program the CSI_RX_IF controller.

12.6.1.4.8.2 Controller Configuration

The CSI_RX_IF streams will default to accept all virtual channels and all data types after reset, so the user must program the stream configuration and pixel interface to match the system use case.

Note

The CSI_RX_IF controller can be configured so that the reset values can adopt the users Power_On state. This can reduce the programming steps required by the system.

Copyright © 2026 Texas Instruments Incorporated

The CSI_RX_IF controller is designed to operate all the defined streams with all virtual channels and all data types passed to the pixel interface. Also, the connection to the front interface adopts reset values that will allow the connection of the lanes to expect no remapping.

The basic system configuration steps are then programming the number of enabled data lanes and starting the streams.

The system can also decrease the virtual channel and data type processed by the stream by configuring the data config (CSIRX_STREAM0_DATA_CFG - CSIRX_STREAM3_DATA_CFG) registers.

The user must perform a read from the stream config register (CSIRX_STREAM0_CFG - CSIRX_STREAM3_CFG) at power up to identify the number of streams and pixel interface mode. The stream FIFO depth must be determined from the system configuration information defined at build time to ensure that any programmed [31-16] FIFO_FILL level is valid for the available FIFO depth.

12.6.1.4.8.3 Power on Configuration

The CSI_RX_IF requires the system to perform the configuration of the DPHY_RX interface before starting the streams of the controller. The default configuration of the controller and each stream can be identified by reading the device ID CSIRX_DEVICE_CONFIG register.

The FW must read the CSIRX_DEVICE_CONFIG register at power up. This will provide the FW the number of streams that will need to be configured, and all the default system information for the FIFO structures in the available streams.

Table 12-336. CSIRX_DEVICE_CONFIG bitfield details

STREAMx_NUM_PIXELS	The width of the pixel interface and the bits per pixel for the selected datatype will determine how many pixels can be output in a single cycle. Default will be 1 pixel per clock. 00 -> 1 pixel per clock 01 -> 2 pixels per clock 10 -> 4 pixels per clock
DATAPATH_SIZE	Internal Datapath width 00 - 32 bit, all other values are reserved.
NUM_STREAMS	Number of Stream interfaces (1-4) = (value+1)
MAX_LANE_NB	Max Number of Lanes (1-4) = (value+1)

Figure 12-410 shows the minimal sequence of registers that will configure the DPHY_RX and then start the stream; this will output all the pixel information for all virtual channels and all data types detected in the link data stream.

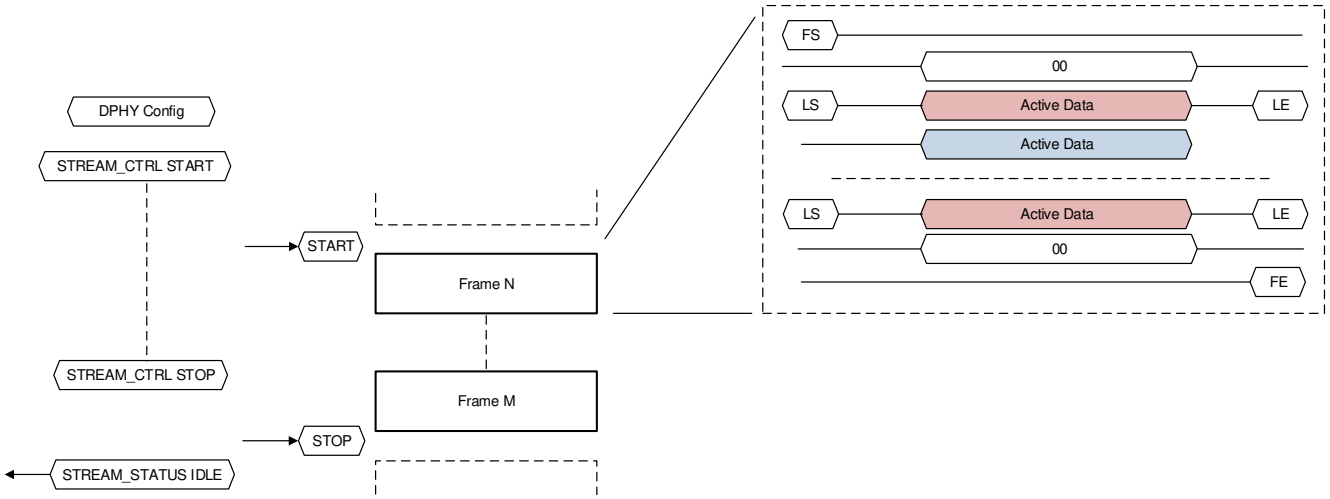


Figure 12-410. Minimal Stream Control - Start and Stop

Copyright © 2026 Texas Instruments Incorporated

12.6.1.4.8.4 Stream Start and Stop

The CSI_RX_IF will perform management of the stop and halt control to the pixel stream during functional operation to allow any stream to change the virtual channel or data type information that the stream will process.

- The FW will use the stream control register (CSIRX_STREAM0_CTRL - CSIRX_STREAM3_CTRL) to perform a stop (set bit 1) and wait for the end of any current frame, and wait for the stream to return its state to IDLE, which can be read from stream status register register (CSIRX_STREAM0_STATUS - CSIRX_STREAM3_STATUS)[31] RUNNING = 1.
- The FW will use the stream control register to perform an abort and wait for the end of any current line, and wait for the stream to return its state to IDLE, which can be read from register STREAM_STATUS[31] RUNNING = 1.

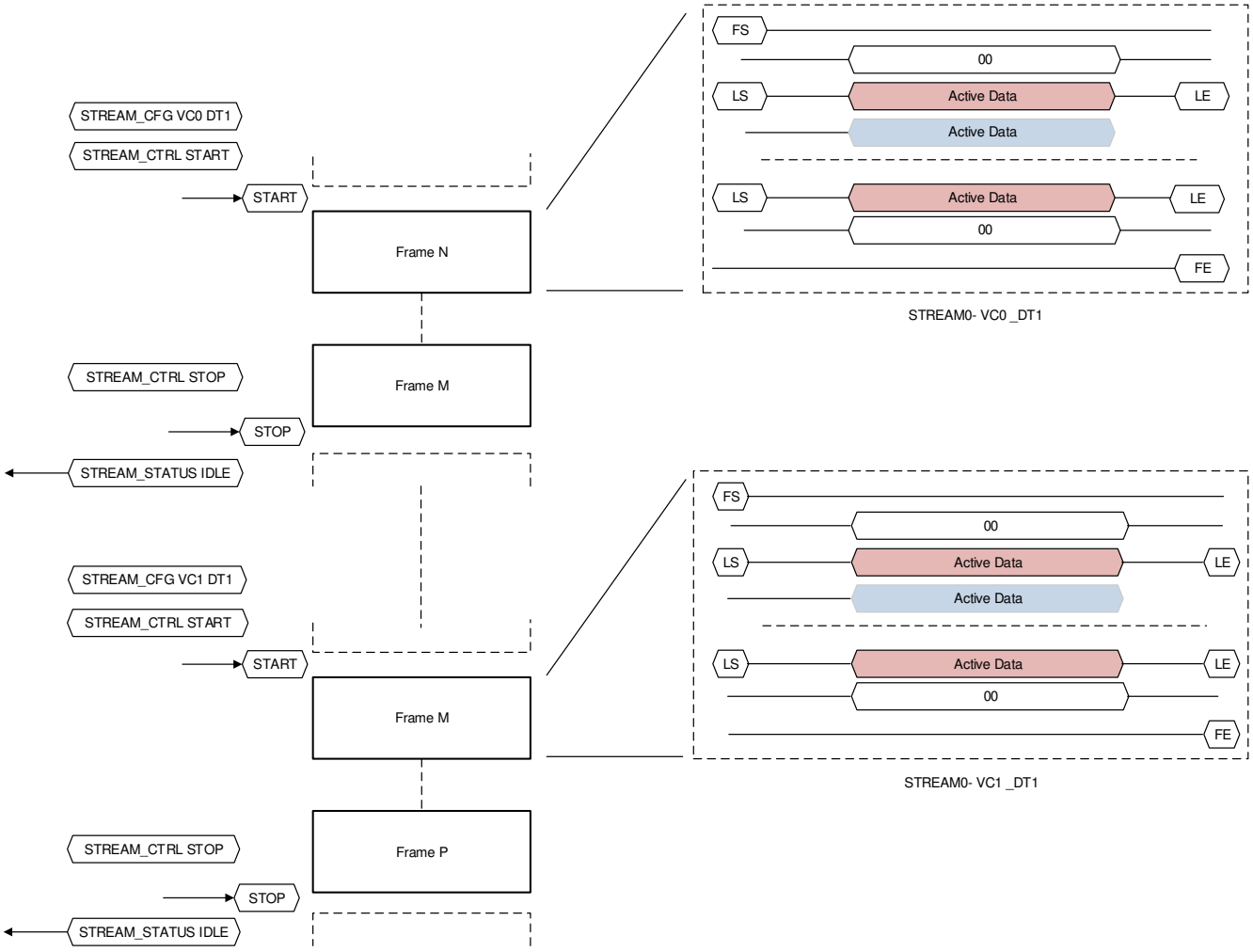


Figure 12-411. Stream Reconfiguration Using Start Stop Start Flow

The CSI_RX_IF will use the monitor control or STOP mechanism to stop the stream processing at the end of the active frame. The stream monitor, configuration and interrupt registers can be redefined and then the stream restarted. The pixel interface will begin processing at the next frame start.

Copyright © 2026 Texas Instruments Incorporated

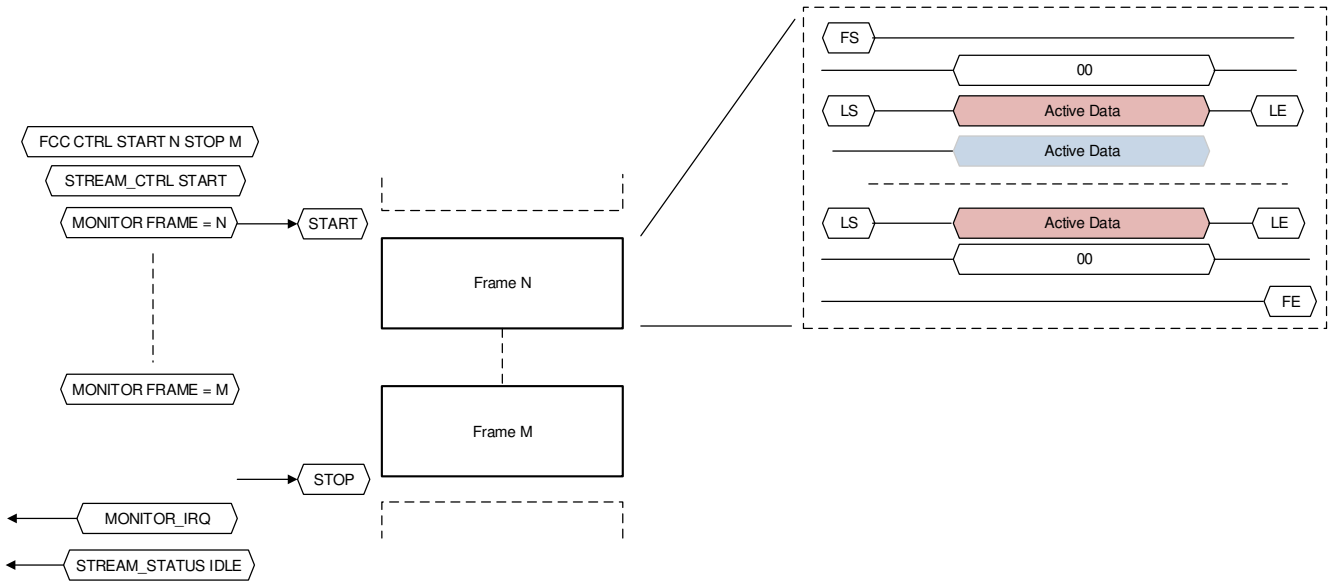


Figure 12-412. Stream Start and Stop Using Monitor Control

12.6.1.4.8.5 Error Control With Soft Resets

The CSI_RX_IF will perform control of the soft reset either for error event recovery or to clear a stream FIFO or internal state machine.

- The FRONT block can be soft reset if the DPHY_RX becomes unresponsive and the controller wishes to maintain its configuration. In this case the DPHY_RX resets can be applied and the DPHY_RX enabled to begin the transfer again.
- The Protocol block can be soft reset if the FRONT soft reset is required, and the protocol is not in the IDLE state.
- The stream soft resets (CSIRX_STREAM0_CTRL - CSIRX_STREAM3_CTRL)[4] SOFT_RST can be used to clear the stream to the stop state and reset all the stream state machines and FIFO. If the system has a failure and wishes to clear the stream FIFO and return to a safe state on the pixel interface, the stream soft reset should be asserted.

12.6.1.4.8.6 Stream Error Detected – No Error Bypass Mode

The CSI_RX_IF will default to stop processing the frame whenever a header Reserved DT or ECC error is detected, or when a long packet payload CRC is identified.

The CSI_RX_IF will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSIRX_ERROR_IRQS.

The CSI_RX_IF will stop processing the current frame and stop the stream.

Copyright © 2026 Texas Instruments Incorporated

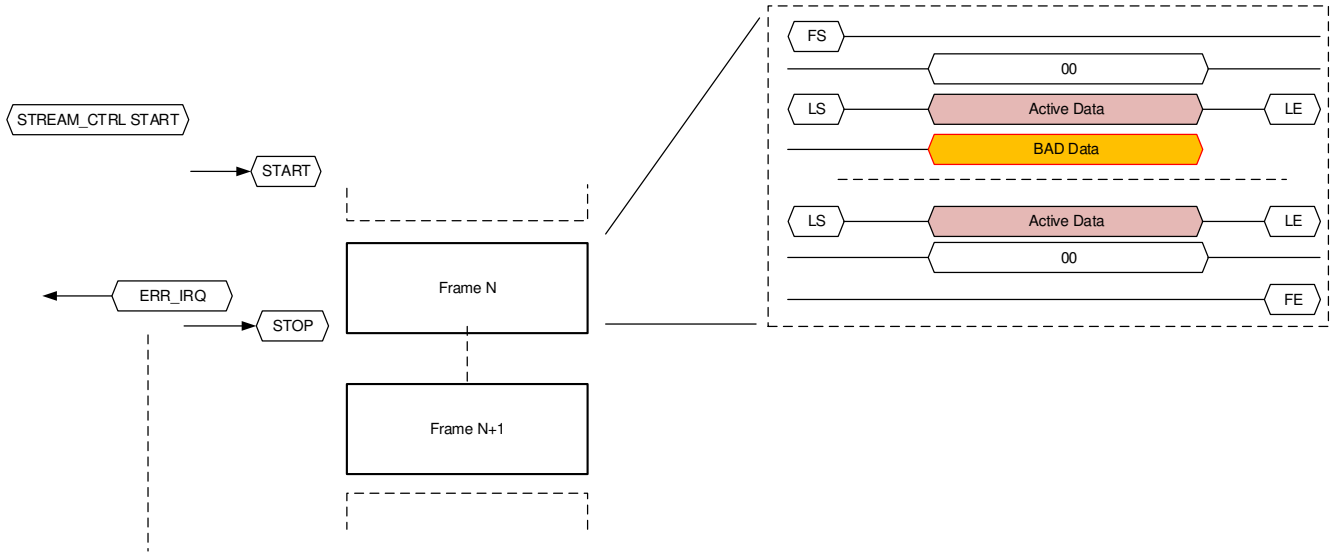


Figure 12-413. Stream Error Detection Causing Stop

12.6.1.4.8.7 Stream Error Detected – Error Bypass Mode

The CSI_RX_IF will detect the error condition and capture the associated VC, DT and WC information for the packet header before generating an error interrupt flag on CSIRX_ERROR_IRQS.

The CSI_RX_IF will continue processing the current frame and continue the stream when a header Reserved DT or a long packet payload CRC error is detected.

The CSI_RX_IF will stop processing the current frame and continue the stream when a packet header ECC error is detected, as the current frame synchronisation cannot be maintained.

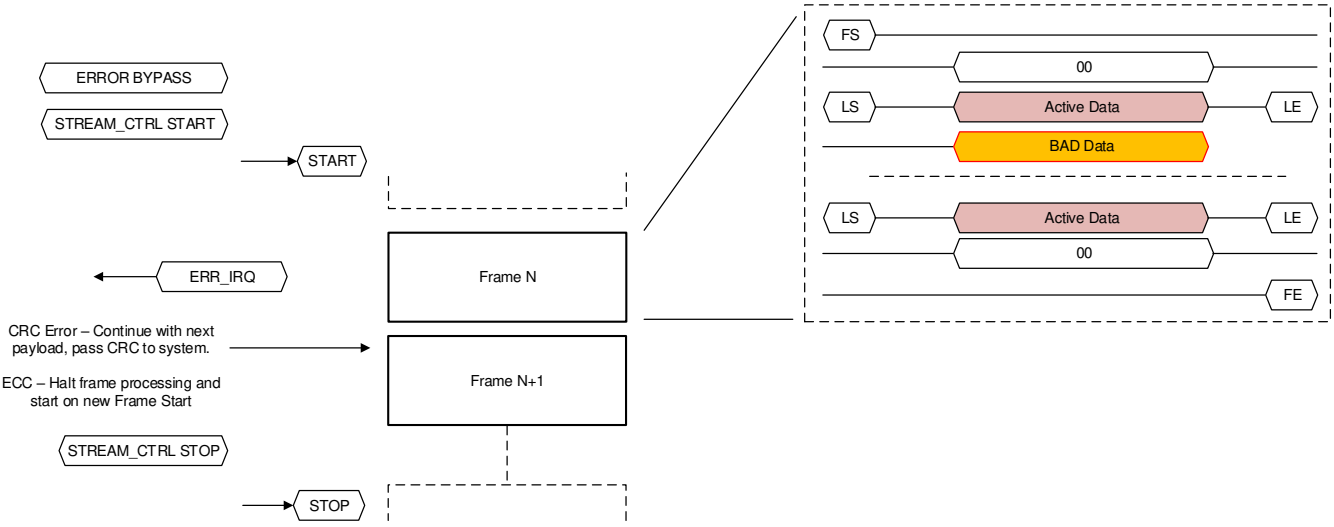


Figure 12-414. Stream Error Bypass - CRC Error During Payload

12.6.1.4.8.8 Stream Error Detected – Soft Reset Recovery

The CSI_RX_IF will stop processing the current frame and continue the stream when a packet header ECC error is detected, as the current frame synchronisation cannot be maintained.

The system can perform a soft reset on the individual stream to clear the pixel interface and payload FIFO and allow the system to restart the stream from the next frame start.

Copyright © 2026 Texas Instruments Incorporated

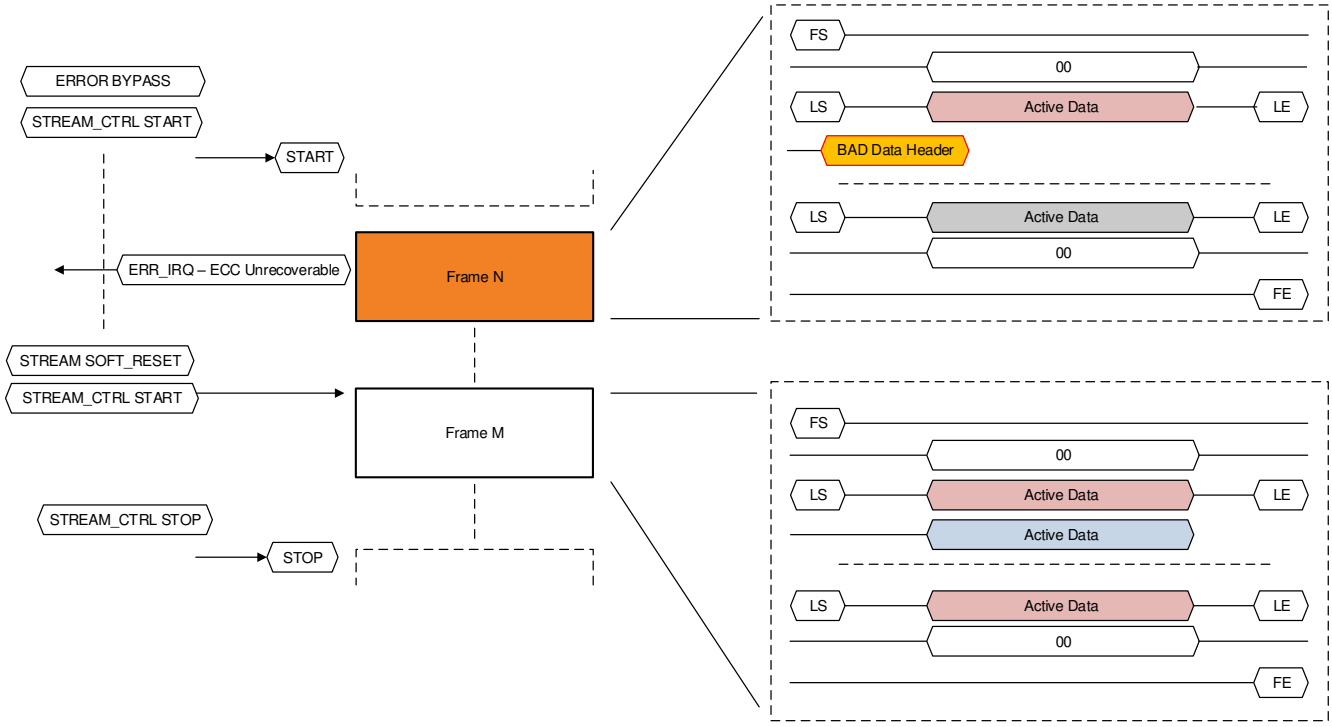


Figure 12-415. Stream Soft Reset After ECC Non-Recoverable Error

12.6.1.4.8.9 Stream Monitor Configuration

The CSI_RX_IF will use the monitor control (CSIRX_STREAM0_MONITOR_CTRL - CSIRX_STREAM3_MONITOR_CTRL) or STOP mechanism to stop the stream processing at the end of the active frame. The stream monitor, configuration and interrupt registers can be redefined and then the stream restarted. The pixel interface will begin processing at the next frame start.

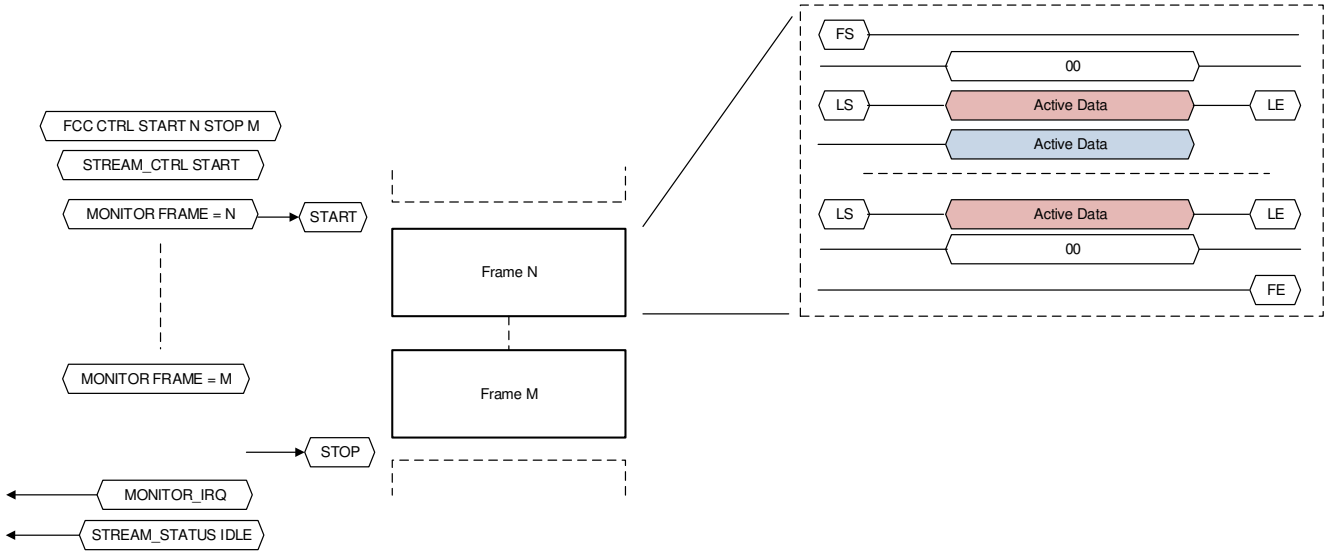


Figure 12-416. Stream Start and Stop Using Monitor Control

The FW will use the FCC config control register (CSIRX_STREAM0_FCC_CFG - CSIRX_STREAM3_FCC_CFG) to perform a start and stop as soon as a frame count is detected. The FCC config control register will define the start and stop frame count and the stream will identify when

Copyright © 2026 Texas Instruments Incorporated

these values are matched. The stream output will begin when start frame is detected and then stop once the stop frame is reached. The virtual channel can be defined in CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL[4-1] FCC_VC, and must match the virtual channels that are available to the stream in the CSIRX_STREAM0_DATA_CFG - CSIRX_STREAM3_DATA_CFG register.

The frame capture is enabled when CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL[0] FCC_EN is set ‘1’

The FW can check the current frame counter value from the CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL[31:16] FRAME_COUNTER

12.6.1.4.8.10 Stream Monitor Frame Capture Control

To use Monitor Frame Capture Control Start and Stop operations:

1. Set Stop and Start values in config register (CSIRX_STREAM0_FCC_CFG - CSIRX_STREAM3_FCC_CFG). Set to ‘0’ for free-running.
   a. [31:16] FRAME_COUNT_STOP -> define desired value
   b. [15:0] FRAME_COUNT_START -> define desired value
2. Set the Virtual Channel to the one that is enabled and the enable in register CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL.
   a. [4:1] FCC_VC -> define VC used
   b. [0] FCC_EN -> set to ‘1’

This will allow visual checking, by setting the Start/Stop as the following examples, this is the behaviour to be observed:

- Stop = 1 / Start = 1: Output (i.e. display) will show a Frame of the Input (i.e. camera)
- Stop = F / Start = 1: Output will show live image from the input during 14 frames and then will freeze in the last one.

Optionally software can read back the frame count value from register (CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL).

In order to do that, software will first need to enable the monitor control register.

1. Define the same Virtual Channel as above and enable the monitor control register (CSIRX_STREAM0_MONITOR_CTRL - CSIRX_STREAM3_MONITOR_CTRL).
   a. [15] FRAME_MON_EN -> set to ‘1’
   b. [14:11] FRAME_MON_VC -> define VC used
2. Read frame_counter containing the current frame number being processed from CSIRX_STREAM0_FCC_CTRL - CSIRX_STREAM3_FCC_CTRL (only when frame numbers are coming from the protocol, not when internal counter)
   a. [31:16] FRAME_COUNTER -> read frame number value

The easiest way to check that the frame number matches the Start and/or Stop values defined, is to use the interrupts in CSIRX_MONITOR_IRQS register. To be able to do that you will need to allow those interrupts through the mask register CSIRX_MONITOR_IRQS_MASK_CFG.

1. By default, the CSIRX_MONITOR_IRQS_MASK_CFG register is set to all ‘0’s, meaning all interrupts are disabled.
   a. [4] STREAM0_FCC_STOP_IRQM -> set to ‘1’
   b. [3] STREAM0_FCC_START_IRQM -> set to ‘1’
2. Create an interrupt handler to read from CSIRX_MONITOR_IRQS.
   a. [4] STREAM0_FCC_STOP_IRQ -> read, if ‘1’, Stop interrupt triggered
   b. [3] STREAM0_FCC_START_IRQ -> read, if ‘1’, Start interrupt triggered

Copyright © 2026 Texas Instruments Incorporated

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI_RX_IF has been configured and enabled if software is using Frame Counter Control features.

Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.

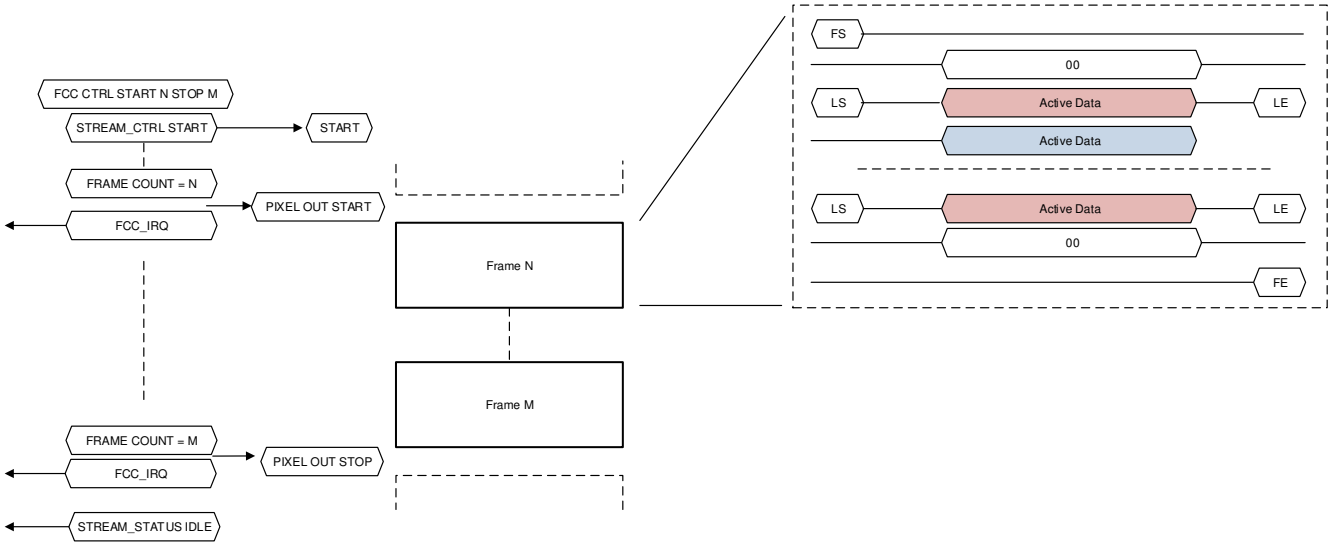


Figure 12-417. Stream Frame Capture Control Flow Diagram

12.6.1.4.8.11 Stream Monitor Timer interrupt

To use the Stream Monitor Timer operations:

1. Set Count value in timer register (CSIRX_STREAM0_TIMER - CSIRX_STREAM3_TIMER). If set to 0 won't count but interrupt will still trigger every EOF or SOF.
   a. [24-0] COUNT -> define desired value
2. Set the Virtual Channel to the one that is enabled, define timer_eof and set enable in the monitor controll register (CSIRX_STREAM0_MONITOR_CTRL - CSIRX_STREAM3_MONITOR_CTRL).
   a. [15] FRAME_MON_EN -> set to '1'
   b. [14:11] FRAME_MON_VC -> define VC used
   c. [10] TIMER_EOF -> set to '1' to count from EOF, set to '0' to count from SOF
   d. [9] TIMER_EN -> set to '1'
   e. [8:5] TIMER_VC -> define VC used

Software should now use the interrupts in register CSIRX_MONITOR_IRQS. To be able to do that you will need to allow those interrupts through the mask register CSIRX_MONITOR_IRQS_MASK_CFG.

1. By default, the CSIRX_MONITOR_IRQS_MASK_CFG register is set to all '0', meaning all interrupts are disabled.
   a. [0] STREAM0_TIMER_IRQM -> set to '1'
2. Create an interrupt handler to read from monitor_irqs.
   a. [0] STREAM0_TIMER_IRQ -> read, if '1', timer interrupt is triggered

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI_RX_IF has been configured and enabled.

Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.

Copyright © 2026 Texas Instruments Incorporated

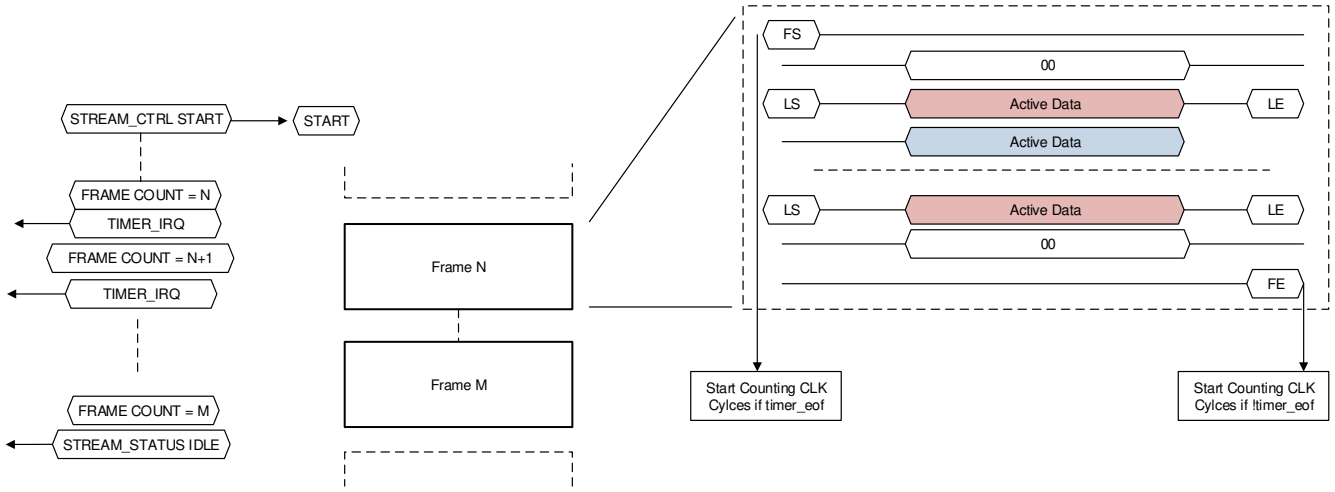


Figure 12-418. Timer Interrupt Flow Diagram

12.6.1.4.8.12 Stream Monitor Line/Byte Counters Interrupt

To use the stream Monitor Line and Byte counters operations:

1. Set LineCount and ByteCount values in register (CSIRX_STREAM0_MONITOR_LB - CSIRX_STREAM3_MONITOR_LB).
   a. [31:16] LINE_COUNT -> define desired value
   b. [15:0] BYTE_COUNT -> define desired value
2. Set the Virtual Channel to the one that is enabled, and set enable in register CSIRX_STREAM0_MONITOR_CTRL.
   a. [15] FRAME_MON_EN -> set to ‘1’
   b. [14:11] FRAME_MON_VC -> define VC used
   c. [4] LB_EN -> set to ‘1’
   d. [3:0] LB_VC -> set to VC used

Software should now use the interrupts in register CSIRX_MONITOR_IRQS. To be able to do that you will need to allow those interrupts through the mask register CSIRX_MONITOR_IRQS_MASK_CFG.

1. By default, the CSIRX_MONITOR_IRQS_MASK_CFG register is set to all ‘0’, meaning all interrupts are disabled.
   a. [7] STREAM0_LINE_CNT_E RROR_IRQM -> set to ‘1’
   b. STREAM0_LB_IRQM [1] -> set to ‘1’
2. Create an interrupt handler to read from CSIRX_MONITOR_IRQS.
   a. [7] STREAM0_LINE_CNT_E RROR_IRQ -> read, if ‘1’, line count error interrupt is triggered
   b. [1] STREAM0_LB_IRQ -> read, if ‘1’, line/byte counter interrupt is triggered

Note that the interrupt will trigger on each frame when it reaches byte number BYTE_COUNT in line number LINE_COUNT.

To ensure correct functionality, it is highly recommended that the input device (i.e. camera) should be configured after the CSI_RX_IF has been configured and enabled.

Software must make sure all the Virtual Channel fields in the registers explained above are set to the correct VC being used by the stream.

Copyright © 2026 Texas Instruments Incorporated

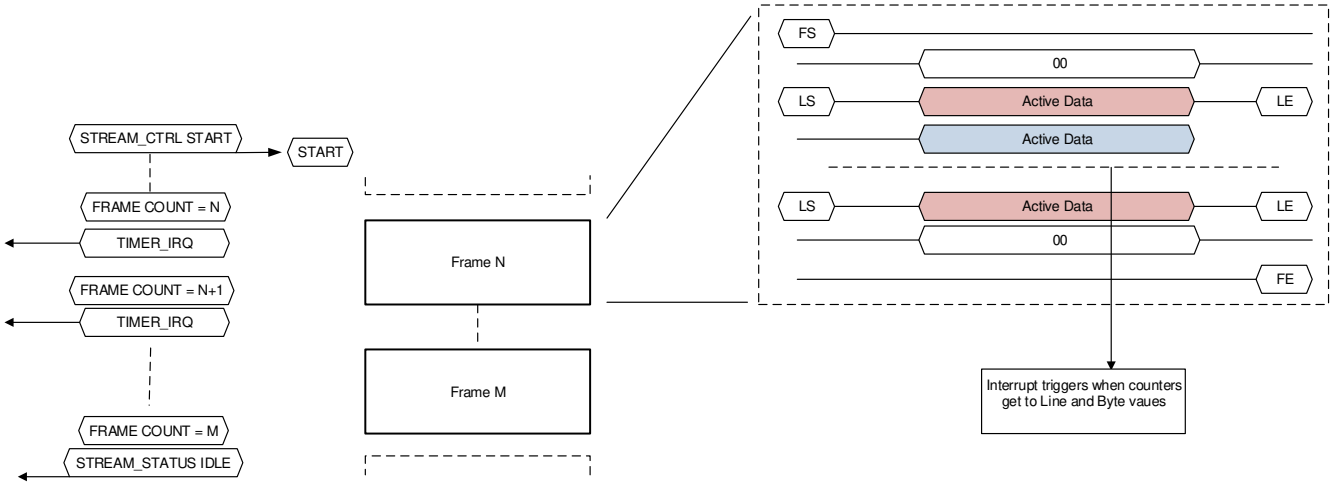


Figure 12-419. Line/Byte Counters Interrupt Flow Diagram

12.6.1.4.8.13 Example Controller Programming Sequence (Single Stream Operation)

The single stream operation will provide the smallest multi-lane IP configuration with the reduced registers configuration removing some control and status registers. The stream will not require a large FIFO configuration and the clock rates will be matched to simplify the implementation to single pixel transfers using all the available interface bits, (i.e. up to 32).

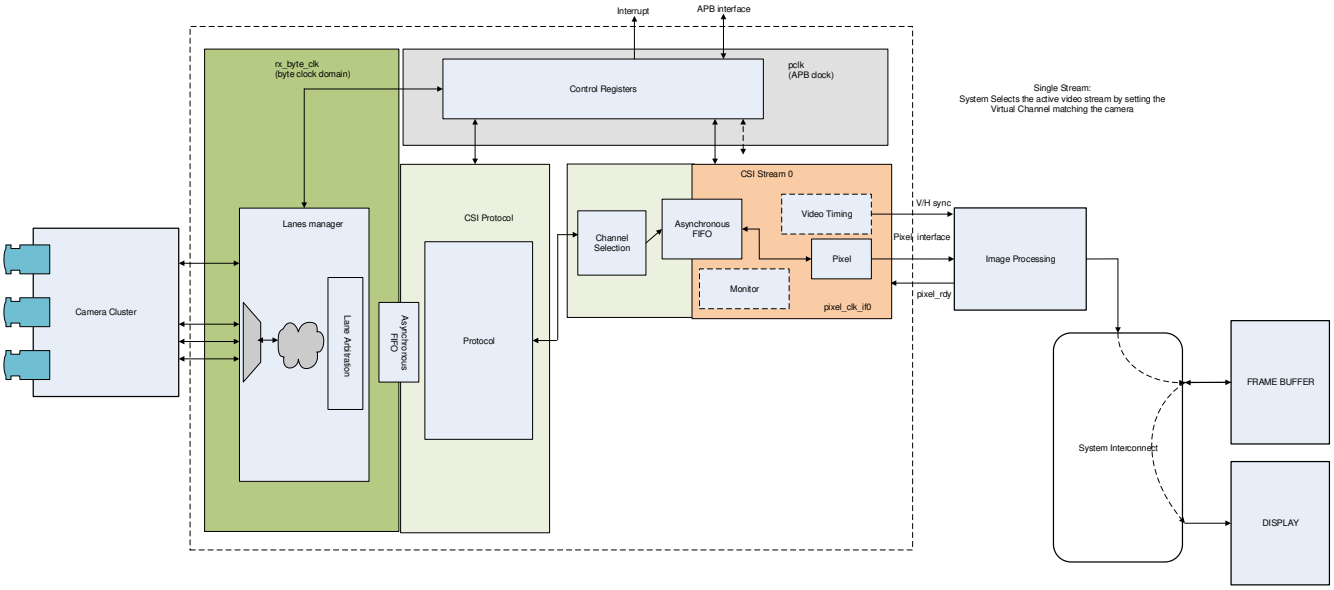


Figure 12-420. Basic Single Stream Use Case Illustration

For single pixel stream configuration, elastic buffer, RAW8 Data type on VC2, one pixel per cycle:

1. Configure the number of DPHY_RX lanes:
   a. See CSIRX_STATIC_CFG register
2. Set Error Interrupt mask:
   a. See CSIRX_ERROR_IRQS_MASK_CFG register
3. Set the Pixel Interface and FIFO configuration. See CSIRX_STREAM0_CFG - CSIRX_STREAM3_CFG.
   a. Set [9-8] FIFO_MODE to select the elastic buffer configuration -> ‘1’

Copyright © 2026 Texas Instruments Incorporated

b. Set the FIFO fill level that will determines the FIFO depth that must be reached before data is output -> 0x0000
c. Set the number of pixels to be output on each cycle -> ‘0’.
d. Set the type of stream interface to “pixel” mode -> ‘0’

4. Select the virtual channel and data types to be processed. See CSIRX_STREAM0_DATA_CFG - CSIRX_STREAM3_DATA_CFG register.
   a. Set [31-16] VC_SELECT and virtual channel bits if not supporting all virtual channels -> 1h, 2h
   b. Set [7] ENABLE_DT0 and [5-0] DATATYPE_SELECT0 values for one or both data types (default all DT) -> 0h, 1h, 2Ah
5. Enable the stream to begin processing the incoming data from the DPHY_RX. See CSIRX_STREAM0_CTRL - CSIRX_STREAM3_CTRL.
   a. Set [0] START bit -> ‘1’

The stream will begin to pass pixel data matching the configuration on the next frame start.

12.6.1.4.8.14 CSI_RX_IF Programming Restrictions

The following CSI_RX_IF programming restrictions apply:

- Full line buffer mode is not supported.
- LS/LE detection has a restriction to be disabled
- Only use Pixel transmit mode can be used, never packet mode
  - In pixel mode only single, dual, or quad mode can be used. It is not allowed to mix these based on data format or virtual channels.
- LSB alignment only
- Video port(VP) stream interfaces must be programmed in the CSI_RX_IF to meet the following restrictions:
  - Raw 8-16 formats only
  - Dual pixel mode
  - Filtering for single virtual channel and single data type

Note

CSI_RX_IF has a limitation that any line must fully be exported on its stream interface before a new line or even end of frame is sent on the D-PHY interface. To meet this requirement, software will need to take into account export rates on stream interface(single, dual, quad modes) at 500MHz versus D-PHY 32-bits @ byte clock frequency to know when a new line or end of frame can be sent in after last line was sent over D-PHY interface.

Table 12-337 shows the CSI_RX_IF programming requirements for pixel modes and data sizes.

Table 12-337. CSI_RX_IF Programming requirements for pixel modes and data sizes

Format	Pixel transmit mode used	Size	Details
RAW(6-8), user defined 8-bit	Single	0	
RAW(6-8), user defined 8-bit	Dual	1	
RAW(6-8), user defined 8-bit	Quad	2	
RAW(10-16), user defined 16-bit	Single	1	
RAW(10-16), user defined 16-bit	Dual	2	
RAW20	Single	2	
YUV422-8	Single	0	must set dual mode=0
YUV422-8	Dual	0	must set dual mode=1
YUV422-10	Single	1	
YUV420-8	Single	0	must set dual mode=0
YUV420-8	Dual	0	must set dual mode=1

Copyright © 2026 Texas Instruments Incorporated

Table 12-337. CSI_RX_IF Programming requirements for pixel modes and data sizes (continued)

Format	Pixel transmit mode used	Size	Details
YUV420-10	Single	1	

12.6.1.4.8.15 CSI_RX_IF Real-time operating requirements

Care must be taken on blanking requirements for next line, end of frame, or start of frame. The previous packets(long or short) must be fully consumed before any new line, end of frame, or start frame can be sent over the CSI interface. Software will get various error conditions if these are not met. Below are some sample equations to determine this blanking time. Note that this is not blanking data. High level details are that the internal FIFO must be completely empty prior to sending any thing new into it. below are some sample equations to determine how much time is needed to empty the FIFO and blanking time needed. Byte clock rates, pixel data type, and export modes play into the calculation as well.

$$\text{Byte_clock_rate} \times 8 \times \#\text{lanes} \times \text{Tcsi} = \text{pixel_clock_rate} \times \text{BPP} \times \text{Tpix} = \text{total bits} \quad (39)$$

BPP = Bits Per Pixel. This number is how many bits are sent per pixel clock cycle is based on single, dual, quad modes and data type.

Tcsi = Time for csi interface to complete 1 line
Tpix = Time for pixel interface to complete 1 line
Tpix16 = Tpix+16/pixel_clock_rate

(Tpix16-Tcsi) = required blanking time on CSI interface
Tpix16-Tcsi <= 0 implies 0

Copyright © 2026 Texas Instruments Incorporated

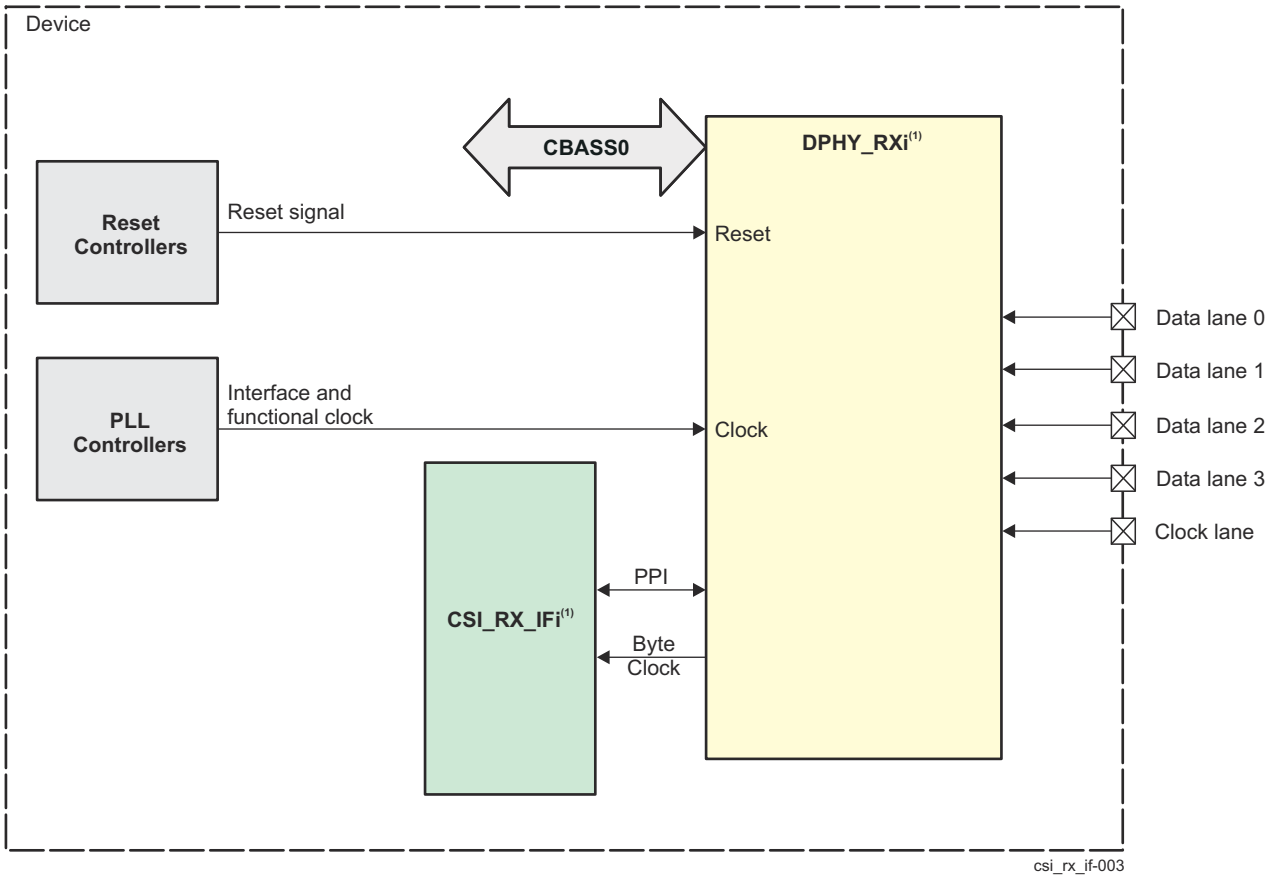
12.6.2 MIPI D-PHY Receiver (DPHY_RX)

The following sections describe the MIPI D-PHY receiver (DPHY_RX) module(s) in the device.

12.6.2.1 DPHY_RX Overview

The integration of the DPHY_RX camera physical port module allows the device to grab video streams from external sensor cameras and other CSI2 compliant sources.

Figure 12-421 shows the DPHY_RX module overview.



A. i represents a DPHY_RX instance. See the device datasheet for available domains and DPHY_RX instances.

Figure 12-421. DPHY_RX Module Overview

12.6.2.1.1 DPHY_RX Features

The DPHY_RX module supports the following features:

- Compliant to MIPI D-PHY standard v1.2
- Supports up to 4 data and 1 clock lanes
- Supports up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane
- Clock lane control / interface logic type is **CIL-SCNN** for HS and low power receiving:
  1. **(S)** Peripheral
  2. **C**lock
  3. **N**/A forward, **N**/A reverse escape mode features
- Data lane control / interface logic type is **CIL-SFAN** for HS and low power receiving:
  1. **(S)** Peripheral
  2. **F**orward direction only for high speed mode
  3. **A**ll forward direction escape mode features are supported

Copyright © 2026 Texas Instruments Incorporated

4. **N**o reverse direction escape mode features are supported

- Data lanes can be independently operated in HS or ULP mode
- Swapping of DP/DN signals within each clock/data pair (Facilitated by CSI_RX_IF controller)

12.6.2.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

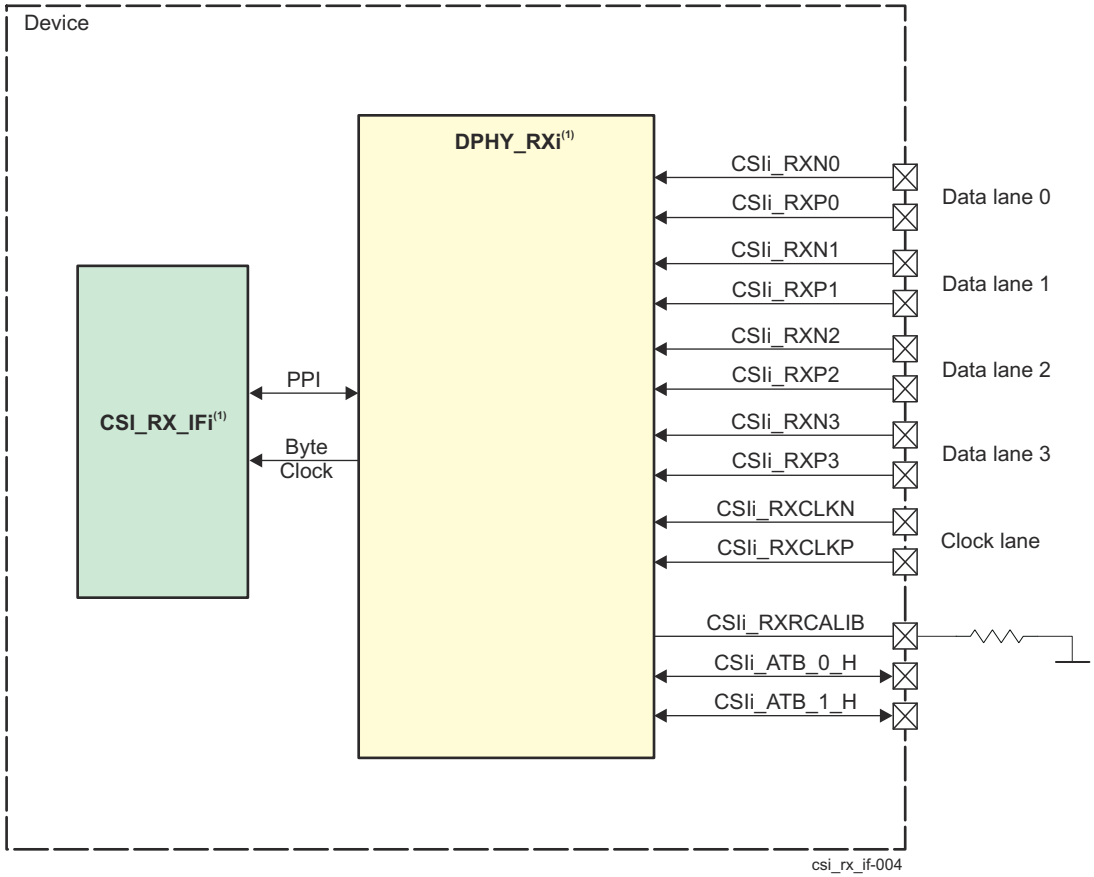
Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.6.2.2 DPHY_RX Environment

This section describes the DPHY_RX application fields from an environment point of view (external connections).



A. i represents a DPHY_RX instance. See the device datasheet for available domains and DPHY_RX instances.

Figure 12-422. DPHY_RX Enviroment

Table 12-338 describes the external signals of the DPHY_RX.

Copyright © 2026 Texas Instruments Incorporated

Table 12-338. DPHY_RX I/O Signals

Device Level Signal	I/O	Description
		DPHY_RXi[1]
CSIi[1]_RXN0	I	Lane 0 Receive Differential Data (Negative)
CSIi[1]_RXP0	I	Lane 0 Receive Differential Data (Positive)
CSIi[1]_RXN1	I	Lane 1 Receive Differential Data (Negative)
CSIi[1]_RXP1	I	Lane 1 Receive Differential Data (Positive)
CSIi[1]_RXN2	I	Lane 2 Receive Differential Data (Negative)
CSIi[1]_RXP2	I	Lane 2 Receive Differential Data (Positive)
CSIi[1]_RXN3	I	Lane 3 Receive Differential Data (Negative)
CSIi[1]_RXP3	I	Lane 3 Receive Differential Data (Positive)
CSIi[1]_RXCLKN	I	Lane 3 Receive Differential Clock (Negative)
CSIi[1]_RXCLKP	I	Lane 3 Receive Differential Clock (Positive)
CSIi[1]_RXRCALIB	A	Pin for external calibration resistor. An external resistor must be connected between this pin and package ground. Refer to the device-specific Datasheet for a recommended resistor value.

(1) i represents a DPHY_RX instance. See the device datasheet for available domains and DPHY_RX instances

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.6.2.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.6.2.4 DPHY_RX Functional Description

12.6.2.4.1 DPHY_RX Programming Guide

12.6.2.4.1.1 Overview

This section details specific steps on how to program the DPHY_RX

12.6.2.4.1.2 Initial Configuration Programming

The DPHY_RX operation shows the registers that are written during the startup sequence.

This section provides description and timing diagrams for DPHY_RX operation. Unless otherwise described in this subsection, the DPHY_RX PPI interface is compliant with the timing diagrams described in the MIPI D-PHY v1.2 specification.

12.6.2.4.1.2.1 Start-up Sequence Timing Diagram

The common configuration pins noted in Figure 12-423 are ipconfig_cmn, pll_ipdiv, pll_fbdiv, pll_opdiv, psm_clock_freq_cmn. Drive these pins as per the pin description given in PHY pin list.

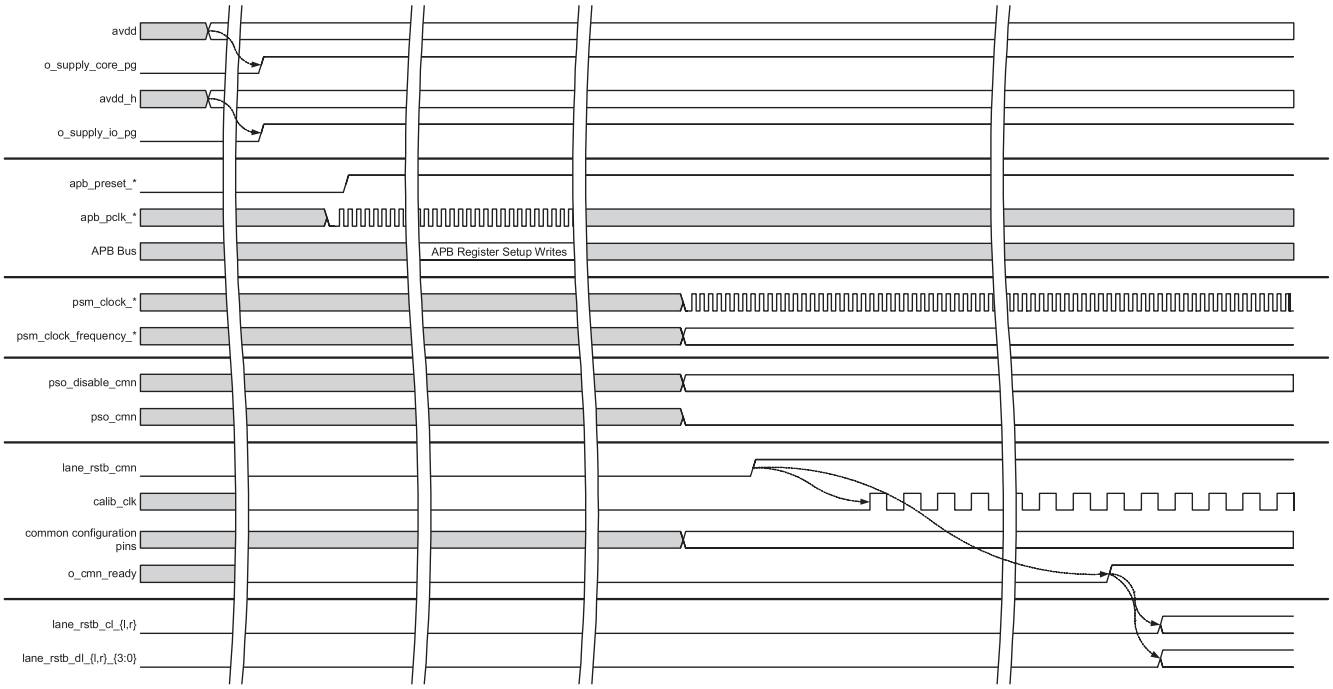


Figure 12-423. Common Power Up and Initialization Timing Diagram

Copyright © 2026 Texas Instruments Incorporated

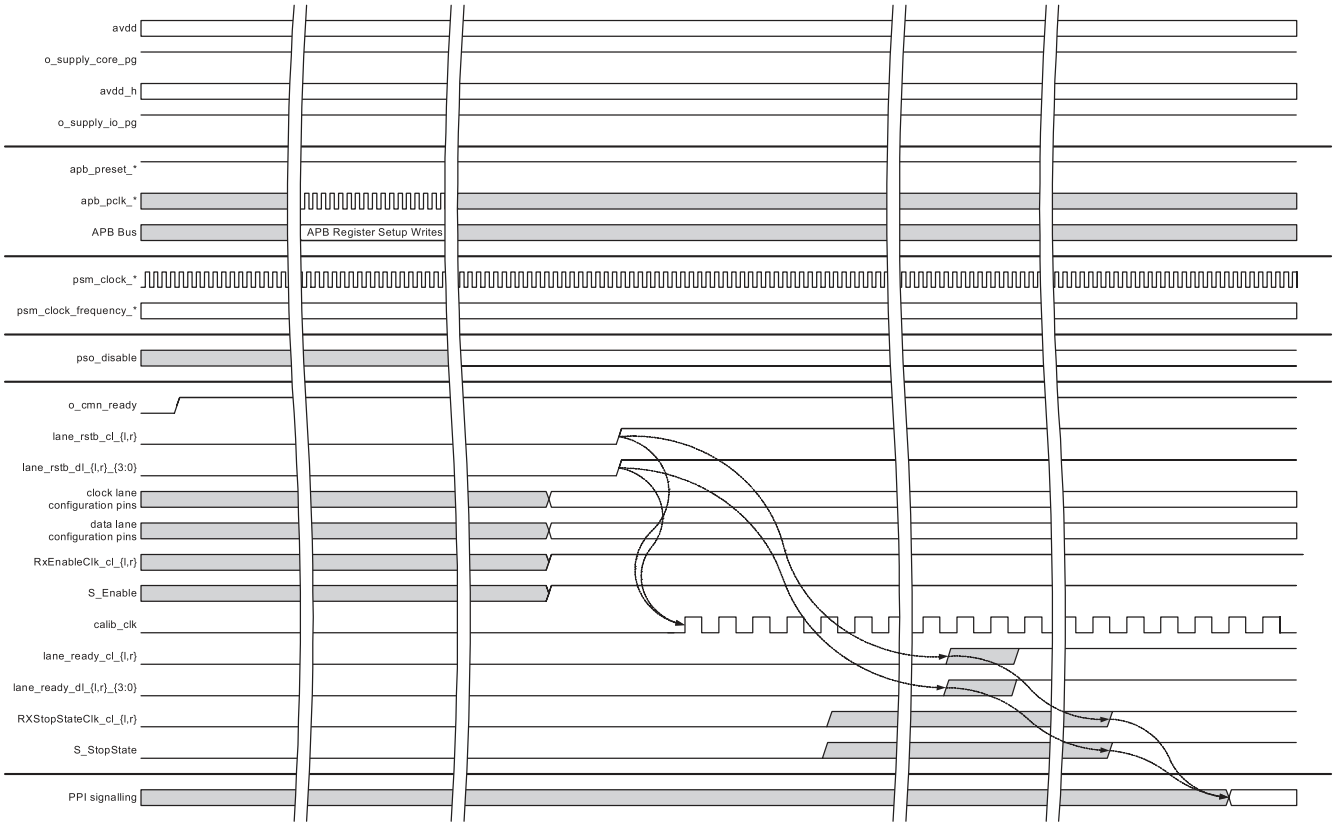


Figure 12-424. Lane Power Up and Initialization Timing Diagram

Copyright © 2026 Texas Instruments Incorporated

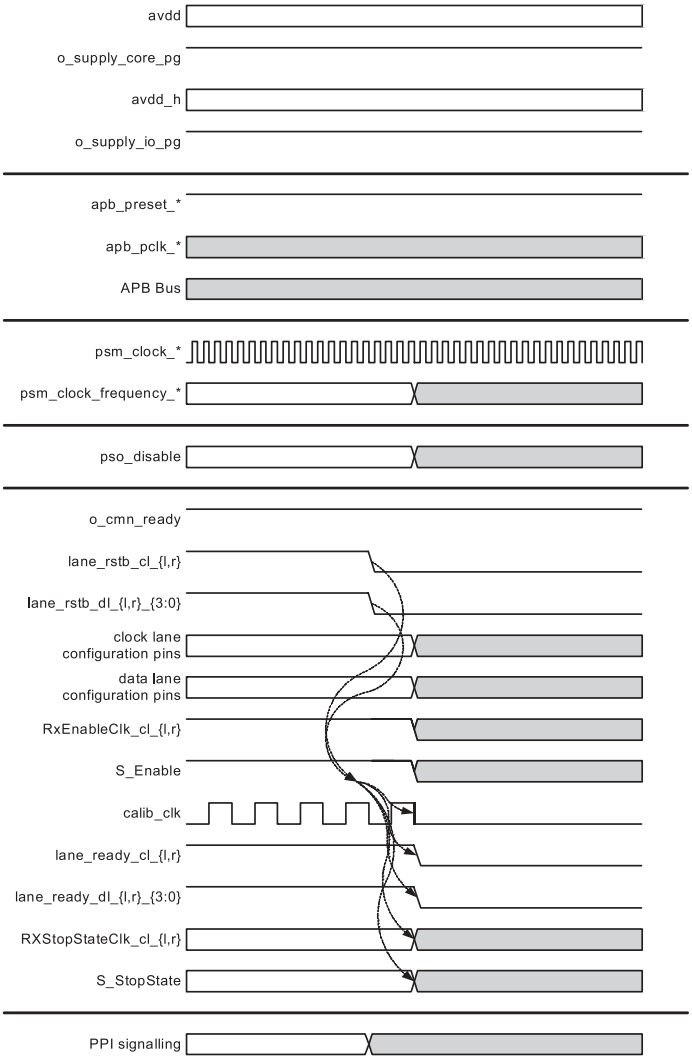


Figure 12-425. PHY Disable Timing Diagram

For initial set up, the DPHY_RX must be configured/set (registers and configuration input pins) for the common module prior to releasing it from reset, and the lane modules prior to releasing them from reset. Registers shall be configured (as required) between releasing the APB from reset and releasing the common / lanes from reset.

Note that in some cases, the option exists to configure a function using either a pin or a register. In such cases, both options will be specified and the you can select the preferred option.

12.6.2.4.1.3 Common Configuration

Table 12-339. Common Configuration-Related Setup

Configuration Register / Pin	Configuration Requirement
PHY Pin: psm_clock_freq PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT1	Set the PMA state machine clock frequency divider. Set either the pin or the register, as specified in the respective description, for the required PMA state machine clock.
PHY Pin: ipconfig_cmn PHY Register: DPHY_RX_MMR_SLV_LANE[11-9] IPCONFIG_CMN	Set the clock lane configuration. Set as specified in the description for the required clock lane configuration.

Copyright © 2026 Texas Instruments Incorporated

Table 12-339. Common Configuration-Related Setup (continued)

Configuration Register / Pin	Configuration Requirement
PMA Register: DPHY_RX_VBUS2APB_CMN0_CMN_DIG_TBIT2[0] O_CMN_SSM_EN, DPHY_RX_VBUS2APB_CMN0_CMN_DIG_TBIT2[10] O_CMN_RX_MODE_EN	Enable the startup state machines for TX mode of operation. Set both register bits to 1'b1.
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT2	Set the required power island phase 2 time. Set the register to 32'hAAAAAAAA.
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT3[7:4] POWER_SW_2_TIME_CL_R, DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT3[3:0] POWER_SW_2_TIME_CL_L	Set the required power island phase 2 time. Set the register to 8'hAA

12.6.2.4.1.4 Lane Configuration

Table 12-340. Lane Configuration-Related Setup

Configuration Register / Pin	Configuration Requirement
PHY Register: DPHY_RX_VBUS2APB_PCS_TX_DIG_TBIT0	Set the lane band control value. Set the register fields, as specified in the register description for the required data rate.

12.6.3 MIPI D-PHY Transmitter (DPHY_TX)

This section describes the features and functions of the shared MIPI D-PHY Transmitter (DPHY_TX).

12.6.3.1 DPHY_TX Subsystem Overview

The DPHY_TX module provides one option for video output interfacing by implementing a four-lane MIPI D-PHY Transmitter.

The device includes one or more instances of DPHY_TX module. See Module Integration for specifics.

12.6.3.1.1 DPHY_TX Features

The DPHY_TX0 module supports the following main features:

- Compliance to MIPI D-PHY standard version 1.2.
- Supports up to 4 data lanes and 1 clock lane.
- Supports High Speed up to 2.5 Gbps (with deskew) and 1.5 Gbps (without deskew) per data lane.
- Supports Escape mode:
  - Remote triggers
  - LP-DT up to 10 Mbps
  - ULPS mode
- Clock Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-MCNN**: [HS-TX, LP-TX]
  1. **(M)** Controller
  2. **C**lock
  3. **N**/A forward, **N**/A reverse escape mode features.
- Data Lane **C**ontrol / **I**nterface **L**ogic type is **CIL-MFAA**: [HS-TX, LP-TX, LP-RX, LP-CD]
  1. **(M)** Controller
  2. **F**orward direction only for High Speed mode
  3. **A**ll forward direction escape mode features are supported
  4. **A**ll reverse direction escape mode features are supported.
- Data Lanes can be independently operated in HS or ULP mode.
- Includes a CMN block with reference generators / resistor calibration and an integrated PLL.
- Fault detection:

Copyright © 2026 Texas Instruments Incorporated

- Contention detection (for example, when RX and TX sides of same link drive opposite LP levels).
- Sequence error detection (corruption on lanes).

12.6.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.6.3.2 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

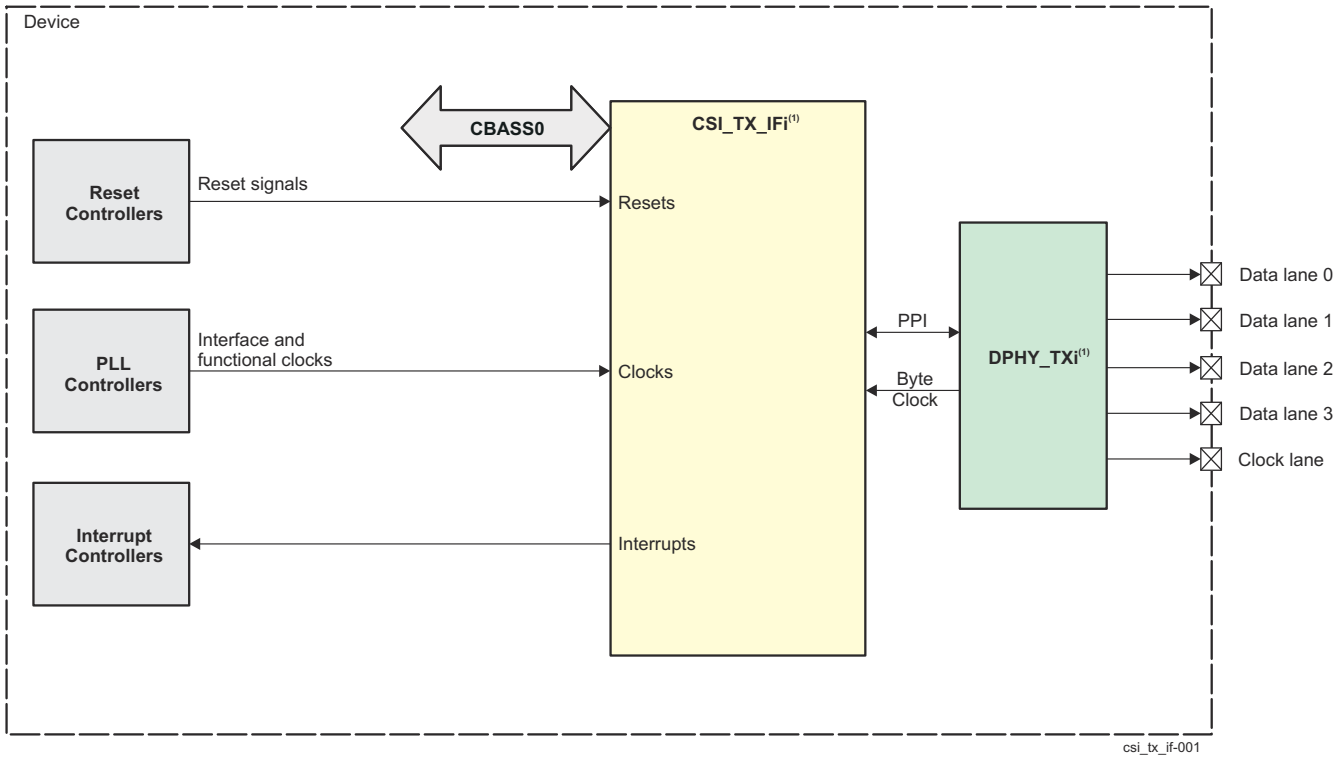
12.6.4 Camera Streaming Interface Transmitter (CSI_TX_IF)

The following sections describe the camera streaming transmitter interface (CSI_TX_IF) module in the device.

12.6.4.1 CSI_TX_IF Overview

The integration of the CSI_TX_IF module allows the device to stream out video data from memory, or retransmit from the CSI receivers as an optional loopback output for diagnostics, debug, and test purposes.

Figure 12-426 shows the CSI_TX_IF module overview.



A. i = 0 to 1

Figure 12-426. CSI_TX_IF Module Overview

12.6.4.1.1 CSI_TX_IF Ports

This section describes the CSI_TX_IF ports related to clocks, resets, and hardware requests.

Table 12-341. CSI_TX_IF Clocks and Resets

Clocks	
Module Clock Input	**Description**
CSI_TX_MAIN_CLK	Main functional(sometimes referred to as pixel clock) clock.
CSI_TX_VBUS_CLK	The VBUS clock runs at always half the speed of the CSI_TX_MAIN_CLK.
CSI_TX_ESC_CLK	20 MHz max clock input for low speed data transmission and some control signals.
DPHY_TXBYTECLKHS	The byte clock is the clock supplied by the DPHY_TX.
Resets	
Module Reset Input	**Description**
CSI_TX_RST	Asynchronous module global reset, driving all collateral asynchronous resets of the 4 clock domains to the low state.

Copyright © 2026 Texas Instruments Incorporated

Table 12-342. CSI_TX_IF Hardware Requests

Interrupt Requests		
Module Interrupt Signal	**Description**	**Type**
CSI_TX_IF_CSI_INTERRUPT_0	Global interrupt that various re-synchronized sources converge into interrupt generation.	Level
CSI_TX_IF_CSI_LEVEL_0	Error interrupt that is generated under the following conditions: • Retransmit error on Stream2 (packet error on the stream interface) • Retransmit error on Stream3 (packet error on the stream interface)	Level
CSI_TX_IF_CSI_FATAL_0	ASF port fatal interrupt. Level sensitive.	Level
CSI_TX_IF_CSI_NONFATAL_0	ASF port non-fatal interrupt. Level sensitive.	Level
CSI_TX_IF_CDNS_RAM_CORR_LEVEL_0	Interrupt on internal FIFO RAM	Level
CSI_TX_IF_CDNS_RAM_UNCORR_LEVEL_0	Interrupt on internal FIFO RAM	Level
CSI_TX_IF_CORR_LEVEL_0	Interrupt on internal FIFO RAM	Level
CSI_TX_IF_UNCORR_LEVEL_0	Interrupt on internal FIFO RAM	Level

12.6.4.2 CSI_TX_IF Features

The CSI_TX_IF module supports the following features:

- Compliant to MIPI CSI v1.3, MIPI CSI v2.1, and MIPI D-PHY v2.1
- Data rate up to 2.5 Gbps per lane (wire rate)
- Supports 1, 2, 3, or 4 Data Lane connection to DPHY_TX
- Programmable formats including YUV422, RGB, Raw, and User Defined (over 25 different formats supported); Limitations apply depending on the stream used
- 16 virtual channel support
- Configurable input streams supported:
  - **Stream0**: DMA interface through a 128-bit PSI_L connection for transfers from memory **OR** CSI_RX retransmit:
    - 128bit wide pixel data with bursting
    - ByteValid per byte in Last Data Phase (LDP)
    - 32 thread IDs supported (virtual channel & data type combinations); Flexible number of threads (32 Max).
    - Unpacking PSI_L data and converting to CSI video format
    - Internal FF based FIFO and external RAM based buffer
  - **Stream1**: Color bar video data generator
    - 2 pixel wide
    - YUV422 8-bit format support only
    - Configurable frame/line size via registers
    - 1 programmable virtual channel
    - Internal FF based FIFO; No external buffer.
- Functional and data path error interrupts
- ECC support on external RAMs

12.6.4.2.1 CSI_TX_IF Legacy Compatibility

The following lists changes that must be considered compared to previous generation CSI_TX_IF:

- YUV420 is not supported, YUV420 configuration registers/bitfields are not used
- Word count must be programmed in the CSI_TX_IF, not in the TX Core.
- CSI_TX_IF compatibility mode must be used in Stream0 when retransmitting from CSI_RX_IF

Copyright © 2026 Texas Instruments Incorporated

12.6.4.3 CSI_TX_IF Environment

The CSI_TX_IF has no dedicated pins. At the device level, the CSI_TX_IF video output goes through the shared (with DSS_DSI) DPHY_TX. See chapter *Shared MIPI D-PHY Transmitter (DPHY_TX)*.

12.6.4.4 CSI_TX_IF Functional Description

12.6.4.4.1 CSI_TX_IF Block Diagram

Figure 12-427 depicts a simplified internal block diagram of the CSI_TX_IF and its surroundings.

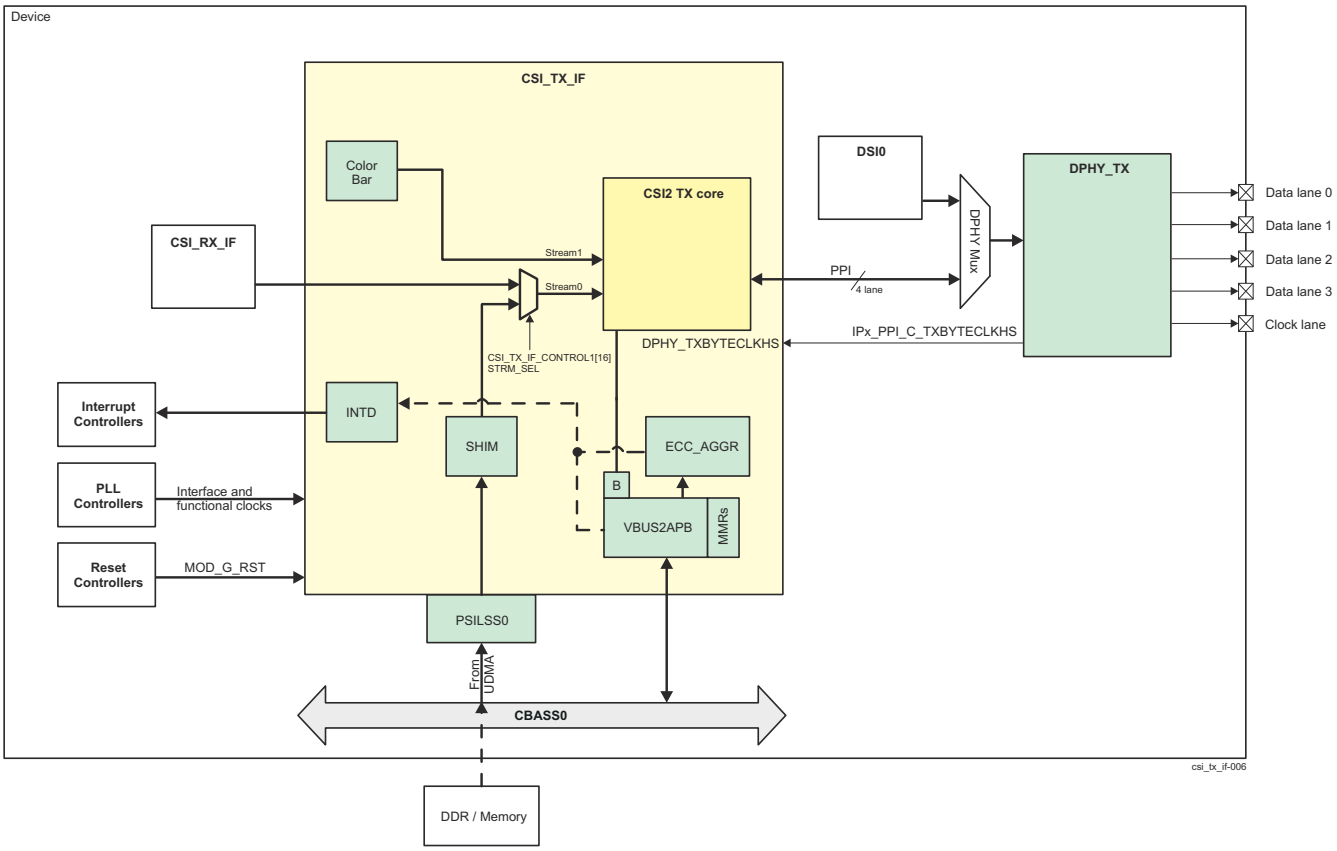


Figure 12-427. CSI_TX_IF Block Diagram

The CSI2 core streams 4 channels of data to the several export paths.

Stream0 (from PSI_L) goes through the PSI_L that converts 128(4x32) bits pixel data and converts it to single 32-bit CSI streaming protocol of pixel data. Each 128 bit set of PSI_L data is split into N M-bit data transfers based on the SIZE_CFG bitfield setting in the CSI_TX_IF_DMACNTX_j register for a given thread. See memory organization details in *CSI_TX_IF Data Memory Organization Details*. Via register mapping in the CSI_TX_IF_DMACNTX_j register the PSI_L thread ID is mapped to a virtual channel and a data type. The CSI_TX_IF_DMACNTX_j register is the thread filter. There are 32 dma context registers. Figure 12-428 shows the stream to PSI_L thread mapping.

Copyright © 2026 Texas Instruments Incorporated

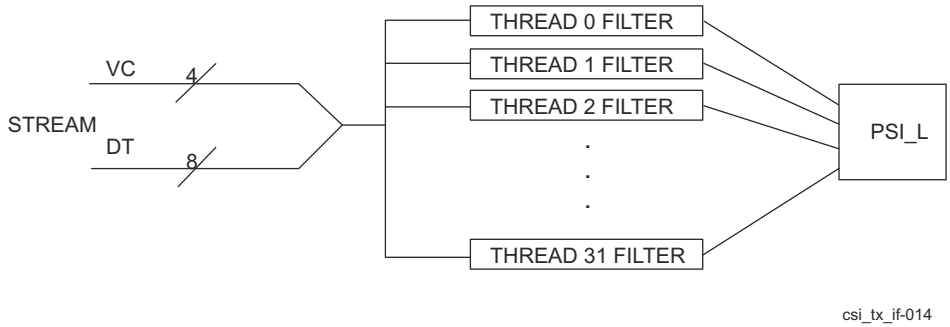


Figure 12-428. CSI_TX_IF PSI_L thread mapping

Software must take the following notes in consideration

- Interleaved data formats on single virtual channel and frame are supported.
- All PSI_L threads that open a frame must be closed before a new PSI_L thread can start a new frame. Until all channels close the frame other threads will be stalled.
- DATA is assumed in a specific organization for CSI_TX_IF to export data correctly. See memory organization details in *CSI_TX_IF Data Memory Organization Details*.
- Threads can only switch after full lines are received. PSI_L pushback will occur on all other threads.
- Thread priority is 0 - 31 and after a line is received will be re-arbitrated in that order for the next thread to send.

Registers CSI_TX_IF_L2L_DELAY_j control the line start to next line start. Also register CSI_TX_IF_L2L_DELAY_j control the last line of frame to next start of frame first line start of next frame. Buffering of PSI_L data is 2k×128 RAM.

Stream0 (from CSI_RX_IF) is a loopback stream from the CSI receiver muxed with PSI_L data stream.

Stream1 streams data from the Color Bar generator. It supports YUV422 8bit format only. CSI_TX_IF_COLOR_PARAM register is used to control height and width information. Width is divided by 8 for each color section. Colors are defined below. Transfers are done on the Stream1 interface in 32-bit packed 8-bit pixel format(required programming in CSI_TX_IF core). Virtual channel and data type is configurable in CSI_TX_IF_COLOR_CNTL register. Data selection inside CSI_TX_IF controller maps data type information. CSI_TX_IF_COLOR_START_DELAY register can configure delays from enabled to start. CSI_TX_IF_COLOR_LINE_DELAY, and CSI_TX_IF_COLOR_FRAME_DELAY register can delay last line to first line of new frame. Once enabled, it will continuously send out frames until disabled. Once disabled, it will stop at end of current frame. Table 12-343 shows the Color Bar format.

Table 12-343. CSI_TX_IF Color Bar format details

Color	Y	CB	CR
White	235	128	128
Yellow	162	44	142
Cyan	131	156	44
Green	112	72	58
Magenta	84	184	198
Red	65	100	212
Blue	35	212	114
Black	16	128	128

Copyright © 2026 Texas Instruments Incorporated



Figure 12-429. CSI_TX_IF PSI_L Color Bar sample output

Note

It is a requirement that only PSI_L/DMA, color bar, or CSI_RX_IF loopback streams are not active at the same point in time. The intended use model is not to have multiple combinations active at any given time. The main issue with this is that RAMs are not sized for concurrent operation and will likely overflow resulting in data loss.

12.6.4.4.2 CSI_TX_IF Hardware and Software Reset

An active low asynchronous hardware reset is provided to CSI_TX_IF by device LPSC. It is internally re-synchronized to the functional clock domain.

A software reset is triggered by configuring the CSI_TX_IF_TX_CONF[1] SOFT_RESET_REQUEST bit-field for protocol reset and/or module reset.

12.6.4.4.3 CSI_TX_IF Clock Configuration

There are four clock domain in the CSI_TX_IF.

1. The CSI_TX_MAIN_CLK(or also referred to as pixel clock) runs most of the logic. It needs to be same frequency as CSI_RX_MAIN_CLK main clock (500MHz). When CSI_TX_MAIN_CLK is operating lower than 312.5MHz, then the clock is essentially limiting the clock rate of the DPHY_TX. Said another way, CSI_TX_MAIN_CLK must be at least the DPHY_TXBYTECLKHS rate else FIFOs will overflow, crashing the module.
2. The CSI_TX_VBUS_CLK is the interface configuration clock that runs at half the speed of the CSI_TX_MAIN_CLK (250MHz).
3. The DPHY_TXBYTECLKHS is the clock supplied by the DPHY_TX PLL and is divided down to byte clock. The DPHY_TX is designed for max of 10gbps. This translates to a max byte clock of 312.5MHz. The clock is inactive when DPHY_TX is not in HS operation.
4. The CSI_TX_ESC_CLK escape clock runs at 20MHz. CSI_TX_ESC_CLK is rarely used by the CSI_TX_IF, usually is only to clock in some low speed control signals from DPHY_TX. The ESC interface is a low speed DPHY common link to the camera/sensor.

Table 12-344 shows the CSI_TX_IF and DPHY_TX inter-clock dependencies.

Table 12-344. CSI_TX_IF Inter-clock Dependencies

	CSI_TX_MAIN_CLK	CSI_TX_VBUS_CLK	DPHY_TXBYTECLKHS	CSI_TX_ESC_CLK
Min freq	DPHY_TXBYTECLKHS freq	CSI_TX_MAIN_CLK / 2 freq	N/A	N/A
Max freq	500MHz	CSI_TX_MAIN_CLK / 2 freq	312.5MHz	20MHz

Copyright © 2026 Texas Instruments Incorporated

12.6.4.4.4 CSI_TX_IF Interrupt Events

12.6.4.4.4.1 CSI_TX_IF Interrupt Events

This section describes the register configuration of the interrupt events that can trigger the several CSI_TX_IF interrupt signals. For detailed description and mapping of the interrupt to the device interrupt processors see *CSI_TX_IF Integration*.

The interrupts are generally handled within the INTD module of the CSI_TX_IF, although there are several interrupt registers in the ECC_AGGR for ECC errors and in the VBUS2APB for stream monitoring errors/flags.

Table 12-345 lists the event generation and corresponding registers of the CSI_TX_IF controller.

Table 12-345. CSI_TX_IF Interrupt Events Cross Table

Event	Mask Register	Status Register	Description
CSI_TX_IF_MAIN_0_CSI_INTERRUPT	CSI_TX_IF_IRQ_MASK CSI_TX_IF_DPHY_IRQ_MASK	CSI_TX_IF_IRQ CSI_TX_IF_DPHY_IRQ_MASK	Global interrupt that various re-synchronized sources converge into interrupt generation. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CSI_LEVEL	CSI_TX_IF_IRQ_MASK CSI_TX_IF_DPHY_IRQ_MASK	CSI_TX_IF_IRQ CSI_TX_IF_DPHY_IRQ_MASK	Error interrupt that is generated under the following conditions: • Retransmit error • Unselected stream error. An unselected DMA path or CSI_RX_IF loopback path is sending data when not selected Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CORR_LEVEL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_SRAM_CORR_FAULT_STATUS	Interrupt on internal FIFO RAM. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_UNCORR_LEVEL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_SRAM_UNCOR R_FAULT_STATUS	Interrupt on internal FIFO RAM. Read the status register bitfields to trace the source of the event.
CSI_TX_IF_MAIN_0_CSI_FATAL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSI_TX_IF_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_TX_IF_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.
CSI_RX_CSI_NONFATAL	CSI_TX_IF_ASF_INT_MASK	CSI_TX_IF_ASF_INT_STATUS	ASF port fatal interrupt. Level sensitive. Set CSI_TX_IF_ASF_FATAL_NONFATAL_SELECT for whether fatal or non-fatal ASF interrupt is triggered. If any of the CSI_TX_IF_ASF_INT_STATUS bit is set, the CSI_RX_CSI_FATAL or CSI_RX_CSI_NONFATAL event signal is asserted.

12.6.4.4.5 CSI_TX_IF Data Memory Organization Details

Figure 12-430 shows the YUV422-8 data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

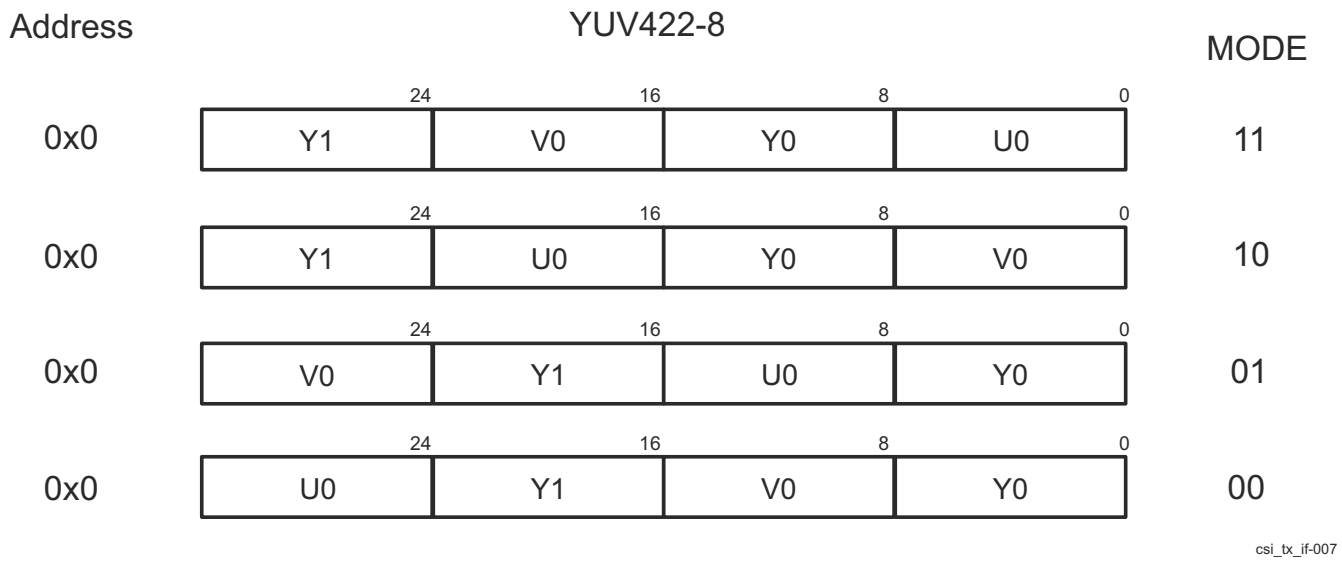


Figure 12-430. CSI_TX_IF YUV422-8 Memory Data Organization

Figure 12-431 shows the YUV422-10 data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

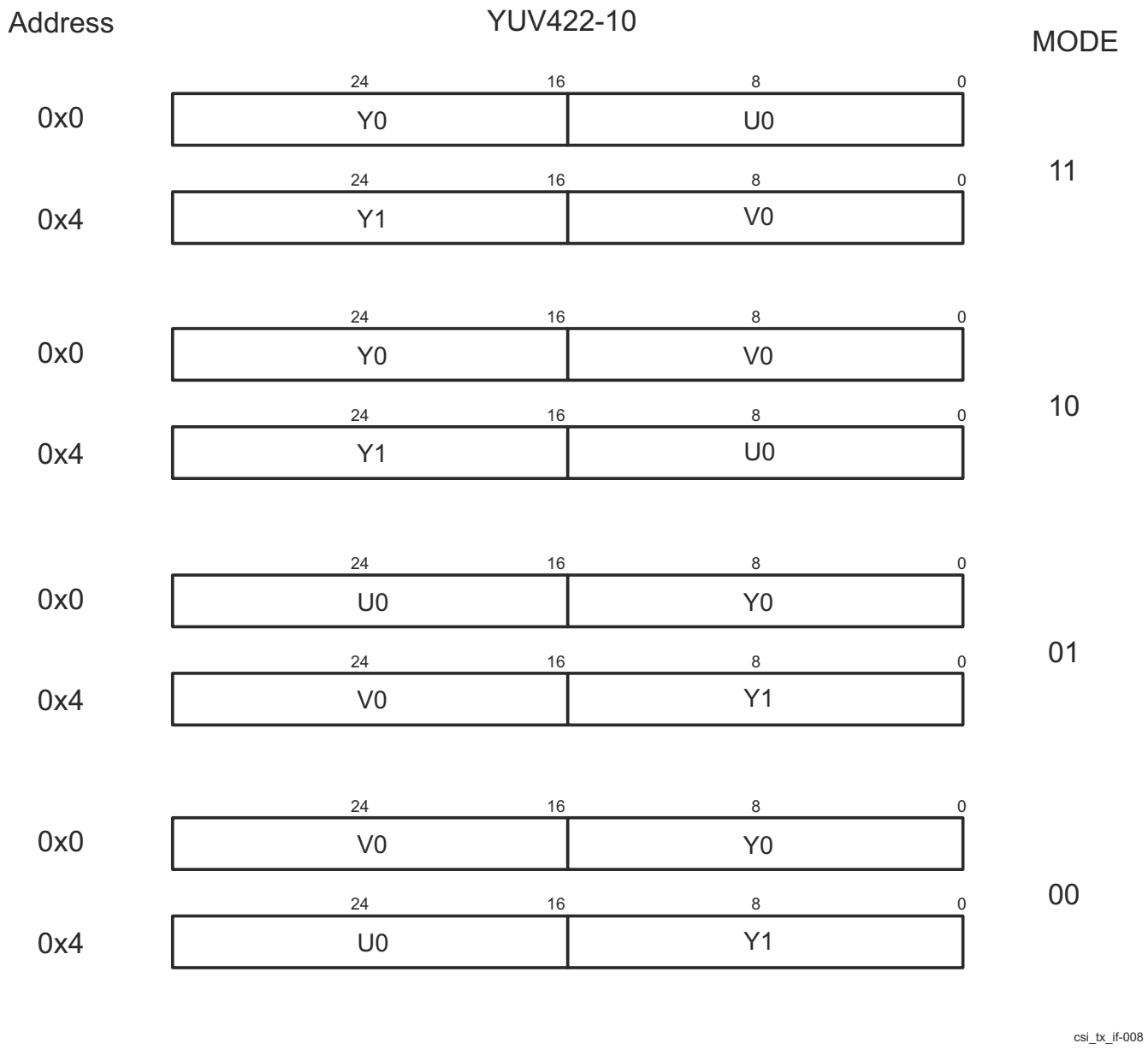


Figure 12-431. CSI_TX_IF YUV422-10 memory data organization

Figure 12-432 shows the RAW data organization in memory. User defined data types behave the same as RAW data types.

Copyright © 2026 Texas Instruments Incorporated

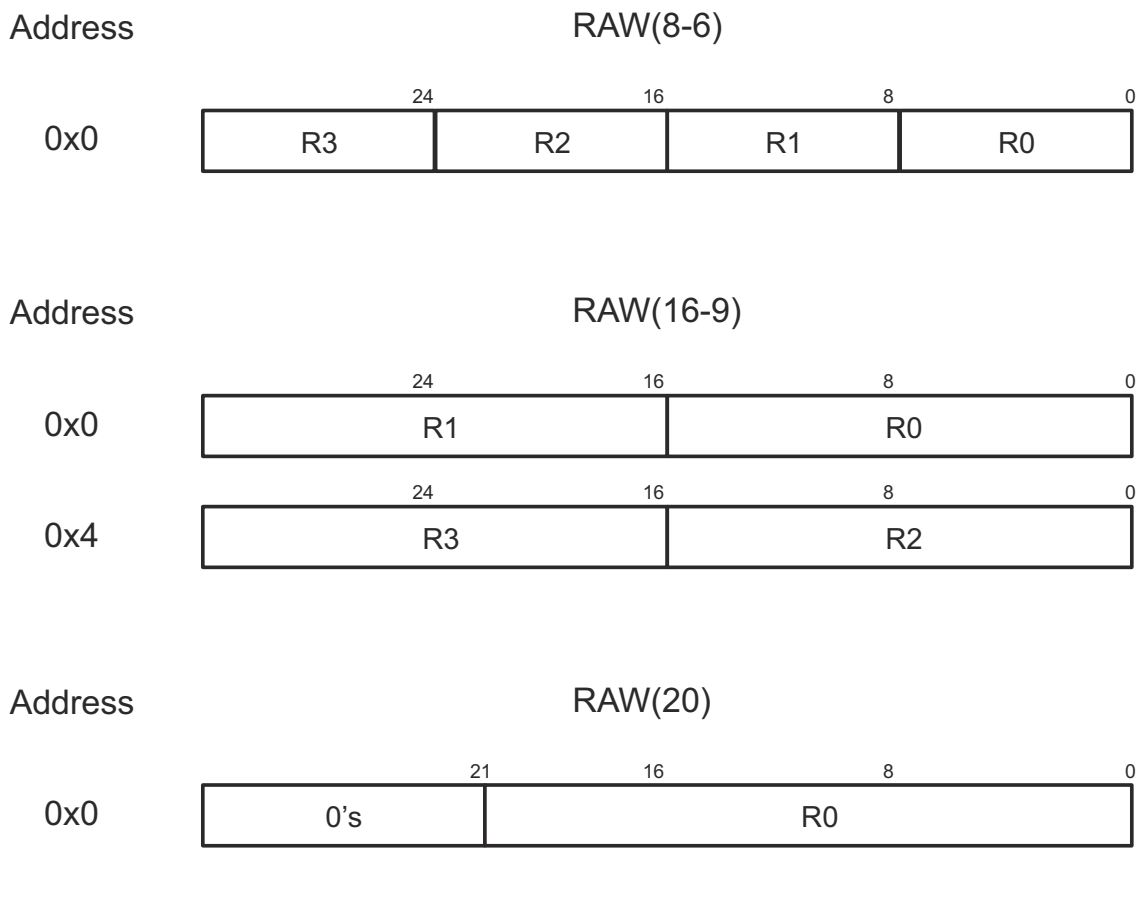


Figure 12-432. CSI_TX_IF RAW (UNPACKED) memory data organization

Figure 12-433 shows the RGB data organization in memory.

Copyright © 2026 Texas Instruments Incorporated

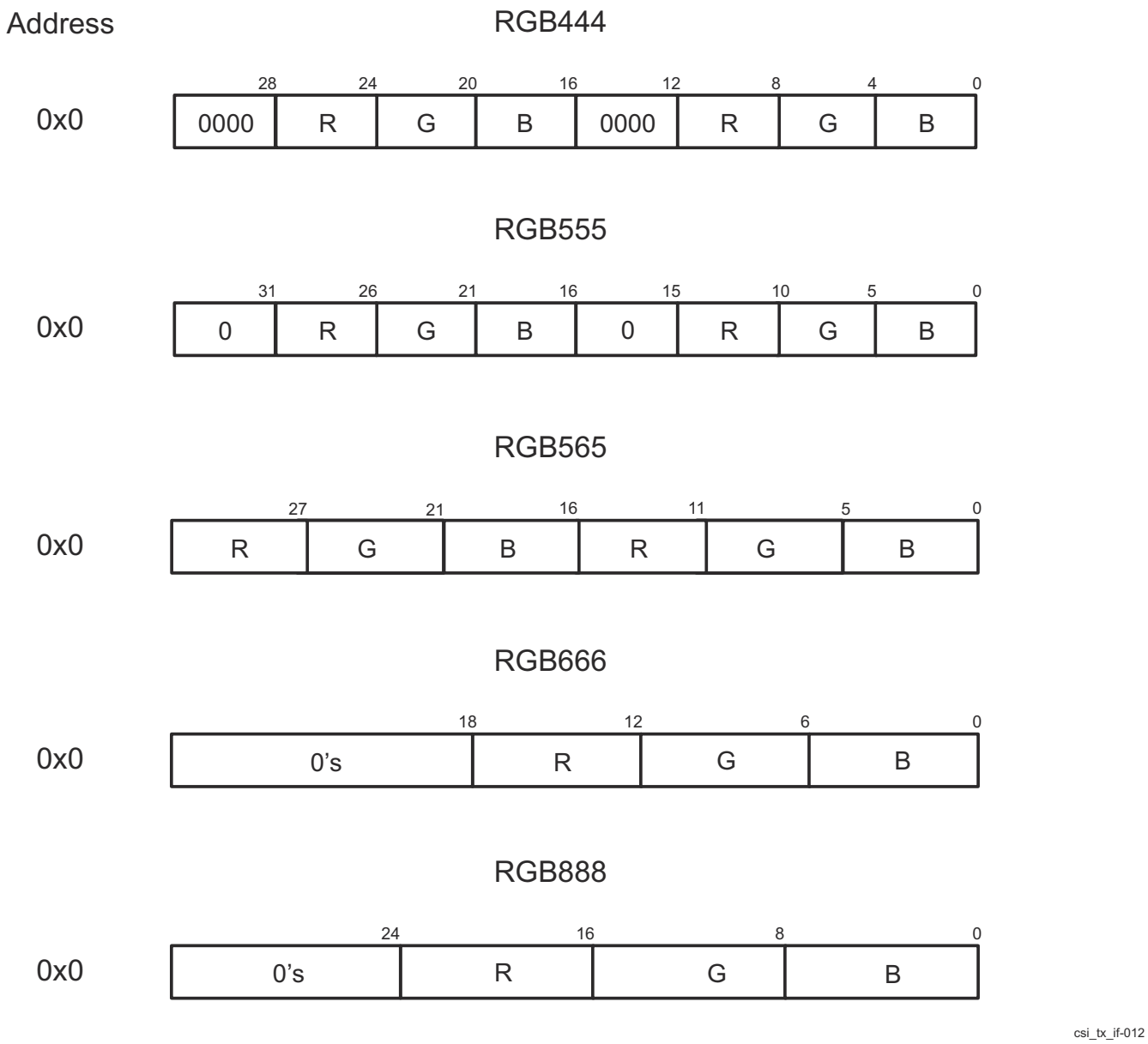
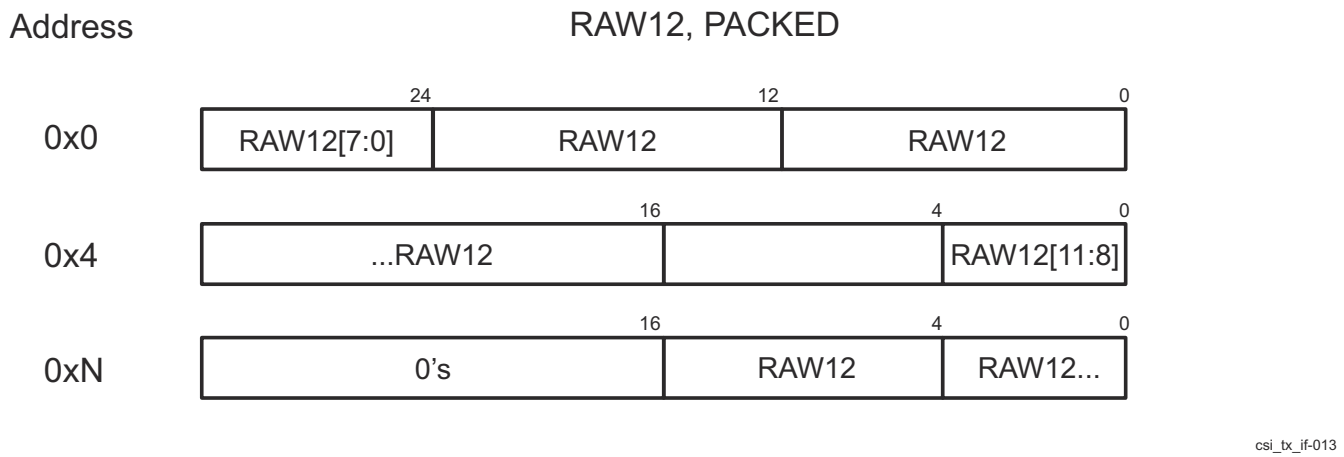


Figure 12-433. CSI_TX_IF RGB memory data organization

Figure 12-434 shows the RAW12 (PACKED) data organization in memory.

Copyright © 2026 Texas Instruments Incorporated



A. Where data does not fill out to a byte boundary, it is zero extended.

Figure 12-434. CSI_TX_IF RAW12 (PACKED) memory data organization

12.6.4.4.6 CSI_TX_IF PSI_L (DMA) Interface

The PSI_L (DMA) interfaces is common for both CSI_RX_IF and CSI_TX_IF, see chapter *Camera Streaming Interface Receiver (CSI_RX_IF) and MIPI DPHY Receiver (DPHY_RX)*, subsection *CSI_RX_IF PSI_L (DMA) Interface*.

12.6.4.4.7 CSI_TX_IF ECC Protection Support

The CSI_TX_IF has two ECC aggregators, one for byte clock (DPHY_TXBYTECLKHS) and the other one for all other clocks. The ECC is a mechanism for providing increased system reliability (via reduction of memory soft errors) by allowing single bit errors to be detected and corrected and double bit errors to be detected.

The ECC protection on the CSI_TX_IF RAM provides Single Error Correction and Double Error Detection (SEC/DED). This logic detects and corrects a single bit error (1 bit error per ECC word or per ECC data segment). For memories that contain critical and/or persistent data, automatic (immediate or delayed) write-back of the corrected data to the corresponding memory address is supported. In addition, the ECC also supports multiple options for partial word writes, such as read-modify-write or multiple ECC code segments per word.

The ECC protection also provides Double Error Detection (DED). This logic only detects (does not correct) double errors (2 bit errors per ECC word or per ECC data segment).

The ECC aggregators on the CSI_TX_IF subsystem level consolidates the ECC configuration and status bits for all the ECC supported memories in the subsystem. They provide a single EOI-handshake based interrupt to the interrupt processors (for both single and double error detections) and a standard 32-bit VBUS interface for configuring and querying the ECC register set, see *CSI_RX_IF_ECC Registers*. For complete details on the features and functions of the ECC aggregator, see *ECC Aggregator*.

12.6.4.5 CSI_TX_IF Programming Guide

Note

Software must keep the CSI_TX_IF_CONTROL1[0] PIXEL_RESET register in the asserted state in the control register until all CSI_TX_IF controller registers are programmed. Only after all CSI_TX_IF controller programming can software de-assert CSI_TX_IF_CONTROL1 register. If software has to reprogram any CSI_TX_IF controller registers it MUST have the control CSI_TX_IF_CONTROL1 register reset asserted.

It is a restriction that LS/LE detection is disabled.

Copyright © 2026 Texas Instruments Incorporated

12.6.4.5.1 CSI_TX_IF Programming (Configuration Mode)

Although it is always possible to read/write all control registers, it strongly recommended writing to the control registers only while configuration mode is active. To enter configuration mode, the CSI_TX_IF_TX_CONF[2] CONFIGURATION_REQUEST bit must be set. CSI_TX_IF enters configuration mode when transmission of any ongoing frame has ended and the configuration request bit is set. Configuration mode is the default state after reset. While configuration mode is active, the CSI_TX_IF_STATUS[2] CONFIGURATION_ACTIVE bit is set. Traffic at the pixel interface is ignored during configuration mode, and it is safe to write to control registers and change configuration.

To exit configuration mode, the CONFIGURATION_REQUEST bit in the CSI_TX_IF_TX_CONF register must be cleared. The CSI_TX_IF then starts normal operation, with first new frame incoming after configuration mode exit.

During configuration mode, all interrupts are disabled.

Note

A mandatory assertion of soft reset occurs following the de-assertion of CONFIGURATION_REQUEST. This ensures the controller is in a known state and all the parameters set in the configuration registers have been loaded correctly into the internal state. The DPHY_TXBYTECLKHS must be active before the configuration request is completed.

12.6.4.5.2 CSI_TX_IF System Initialization Programming

The power up/reset release of the system will clear all the register values to their reset conditions. Software must perform programming of the virtual channel and data type registers to match the system operation if the reset conditions do not match the required values. Software must ensure that all the data type select registers are programmed with valid values for the pixel_dt_sel_if parameter used.

The CSI_TX_IF registers for all the DPHY_TX related control and delays must be set before starting the system. The DPHY_TX delays must be calculated using the system clock periods associated with CSI_TX_ESC_CLK and DPHY_TXBYTECLKHS. The wait burst time must be calculated using the relevant DPHY_TX data sheet to ensure that all the lanes begin new requests at the same point.

Software must configure the FIFO fill level control registers, if required to suit clock ratios used and the desired payload control configurations. When all registers are programmed the configuration can be made active. The pixel streams can then start requests.

Note

The CSI_TX_IF is configured at design time, such that the reset values adopt a generic preferred Power_On state. This reduces the programming steps required by the system for the general use case scenario.

12.6.4.5.3 CSI_TX_IF Lane Control Programming

The CSI_TX_IF links to the DPHY_TX clock and data lanes. After power up, software must identify that the DPHY_TX is powered on and the DPHY_TX PLL has locked. The DPHY_TX controls for swapping the DP/DN control can be modified using the CSI_TX_IF_DPHY_CFG1 register if required.

Software must then program the enable for the clock lane and each data lane that is required. The DPHY_TX lane signals can be checked using the CSI_TX_IF_DPHY_STATUS register to identify that the lane is in STOPSTATE and is ready for High Speed transmission. This register can also be used to identify when ULPS is active.

12.6.4.5.4 CSI_TX_IF Virtual Channel and Data Type Management

The following section provides information on how the configuration of virtual channels and data types may be used in conjunction with one or more stream inputs to meet the system requirements.

Copyright © 2026 Texas Instruments Incorporated

12.6.4.5.4.1 CSI_TX_IF Data Type Interleaving

The CSI_TX_IF can support interleaving of data types by configuring the pixel_dt_sel_if inputs at the pixel interface to select the required data type and pixel count values.

12.6.4.5.4.2 CSI_TX_IF Data Type Interleaving with Multiple Interfaces

The CSI_TX_IF will perform data type interleaving with one, two, or more pixel interfaces. In this scenario, Stream0 will be the master and one or more streams may be ganged together as slaves to the master stream. Therefore, for all the ganged streams, the virtual channel signals must be configured to the same value and the pixel streams must have the same active frame valid cycle (that is the virtual_channel_if* and frame_valid_if* must be wired in parallel). The STREAM_IF_*_SLAVE_MODE bit in the STREAM_IF_*_CFG registers must also be set for all slaved streams. This ensures that only one frame start and frame end are generated for all the ganged streams. Note that the pixel 0 interface is always the master and slave bits exist for the other streams, in the control registers, to determine if they are to operate in master or slave mode.

In the case where streams are ganged together as master and slave(s), all ganged interfaces must have the same virtual channel value, but the pixel_dt_sel_if inputs must be set to different values. Again, each of the interfaces can have the pixel_dt_sel_if input change on a line by line basis, so that interleaving is done with all the ganged streams.

Note

Each stream must have exclusive use of the data types allocated to it.

Data type interleaving is only done when the frame is valid (CSI_TX_IF_DEBUG_PROT*_FSM[11] FRAME_VALID_IF* bitfield input is asserted). The virtual channel will be sampled on the first cycle after the rising edge of the FRAME_VALID_IF input and this virtual channel will be used for all the packets originating from that stream during that frame. The FRAME_VALID_IF input must be the same on all ganged stream interfaces.

12.6.4.5.4.3 CSI_TX_IF Virtual Channel Interleaving

The CSI_TX_IF can support interleaving of virtual channels by using two or more pixel interfaces and configuring the value of the virtual channels signal differently on each interface. The data streams can then be controlled using the pixel_dt_sel_if inputs to select the data type for the lines. In this scenario, each pixel interface must have exclusive use of the virtual channel. Each stream will generate its own frame start and frame end synchronisation packets. In this case, all the stream protocol modules employing virtual channel interleaving must be configured as masters by clearing the STREAM_IF_*_SLAVE_MODE bit in the STREAM_IF_*_CFG registers.

Note that the virtual channel control input is sampled at the rising edge of the frame_valid_if input, and thereafter remains fixed for that frame, therefore the virtual channel cannot be switched on a line-by-line basis for any single pixel interface. While it is possible to switch the Virtual Channel control input from frame to frame, the timing-critical switching required here would indicate a requirement for hardware rather than software control.

12.6.4.5.4.4 CSI_TX_IF Virtual Channel and Data Type Interleaving

The CSI_TX_IF may be configured to perform a combination of the scenarios outlined in the previous sections if sufficient streams are available. One or more streams may be ganged with Stream0 to perform data type interleaving on the same virtual channel, whilst the remaining streams may be left autonomous to perform virtual channel interleaving.

In this case, the same rules governing the use of the virtual channels and data types must be observed. In this scenario, the virtual channel signals will be configured to the single value required for the ganged group (as they need to be controlled within the same frame valid active signal) and different values for those left to be autonomous (as they may generate their own frame start and frame end events).The slave stream for each virtual channel can have a different data type selected, and this may change on a line-by-line (or packet-by-packet) basis.

Copyright © 2026 Texas Instruments Incorporated

12.6.4.5.5 CSI_TX_IF Line Control

The line control block uses the DPHY_TXBYTECLKHS from the DPHY_TX PPI interface to take the line data from the packet interface FIFO and transfer the bytes to the active lanes of the DPHY_TX. The block detects when the DPHY_TX is ready to transmit high speed data and the pixel stream has packets available.

The line control block performs pixel stream arbitration using the line control arbitration block when more than one pixel stream is generated in the configuration. The block performs all the clock and data lane control required to activate the high-speed transmission and return the lane to ULPS state as required.

12.6.4.5.5.1 CSI_TX_IF Line Control Arbitration

The CSI_TX_IF controller uses a simple arbitration scheme for configurations with more than one pixel interface.

The arbitration has 2 levels; the first is a round-robin scheme and if no streams are granted by this a priority based scheme is used. The arbitration will select the first stream making a request if there is only one stream waiting or the first in the round robin after a stream has completed transmission. This means that there may be one cycle where the round-robin selector moves across a stream position that is not requesting to transmit.

The re-arbitration boundary is between packets except where line sync is used. In this case the line start packet, long packet and line end are grouped together.

The selected stream will pass the line data from the stream FIFO to the line control block for distribution to the active PPI lanes of the DPHY_TX.

12.6.4.5.6 CSI_TX_IF Lane Manager FSM

Data distribution to the available lanes is controlled via the lane manager FSM.

The state descriptions can be found in Table 12-346.

Table 12-346. CSI_TX_IF Lane Manager FSM State Description

State (line_fsm_st_r)	Description
LINE_FSM_IDLE	Wait for new transmission to be ready. When start_hs_transmission_c is asserted then go to the transmission state per enabled lanes (lanes_enable_r)
LINE_FSM_BURST_1L	Transmit burst data over lane 0 until end of the burst. When end then go to the LINE_FSM_BURST_END state if EPD is not enabled, otherwise go to the LINE_FSM_LRTE_EPD_SPACER state.
LINE_FSM_BURST_2L	Transmit burst data over lanes 0 and 1 until end of the burst. When end then go to the LINE_FSM_BURST_END state if EPD is not enabled, otherwise go to the LINE_FSM_LRTE_EPD_SPACER state.
LINE_FSM_BURST_4L	Transmit burst data over all lanes until end of the burst. When end then go to the LINE_FSM_BURST_END state if EPD is not enabled, otherwise go to the LINE_FSM_LRTE_EPD_SPACER state.
LINE_FSM_LRTE_EPD_SPACER	Insert spacer packets across each active lane. When finished, if EPD Option 1 then go to the LINE_FSM_LRTE_EPD_PDQ state, otherwise go to the LINE_FSM_IDLE state.
LINE_FSM_LRTE_EPD_PDQ	Tell D-PHY to initiate HS-IDLE state and insert PDQ.
LINE_FSM_BURST_END	Wait for being ready for new burst transmission. In this state counter counts until WAIT_BURST_TIME value in the register is reached.

12.6.4.5.7 CSI_TX_IF Data Lane Control FSM

Data lanes control uses a simple FSM to sequence entry and exit of Ultra Low Power mode. This FSM observes inputs from the control registers and based on these inputs it generates PPI outputs for ULP mode.

Table 12-347 shows the state descriptions.

Table 12-347. CSI_TX_IF Data Lanes Control FSM State Description

State (ulps_data_fsm_st_r)	Description
DATA_LN_IDLE	Idle state. In this state HS transmission can be issued if the Clock Lane is in HS mode.
DATA_LN_ULPS_REQ	Request to enter ULP state. Waiting for activation ULP confirmed by the ppi_ulps_active_not_dl input.

Copyright © 2026 Texas Instruments Incorporated

Table 12-347. CSI_TX_IF Data Lanes Control FSM State Description (continued)

State (ulps_data_fsm_st_r)	Description
DATA_LN_ULPS_ACTIVE	ULPS for Data Lanes is active. Waiting for the ulps_req_sync signal being low to exit ULPS.
DATA_LN_ULPS_EXIT	ULPS exiting, waiting for the time defined in the CSI_TX_IF_DPHY_ULPS_WAKEUP register.

12.6.4.5.8 CSI_TX_IF Application Examples

This section details specific steps on how to program the CSI_TX_IF.

12.6.4.5.8.1 CSI_TX_IF D-PHY Control and Configuration

The DPHY_TX will use a PLL to provide the bit clock for transmission. The CSI_TX_IF controller uses the associated byte clock as a primary clock source and this must be active and stable before the CSI_TX_IF is ready to transmit high speed data.

Software must perform the programming of the PLL dividers and monitor the lock status. Software must also ensure the correct sequence of programming the PLL, DPHY_TX and CSI_TX_IF to guarantee the clocks are active and resets are released, following the start-up sequence described in DPHY_TX chapter (See chapter *Shared MIPI D-PHY Transmitter (DPHY_TX)*).

12.6.4.5.8.2 CSI_TX_IF Clock and Data Lane Enable

The DPHY_TX lanes are controlled using the CSI_TX_IF_DPHY_CFG register. The clock enable will be asserted once the DPHY_TX is configured, PLL is programmed and locked and the lanes are enabled. Enabling the clock lane will then make the TX clock module stay in stop state with the DP/DN pads in LP11. The TX clock high speed ready state will be detected once the TX clock lane has completed the preamble and begun to drive the high speed clock.

The configured number of data lanes will be enabled at the same time as enabling the TX clock. The data lane will remain in LP until the high-speed clock is stable and the CSI_TX_IF makes the request for the data lane transitions from stop state to active (LP11-LP01-LP00).

12.6.4.5.8.3 CSI_TX_IF DP/DN Signal Swap

The DPHY_TX may have the capability to swap the data pin polarity or invert the clock lane polarity - the CSI_TX_IF_DPHY_CFG1 register is available within the controller for driving these pad level controls (using the DPHY_DIFF_INVERT_* fields). The DPHY_TX will identify any contention issues with the transmitter and flag this using the contention signals and error interrupt - this may be used by software to determine if the data pin or clock lane polarity switching is required or not (by optimizing to the minimum, preferably zero, rate of occurrence of contention issues).

Int the DPHY_TX, the IO pads for the clock and data lanes support swapping of the polarity of the signals DP-DN. All dphy_differential_invert_* outputs should be treated as asynchronous. Note that, if these signals need to be asserted, they should be asserted before releasing the DPHY_TX reset.

Each lane can be controlled independently with the CSI_TX_IF_DPHY_CFG and CSI_TX_IF_DPHY_CFG1 registers.

12.6.4.5.9 CSI_TX_IF DPHY_TX Status

The status of clock and data lanes is available in the read only CSI_TX_IF_DPHY_STATUS register. The clock lane fields show the condition of the ppi_tx_ulps_active_not_cl and ppi_stopstate_cl PPI inputs. The data lane fields show the ppi_ulps_active_not_dl* and ppi_stopstate_dl* PPI inputs.

All the status signals are based on the active enabled lane configuration.

The DPHY_TX can signal error conditions for each lane for contention in low power transitions and signal errors at the start of high speed transitions, using the CSI_TX_IF_DPHY_IRQ register. The signals can be used to generate an error interrupt event if the relevant bits are set in the CSI_TX_IF_DPHY_IRQ_MASK register. The flag can be cleared by writing to the CSI_TX_IF_DPHY_IRQ register.

Copyright © 2026 Texas Instruments Incorporated

12.6.4.5.10 CSI_TX_IF ULPS Operation

Each lane can be put in ultra-low power state (ULPS) by software configuration. The ULPS mode requires all the following conditions:

- The lane must be in stop state.
- For data lanes, no data must be pending in the CSI_TX_IF module.

12.6.4.5.11 CSI_TX_IF System Frame Rate Measurement

The CSI_TX_IF provides an interrupt event to signal when the frame start (FS) and frame end (FE) packets are sent to the DPHY_TX. These events are generated on a stream basis for each virtual channel and allow a system configured to perform virtual channel interleaving to capture the events, and to derive the frame rate achieved based on the flow control and clock configuration.

The system performance will be impacted by the FIFO sizing and any flow control mechanism used by the pixel streams, so this provides a mechanism for checking that the system is scaled correctly.

12.6.4.5.12 CSI_TX_IF Configuration for PSI_L

Table 12-348 lists the settings that must be programed before enabling CSI_TX_IF controller and sending PSI_L data.

Table 12-348. CSI_TX_IF Configuration table for PSI_L

Format	Pixel transmit mode used	Size	Details
YUV420-8	single	0	Must set YUV420 mode in CSI_TX_IF_DMACNTX_j
YUV420-10	single	1	Must set YUV420 mode in CSI_TX_IF_DMACNTX_j
YUV422-8	single	0	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
YUV422-10	single	1	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
YUV422-8	packed	2	Must set YUV422 mode in CSI_TX_IF_DMACNTX_j
RGB888, RGB666	single	2	
RGB565, RGB555, RGB444	single	1	
RAW6-7(1)	single	0	
RAW8(1)	dual	1	
RAW8(1)	quad	2	
RAW10(1)	single	1	
RAW10(1)	dual	2	
RAW12(1)	single	1	No packing
RAW12(1)	dual	2	No packing
RAW12(packed)(1)	single	0	Must set CSI_TX_IF_DMACNTX_j[23] PACK12_CFG = 1
RAW12(packed)(1)	dual	1	Must set CSI_TX_IF_DMACNTX_j[23] PACK12_CFG = 1
RAW14(1)	single	1	
RAW14(1)	dual	2	
RAW16(1)	single	1	
RAW20(1)	single	2	

(1) User defined/blanking/generic data types are programmed the same way as RAW data types.

12.6.4.5.13 CSI_TX_IF Configuration for Color Bar

The following lists the Color Bar configuration requirements:

Copyright © 2026 Texas Instruments Incorporated

- Color bar must be set to packed YUV422-8 format, this is an extended datatype in CSI_TX_IF controller.
- Software must program CSI_TX_IF controller before enabling the color bar from the CSI_TX_IF_COLOR_CNTL[0] EN bit-field.
- Program CSI_TX_IF_COLOR_LINE_DELAY, CSI_TX_IF_COLOR_FRAME_DELAY, CSI_TX_IF_COLOR_START_DELAY, and CSI_TX_IF_COLOR_PARAM registers with appropriate configurations. Update CSI_TX_IF_COLOR_CNTL with virtual channel and data type desired and enable it.

12.6.4.5.14 CSI_TX_IF Error Recovery

When an underflow error occurs, software must reset the entire CSI_TX_IF by SoC PSC reset and restart over.

12.6.4.5.15 CSI_TX_IF Power up/down Sequence

The following lists the power up/down sequence requirements:

1. Software must ensure no more PSI_L traffic is directed toward the CSI_TX_IF module.
2. Software must then go through a tear down process on PSI_L gasket inside CSI_TX_IF wrapper. Refer to chapter *PSI_L* for tear down process.
3. Software must then check the stream idle values so that they are idle in the CSI_TX_IF_CONTROL1 register.
4. Software must then stop all traffic inside the CSI_TX_IF controller and wait till it is idle.
5. After that software can then proceed to do PSC power down routine.

12.7 Timer Modules

12.7.1 Global Timebase Counter (GTC)

This section describes the global timebase counter (GTC) in the device.

12.7.1.1 GTC Overview

The GTC module provides a continuous running counter that can be used for time synchronization and debug trace time stamping.

12.7.1.1.1 GTC Features

The GTC supports the following features:

- 64-bit up counter
- No rollover during the lifetime of the device
- Compatible with ARMv8 system counter requirements:
  - Disabled at power-up
  - Register definition and memory map aligned to ARMv8 definition
  - Implements memory-mapped counter control and status frames
- Outputs reflected binary (Gray) encoded timer value for system timer bus distribution to other modules
- Selectable counter bit output as a push event that can be used by CPTS modules, timers or interface protocols

12.7.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.7.1.2 GTC Functional Description

12.7.1.2.1 GTC Block Diagram

The GTC is essentially comprised of a 64-bit up counter, a Gray encoder, a 64-bit multiplexer, and memory-mapped control registers. Figure 12-435 shows a high-level block diagram of the GTC.

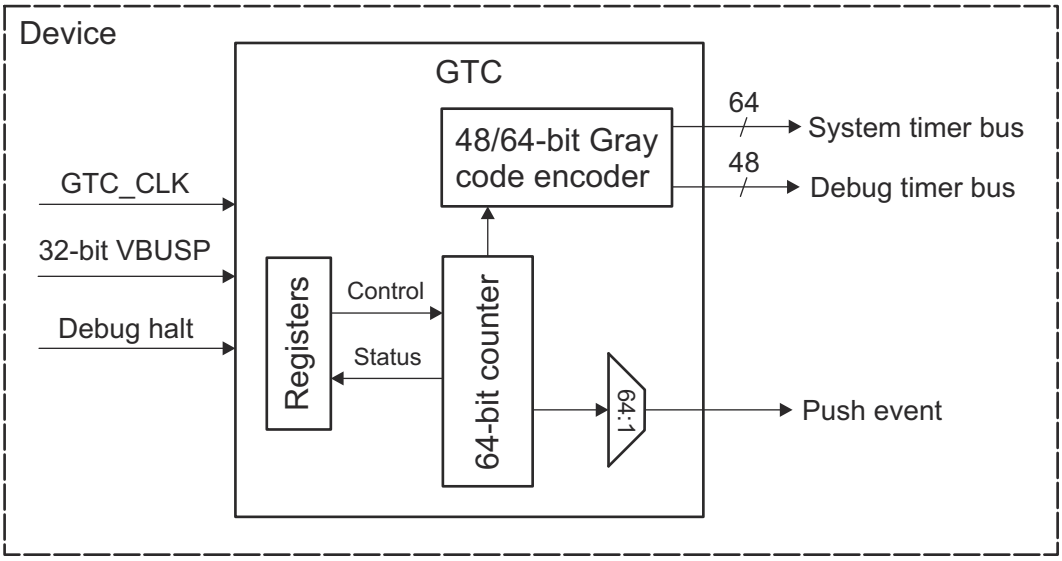


Figure 12-435. GTC Block Diagram

12.7.1.2.2 GTC Counter

The counter is a 64-bit binary up counter that increments on every GTC_CLK rising edge. The source for GTC_CLK is selected through a mux which is controlled via the GTC_CLKSEL[2-0] register bit field.

The counter is disabled by default (at power-on-reset) and increments only when enabled by setting the *GTC_CFG1_CNTCR*[0] EN bit to '1'. The current counter value is readable via the combined COUNTVALUE bit field of both *GTC_CFG1_CNTCV_HI* (upper 32 bits) and *GTC_CFG1_CNTCV_LO* (lower 32 bits) registers.

When counting is disabled, a new counter value can be loaded via a software write to the combined COUNTVALUE bit field. In this case, when the counter gets re-enabled, it will start counting from the last value written to this field.

The counter can be optionally configured to stop incrementing when a debug halt signal is issued, by writing a '1' to the *GTC_CFG1_CNTCR*[1] HDBG bit. This condition is indicated through the *GTC_CFG1_CNTSR*[1] DBGH status bit.

12.7.1.2.2.1 Steps to Clear the Counter Value to Zero

1. Disable the GTC
2. Clear GTC_CNTCV_LO (First)
3. Clear GTC_CNTCV_HI (Second)
4. Enable the GTC

Once the GTC is enabled, the GTC_CNTCV_HI bit will be cleared to zero.

12.7.1.2.3 GTC Register Partitioning

The ARMv8 architecture spec requires specific system-level components, each with one or two register frames. To accommodate this, the GTC memory-mapped registers (MMRs) are divided into four separate regions (4KB each):

- Peripheral MMRs (GTC0_GTC_CFG0): This region includes standard peripheral identification registers and any other control registers not associated with the ARMv8 system timer functions.

Copyright © 2026 Texas Instruments Incorporated

- Counter control MMRs (GTC0_GTC_CFG1): This region includes registers that provide control over the operation of the system timer.
- Counter status MMRs (GTC0_GTC_CFG2): This region includes registers that provide a mechanism for reading the status of the system timer.
- Timer control MMRs (GTC0_GTC_CFG3): This region includes registers that identify and provide a control mechanism for memory-mapped timer implementations. For this device, no memory-mapped timers are implemented and only the minimum registers required to indicate the presence of no timers are necessary.

12.7.2 RTI-Windowed Watchdog Timer (WWDT)

This section describes the Windowed Watchdog Timer (WWDT), implemented by using the Digital Windowed Watchdog (DWWD) function of the Real Time Interrupt (RTI) module in the device.

CAUTION

The RTI module shall be used to serve only as a Digital Windowed Watchdog (DWWD).

The Real Time Interrupt module provides timer functionality for operating systems and for benchmarking code. The module incorporates several counters, which define the timebases needed for scheduling in the operating system.

This module is specifically designed to fulfill the requirements for OSEK (“Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug”; “Open Systems and the Corresponding Interfaces for Automotive Electronics”) as well as OSEK/Time compliant operating systems.

The timers also provide the ability to benchmark certain areas of code by reading the counter contents at the beginning and the end of the desired code range and calculating the difference between the values.

12.7.2.1 RTI Features

The RTI modules include the following main features:

- Windowed Watchdog Timer (WWDT) feature.
- Two independent 64 bit counter blocks (counter block0 or counter block1). Each block consists of
  - One 32 bit up counter
  - One 32 bit free running counter
  - Two capture registers for capturing the prescale and free running counter on a special event.
- Free running counter 0 can be incremented by either the internal prescale counter or by an external event.
- Four configurable compare registers for generating operating system ticks . Each event can be driven by either counter block0 or counter block1.
- Fast enabling/disabling of events.
- RTI clock input derived from any of the available clock sources, selectable in the System Module
- Optional capability to drive a pulse-width modulated signal out on an interrupt line.

12.7.2.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.7.2.3 RTI Functional Description

The MCU_RTI0 and RTI modules are hereinafter referred to as RTI module.

12.7.2.3.1 RTI Digital Windowed Watchdog

Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Note

Digital windowed watchdog (DWWD) timer is implemented using the digital windowed watchdog function of the RTI modules. Real time interrupt functionality is not supported. In this mode, the timer should default to disabled and user can adjust the period as desired before enabling the watchdog.

In addition to the time-out boundary configurable via the digital watchdog (DWD), some applications may also want to configure the start-time boundary of the watchdog. This is enabled by the digital windowed watchdog (DWWD) feature.

Functional Behavior

The DWWD opens a configurable time window in which the watchdog must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate either a reset or a non-maskable interrupt to the CPU. This is controlled by configuring the RTI_RTIDWWDRXNCTRL register. As stated earlier, when the watchdog needs to be enabled by software, the watchdog counter is disabled on a system reset. When the DWWD is configured to generate a non-maskable interrupt on a window violation, the watchdog counter continues to count down. The RTI_INTR_WWD interrupt handler needs to clear the watchdog violation status flag(s) and then service the watchdog by writing the correct sequence in the watchdog key RTI_RTIWDKEY register. This service will cause the watchdog counter to get reloaded from the preload value and start counting down. If the RTI_INTR_WWD handler does not service the watchdog in time, it could count down all the way to zero and wrap around. No second exception for a time out is generated in this case.

Configuration of DWWD

The DWWD preload value (same as DWD preload) can only be configured when the DWWD counter is disabled. The window size and watchdog reaction to a violation can be configured even after the watchdog has been enabled. Any changes to the window size and watchdog reaction configurations will only take effect after the next servicing of the DWWD.

Copyright © 2026 Texas Instruments Incorporated

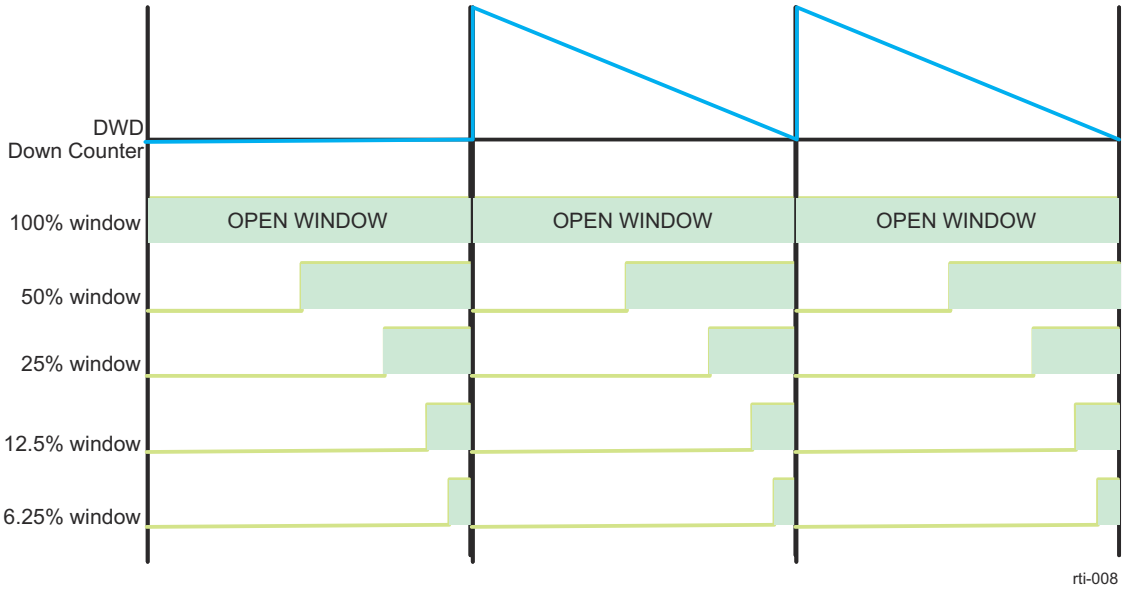


Figure 12-436. RTI Digital Windowed Watchdog Timing Example

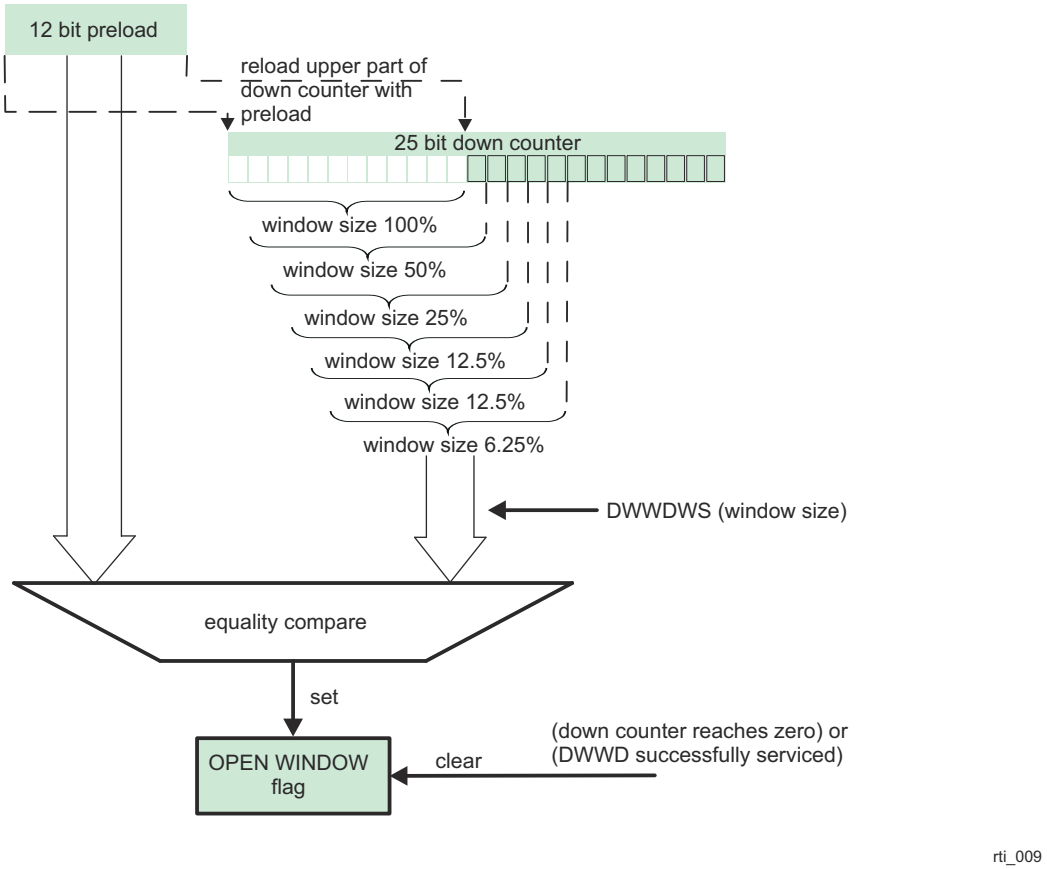


Figure 12-437. RTI Digital Windowed Watchdog Operation Block Diagram

Copyright © 2026 Texas Instruments Incorporated

12.7.2.3.1.1 RTI Debug Mode Behavior

Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Once the system enters debug mode, the behavior of the RTI depends on the RTI_RTIGCTRL[15] COS bit. If the bit is cleared and debug mode is active, all counters will stop operation. If the bit is set to one, all counters will be clocked normally and the RTI will work like in normal mode.

The DWD counter will not decrement in debug mode and will hold its current value, regardless of the RTI_RTIGCTRL[15] COS bit.

Note

The user must not service the watchdog while in debug mode.

12.7.2.3.1.2 RTI Low Power Mode Operation

The operation of the RTI module is guaranteed in run, doze and snooze mode. In sleep or hibernate mode all clocks will be switched off and the RTI module will not work.

In doze and snooze modes all parts of the RTI are active, since it has to be able to wake up the device with compare and timebases interrupts. Capturing events generated by the interrupt module is also possible since in both modes the peripheral modules are able to generate interrupts, which can trigger capture events. The RTI module will generate compare and timebases interrupts. The compare interrupts will periodically wake up the device.

Note

In the special case of doze mode with DPLL off, RTI_FCLK might have a different period than with DPLL enabled, since RTI_FCLK will be derived from the oscillator output. It has to be ensured that the RTI_ICLK to RTI_FCLK ratio is at least 3:1.

The DWD/DWWD remains active when the device enters low power mode as long as the RTI_FCLK is kept active.

Whenever the LPSC that controls an RTI is in any state other than Enable (see , *Module States*), the RTI cannot count or generate interrupts. For more information, see , *Power Control Modules*.

During standard SoC warm reset the RTI and the rest of the SoC are reset. In this case, the RTI counters are stopped and interrupts are not issued. During reset-isolated warm reset, if an R5FSS is put in reset-isolation, then the associated RTI also becomes reset isolated. As such, the R5FSS and the RTI are not reset, but RTI stops counting and cannot generate interrupts until R5FSS is taken out of CLKSTOP (brought to Enable state).

12.7.2.3.2 RTI Digital Watchdog

Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Some applications might use a digital watchdog (DWD) integrated in the RTI module. The digital watchdog generates resets after a programmable period, if no correct key sequence is written to the RTI_RTIWDKEY register. Figure 12-438 shows the digital watchdog functional block.

Copyright © 2026 Texas Instruments Incorporated

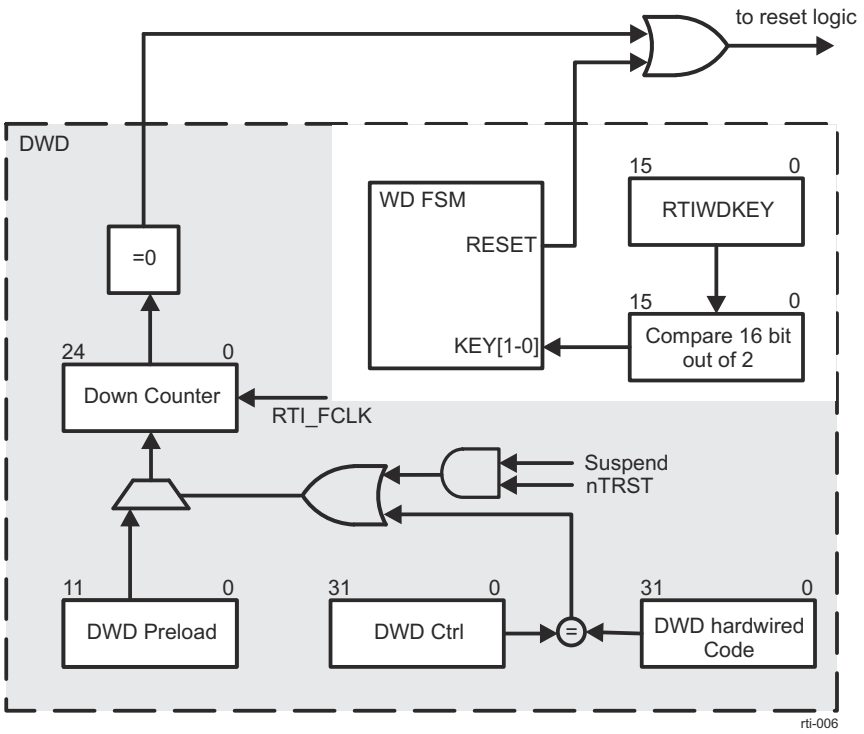


Figure 12-438. RTI Digital Watchdog Functional Block Diagram

The digital watchdog functionality is implemented such that it can be enabled by software.

The DWD starts counting down from the reset value of the RTI_RTIDWDCNTR (DWD Counter Register). The DWD preload register can be configured at any time by the application according to the desired time-out period.

When enabled by software, the digital watchdog is disabled after system reset. If it should be used, it has to be enabled by writing A98559DAh to the RTI_RTIDWDCTRL register. The DWD timeout period must be configured using the DWD preload register before the DWD is enabled. The DWD cannot be disabled by the application once it is enabled.

Note

When the DWD is enabled by software, any system reset will disable the DWD. This reset could have been generated by the watchdog itself.

If the correct key sequence is written to the RTI_RTIWDKEY register (E51Ah followed by A35Ch), the 25-bit DWD Down Counter is reloaded with the 12-bit preload value stored in RTI_RTIDWDPRLD register. If any incorrect value is written to the RTI_RTIWDKEY register, a watchdog reset will occur immediately. A reset will also be generated, when the DWD Down Counter is decremented to 0.

The user has to take into account that the write to the RTI_RTIWDKEY register takes 3 RTI_ICLK cycles. This needs to be considered for the DWD expiration calculation.

The DWD Down Counter will be decremented with RTI_FCLK frequency. If the RTI_FCLK is switched off via the disable registers of the Clock management, the DWD counter stops decrementing. The DWD module cannot generate a reset under this condition.

Copyright © 2026 Texas Instruments Incorporated

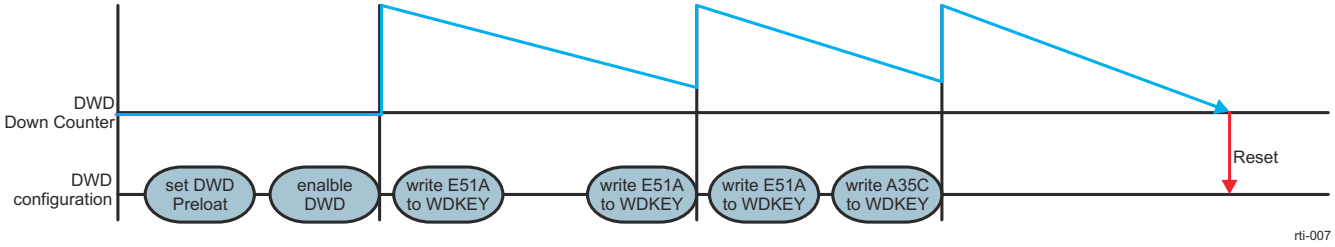


Figure 12-439. RTI Digital Watchdog Operation

The expiration time of the DWD Down Counter can be determined with following equation:

$$t_{exp} = (RTI_RTIDWDPRLD + 1) \times 2^{13} / RTI_FCLK \quad (40)$$

where RTI_RTIDWDPRLD = 0...4095

12.7.2.3.3 RTI Counter Operation

Note

Some of the RTI features described in this section may not be supported on this family of devices. For more information, see *RTI Not Supported Features*.

Figure 12-440 shows the RTI module counter blocks. The RTI module supports two counter blocks.

Copyright © 2026 Texas Instruments Incorporated

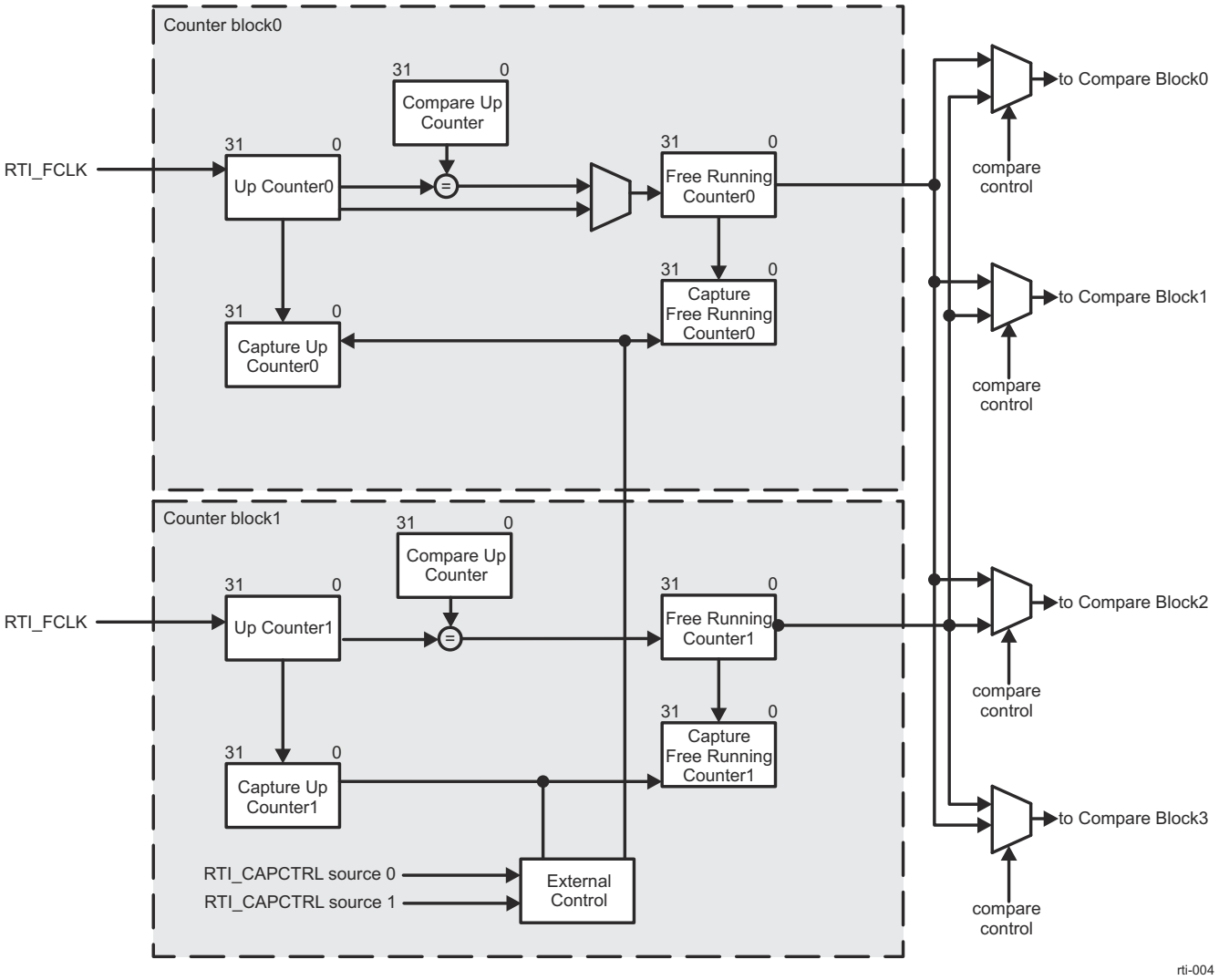


Figure 12-440. RTI Counters Block Diagram

Each block consists of two 32 bit up counters - Up Counter (UC) and Free Running Counter (FRC). The Up Counter (RTI_RTICAUC0 or RTI_RTICAUC1 register) is driven by the RTI_FCLK and counts up until the compare value in the Compare Up Counter register (RTI_RTICPUC0 or RTI_RTICPUC1) is reached. When the compare matches, the second counter (RTI_RTICAFRC0 or RTI_RTICAFRC1 register), which is a free running counter, is incremented. At the same time UCx is reset to zero.

To ensure the consistency of the counters, when both counter value have to be determined, the Free Running Counter has to be read first. This will ensure that at the CPU read cycle, the Up Counter value is stored in the counter register. The second read is done on the Up Counter register, which holds then the value of the counter cycle of the previous read on the Free Running Counter register.

Both blocks provide also a capture feature on external events. Two capture sources can trigger the capture event. Which event triggers block 0 or block 1 is configurable from the RTI_RTICAPCTRL register. The sources are coming from the interrupt manager, in order to be able to generate a capture event when one of the peripheral modules has generated an interrupt. The peripheral, which can generate an event is configured in the interrupt manager. When the event is detected, UCx and FRCx are stored in Capture Up Counter (RTI_RTICAUC0 or RTI_RTICAUC1) and Capture Free Running Counter (RTI_RTICAFRC0 or RTI_RTICAFRC1) registers. The read order of the captured values has to be like the order of the actual counters. So the CAFRCx has to be read first and the CAUCx registers has to be read after the CAFRCx value

Copyright © 2026 Texas Instruments Incorporated

was determined. While the CAFRCx is read the CAUCx value is loaded into a shadow register to ensure data consistency, if during the two reads of the captured data another capture event happens. If the application fails to read the two registers before a second capture event happens, the previous data will be overwritten.

Figure 12-441 shows the block diagram for one compare block. The RTI module supports four compare blocks.

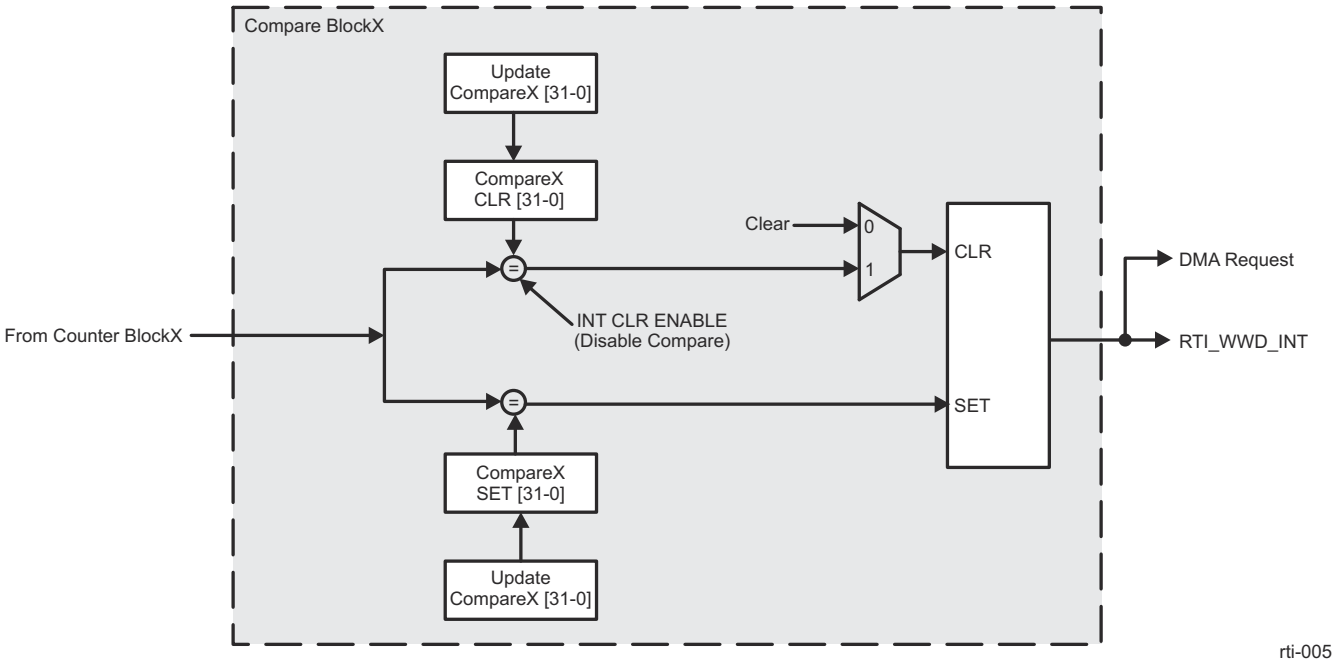


Figure 12-441. RTI Compare Block Diagram

In order to generate interrupt requests to the interrupt manager, there are four compare registers (RTI_RTICOMP0, RTI_RTICOMP1, RTI_RTICOMP2, and RTI_RTICOMP3). Each of the compare registers can be configured to work either on FRC0 (Counter block0) or FRC1 (Counter block1). When the counter value matches the compare value, an interrupt is generated. This sets an interrupt request line to the interrupt manager. The compare value gets updated automatically with the value stored in Update Compare (RTI_RTIUDCP0, RTI_RTIUDCP1, RTI_RTIUDCP2, and RTI_RTIUDCP3) registers when the compare matches. This gives the ability to generate periodic interrupts/DMA requests without having to update the compare value by software.

An optional feature allows an application to program another compare value which is then used to clear the interrupt request line. This feature is supported by four compare clear registers (RTI_RTICOMP0CLR, RTI_RTICOMP1CLR, RTI_RTICOMP2CLR, and RTI_RTICOMP3CLR). When the counter value matches the compare clear value, the interrupt line is cleared. This clears the interrupt request line to the interrupt manager. The compare clear value gets updated automatically with the value stored in Update Compare (RTI_RTIUDCPx) registers when the compare matches.

12.7.3 Real-Time Clock (RTC)

12.7.3.1 RTC Overview

The basic purpose for the RTC is to keep time of day. The other equally important purpose of RTC is for Digital Rights management. Some degree of tamper proofing is needed to ensure that simply stopping, resetting, or corrupting the RTC does not go unnoticed so that if this occurs, the application can re-acquire the time of day from a trusted source. The final purpose of the RTC is to wake the rest of chip up from a power down state.

12.7.3.1.1 RTC Features

The real-time clock (RTC) provides the following features:

Copyright © 2026 Texas Instruments Incorporated

- With Analog IP block
  - 2 digital voltage domains with integrated LVL/ISO cells
    - CORE or volatile domain, same as core SOC
    - RTC or Always ON domain
  - Without Analog IP block
    - 1 digital voltage domain, no lvl.iso
- 15 bit 32768Hz counter
- 48 bit seconds counter with +/- 1 30uS upto +/- 1 S drift adjustment every 4048 Seconds
- 256 bits of Scratch PAD
- 1 ON_OFF compare event, 48-bits
- 1 OFF_ON compare event, 48-bits
- 2 event outputs OFF_ON and ON_OFF to CORE
- Support active external 32768 Hz and inactive/gated 32768 Hz
- This allows HOST PROCESSOR to use some of the counter registers as general purpose
- CORE protection logic which enables loose ISO assertion
- Functional lockout, unlock by special vbusp sequence
- DFT lockout, unlock by special 1500 sequence
- HOST PROCESSOR can issue OFF now cmd
- HOST PROCESSOR can update MMRs without polling, thanks to HW Shadow/Auto Sync
- HOST PROCESSOR can read time without polling, thanks to HW Shadow/Auto Sync
- External IOs *(Optional)*
  - 4 wakeup inputs with active high/low including optional debounce
    - Async wake up supported
  - 1 pmic_enable output
- Analog Interface *(Optional)*
  - 32 bit Analog Config output
  - ISO input
  - 32KHz input

12.7.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.7.3.2 RTC Integration

This device includes a Real-Time (RTC) module to allow easy tracking of time and date and the generation of real time alarms.

The following tables provide pin level detail for each of the interfaces on this module.

Table 12-349 describes the 1500 Interface for the DFT unlock circuit.

Table 12-349. 1500 Interface

Signal Name	Direction	Default Value	Description
jtag_wrck	In	1'd0	1500 wrapper clock
jtag_wrstn_n	In	1'd1	1500 wrapper reset (active low)
jtag_wsi	In	1'd0	1500 wrapper serial input
jtag_selectwir	In	1'd0	1500 select wrapper instruction register
jtag_shiftwr	In	1'd0	1500 shift wrapper register
jtag_updatewr	In	1'd0	1500 update wrapper register

Copyright © 2026 Texas Instruments Incorporated

Table 12-349. 1500 Interface (continued)

Signal Name	Direction	Default Value	Description
jtag_capturewr	In	1'd0	1500 capture wrapper register
jtag_wso	Out	1'd0	1500 wrapper serial out

Table 12-350 describes the PMIC Control External Wakeup and PMIC ON/OFF Control IOs.

Table 12-350. PMIC Control IOs

Signal Name	Direction	Default Value	Description
pmic_ext_wakeup0_io	In	1'd0	RTC Wakeup0
pmic_ext_wakeup1_io	In	1'd0	RTC Wakeup1
pmic_ext_wakeup2_io	In	1'd0	RTC Wakeup2
pmic_ext_wakeup3_io	In	1'd0	RTC Wakeup3
pmic_pmic_enable_io	Out	1'd1	RTC PMIC Enable 1 = ON 0 = OFF

Table 12-351. Analog Interface

Signal Name	Direction	Default Value	Description
ana_osc32k_clk	In	NA	Analog 32768 buffered Crystal clock If no analog, provide a near 32768 clock
ana_iso	In	1'd0	Analog ISO 1 = Isolated 0 = Not Isolated If no analog, tie 1'b0
ana_porz	In	NA	Analog PowerOnReset 1 = not asserted 0 = asserted This will assert when ON/BAT domain is powering up If no analog, then you can connect same reset which was used for rst_mod_g_rst_n You can also use por_rst_n class, it's a don't care
ana_cfg[31:0]	Out	TBD	Analog config If no analog, NC

Table 12-352. PSC Interface

Signal	Direction	Default Value	Description
pwri_clkstop_idle	Out	1'd1	The IP is in IDLE state, output of an IP
pwrs_clkstop_req	In	1'd0	Clock stop request
pwrs_clkstop_ack	Out	1'd0	Clock stop acknowledge

Copyright © 2026 Texas Instruments Incorporated

Table 12-352. PSC Interface (continued)

Signal	Direction	Default Value	Description
pwrw_clkstop_wakeup	Out	1'd0	Wakeup It requires clock stop protocol to be active and active vclk_clk
pwr_clk_clk_en	In	1'd1	Clock-gate enable to IP
pwr_clk_clk_en_ack	Out	pwr_clk_clk_en	Clock-gate enable acknowledge from IP

Table 12-353 describes the Main IP DFT Interface.

Table 12-353. IP DFT Interface

Signal Name	Direction	Default Value	Description
dft_partition_en	In	1'd0	dft_partition_en
dft_scan_en	In	1'd0	dft_scan_en
dft_clk_force_en	In	1'd0	dft_clk_force_en
dft_async_rst_sel	In	1'd0	dft_async_rst_sel
dft_test_async_rst_n	In	1'd1	dft_test_async_rst_n
dft_tft_mcp_dis	In	1'd0	Assert High to block MCP path
dft_local_clk_en	In	1'd0	dft_local_clk_en
dft_mode_atpg	In	1'd0	ATPG

The reset shown in Table 12-354 only goes to the core domain.

Table 12-354. Reset Interface

Signal Name	Direction	Default Value	Description
rst_mod_g_rst_n	In	1'd1	Module level main reset

Table 12-355 describes the Main on only VBUSP target port

Table 12-355. VBUSP Target Port

Signal Name	Direction	Default Value	Description
pr1_slv_req	In	1'b0	Request
pr1_slv_dir	In	1'b0	Direction
pr1_slv_address[31:0]	In	32'd0	Address
pr1_slv_xcnt[2:0]	In	3'd4	Good Byte Count
pr1_slv_byten[3:0]	In	{4{1'b1}}	Byte Enables
pr1_slv_wdata[31:0]	In	32'd0	Write Data
pr1_slv_wready	Out	1'b1	Write Ready
pr1_slv_rdatap[31:0]	Out	32'd0	Read Data (Pipelined)
pr1_slv_rready	Out	1'b1	Read Ready
pr1_slv_emudbg	In	1'b0	Emulation Debug Attribute

Copyright © 2026 Texas Instruments Incorporated

12.7.3.3 RTC Functional Description

12.7.3.3.1 RTC Block Diagram

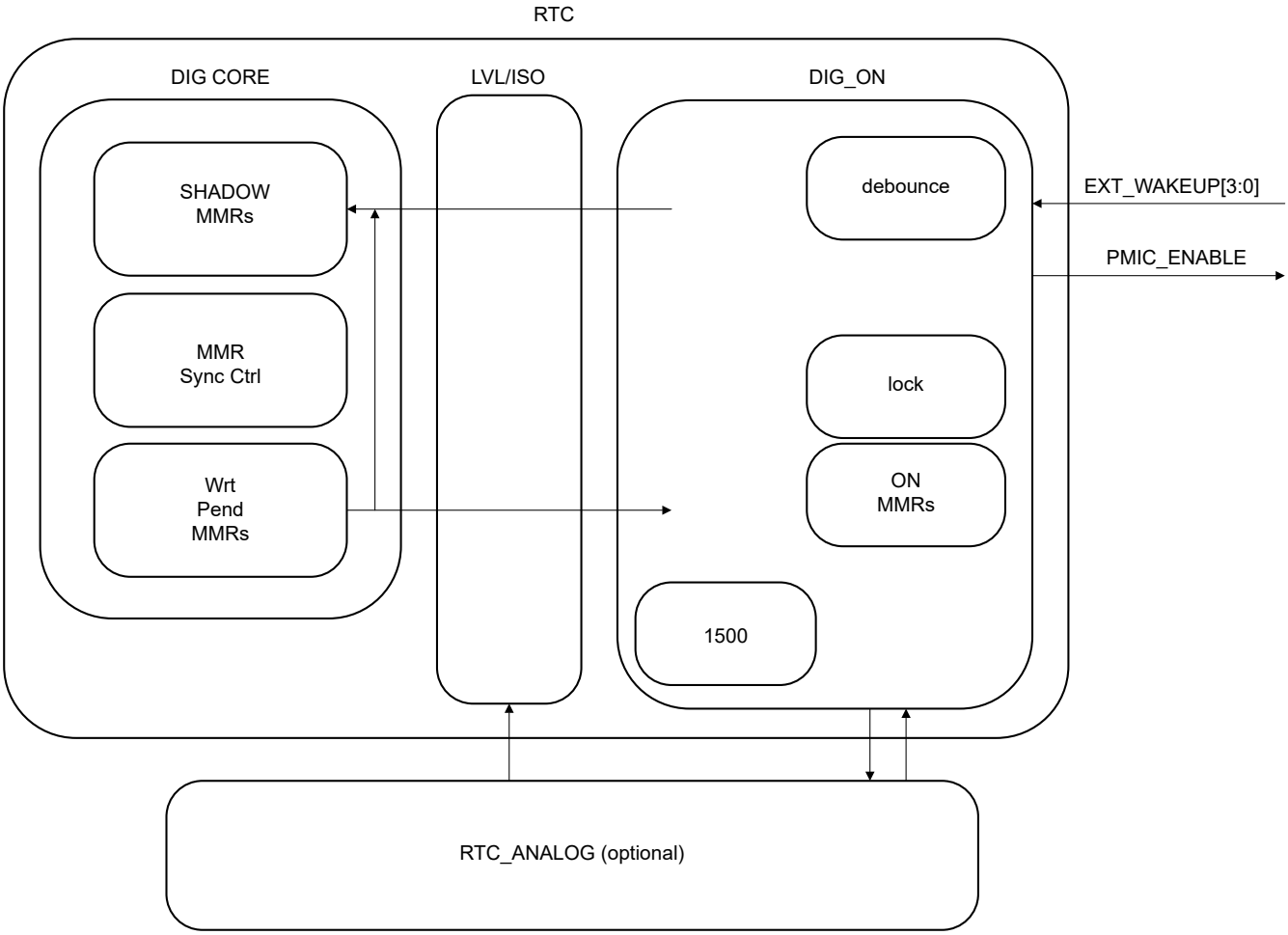


Figure 12-442. RTC Block Diagram

The RTC is composed of 3 major blocks, DIG_CORE, DIG_ON, and LVL_ISO.

12.7.3.3.1.1 DIG_CORE uARCH

The DIG_CORE block is on the CORE power domain which will power OFF/ON when the SOC powers OFF/ON. It contains, CORE IF logic, Shadow MMRs and MMR Sync logic.

The MMR Sync logic has 2 modes of operation

- Refresh/Sync after Core Reset
- Sync after HOST PROCESSOR Updates Shadow MMRs

Refresh/Sync after Core Reset Deassertion

After every Core Reset event, the MMR Sync logic we refresh ALL Shadow MMRs by copying the contents of ON MMRs into the Shadow MMRs.

Core Reset DeAssertion

The internal state machine Reads All MMRs in linear order

- If(GENRAL_CTL..32K_CLK_EN = 1)

Copyright © 2026 Texas Instruments Incorporated

  - Max delay of 2 32768Hz period delay
- If(GENRAL_CTL..32K_CLK_EN = 0)
  - Max delay 1.2uS = 32 * vbus_clk period delay(27MHz)
  - It will not wait for high to low 32768 event to read MMRs

The internal state machine Sets SYNCPEND.RD_DONE bit when the copy from the ON domain is completed. The host processor needs to wait for this status bit assertion before it reads the other MMRs

Sync after HOST PROCESSOR Updates Shadow MMRs

host processor UNLOCK

host processor Wrts to MMRs

The ON domain MMRs will get updated after a finite delay
- If(GENRAL_CTL..32K_CLK_EN = 1)
  - Max delay of 2 32768Hz period delay
- If(GENRAL_CTL..32K_CLK_EN = 0)
  - Max delay 8uS = 32 * vbus_clk period delay
  - It will not wait for high to low 32768 event to read MMRs

The internal state machine sets SYNCPEND.WR_DONE bit. This needs to get cleared before the next **HOST PROCESSOR Updates Shadow MMRs**

Note

The CORE domain has its own copy of the sub-second counter and second counter along with the compensation logic. The 2 domains will always be in sync after a reset or after host processor does a update to the counters or compensation

12.7.3.3.1.2 DIG_ON uARCH

The DIG_ON block is in the always ON domain, the power is provided by the ANALOG Block. It contains the always ON MMRs which will preserve state of the Shadow MMRs when the CORE is OFF state.

The 15 bit 32786Hz counter and the 48 bit second counter maintains count when the CORE is OFF. It has 2 comparators one for OFF_ON and one for ON_OFF event. It has External Wakeup events which can get de-bounced to control the PMIC_ENABLE. It controls the PMIC_ENABLE output which can change state based on ON_OFF, OFF_ON, and HOST PROCESSOR OFF now events.

The functional lockout and the dft lockout (1500) logic resides in this domain.

12.7.3.3.1.3 ISO_LVL uARCH

The ISO_LVL module constrains all DIG_CORE <-> DIG_ON Level/Isolation cells between these 2 domains. They use tibox to instantiate them. If No Analog support, then SOC integration team will not add in these RTC.

12.7.3.3.1.4 DIG_CORE to DIG_ON updates uARCH

Note

If No Analog support, then DIG_CORE and DIG_ON will be on a common V domain.

This section defines how the DIG_CORE Shadow MMRs maintain coherences with the DIG_ON Live MMR domain. All vbusp transactions terminate in the DIG_CORE block. All vbusp write transaction which require DIG_ON updates is completed by the Shadow to Live HW sequencer.

This HW sequencer has 2 modes of operation.
- Immediate Write Mode
  - When GENRAL_CTL.32K_OSC_DEP_EN = 0
  - It will start the write updates immediately

Copyright © 2026 Texas Instruments Incorporated

- Delayed Write Mode
  - When GENRAL_CTL.32K_OSC_DEP_EN = 1
  - It will start the write updates after the first 32khz_clk high to low transition

In Delayed Write Mode

For example, HOST PROCESSOR updates OFF_ON_S_CNT_LSW and OFF_ON_S_CNT_MSW.

After the HOST PROCESSOR completes the OFF_ON_S_CNT_MSW, the HW sequencer will start the process of updating the DIG_ON domain at the first available 32khz_clk is low cycle.

Note

32khz_clk is already low, the circuit will wait next low cycle to insure we have the full low period to do all pending updates. The HOST PROCESSOR can observe the SYNCPEND.WR_DONE bit, when set the write operations has completed to the DIG_ON domain

Note

If GENRAL_CTL..32K_CLK_EN = 0, then the HW sequencer will not wait for 32khz_clk high to low event, since this will force 32khz_clk low. This removes the dependency of an active 32768Hz clock

ALL MMR write transaction require full 4 Bytes writes in order to reduce the logic size of the DIG_ON domain.

RULE: Require 4 Bytes writes (less logic and iso)

Note

ON domain MMRs have a write lock protection circuit to prevent corruption of the DIG_ON domain during a power down event since ISO assertion can be late.

Due to this fact, the HOST PROCESSOR must first unlock write access by performing the unlock sequence, by simply writing Kick0 and Kick1 with the correct pattern and sequence. After this is done, the RTC is in an unlock date. The HOST PROCESSOR will need to relocking by writing any pattern to Kick1

12.7.3.3.2 CPU Interrupt Support

12.7.3.3.2.1 CPU Interrupts

The RTC can produce one HW event output active high level and active high pulse. This one output will assert on 3 internal HW events.

1. Timer ON_OFF event
2. Timer OFF_ON event
3. External Wakeup event

HOST PROCESSOR can read the Interrupt MMR status to determine which events caused the RTC event output to assert. vbus_clk can be on or off, the active high level will assert on either scenario. The 32768Hz clock must be active to cause a ON_OFF or OFF_ON event. The External Wakeup event can occur without 32768Hz clock if debounce is not enabled.

The Interrupt MMR programing model uses the standard Keystone3 schema which allows HOST PROCESSOR to issue debug event. Also, the enabling of the events are backed up in the ON domain, so HOST PROCESSOR will not enable to enable events on every power cycle of the CORE domain.

12.7.3.3.3 Programming Usage Guide

12.7.3.3.3.1 No Analog Support

DIG_ON domain MMR Write Rules

- UnLock DIG_CORE (Wrt Kick0 and Wrt Kick1)

Copyright © 2026 Texas Instruments Incorporated

  - Keep it UnLock
- Write all MMRs in the same address order has MMRs from low to high
  - For example . Wrt Scratch Storage 0 through Scratch Storage 7
- HOST PROCESSOR poles until SYNCPEND.WR_PEND is 0
  - Max completion is 60 uSeconds when GENRAL_CTL. 32K_OSC_DEP_EN is set
  - Max completion is 10 uSeconds when GENRAL_CTL. 32K_OSC_DEP_EN is clr
- HOST PROCESSOR is not allowed to perform new write without waiting for clearing of SYNCPEND.WR_PEND

After every new power up of the SOC

- Unlock RTC by writing correct pattern in Kick0 and Kick1
- Keep it UnLock
- HOST PROCESSOR can verify that 32Khz_osc_clk is toggling by reading SYNCPEND. 32K_CLK_OBS
- Should not be required, since the 32Khz_osc_clk will always will be active
- HOST PROCESSOR sets GENRAL_CTL.32K_OSC_DEP_EN
- Wait for 60 uSec
- Initialize ALL RTC MMRs in the DIG_ON domain in the same order as the address map
- The RTCs MMRs do not have a reset state in general, only a few required MMRs bits have one.
- HOST PROCESSOR must initialize all bits of the MMRs which reside in DIG_ON domain
  - Ie wrt <value>@0x00, <value>@0x04, <value>@0x08 ….<value>@0x50
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null
  - This means all DIG_ON domain MMRs have been written to

RTC is now in an operational state and fully configured.

32768Hz time adjustment

- HOST PROCESSOR UnLock RTC(required for writes to DIG_ON domain)
- HOST PROCESSOR can readSUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- HOST PROCESSOR can write new time by writing to SUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null

32768Hz time drift adjustment

- HOST PROCESSOR UnLocks RTC(required for writes to DIG_ON domain)
- HOST PROCESSOR can readSUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- SW determines drift rate
- HOST PROCESSOR updates or enables drift adjustment by HOST PROCESSOR write new COMP_LSB and COMP_MSB
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null

12.7.3.3.3.2 With Analog Support

DIG_ON domain MMR Write Rules

- UnLock DIG_CORE (Wrt Kick0 and Wrt Kick1)
- Write all MMRs in the same address order has MMRs from low to high
- For example . Wrt Scratch Storage 0 through Scratch Storage 7
- ReLock DIG_CORE (Wrt 0x0000_0000 to Kick1)
- HOST PROCESSOR poles until SYNCPEND.WR_PEND is 0
  - Max completion is 60 uSeconds when GENRAL_CTL. 32K_OSC_DEP_EN is set
  - Max completion is 10 uSeconds when GENRAL_CTL. 32K_OSC_DEP_EN is clr
- HOST PROCESSOR is not allowed to perform new write without waiting for clearing of SYNCPEND.WR_PEND

First Time Powered ON(DIG_ON)

Copyright © 2026 Texas Instruments Incorporated

- Unlock RTC by writing correct pattern in Kick0 and Kick1
- HOST PROCESSOR updates ANALOG_CTRL MMR
  - Set the correct parameters
  - Enable 32768Hz OSC
  - Wait for 3 seconds to allow the 32768Hz OSC to lock
- HOST PROCESSOR can verify that 32Khz_osc_clk is toggling by reading SYNCPEND. 32K_CLK_OBS
- HOST PROCESSOR sets GENRAL_CTL.32K_OSC_DEP_EN
- HOST PROCESSOR ReLock DIG_CORE (Wrt 0x0000_0000 to Kick1)

Initialize ALL RTC MMRs in the DIG_ON domain in the same order as the address map except for the first 2 writes which should always be to the 2 KICK0/KICK1 MMRs to unlock write transaction and the last write which should relock by writing 0x0 to KICK1

- The RTCs MMRs do not have a reset state in general, only a few required MMRs bits have one.
- HOST PROCESSOR must initialize all bits of the MMRs which reside in DIG_ON domain
- HOST PROCESSOR UnLock
- Ie wrt <value>@0x00, <value>@0x04, <value>@0x08 ….<value>@0x50
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null
- This means all DIG_ON domain MMRs have been written to

RTC is now in an operational state and fully configured

32768Hz time adjustment

- HOST PROCESSOR UnLock RTC(required for writes to DIG_ON domain)
- HOST PROCESSOR can readSUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- HOST PROCESSOR can write new time by writing to SUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null

32768Hz time drift adjustment

- HOST PROCESSOR UnLocks RTC(required for writes to DIG_ON domain)
- HOST PROCESSOR can readSUB_S_CNT(1st), S_CNT_LSW(2nd), S_CNT_MSW(3rd)
- SW determines drift rate
- HOST PROCESSOR updates or enables drift adjustment by
  - HOST PROCESSOR write new COMP_LSB and COMP_MSB
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null

Update OFF_ON and/or ON_OFF times

- HOST PROCESSOR poll SYNCPEND.WR_PEND until null (ACTION PRE vs POST)
- HOST PROCESSOR UnLocks RTC(required for writes to DIG_ON domain)
- HOST PROCESSOR update OFF_ON_S_CNT_LSW/MSW and/or ON_FF_S_CNT_LSW/MSW
- HOST PROCESSOR can update SCRATCH0/7
- HOST PROCESSOR update GENRAL_CTL to enable if required
- HOST PROCESSOR ReLock RTC(writing 0x0000_0000 to KICK1)
- HOST PROCESSOR poll SYNCPEND.WR_PEND until null

Note

The new OFF time must be at least 2 seconds in the future

OFF NOW

- HOST PROCESSOR UnLocks RTC(required for writes to DIG_ON domain)
- Optional (HOST PROCESSOR can update SCRATCH0/7)

Copyright © 2026 Texas Instruments Incorporated

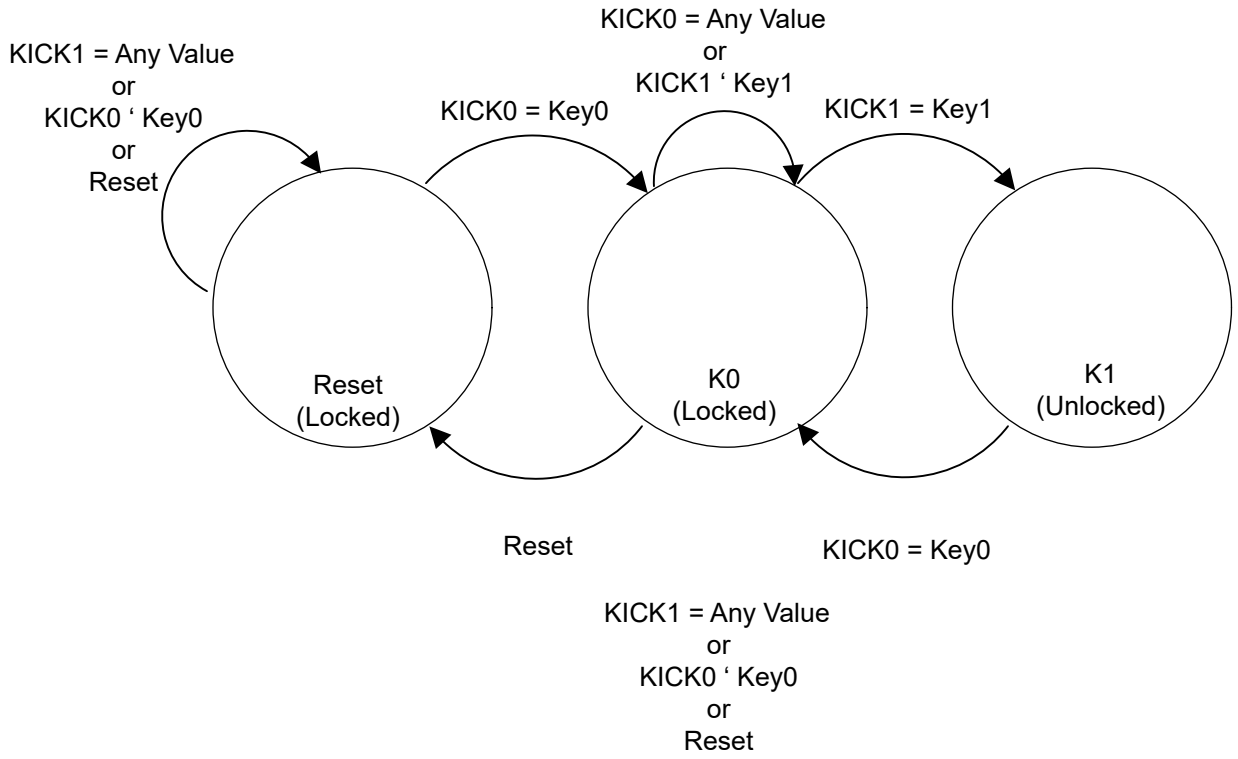
- HOST PROCESSOR issues a SW_OFF in GENRAL_CTL
- HOST PROCESSOR does a dead loop, wait for power OFF

Note

Note any Write to SW_OFF will cause a Auto ReLock

12.7.3.3.3.2.1 MMR Spurious WRT Protection

The module also contains a kicker mechanism (Figure 12-443) to prevent any spurious writes from changing the register values. This mechanism requires two MMR writes to the Kick0 and Kick1 registers with exact data values before the kicker lock mechanism is released. Once released, the MMRs are writeable. The Kick0 data is 83E7 0B13h; the Kick1 data is 95A4 F1E0h. Note that it remains in an unlocked state until an OCP reset or invalid data pattern is written to one of the Kick0 or Kick1 registers.



- Key0 = 83E7 0B13h
- Key1 = 95A4 F1E0h
- Locked = Write protection enabled
- Unlocked = Write protection disabled

Figure 12-443. Kick Register State Machine Diagram

- S0 is the Reset/Idle state
- S1 is an wrt cycle of 83E7 0B13h at Kick0 completed state
- S2 is the UNLOCK MMR WRT state
- S0 → S1 when wrt cycle of 83E7 0B13h at Kick0
- S1 → S2 when wrt cycle of 95A4 F1E0h at Kick1
- S1 → S0 when reset event
- S2 → S0 when reset event OR OCP wrt cycle of NOT 83E7 0B13h at Kick0 OR OCP wrt cycle at Kick1
- S2 → S1 when wrt cycle of 83E7 0B13h at Kick0

Copyright © 2026 Texas Instruments Incorporated

12.7.3.3.3.2.2 Crystal Compensation

In order to compensate for any inaccuracy of the 32768HZ oscillator, the HOST can perform a calibration of the oscillator frequency, calculate the drift compensation versus one-hour period and load the compensation registers with the drift compensation value. If the COMP_REG value is positive, compensation occurs after the second change event. COMP_REG cycles are removed from the next second. If the COMP_REG value is negative, compensation occurs before the second change event. -COMP_REG cycles are added to the current second. This enables to compensate with a 132 kHz period accuracy each 4096 Seconds. The waveform below summarizes positive and negative compensation effect.

Copyright © 2026 Texas Instruments Incorporated

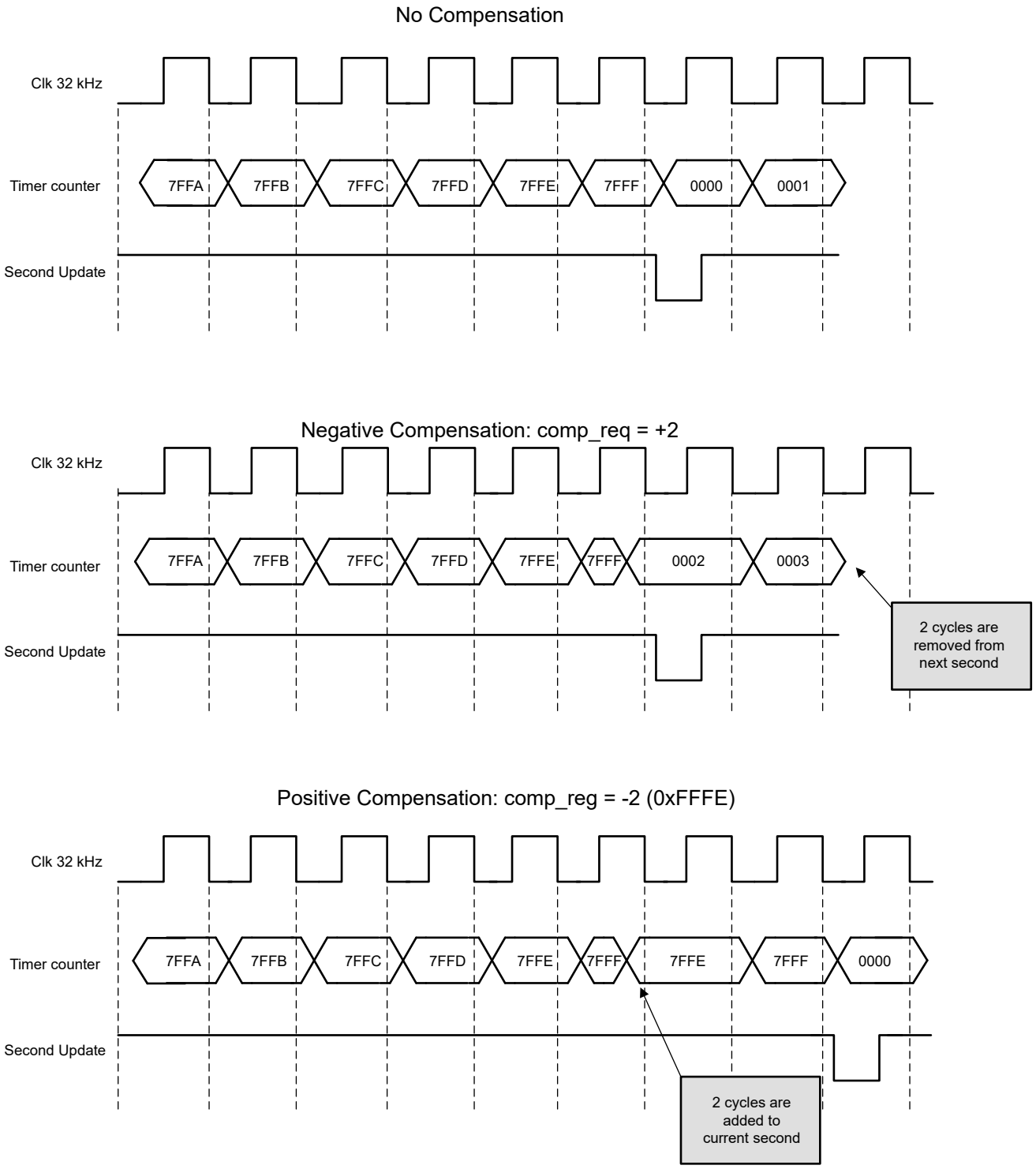


Figure 12-444. Compensation Illustration

12.7.3.3.4 Scratch Registers

The RTC provides eight general-purpose registers (SCRATCHx_REG) that can be used to store 32-bit words -- these registers have no functional purpose for the RTC. Software using the RTC may find the SCRATCHx registers to be useful in indicating RTC states. For example, the SCRATCHx_REG registers may be used to indicate write-protection lock status or unintentional power downs. To indicate write-protection, the software

Copyright © 2026 Texas Instruments Incorporated

should write a unique value to one of the SCRATCHx_REG registers when write-protection is disabled and another unique value when write-protection is enabled again. In this way, the lock-status of the registers can be determined quickly by reading the SCRATCH register. To indicate unintentional power downs, the software should write a unique value to one of the SCRATCHx_REG registers when RTC is configured and enabled. If the RTC is unintentionally powered down, the value written to the SCRATCH register is cleared. For more information, *see RTCSS Registers*.

12.7.3.3.5 KS3 Clock Stop Protocol

The IP is complaint to KS3 Clock Stop Protocol, please refer to its spec.

For this IP

- pwri_clkstop_idle = !(RD_PEND or WR_PEND or EVENT_OFF_ON or EVENT_ON_OFF)
- If any of these MMRs are set, pwri_clkstop_idle = 0 or busy
- pwrs_clkstop_ack will not assert per spec unless pwrs_clkstop_req == 1 and pwri_clkstop_idle == 1
- pwrw_clkstop_wakeup will not assert unless pwrs_clkstop_ack == 1 AND vclk_clk is active

  It will assert rtc_event_pend_intr asserts
- The integration guideline is to always use rtc_event_pend_intr for wakeup event since it has NO dependencies on clock stop protocol nor active vclk_clk is required, just 32k_clk needs to be active

Copyright © 2026 Texas Instruments Incorporated

12.7.4 Timers

This section describes the Timer modules for the device.

12.7.4.1 Timers Overview

All timers include specific functions to generate accurate tick interrupts to the operating system.

Each timer can be clocked from several different independent clocks. The selection of clock source is made from the TIMERn_CLKSEL registers, where n is in the instance .

In the MCU domain the device provides up to 4 timer pins, one for each instance, to be used as MCU Timer Capture inputs or as MCU Timer PWM outputs. (See *Module Integration* and the device specific datasheet to see if MCU Timers apply)

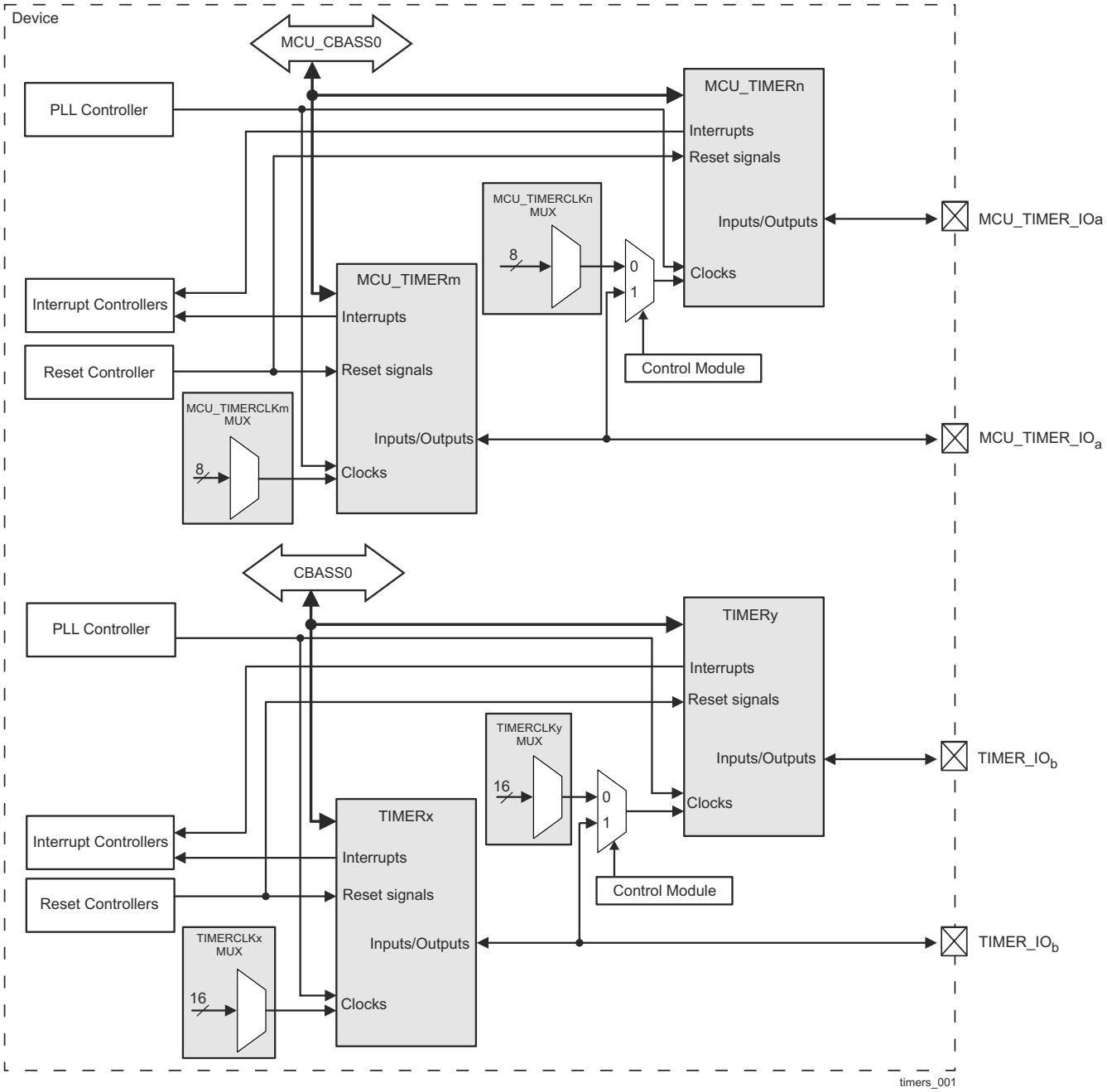
In the MAIN domain the device provides up to 11 timer pins, one for each instance, to be used as Timer Capture inputs or as Timer PWM outputs. (See device specific datasheet to see details on timer pins.)

Each odd numbered timer instance from each of the domains may be optionally cascaded with the previous even numbered timer instance from the same domain to form up to a 64-bit timer. For example, TIMER1 may be cascaded to TIMER0, MCU_TIMER1 may be cascaded to MCU_TIMER0, etc.

When cascaded, TIMERi acts as a 32-bit prescaler to TIMERi+1, as well as MCU_TIMERn acts as a 32-bit prescaler to MCU_TIMERn+1. TIMERi / MCU_TIMERn must be configured to generate a PWM output edge at the desired rate to increment the TIMERi+1/ MCU_TIMERn+1 counter.

Figure 12-445 is an overview of the timers.

Copyright © 2026 Texas Instruments Incorporated



m = even instances of MCU Timer. See the device datasheet for valid domains and instances.
x = even instances of Timer. See the device datasheet for valid domains and instances

Note

n = odd instances of MCU Timer. See the device datasheet for valid domains and instances.

Note

y = odd instances of Timer. See the device datasheet for valid domains and instances

Figure 12-445. Timers Overview

Copyright © 2026 Texas Instruments Incorporated

12.7.4.1.1 Timers Features

The following are the main features of the timer controllers:

- Target interface supports:
  - 32-bit data bus width
  - 32-bit access supported
  - 10-bit address bus width
  - Write nonposted transaction mode supported
- Interrupts generated on overflow, compare, and capture
- Free-running 32-bit upward counter
- Compare and capture modes
- Autoreload mode
- Start/stop mode
- Programmable divider clock source (2^n, where n = [0-8])
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- On-the-fly read/write register (while counting)
- Generates a 1-ms tick clock with a 32.768 kHz functional clock sourced from the LFOSC

12.7.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.7.4.2 Timers Environment

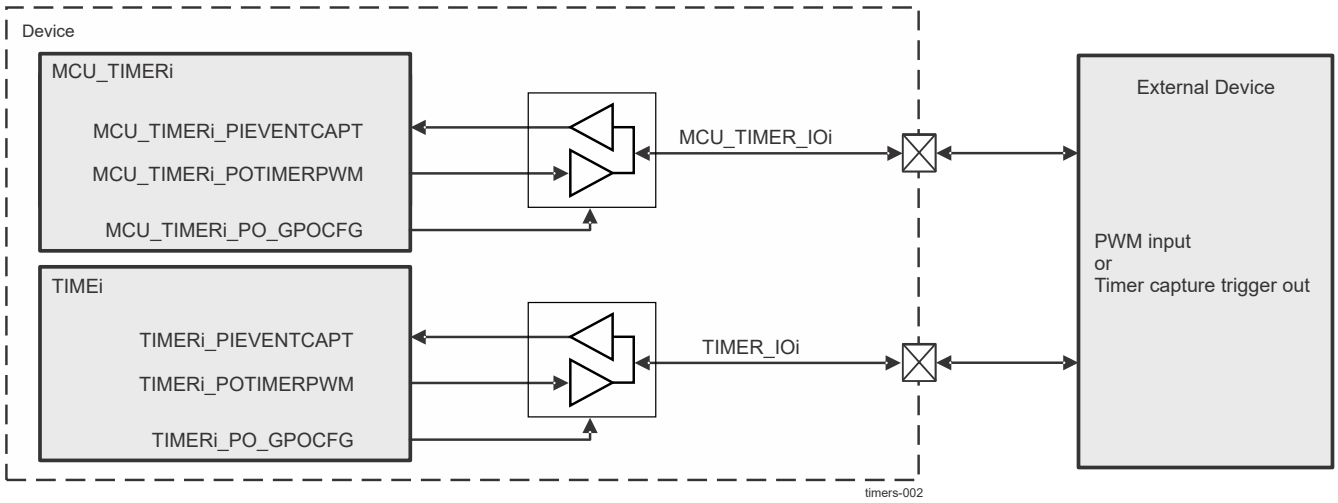
The MCU_TIMER[3-0] and TIMER[11-0] modules are hereinafter referred to as timer module.

This section describes the timers external connections (environment).

12.7.4.2.1 Timer External System Interface

Each timer can send or receive stimulus to/from the external (off-chip) system. In the device all timers are configured to output a PWM pulse or receive an external event signal used as a trigger to capture the current timer count.

Figure 12-446 shows the external system interface for the timers, and Table 12-356 describes the timer inputs and outputs.



Note

i represents a TIMER instance. See the device datasheet for available domains and TIMER instances.

Figure 12-446. Timers External System Interface

Table 12-356. Timer I/O Signals

Module Pin	Device Level Signal	I/O[1]	Description	Module Pin Reset Value[2]
		MCU_TIMERi[2]		
MCU_TIMER_IOi[2]	MCU_TIMERi[2]_PIEVENTCAPT MCU_TIMERi[2]_POTIMERPWM	I/O	MCU_TIMERi[2] trigger input or MCU_TIMERi[2] output	0
		TIMERi[2]		
TIMER_IOi[2]	TIMERi[2]_PIEVENTCAPT TIMERi[2]_POTIMERPWM	I/O	TIMERi[2] trigger input or TIMERi[2] PWM output	0

(1) When configured for that function; I = Input, O = Output
(2) i represents a TIMER instance. See the device datasheet for available domains and TIMER instances.

Note

Each MCU_TIMER_PO_GPOCFG and TIMER_PO_GPOCFG signals are used as an output enable to control the function of the MCU_TIMER_IO[3-0], and respectively TIMER_IO[11-0], as the PWM output (PO_GPOCFG = 0) or capture input (PO_GPOCFG = 1).

Copyright © 2026 Texas Instruments Incorporated

Note

For more information about device level signals (pull-up/down resistors, buffer type, and others), see tables *Pin Attributes* in the device-specific Datasheet.

12.7.4.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.7.4.4 Timers Functional Description

Each timer contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on-the-fly (while counting). Each timer includes compare logic to allow an interrupt event on a programmable counter matching value. A dedicated output signal can be pulsed or toggled on either an overflow or a match event. This offers time-stamp trigger signaling or PWM signal sources. A dedicated input signal can be used to trigger an automatic timer counter capture or an interrupt event on a programmable input signal transition. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged into one module interrupt line and one wake-up line.

Each internal interrupt source can be independently enabled and disabled by a dedicated bit in the DMTIMER1MS_IRQSTATUS_SET and DMTIMER1MS_IRQSTATUS_CLR register for the interrupt features, and a dedicated bit of the DMTIMER1MS_IRQWAKEEN register for a wake-up. In addition, timers have a mechanism implemented to generate an accurate tick interrupt.

For each timer implemented in the device, there are two possible clock sources:

- 32-kHz clock (sourced from the LFOSC)
- System clock

For more information of the selection of the input clock source, see *Clocking*.

Note

Application software must ensure that DMTIMER bus interface clock frequency be greater than input clock frequency before accessing DMTIMER registers. Failure to do so will result in bus hang condition.

Each timer supports three functional modes:

- Timer mode
- Capture mode
- Compare mode

The capture and compare modes are disabled by default after core reset.

12.7.4.4.1 Timer Block Diagram

Figure 12-447 is a block diagram of the timers.

Copyright © 2026 Texas Instruments Incorporated

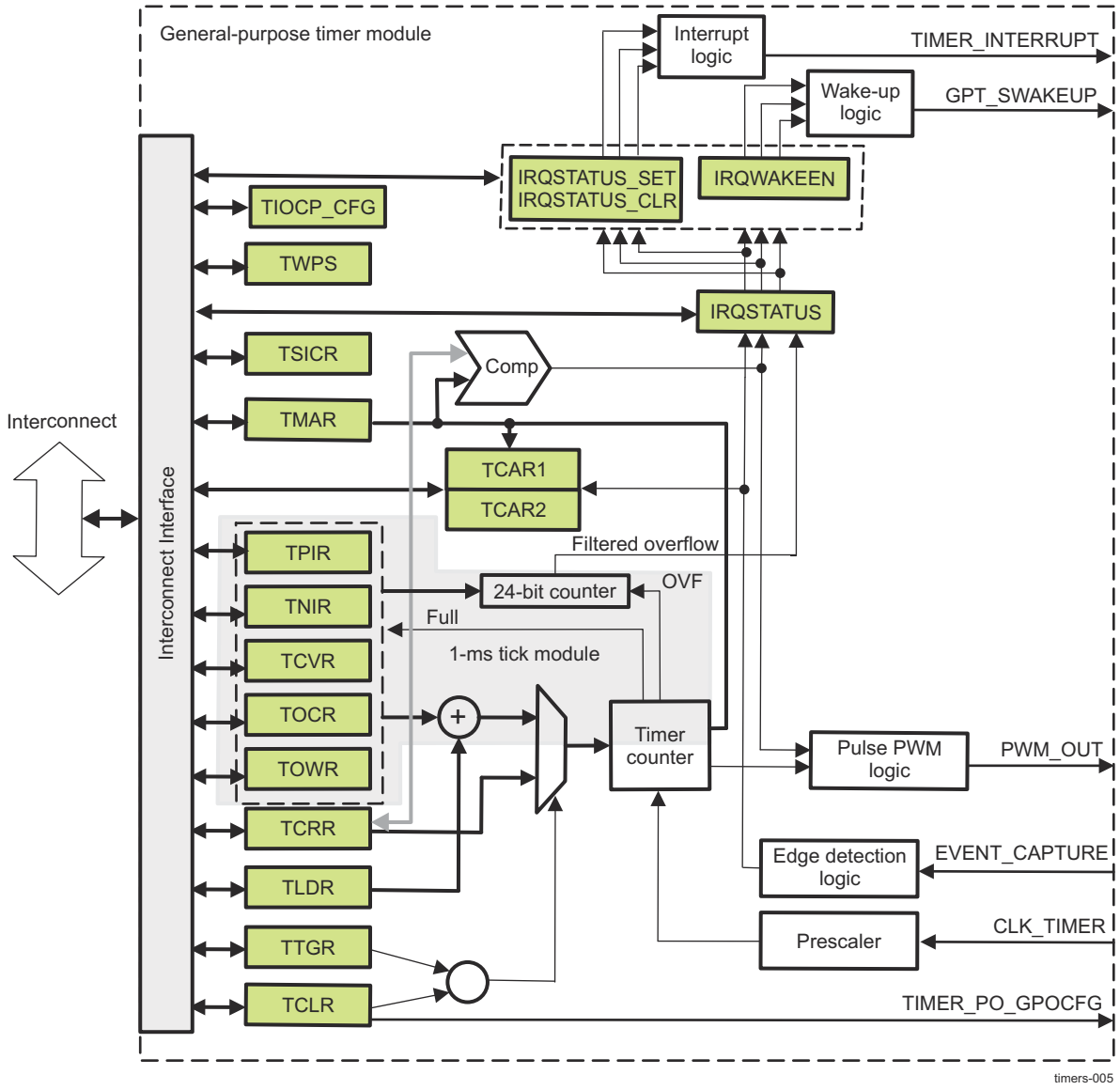


Figure 12-447. Timer Block Diagram

12.7.4.4.2 Timer Power Management

Note

Some of the timers features described in this section may not be supported on this family of devices. For more information, see *Module Integration* for specifics..

The way the timer acknowledges the LPSC clock stop request is configurable through the DMTIMER1MS_TIOCP_CFG[3-2] IDLEMODE bit field.

Table 12-357 lists the IDLEMODE settings and the related acknowledgment modes.

Table 12-357. IDLEMODE Settings

IDLEMODE Value	Selected Mode	Description
00	Force-idle	The clock stop request is unconditionally acknowledged from the timer, regardless of its internal operations. This mode must be used carefully, because it does not prevent the loss of data when the clock is switched off.

Copyright © 2026 Texas Instruments Incorporated

Table 12-357. IDLEMODE Settings (continued)

IDLEMODE Value	Selected Mode	Description
01	No-idle	The clock stop request is never acknowledged from the timer. This mode is safe from a module point of view but is not efficient from a power-saving perspective because the clocks remain active.
10	Smart-idle	The timer acknowledges the clock stop request, basing its decision on its internal activity. The acknowledge signal is asserted only when all pending transactions and interrupt requests are treated. This is the best approach to efficient system power management.
11	Smart-idleWakeup	The module behaves like in Smart-idle mode, with the exception, that it can issue a wake-up request in sleep mode, if the functional clock is not cut off.

12.7.4.4.2.1 Wake-Up Capability

Note

Some of the timers features described in this section may not be supported on this family of devices. For more information, see *Timers Not Supported Features*.

If the DMTIMER1MS_TIOCP_CFG[3-2] IDLEMODE bit field sets the smart-idle mode or smart-idle with wake-up mode, the timer evaluates its internal capability to have the interface clock switched off. When there is no further internal activity (no pending interrupt sources: match, overflow, or timer capture events), the clock stop acknowledge signal is asserted and the timer enters sleep mode, ready to issue a wake-up request if configured in smart-idle with wake-up mode.

Figure 12-448 shows the wake-up request generation.

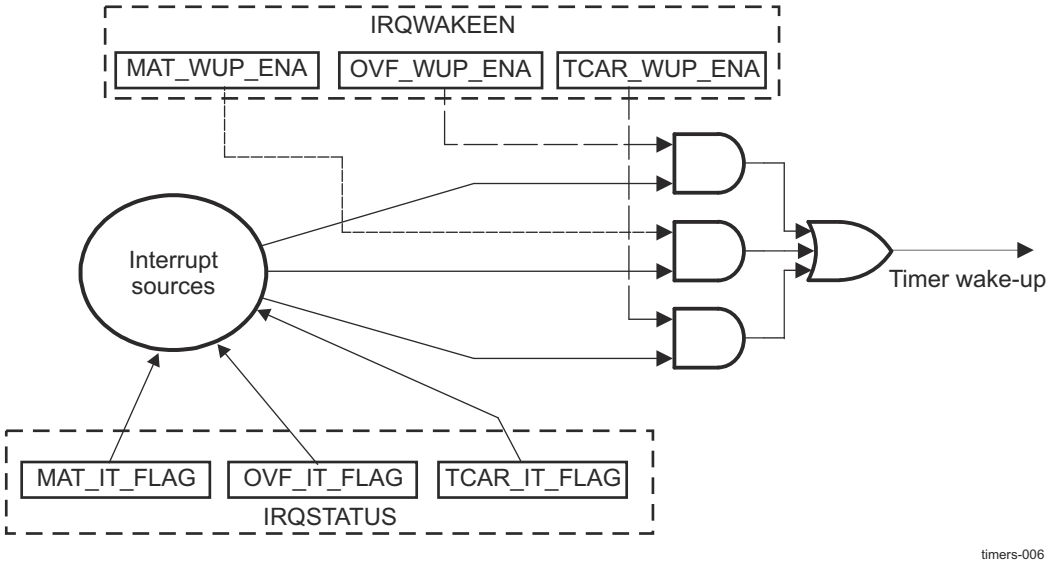


Figure 12-448. Wake-Up Request Generation

The timer wake-up-enable register allows masking of the expected source of the wake-up event that generates a wake-up request. The register is synchronously programmed with the interface clock before the Clock magagemet sends a clock stop request. The expected source of the wake-up event is an overflow (DMTIMER1MS_TCRR), a timer match (the compare result of DMTIMER1MS_TCRR and DMTIMER1MS_TMAR matches the counter value), and a timer capture (detection of an external pulse transition of the correct polarity on the TIMER_PIEVENTCAPT).

When the wake-up event is issued, the associated interrupt status bit is set in the timer status register (DMTIMER1MS_IRQSTATUS). The pending wake-up event is reset when the set status bit is overwritten with 1.

Copyright © 2026 Texas Instruments Incorporated

Note

The status bit must be reset to re-enter idle mode.

12.7.4.4.3 Timer Software Reset

DMTIMER1MS_TIOCP_CFG[0] SOFTRESET bit can initiate a software reset of the timer. This bit is autocleared to 0 when the reset is complete.

Before accessing or using the timer, the local host must ensure that internal reset is released by reading the DMTIMER1MS_TIOCP_CFG[0] SOFTRESET bit. This bit monitors the internal reset status.

12.7.4.4.4 Timer Interrupts

The timer can issue an overflow interrupt, a timer match interrupt, and a timer capture interrupt. Each internal interrupt source can be independently enabled and disabled in the interrupt-enable register (DMTIMER1MS_IRQSTATUS_SET) and disabled in the interrupt-disable register (DMTIMER1MS_IRQSTATUS_CLR). When the interrupt event is issued, the associated interrupt status bit is set in the timer status register (DMTIMER1MS_IRQSTATUS).

12.7.4.4.5 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the timer control register (the DMTIMER1MS_TCLR[0] ST bit). The timer counter register (DMTIMER1MS_TCRR) can be loaded when stopped or on-the-fly (while counting). DMTIMER1MS_TCRR can be loaded directly by a DMTIMER1MS_TCRR write access with a new timer value. DMTIMER1MS_TCRR can also be loaded with the value held in the timer load register (DMTIMER1MS_TLDR) by a trigger register (DMTIMER1MS_TTGR) write access. The loading of DMTIMER1MS_TCRR is done regardless of the written value of DMTIMER1MS_TTGR. The value of DMTIMER1MS_TCRR can be read when stopped or captured on-the-fly by a DMTIMER1MS_TCRR read access. The timer is stopped and the counter value is set to 0 when the module reset is asserted. The timer is maintained at stop after the reset is released.

In one-shot mode (the DMTIMER1MS_TCLR[1] AR bit is set to 0), the counter is stopped after counting overflow occurs (the counter value remains at 0).

When the autoreload mode is enabled (the DMTIMER1MS_TCLR[1] AR bit is set to 1), DMTIMER1MS_TCRR is reloaded with the value of DMTIMER1MS_TLDR after a counting overflow occurs.

CAUTION

Do not put the overflow value (0xFFFF FFFF) in the DMTIMER1MS_TLDR register because it can lead to undesirable results.

An interrupt can be issued on overflow if the overflow interrupt-enable bit is set in the timer interrupt-enable register (the DMTIMER1MS_IRQSTATUS_SET[1] OVF_EN_FLAG bit is set to 1). A dedicated output pin (POTIMERPWM) can be programmed in the DMTIMER1MS_TCLR[12] PT bit through the DMTIMER1MS_TCLR[11-10] (PT and TRG bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs. The DMTIMER1MS_TCLR[12] PT bit selects pulse/toggle modulation (the DMTIMER1MS_TCLR[11-10] TRG bit field selects trigger mode).

Figure 12-449 shows the DMTIMER1MS_TCRR timing value.

Copyright © 2026 Texas Instruments Incorporated

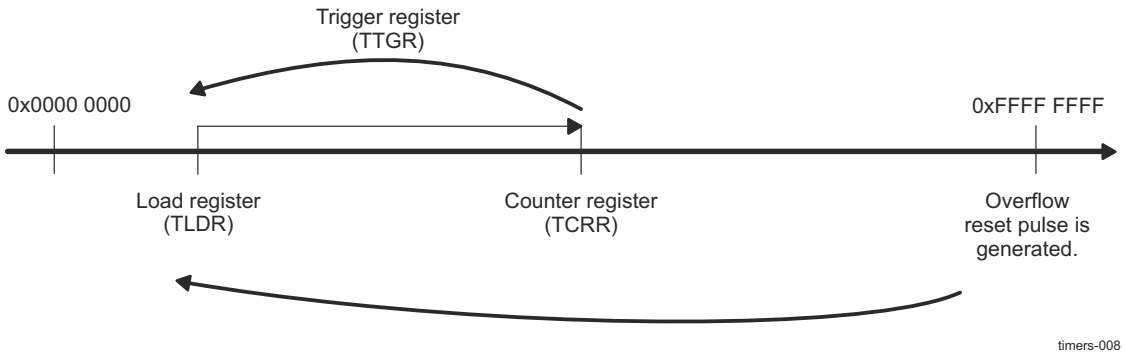


Figure 12-449. DMTIMER1MS_TCRR Timing Value

12.7.4.4.5.1 1-ms Tick Generation

The interrupt period is not exactly 1 ms, because the timer input clock is 32.768 kHz. If the clock counts up to 32, it obtains a 0.977-ms period; if it counts up to 33, it obtains a 1.007-ms period. For large granularity, the error is cumulative and can generate important deviations from the standard value.

To minimize the error between a true 1-ms tick and the tick generated by the 32.768-kHz timer, the sequencing of periods less than 1 ms and periods greater than 1 ms must be shuffled. An additional 1-ms block is used to correct this error. See Figure 12-450.

Note

The only available source for a 32.768-kHz clock is the LFOSC. The HFOSC is not capable of generating a true 32.788-kHz frequency.

In this implementation, the increment sequencing is automatically managed by the timer to minimize the error. The user must define only the value of the timer positive increment register (the DMTIMER1MS_TPIR[31-0] POSITIVE_INC_VALUE bit field) and the timer negative increment register (the DMTIMER1MS_TNIR[31-0] NEGATIVE_INC_VALUE bit field). An automatic adaptation mechanism is used to simplify the programming model.

Copyright © 2026 Texas Instruments Incorporated

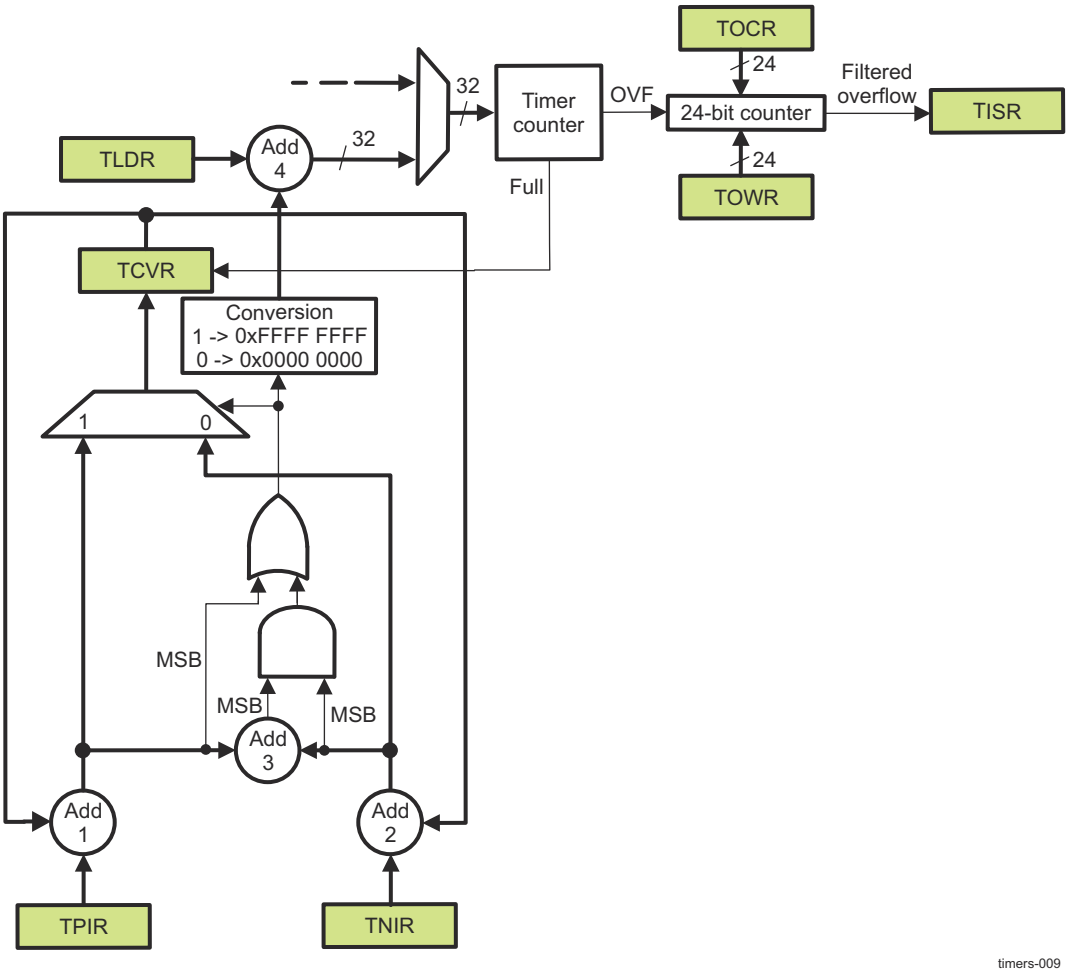


Figure 12-450. Block Diagram of the 1-ms Tick Module

The DMTIMER1MS_TPIR, DMTIMER1MS_TNIR, and DMTIMER1MS_TCVR registers and adders Add1, Add2, and Add3 are used to define whether the next value loaded in the timer counter register (the DMTIMER1MS_TCRR[31-0] TIMER_COUNTER bit field) is the value of the DMTIMER1MS_TLDR[31-0] LOAD_VALUE bit field (period less than 1 ms) or the value of DMTIMER1MS_TLDR[31-0] LOAD_VALUE –1 (period greater than 1 ms).

Table 12-358 lists the value loaded in the DMTIMER1MS_TCRR according to the sign of the result of Add1, Add2, and Add3.

MSB = 0: Positive value; MSB = 1: Negative value

Table 12-358. Value Loaded in DMTIMER1MS_TCRR to Generate 1-ms Tick

Add1 MSB	Add2 MSB	Add3 MSB	Value of DMTIMER1MS_TCRR Register
0	0	0	DMTIMER1MS_TLDR[31-0] LOAD_VALUE bit field
0	0	1	DMTIMER1MS_TLDR[31-0] LOAD_VALUE bit field
0	1	0	DMTIMER1MS_TLDR[31-0] LOAD_VALUE bit field
0	1	1	DMTIMER1MS_TLDR[31-0] LOAD_VALUE –1
1	0	0	N/A
1	0	1	N/A
1	1	0	DMTIMER1MS_TLDR[31-0] LOAD_VALUE –1
1	1	1	DMTIMER1MS_TLDR[31-0] LOAD_VALUE –1

Copyright © 2026 Texas Instruments Incorporated

The values of the DMTIMER1MS_TPIR and DMTIMER1MS_TNIR registers are calculated using the following formulas:

- Positive increment value = (INTEGER[F_{clk} × T_{tick}] + 1) × 1e6) – (F_{clk} × T_{tick} × 1e6)
- Negative increment value = (INTEGER[F_{clk} × T_{tick}] × 1e6) – (F_{clk} × T_{tick} × 1e6)

Note

F_{clk} clock frequency (kHz)

T_{tick} tick period (ms)

The timer overflow counter register (DMTIMER1MS_TOCR) and the timer overflow wrapping register (DMTIMER1MS_TOWR) are used to filter interrupts. When the timer overflows, it increments the 24-bit DMTIMER1MS_TOCR. When the values in the 24-bit DMTIMER1MS_TOCR match the values in the 24-bit DMTIMER1MS_TOWR and the timer overflow is asserted, the DMTIMER1MS_TOCR is reset and an interrupt is generated to the DMTIMER1MS_IRQSTATUS register.

Note

DMTIMER1MS_TOWR has to be set to requested value. For example, if no interrupt needs to be masked DMTIMER1MS_TOWR must be set to 0, if one interrupt needs to be masked DMTIMER1MS_TOWR must be set to 1, if two interrupts need to be masked DMTIMER1MS_TOWR must be set to 2 and so on.

It is important to have in mind that the case when FFFFFF interrupts need to be masked is not possible.

With the conversion block in reset state (the positive increment register, negative increment register, and counter value register are zeroed), the programming model and the behavior of timers remain unchanged.

For 1-ms tick with a 32.768-kHz clock:

- DMTIMER1MS_TPIR[31-0] POSITIVE_INC_VALUE = 232,000
- DMTIMER1MS_TNIR[31-0] NEGATIVE_INC_VALUE = –768,000
- DMTIMER1MS_TLDR[31-0] LOAD_VALUE = 0xFFFF FFE0

Note

Any value of the tick period can be generated with the appropriate value of the DMTIMER1MS_TPIR, DMTIMER1MS_TNIR, and DMTIMER1MS_TLDR.

By default, the DMTIMER1MS_TPIR, DMTIMER1MS_TNIR, DMTIMER1MS_TCVR, DMTIMER1MS_TOCR, and DMTIMER1MS_TOWR and the associated logic are in reset mode (all 0s) and have no effect on the programming model.

12.7.4.4.6 Timer Capture Mode Functionality

When a transition is detected on the module input pin (PIEVENTCAPT), the timer value in the DMTIMER1MS_TCRR can be captured and saved in the DMTIMER1MS_TCAR1 or DMTIMER1MS_TCAR2 register function of the mode selected in the DMTIMER1MS_TCLR[13] CAPT_MODE bit. The edge detection circuitry monitors transitions on the input pin (PIEVENTCAPT).

The rising edge, falling edge, or both, can be selected in the DMTIMER1MS_TCLR[9-8] TCM bit field to trigger the timer counter capture. The module sets the DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG bit when an active edge is detected, and at the same time, the counter value DMTIMER1MS_TCRR is stored in timer capture register DMTIMER1MS_TCAR1 or DMTIMER1MS_TCAR2, as follows:

- If the DMTIMER1MS_TCLR[13] CAPT_MODE bit is 0, on the first enabled capture event, the value of the counter register is saved in the DMTIMER1MS_TCAR1 register, and the next events are ignored (no update

Copyright © 2026 Texas Instruments Incorporated

on the DMTIMER1MS_TCAR1 register and no interrupt triggering) until the detection logic is reset or the DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG is cleared by writing 1 to it.

- If the DMTIMER1MS_TCLR[13] CAPT_MODE bit is 1, on the first enabled capture event, the value of the counter register is saved in the DMTIMER1MS_TCAR1 register, and on the second enabled capture event, the value of the counter register is saved in the DMTIMER1MS_TCAR2 register. If a capture interrupt is enabled, the interrupt triggers on the second event capture. All other events are ignored (no update on DMTIMER1MS_TCAR1/DMTIMER1MS_TCAR2 and no interrupt triggering) until the detection logic is reset or the DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG bit is cleared by writing 1 to it. This mechanism is useful for period calculation of a clock, if that clock is connected to the PIEVENTCAPT input pin.

The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is served. The DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG bit is cleared by writing 1 to it or when the edge detection mode bits (the DMTIMER1MS_TCLR[9-8] TCM bit field) are changed from no-capture mode detection to any other mode. The timer functional clock (input to prescaler) is used to sample the input pin (PIEVENTCAPT). A negative or positive pulse input can be detected when the pulse time is greater than the functional clock period. An interrupt is issued on edge detection if the capture interrupt-enable bit is set in the DMTIMER1MS_IRQSTATUS_SET[2] TCAR_EN_FLAG bit. See the examples in Figure 12-451 and Figure 12-452.

In Figure 12-451, the value of the DMTIMER1MS_TCLR[9-8] TCM bit field is 1h, and the DMTIMER1MS_TCLR[13] CAPT_MODE bit is 0. Only the rising edge of PIEVENTCAPT triggers a capture in the DMTIMER1MS_TCAR1 and DMTIMER1MS_TCAR2 registers, and only the DMTIMER1MS_TCAR1 register updates.

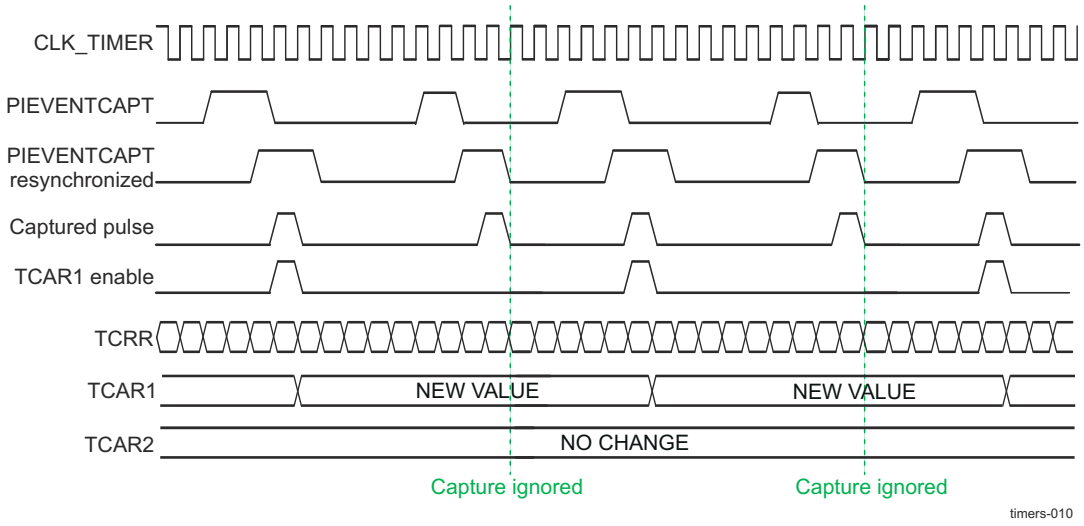


Figure 12-451. Capture Wave Example for DMTIMER1MS_TCLR[13] CAPT_MODE = 0

In Figure 12-452, the value of the DMTIMER1MS_TCLR[9-8] TCM bit field is 1h, and the DMTIMER1MS_TCLR[13] CAPT_MODE bit is 1. Only the rising edge of PIEVENTCAPT triggers a capture in the DMTIMER1MS_TCAR1 register on the first enabled event, and the DMTIMER1MS_TCAR2 register updates on the second enabled event.

Copyright © 2026 Texas Instruments Incorporated

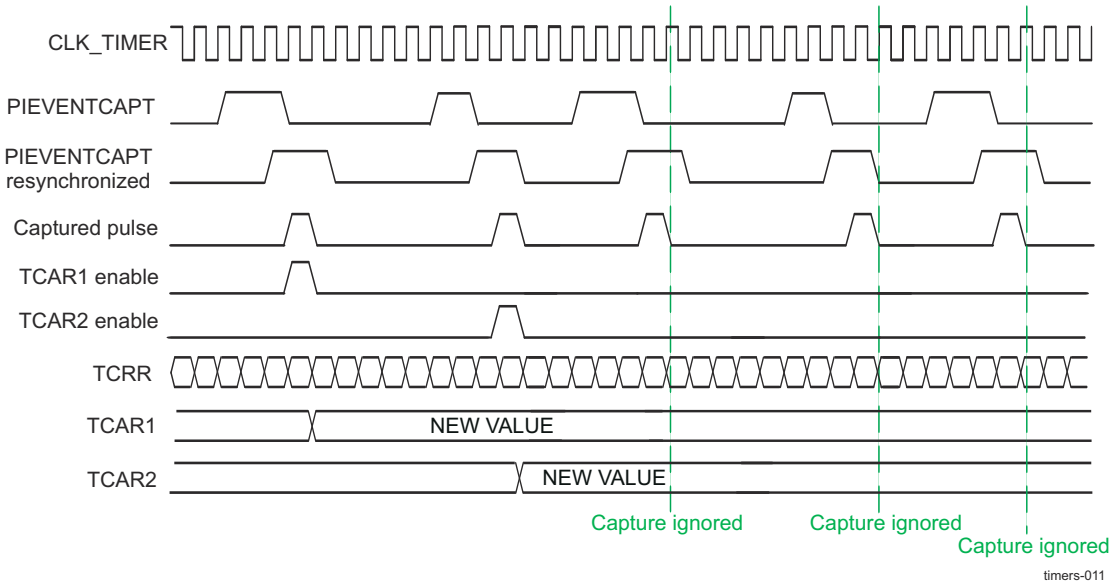


Figure 12-452. Capture Wave Example for DMTIMER1MS_TCLR[13] CAPT_MODE = 1

12.7.4.4.7 Timer Compare Mode Functionality

When the compare-enable register DMTIMER1MS_TCLR[6] CE bit is set to 1, the timer value (the DMTIMER1MS_TCRR[31-0] TIMER_COUNTER bit field) is continuously compared to the value held in the timer match register (DMTIMER1MS_TMAR). The value of the DMTIMER1MS_TMAR[31-0] COMPARE_VALUE bit field can be loaded at any time (timer counting or stopped). When the DMTIMER1MS_TCRR and the DMTIMER1MS_TMAR values match, an interrupt is issued, if the DMTIMER1MS_IRQSTATUS_SET[0] MAT_EN_FLAG bit is set.

To prevent any unwanted interrupts due to reset value matching effect, write a compare value to the DMTIMER1MS_TMAR before setting the DMTIMER1MS_TCLR[6] CE bit.

The dedicated output pin (POTIMERPWM) can be programmed in the DMTIMER1MS_TCLR[12] PT bit through the DMTIMER1MS_TCLR[11-10] TRG bit field to generate one positive pulse (timer clock duration) or to invert the current value (toggle mode) when an overflow or a match occurs.

12.7.4.4.8 Timer Prescaler Functionality

A prescaler can be used to divide the timer counter input clock frequency. The prescaler is enabled when the DMTIMER1MS_TCLR[5] PRE bit is set. The DMTIMER1MS_TCLR[4-2] PTV bit field sets the 2^n division ratio (prescaler value is $2^{(PTV + 1)}$. The prescaler counter is reset when the timer counter is stopped or reloaded on-the-fly.

Table 12-359 lists the prescaler/timer reload values versus contexts.

Table 12-359. Prescaler/Timer Reload Values Versus Contexts

Context	Prescaler	Timer Counter
Overflow (when autoreload is on)	Reset	DMTIMER1MS_TLDR[31-0]
DMTIMER1MS_TCRR write	Reset	DMTIMER1MS_TCRR[31-0]
DMTIMER1MS_TTGR write	Reset	DMTIMER1MS_TLDR[31-0]
Stop	Reset	Frozen

12.7.4.4.9 Timer Pulse-Width Modulation

The timer can be configured to provide a programmable PWM output. The timer PWM (POTIMERPWM) output pin can be configured to toggle on an event. The DMTIMER1MS_TCLR[11-10] TRG bit field determines on

Copyright © 2026 Texas Instruments Incorporated

which register value the PWM pin toggles. Either overflow or both overflow and match can be selected to toggle the timer PWM pin when a compare condition occurs.

Note

In toggle mode, when DMTIMER1MS_TCLR[11-10] TRG = 0x2 (overflow and match), the first event that toggles the PWM line is an overflow event.

The DMTIMER1MS_TCLR[7] SCPWM bit can be programmed to set or clear the timer PWM output signal only while the counter is stopped or the trigger is off. This allows setting the output pin to a known state before modulation starts. Modulation synchronously stops when the DMTIMER1MS_TCLR[11-10] TRG bit field is cleared and overflow occurs. This allows fixing a deterministic state of the output pin when modulation stops.

In Figure 12-453, the internal overflow pulse is set each time the (0xFFFF FFFF – DMTIMER1MS_TLDR[31-0] LOAD_VALUE + 1) value is reached, and the internal match pulse is set when the counter reaches the value of DMTIMER1MS_TMAR. Depending on the value of the DMTIMER1MS_TCLR[12] PT bit and DMTIMER1MS_TCLR[11-10] TRG bit field, the timer provides pulse or PWM event on the output pin (POTIMERPWM).

The DMTIMER1MS_TLDR and DMTIMER1MS_TMAR must keep values below the overflow value (0xFFFF FFFF) by at least two units. If the PWM trigger events are both overflow and match, the difference between the values kept in the DMTIMER1MS_TMAR and the value in the DMTIMER1MS_TLDR must be at least two units. When match event is used, the compare mode DMTIMER1MS_TCLR[6] CE bit must be set.

In Figure 12-453, the DMTIMER1MS_TCLR[7] SCPWM bit is set to 0. In Figure 12-454, the DMTIMER1MS_TCLR[7] SCPWM bit is set to 1. To obtain the desired wave form, start the counter at 0xFFFF FFFE value (to ensure an overflow first) or adjust the line polarity (DMTIMER1MS_TCLR[7] SCPWM bit).

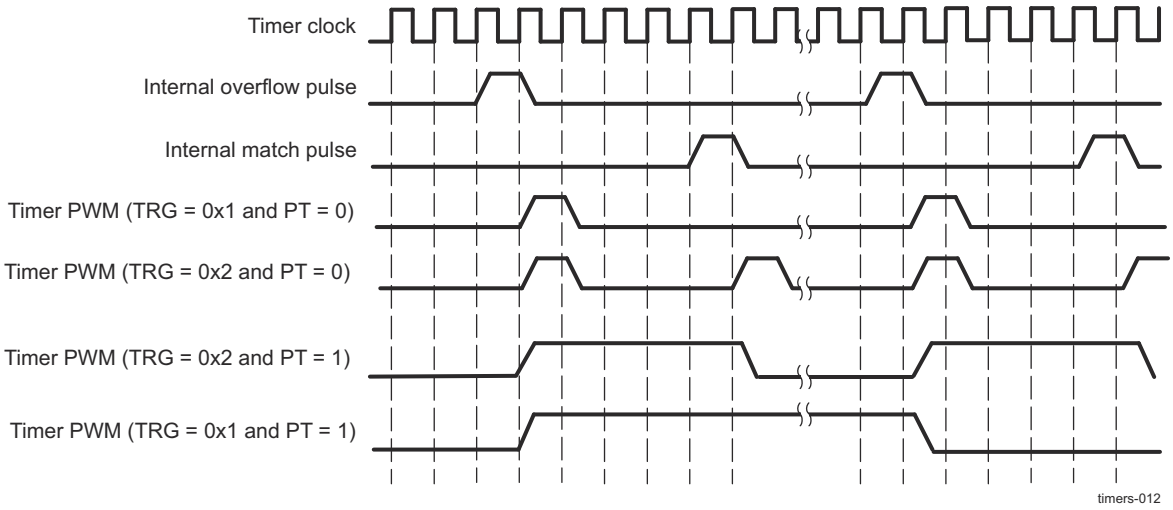


Figure 12-453. Timing Diagram of PWM With DMTIMER1MS_TCLR[7] SCPWM Bit = 0

Copyright © 2026 Texas Instruments Incorporated

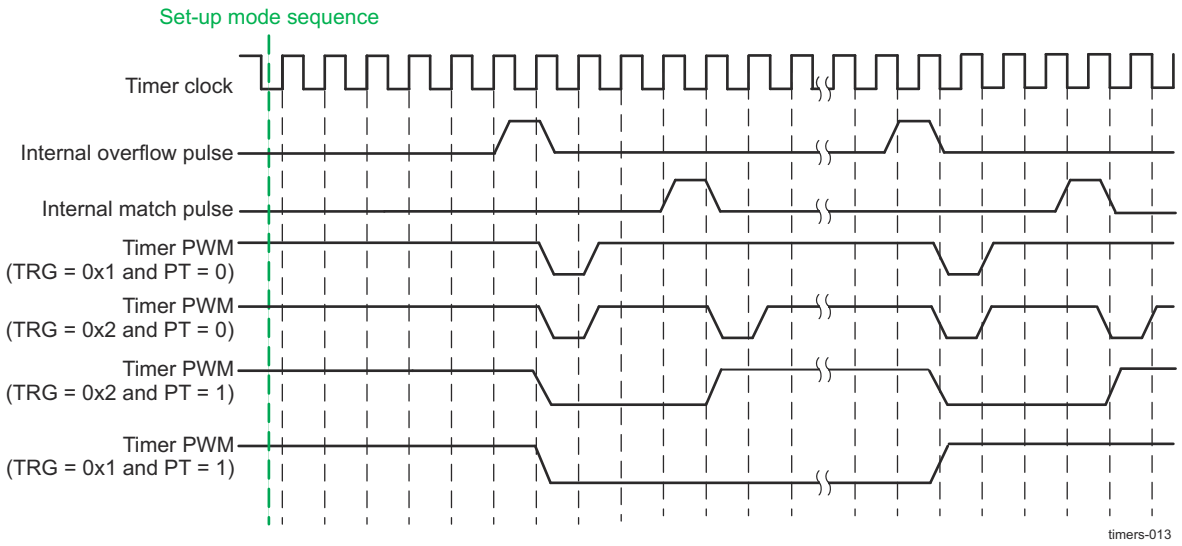


Figure 12-454. Timing Diagram of PWM With DMTIMER1MS_TCLR[7] SCPWM Bit = 1

12.7.4.4.10 Timer Counting Rate

The timer rate is defined by the following values:

- Value of the prescaler fields (the DMTIMER1MS_TCLR[5] PRE bit and DMTIMER1MS_TCLR[4-2] PTV bit field)
- Value loaded into the DMTIMER1MS_TLDR

Table 12-360 lists the prescaler clock ratio values.

Table 12-360. Prescaler Clock Ratio Values

DMTIMER1MS_TCLR[5] PRE	DMTIMER1MS_TCLR[4-2] PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32
1	5	64
1	6	128
1	7	256

Thus, the timer overflow rate is expressed as:

OVF_Rate = (0xFFFF FFFF – DMTIMER1MS_TLDR + 1) × (timer-functional clock period) × PS

With (timer-functional clock period) = 1/(timer-functional clock frequency) and PS = $2^{(PTV + 1)}$ if prescaler is enabled, or PS = 1 if prescaler is disabled.

Copyright © 2026 Texas Instruments Incorporated

CAUTION

Internal resynchronization causes any write to the DMTIMER1MS_TCLR[1] ST bit to have some latency before the register is updated:

2.5 × functional clock cycles write_DMTIMER1MS_TCLR_latency 3.5 × functional clock cycles

Remember to consider this latency whenever the timer must be started or stopped by a software change to the DMTIMER1MS_TCLR[1] ST bit.

CAUTION

- In non-PWM mode, DMTIMER1MS_TLDR must be maintained at less than or equal to 0xFFFF FFFE.
- In PWM mode, DMTIMER1MS_TLDR must be maintained at less than or equal to 0xFFFF FFFD.

For example, with a timer clock input of 32 kHz and the DMTIMER1MS_TCLR[5] PRE bit set to 0, the timer output period is as listed in Table 12-361.

Table 12-361. Value and Corresponding Interrupt Period

DMTIMER1MS_TLDR[31-0] LOAD_VALUE	Interrupt Period
0x0000 0000	37 h
0xFFFF 0000	2 s
0xFFFF FFF0	500 µs
0xFFFF FFFE	62.5 µs

12.7.4.4.11 Timer Under Emulation

During emulation mode, the timer continues to run according to the value of the DMTIMER1MS_TIOCP_CFG[1] EMUFREE bit.

If the DMTIMER1MS_TIOCP_CFG[1] EMUFREE bit is set to 1, timer execution is not stopped in emulation mode and the interrupt is still generated when overflow or match is reached.

If the DMTIMER1MS_TIOCP_CFG[1] EMUFREE bit is set to 0, the prescaler and timer are frozen and both resume on exit from emulation mode. The asynchronous external input pin (MCU_TIMER_IO[9-0] or TIMER_IO[7-0]) is internally synchronized on two timer-clock rising edges.

12.7.4.4.12 Accessing Timer Registers

All accesses are posted mode, under the assumption that freq(timer clock) < freq(interface clock)/4, until software reconfiguration. In addition, it is not recommended to access the timer registers prior to the PLLs generating the timer and interface clocks have been configured and the clocks have stabilized. All registers are 32 bits wide, accessible through the configuration interface with 32-bit access (read/write).

Write operations to the following functional registers must be complete (the MSB must be written even if the MSB data is not used):

- DMTIMER1MS_TCLR
- DMTIMER1MS_TCRR
- DMTIMER1MS_TLDR
- DMTIMER1MS_TTGR
- DMTIMER1MS_TMAR
- DMTIMER1MS_TPIR
- DMTIMER1MS_TNIR
- DMTIMER1MS_TCVR
- DMTIMER1MS_TOCR
- DMTIMER1MS_TOWR

Copyright © 2026 Texas Instruments Incorporated

The following registers are not affected by the posted/nonposted mode selection; the write/read operation is effective and acknowledged (command accepted) after one interface clock cycle from command assertion:

- DMTIMER1MS_TIDR
- DMTIMER1MS_TIOCP_CFG
- DMTIMER1MS_IRQSTATUS
- DMTIMER1MS_IRQSTATUS_RAW
- DMTIMER1MS_IRQSTATUS_SET
- DMTIMER1MS_IRQSTATUS_CLR
- DMTIMER1MS_IRQWAKEEN
- DMTIMER1MS_TWPS
- DMTIMER1MS_TSICR

12.7.4.4.12.1 Writing to Timer Registers

The host uses the configuration interface to write to the following registers synchronously with the timer interface clock:

- DMTIMER1MS_TLDR
- DMTIMER1MS_TCRR
- DMTIMER1MS_TCLR
- DMTIMER1MS_TIOCP_CFG
- DMTIMER1MS_IRQSTATUS
- DMTIMER1MS_IRQSTATUS_SET
- DMTIMER1MS_IRQSTATUS_CLR
- DMTIMER1MS_IRQWAKEEN
- DMTIMER1MS_TTGR
- DMTIMER1MS_TSICR
- DMTIMER1MS_TMAR
- DMTIMER1MS_TPIR
- DMTIMER1MS_TNIR
- DMTIMER1MS_TCVR
- DMTIMER1MS_TOCR
- DMTIMER1MS_TOWR

12.7.4.4.12.1.1 Write Posting Synchronization Mode

This mode is used if the DMTIMER1MS_TSICR[2] POSTED bit is set to 1 (default value).

This mode uses a posted write scheme to update any internal register (DMTIMER1MS_TCLR, DMTIMER1MS_TCRR, DMTIMER1MS_TLDR, DMTIMER1MS_TTGR, DMTIMER1MS_TMAR, DMTIMER1MS_TPIR, DMTIMER1MS_TNIR, DMTIMER1MS_TCVR, DMTIMER1MS_TOCR, and DMTIMER1MS_TOWR). Therefore, the write transaction is immediately acknowledged on the configuration interface, although the effective write operation occurs later because of a resynchronization in the timer clock domain. The advantage is that neither the interconnect, nor the device that requested the write transaction is stalled.

For each register, a status bit is provided in the timer write-posted status (DMTIMER1MS_TWPS) register. In this mode, it is mandatory that software check this status bit before any write access. If a write is attempted to a register with a previous access pending, the previous access is discarded without notice.

The timer module updates the value of the timer counter register synchronously with the interface clock. Consequently, any read access to DMTIMER1MS_TCRR does not add any resynchronization latency; the current value is always available.

Copyright © 2026 Texas Instruments Incorporated

Note

Because the overflow IRQ is generated when the value of DMTIMER1MS_TCRR reaches 0xFFFF FFFF, and not when it changes its value to the value after overflow, it is necessary to wait a delay of (1 × PS × timer functional clock period) before any read access to DMTIMER1MS_TCRR to ensure a correct reading of its content.

Note

If DMTIMER1MS_TTGR register is written during a posted write to DMTIMER1MS_TCRR, the value to be written to DMTIMER1MS_TCRR will be discarded.

If a posted write to DMTIMER1MS_TCVR is started, the user must not write to DMTIMER1MS_TPIR or DMTIMER1MS_TNIR before the DMTIMER1MS_TCVR write is finished, because the value of DMTIMER1MS_TCVR is re-evaluated, so both the value to be written, and the recalculated value will be discarded.

If a write access is pending for a register, reading from this register does not yield a correct result. Software synchronization must be used to avoid incorrect results.

Functional frequency range: freq(timer clock) < freq(interface clock)/4.

12.7.4.4.12.1.2 Write Nonposting Synchronization Mode

This mode is used if the DMTIMER1MS_TSICR[2] POSTED bit is set to 0. It uses a nonposted write scheme to update any internal register. Therefore, the write transaction is not acknowledged on the configuration interface until the effective write operation occurs after the resynchronization in the timer functional clock domain. The drawback is that the interconnect and the device that requested the write transaction are stalled during this period.

The same full resynchronization scheme is used for a read transaction, and the same stall period applies. A register read following a write to the same register is always coherent.

This mode is functional regardless of the ratio between the configuration interface frequency and the timer clock frequency.

12.7.4.4.12.2 Reading From Timer Counter Registers

Note

LSB/MSB accesses cannot be interleaved (that is, the sequence LSB register 1, LSB register 2, MSB register 1, MSB register 2 is not supported).

The DMTIMER1MS_TCRR is a 32-bit “atomic datum” and its 16-bit capture is done on the 16-bit LSB first to allow atomic LSB16 + MSB16 capture. This capture scheme is also performed for the DMTIMER1MS_TCAR1 and DMTIMER1MS_TCAR2 registers as they can be changed due to internal processes too.

12.7.4.4.12.2.1 Read Posted

This mode is functional regardless of the ratio between the configuration interface frequency and the functional clock frequency. The recommended functional frequency range is freq(timer) < freq(interface clock)/4.

Read posted mode is used if DMTIMER1MS_TSICR[2] POSTED = 0x1 or DMTIMER1MS_TSICR[3] READ_MODE is set to 0. This mode uses a posted-read scheme for reading any internal timer register. The read transaction is immediately acknowledged on the configuration interface, prior to the value to be read has been resynchronized. With this method, neither the interconnect nor the device that requested the read transaction are stalled.

Copyright © 2026 Texas Instruments Incorporated

Read posted mode applies to DMTIMER1MS_TCRR, DMTIMER1MS_TCAR1, DMTIMER1MS_TCAR2, DMTIMER1MS_TCVR, and DMTIMER1MS_TOWR, which needs resynchronization from functional to interface clock domains.

Note that in Posted mode, if the DMTIMER1MS_TCRR is read immediately after wake-up and the interface clock is off during idle state, then it is possible to get an old value from just before going to idle state due to the fact the interface clock is needed for synchronization.

In order to avoid this situation, another synchronization mechanism is used for the first read operation after idle state. The DMTIMER1MS_TSICR[4] READ_AFTER_IDLE bit is used to enable/disable the mechanism.

When the synchronization mechanism is disabled (READ_AFTER_IDLE bit is set to 1), first read transaction takes only 2 interface clock cycles, but the read value of DMTIMER1MS_TCRR could be wrong.

When the synchronization mechanism is enabled (READ_AFTER_IDLE bit is set to 0), first read value of DMTIMER1MS_TCRR is correct, but the read transaction takes more than 2 interface clock cycles.

12.7.4.4.12.2.2 Read Non-Posted

This mode is functional regardless of the ratio between the configuration interface frequency and the functional clock frequency. Recommended functional frequency range is freq(timer) >= freq(interface clock)/4.

Read non-posted mode is used if DMTIMER1MS_TSICR[2] POSTED = 0x0 and DMTIMER1MS_TSICR[3] READ_MODE = 0x1. This mode uses a non-posted read scheme for reading internal timer registers. The read transaction is not acknowledged on the configuration interface until the effective read operation occurs, after the resynchronization in the timer clock domain. The result is that both the interconnect and the device that requested the read transaction are stalled during this period.

This mode applies to DMTIMER1MS_TCRR, DMTIMER1MS_TCAR1, DMTIMER1MS_TCAR2, DMTIMER1MS_TCVR, and DMTIMER1MS_TOWR, which need resynchronization from functional to interface clock domains.

12.7.4.4.13 Timer Posted Mode Selection

A choice between two synchronization modes is made taking into account the frequency ratio and the stall periods that can be supported by the system, without impacting the global performance.

The posted mode selection applies only to registers that require synchronization on or from the timer clock domain. For write operation, the registers affected by posted and non-posted selection are DMTIMER1MS_TCLR, DMTIMER1MS_TLDR, DMTIMER1MS_TCRR, DMTIMER1MS_TTGR, DMTIMER1MS_TMAR, DMTIMER1MS_TPIR, DMTIMER1MS_TNIR, DMTIMER1MS_TCVR, DMTIMER1MS_TOCR, and DMTIMER1MS_TOWR. For read operation, the registers affected by this selection are: DMTIMER1MS_TCRR, DMTIMER1MS_TCAR1, DMTIMER1MS_TCAR2, DMTIMER1MS_TCVR, and DMTIMER1MS_TOWR.

The interface clock domain synchronous registers DMTIMER1MS_TIDR, DMTIMER1MS_TIOCP_CFG, DMTIMER1MS_IRQSTATUS, DMTIMER1MS_IRQSTATUS_SET, DMTIMER1MS_IRQWAKEEN, DMTIMER1MS_TWPS, and DMTIMER1MS_TSICR are not affected by posted and non-posted mode selection. The operation (read or write) is effective and acknowledged after one interface clock cycle from the command assertion.

The configuration of posted or non-posted mode can be changed (overwritten) by software by writing in DMTIMER1MS_TSICR[2] POSTED bit. The DMTIMER1MS_TSICR[3] READ_MODE defines how the read operation is performed when the module is configured in non-posted mode (see DMTIMER1MS_TSICR). The following cases are possible:

- DMTIMER1MS_TSICR[2] POSTED = 0x1 and DMTIMER1MS_TSICR[3] READ_MODE = x (don’t care): read and write operations are expected in posted mode.
- DMTIMER1MS_TSICR[2] POSTED = 0x0 and DMTIMER1MS_TSICR[3] READ_MODE = 0x0: the write operation is executed in non-posted mode and read is executed in posted mode.

Copyright © 2026 Texas Instruments Incorporated

- DMTIMER1MS_TSICR[2]POSTED = 0x0 and DMTIMER1MS_TSICR[3] READ_MODE = 0x1: write is executed in non-posted mode and read is executed in non-posted mode.

Copyright © 2026 Texas Instruments Incorporated

12.7.4.5 Timers Low-Level Programming Models

This section describes the low-level hardware programming sequences for the configuration and use of the module.

12.7.4.5.1 Timer Global Initialization

12.7.4.5.1.1 Main Sequence – Timer Module Global Initialization

Table 12-362 identifies the main steps for initializing the timer module when the module is to be used for the first time.

Table 12-362. Timer Module Global Initialization

Step	Register/Bit Field/Programming Model	Value
Execute software reset.	DMTIMER1MS_TIOCP_CFG[0] SOFTRESET	0x1
Wait until reset release?	DMTIMER1MS_TIOCP_CFG[0] SOFTRESET	0x0
Configure idle mode.	DMTIMER1MS_TIOCP_CFG[3-2] IDLEMODE	0x-
Enable wake-up interrupt events.	DMTIMER1MS_IRQWAKEEN[2-0]	0x-
Select posted mode.	DMTIMER1MS_TSICR[2] POSTED	0x-

12.7.4.5.2 Timer Operational Mode Configuration

12.7.4.5.2.1 Timer Mode

12.7.4.5.2.1.1 Main Sequence – Timer Mode Configuration

Table 12-363 lists the steps in the timer mode configuration.

Table 12-363. Timer Mode Configuration

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	DMTIMER1MS_TCLR[1] AR	0x-
Set prescale timer value.	DMTIMER1MS_TCLR[4-2] PTV	0x-
Enable prescaler.	DMTIMER1MS_TCLR[5] PRE	0x1
Enable overflow interrupt.	DMTIMER1MS_IRQSTATUS_SET[1] OVF_EN_FLAG	0x1
Load timer counter value.	DMTIMER1MS_TCRR	0x-
Load timer load value.	DMTIMER1MS_TLDR	0x-
Start the timer.	DMTIMER1MS_TCLR[0] ST	0x1

12.7.4.5.2.2 Timer Compare Mode

12.7.4.5.2.2.1 Main Sequence – Timer Compare Mode Configuration

Table 12-364 lists the steps in the timer compare mode configuration.

Table 12-364. Timer Compare Mode Configuration

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	DMTIMER1MS_TCLR[1] AR	0x-
Set prescale timer value.	DMTIMER1MS_TCLR[4-2] PTV	0x-
Enable prescaler.	DMTIMER1MS_TCLR[5] PRE	0x1
Enable match interrupt.	DMTIMER1MS_IRQSTATUS_SET[0] MAT_EN_FLAG	0x1
Load timer counter value.	DMTIMER1MS_TCRR	0x-
Load timer compare value.	DMTIMER1MS_TMAR	0x-
Enable compare mode.	DMTIMER1MS_TCLR[6] CE	0x1
Start the timer.	DMTIMER1MS_TCLR[0] ST	0x1

Copyright © 2026 Texas Instruments Incorporated

12.7.4.5.2.3 Timer Capture Mode

12.7.4.5.2.3.1 Main Sequence – Timer Capture Mode Configuration

Table 12-365 lists the steps in the timer capture mode configuration.

Table 12-365. Timer Capture Mode Configuration

Step	Register/Bit Field/Programming Model	Value
Initialize capture mode.	See Section 12.7.4.5.2.3.2.	
Enable capture interrupt.	DMTIMER1MS_IRQSTATUS_SET[2] TCAR_EN_FLAG	0x1
Start the timer.	DMTIMER1MS_TCLR[0] ST	0x1
Detect event.	See Section 12.7.4.5.2.3.3.	

12.7.4.5.2.3.2 Subsequence – Initialize Capture Mode

Table 12-366 lists the steps to initialize capture mode.

Table 12-366. Initialize Capture Mode

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	DMTIMER1MS_TCLR[1] AR	0x-
Set prescale timer value.	DMTIMER1MS_TCLR[4-2] PTV	0x-
Enable prescaler.	DMTIMER1MS_TCLR[5] PRE	0x1
Select TIMER[11-0] or MCU_TIMER[3-0] Capture input at device pins TIMER_IO[11-0] for TIMER[11-0] or at pins MCU_TIMER_IO[3-0] for MCU_TIMER[3-0].	DMTIMER1MS_TCLR[14] GPO_CFG	0x1
Select single or second event capture.	DMTIMER1MS_TCLR[13] CAPT_MODE	0x-
Select transition capture mode.	DMTIMER1MS_TCLR[9-8] TCM	0x-

12.7.4.5.2.3.3 Subsequence – Detect Event

Table 12-367 lists the steps in detecting an event.

Table 12-367. Detect Event

Step	Register/Bit Field/Programming Model	Value
Wait until event detected?	DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG	= 0x1
Read timer capture value.	DMTIMER1MS_TCAR1 and/or DMTIMER1MS_TCAR2	
Clear capture interrupt request.	DMTIMER1MS_IRQSTATUS[2] TCAR_IT_FLAG	0x1

12.7.4.5.2.4 Timer PWM Mode

12.7.4.5.2.4.1 Main Sequence – Timer PWM Mode Configuration

Table 12-368 lists the steps in the timer PWM mode configuration.

Table 12-368. Timer PWM Mode Configuration

Step	Register/Bit Field/Programming Model	Value
Select autoreload mode.	DMTIMER1MS_TCLR[1] AR	0x-
Set prescale timer value.	DMTIMER1MS_TCLR[4-2] PTV	0x-
Enable prescaler.	DMTIMER1MS_TCLR[5] PRE	0x1
Select trigger output mode.	DMTIMER1MS_TCLR[11-10] TRG	0x-
Select pulse or toggle modulation PWM mode.	DMTIMER1MS_TCLR[12] PT	0x-
Select TIMER[11-0] or MCU_TIMER[3-0] PWM output at device pins TIMER_IO[11-0] for TIMER[11-0] or at pins MCU_TIMER_IO[3-0] for MCU_TIMER[3-0].	DMTIMER1MS_TCLR[14] GPO_CFG	0x0
Configure PWM output pin default value.	DMTIMER1MS_TCLR[7] SCPWM	0x-

Copyright © 2026 Texas Instruments Incorporated

Table 12-368. Timer PWM Mode Configuration (continued)

Step	Register/Bit Field/Programming Model	Value
Load timer load value.	DMTIMER1MS_TLDR	0x-
Load timer compare value.	DMTIMER1MS_TMAR	0x-
Enable compare.	DMTIMER1MS_TCLR[6] CE	0x1
Start the timer.	DMTIMER1MS_TCLR[0] ST	0x1

12.8 Internal Diagnostics Modules

12.8.1 Dual Clock Comparator (DCC)

This section describes the Dual Clock Comparator (DCC) modules in the device.

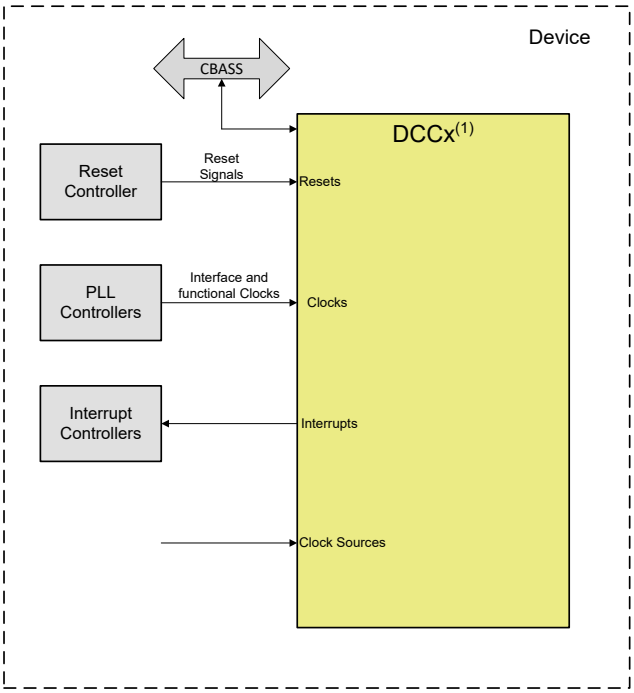
12.8.1.1 DCC Overview

The Dual Clock Comparator (DCC) is used to determine the accuracy of a clock signal during the time execution of an application. Specifically, the DCC is designed to detect drifts from the expected clock frequency. The desired accuracy can be programed based on calculation for each application. The DCC measures the frequency of a selectable clock source using another input clock as a reference.

The device has instances of DCC modules.

Figure 12-455 shows the DCC modules overview.

Copyright © 2026 Texas Instruments Incorporated



1. x represents a valid instance of UART in a domain. See the device data sheet for valid domains and instances.

Figure 12-455. DCC Modules Overview

12.8.1.1.1 DCC Features

The DCC uses two independent clock sources to detect when one is out of specification. Each DCC module implements the following features:

- Two independent counter blocks count clock pulses from each clock source
- Each counter block is programmable, however, for proper operation the counters must be programmed with seed values that respect the ratio of the two clock frequencies
- Configurable timebase for error signal
- Error signal generation when one of the clocks is out of specification
- Clock frequency measurement

12.8.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Copyright © 2026 Texas Instruments Incorporated

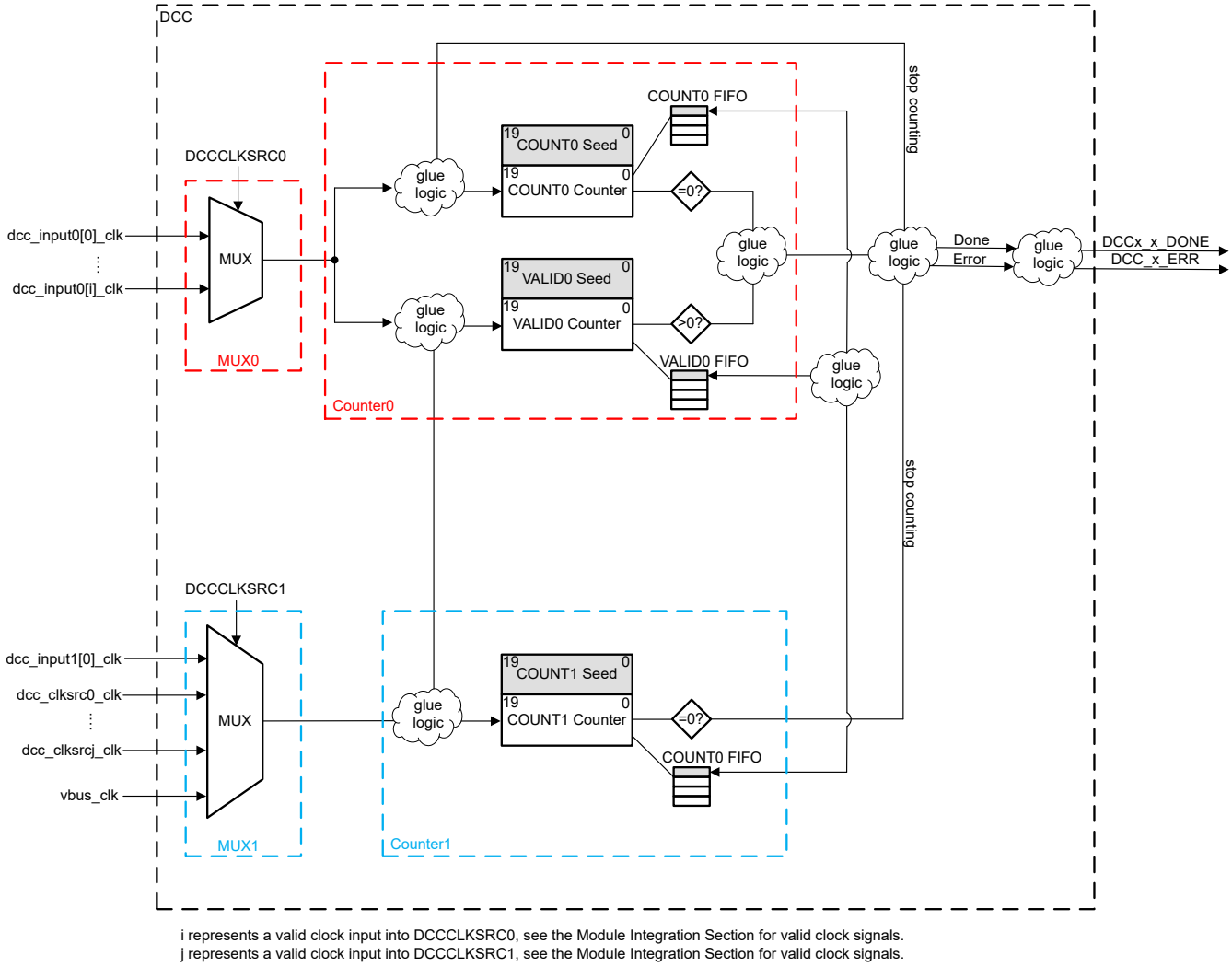
Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.8.1.2 DCC Functional Description

The DCC module supports two selectable clock sources for Counter0 and Counter1. Figure 12-456 shows the DCC functional block diagram.



i represents a valid clock input into DCCCLKSRC0, see the Module Integration Section for valid clock signals.
j represents a valid clock input into DCCCLKSRC1, see the Module Integration Section for valid clock signals.

Figure 12-456. DCC Functional Block Diagram

12.8.1.2.1 DCC Counter Operation

Note

For detailed DCC compute calculations, refer to Continuous Monitor of the PLL Frequency With the DCC App Note.

The DCC has two parallel counters that count clock pulses for two independent clock sources:

- Counter0 generates a fixed-width counting window (VALID0) after a pre-programmed number of pulses (COUNT0). The values for VALID0 and COUNT0 can be programmed in DCC2_DCCVALIDSEED0 and DCC2_DCCCNTSEED0 registers respectively.
- Counter1 generates a fixed-width pulse (1 cycle) after a pre-programmed number of pulses (COUNT1). This pulse sets an error signal if Counter1 does not reach 0 during the time when VALID0 is running. The seed value for COUNT1 can be programmed in DCC2_DCCCNTSEED1 register.

Copyright © 2026 Texas Instruments Incorporated

The error signal is generated by any one of the following conditions:

- Clock1 expires before the COUNT0 reaches 0.
- Clock1 expires after both COUNT0 and VALID0 reach 0.
- Clock1 not present.
- Clock0 not present.

Any of these errors causes the counters to stop counting. An application must then read out the counter values to determine what caused the error. Once the error is detected, the counters are stopped after 3 FICLK and 2 source clock cycles due to the cross clock domain synchronisation.

Reloads or restarts occur under two conditions:

- The module is reset or restarted through software (that is, software starts the module after reset, or software checks an error condition and decides to restart the module).
- COUNT0, COUNT1, and VALID0 all reach 0 without error.

CAUTION

The DCC module does not check jitter for Clock0 or Clock1.

As the counter preset signal is synchronized to either of the source clock domains, the counters begin downcounting after two corresponding source clock cycles.

The error signal is captured to the FICLK domain. There is 1 FICLK period uncertainty on either side of the fixed width counting window (VALID0) in generating the error signal since the counters work in different clock domains. This should be accounted for when setting the count value for VALID0.

Figure 12-457 through Figure 12-461 shows examples of counters relationship and error generation.

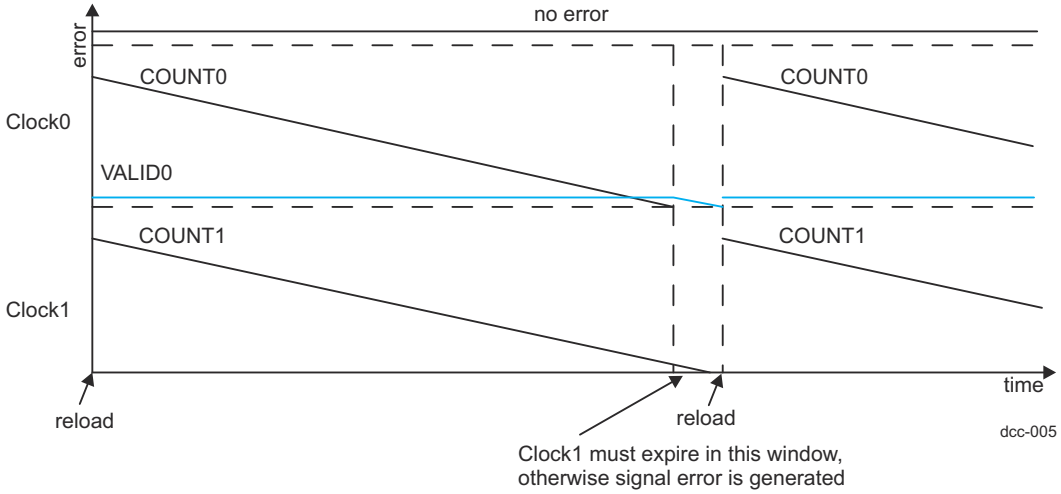


Figure 12-457. DCC Clock0 and Clock1 With no Error

Copyright © 2026 Texas Instruments Incorporated

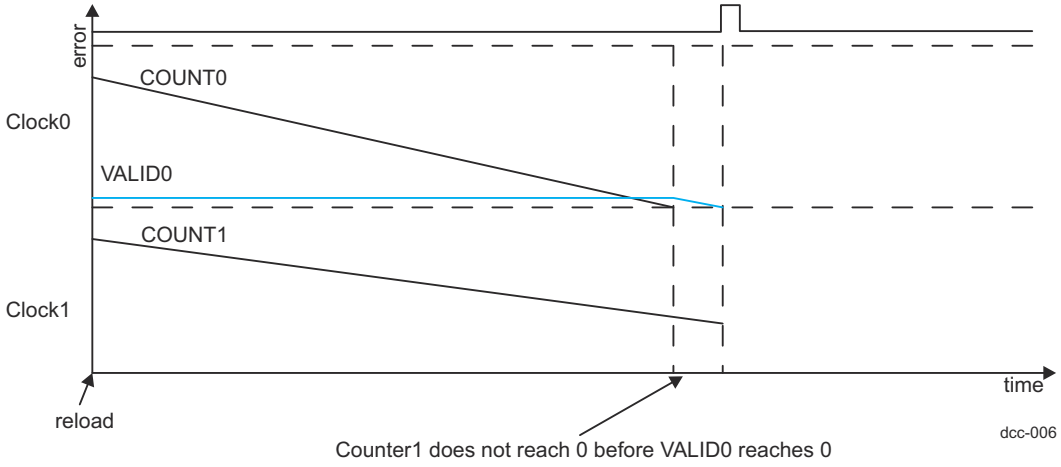


Figure 12-458. DCC Clock1 slower than Clock0 results in an error and stops counting

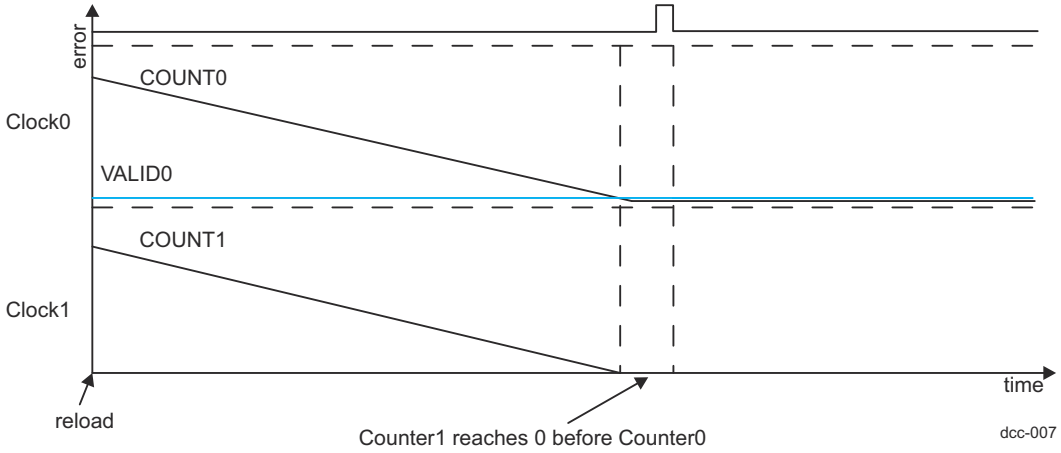


Figure 12-459. DCC Clock1 faster than Clock0 results in an error and stops counting

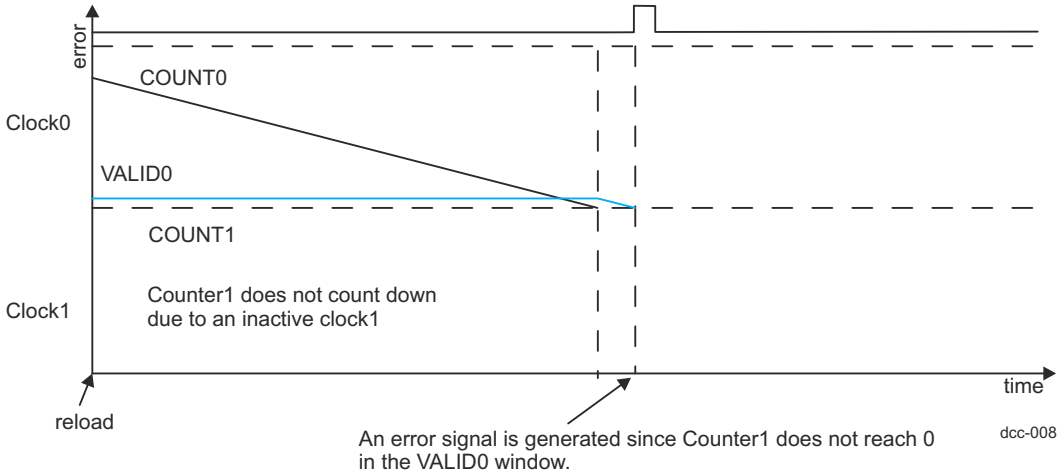


Figure 12-460. DCC Clock1 not present results in an error and stops counting

Copyright © 2026 Texas Instruments Incorporated

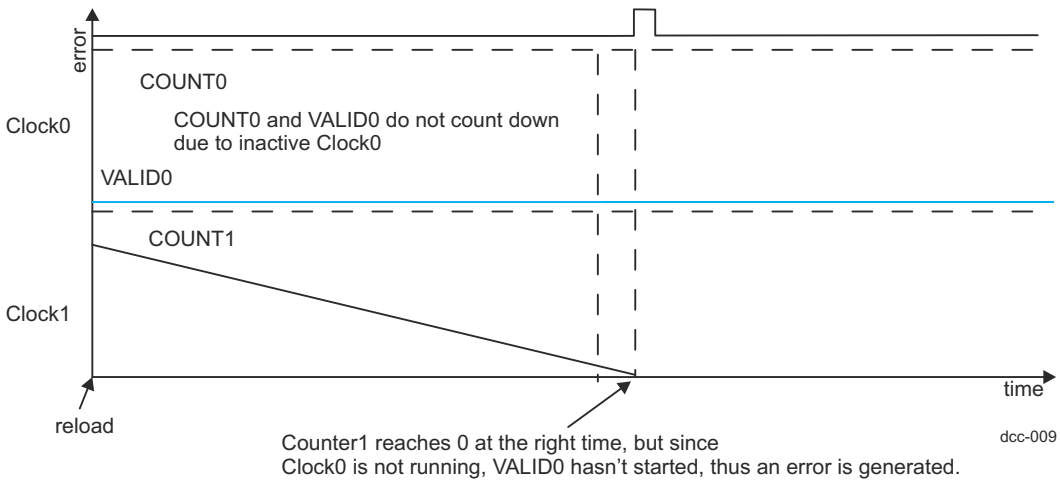


Figure 12-461. DCC Clock0 not present results in an error and stops counting

12.8.1.2.2 DCC Low Power Mode Operation

The DCC module does not function in Low Power Mode. It is the responsibility of software to stop the module before entering low power mode and to restart the module after exiting low power mode.

12.8.1.2.3 DCC Suspend Mode Behavior

The DCC module will continue running regardless of the state of emulation. All registers are readable via emulation reads.

12.8.1.2.4 DCC Single-Shot Mode

The DCC can be programmed to count down one time using single-shot mode. In this mode, the DCC stops operation when both COUNT0 and VALID0 reach 0.

At the end of one sequence in single-shot mode the DCC2_DCCGCTRL[3-0] DCCENA bitfield is set to disabled, which stops further counting. Single-shot mode is enabled from the DCC2_DCCGCTRL[11-8] SINGLESHOT bitfield.

At the end of one sequence in single-shot mode, if there is no error which stops counting, then the done status bit is set in the DCC2_DCCSTATUS[1] DONEFLG bitfield and a done interrupt DCC_x_INT is generated. Software must clear the done bit before restarting the counting.

12.8.1.2.5 DCC Continuous mode

When DCC runs in continuous mode both the counts shall get reloaded with seed value upon completion of counts without error. If the counts end in error DCC stops the operation and counts are not reloaded.

12.8.1.2.5.1 DCC Continue on Error

During debug, if there are events which are causing clocks to be anomalous over short period covering more than one evaluation window then it would be important to capture trajectory of error event and period around such event. To allow capturing the successive error events DCC can be programmed to continue after error. [3-0] CONT_ON_ERR shall be set to value other than "0101" to enable this mode. It is recommended to write "1010" to avoid single soft errors.

12.8.1.2.5.2 DCC Error Count

DCC also counts the number of error pulses generated since reset or since last time the error count is cleared. This is read/write register for CPU to clear when new trace of number of errors is required to be maintained.

Copyright © 2026 Texas Instruments Incorporated

12.8.1.2.6 DCC Control and count hand-off across clock domains

As the counters run in two different selectable clock domains and the register interface runs on the fixed bus clock domain, control signals and counter value hand-off have synchronizers implemented. These add to the margins of error while comparing the counts.

1. With all three counts synchronized to FICLK domain for comparison error detection there is delay in terms of FICLK domain.
2. Based on the error signal, the enable for the counters is synchronized back to the respective clock domains of the counters, which adds latency in terms of clock periods which are different, this would create a skew between start of count. Depending upon frequency ratios of the clocks used, the difference between two could vary.

Application needs to consider the worst case delay differences while measuring the clocks.

12.8.1.2.7 DCC Error Trajectory record

Once the clock errors out, the host can read the counter values to determine the extent of error to analyze type of failure. For short window comparisons this would become difficult, specially if there are back to back errors due to some transient event. Secondly, for random events which can cause an interrupt during the critical phase of application running, then event if not recorded may get overwritten and also not provide meaningful trace of error.

12.8.1.2.7.1 DCC FIFO capturing for Errors

DCC provides the FIFO for capturing COUNT0, VALID0, and COUNT1 information which captures all three counts upon "Error" event. For "Done" event no results are captured by default.

12.8.1.2.7.2 DCC FIFO in continuous capture mode

To track the VALID0 counter values regardless of "Error" or not, FIFOs can be configured to capture the count for each compare window. This is useful in validation and characterization exercise. [11-7] FIFO_NONERR control when set to value other than "0101" this mode is set; it is recommended to write "1010" to avoid single soft errors. Note, this capture is applicable only in continuous mode and not in single shot mode.

12.8.1.2.7.3 DCC FIFO Details

The FIFO is 4 deep for each count and updates new count information for all the non-full FIFOs. Information is updated on every configured trigger of error or cycle completion. If full, the next values are not written till at-least one entry is read. Application owns responsibility to read the FIFOs uniformly to keep synchronisation between three entries of the FIFO. Both empty and full indications for individual FIFOs is provided through the .

12.8.1.2.7.4 DCC FIFO Debug mode behavior

Upon debug access, the FIFO pointers should not advance hence not impacting the functional behavior. This requirement is same as other functional blocks in the device.

12.8.1.2.8 DCC Count read registers

DCC has provision to read the counts during operation. This is performed using , , and registers. Read from these registers in default mode allows reading the present value of count. This is useful when in single shot mode or mode where DCC stops upon error.

These registers can be used to read the FIFO through the [7-4] FIFO_READ configuration. Reads on the empty FIFO shall provide the contents of last pointed location. Application shall track the empty/full conditions of the FIFOs to track the count records consistently.

Regardless of FIFO_READ configuration, the FIFO internally keeps updating records based on configured triggers till full.

Copyright © 2026 Texas Instruments Incorporated

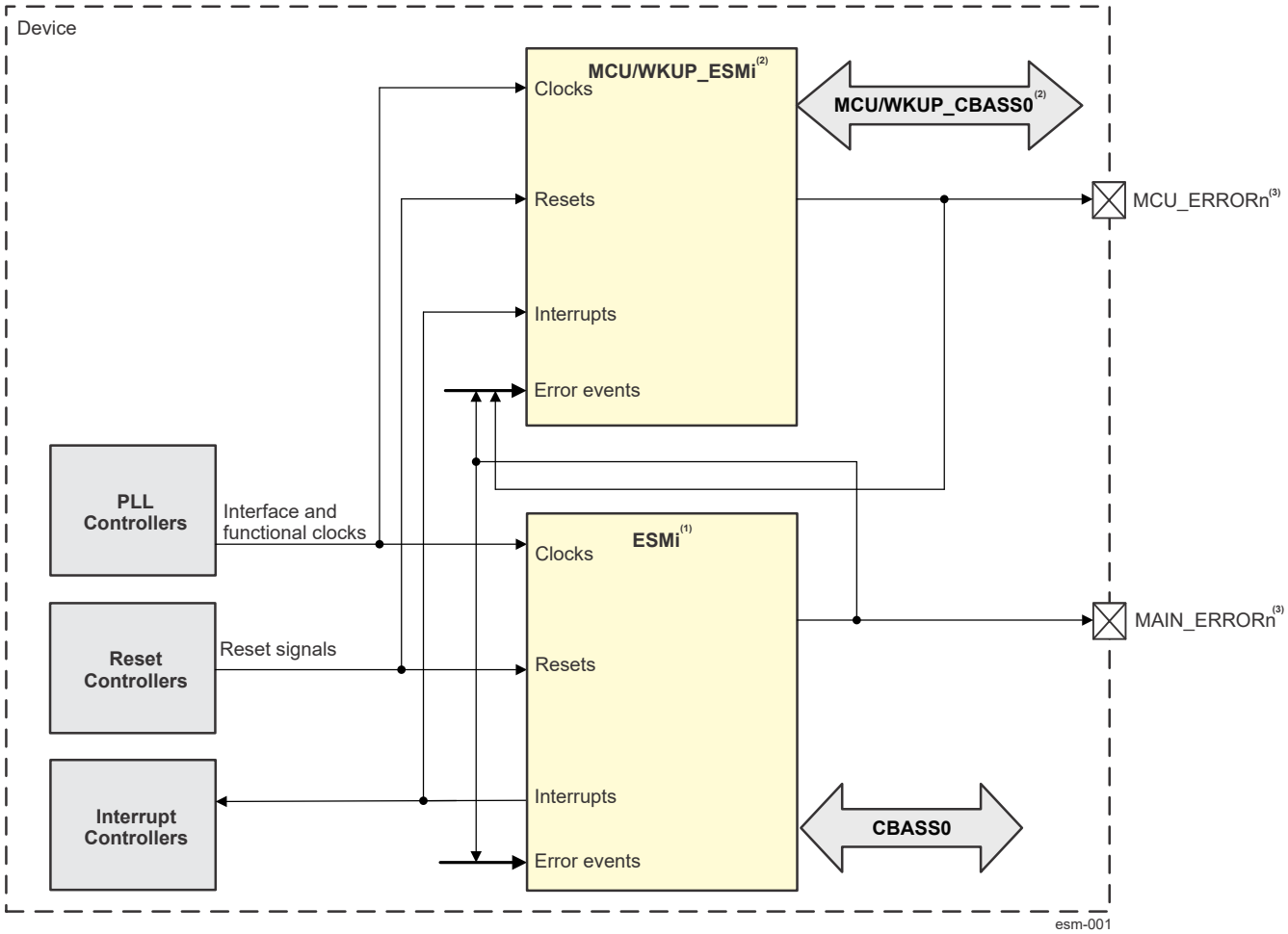
12.8.2 Error Signaling Module (ESM)

This section describes the Error Signaling Module (ESM) in the device.

12.8.2.1 ESM Overview

The Error Signaling Module (ESM) aggregates events and/or errors from throughout the device into one location. It can signal both low and high priority interrupts to a processor to deal with an event and/or manipulate an I/O error pin to signal an external hardware that an error has occurred. Therefore an external controller is able to reset the device or keep the system in a safe, known state.

Figure 12-462 shows the ESM modules overview.



1. i represents a valid instance of ESM in a domain. Consult the device datasheet for available domains and ESM instances.
2. i represents a valid instance of ESM in a MCU or WKUP domain. Consult the device datasheet for available domains and ESM instances.
3. Not all pins are valid for all devices. Consult the device datasheet for specifics.

Note

The MAIN_ERRORn pin, if valid, is used for observation only.

Figure 12-462. ESM Modules Overview

12.8.2.1.1 ESM Features

Each ESM module implements the following features:

Copyright © 2026 Texas Instruments Incorporated

- Up to 1024 error event inputs
  - Implemented in groups of 32 events
  - Level or Pulse inputs (Pulse inputs are triple redundant)
- Selectable low and high priority interrupt error pin prioritization of each error event
- Error pin to signal severe device failure
  - Support of level or PWM modes
- Configurable timebase for error signal
- Error forcing capability
- Internal redundant flops on critical fields

12.8.2.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

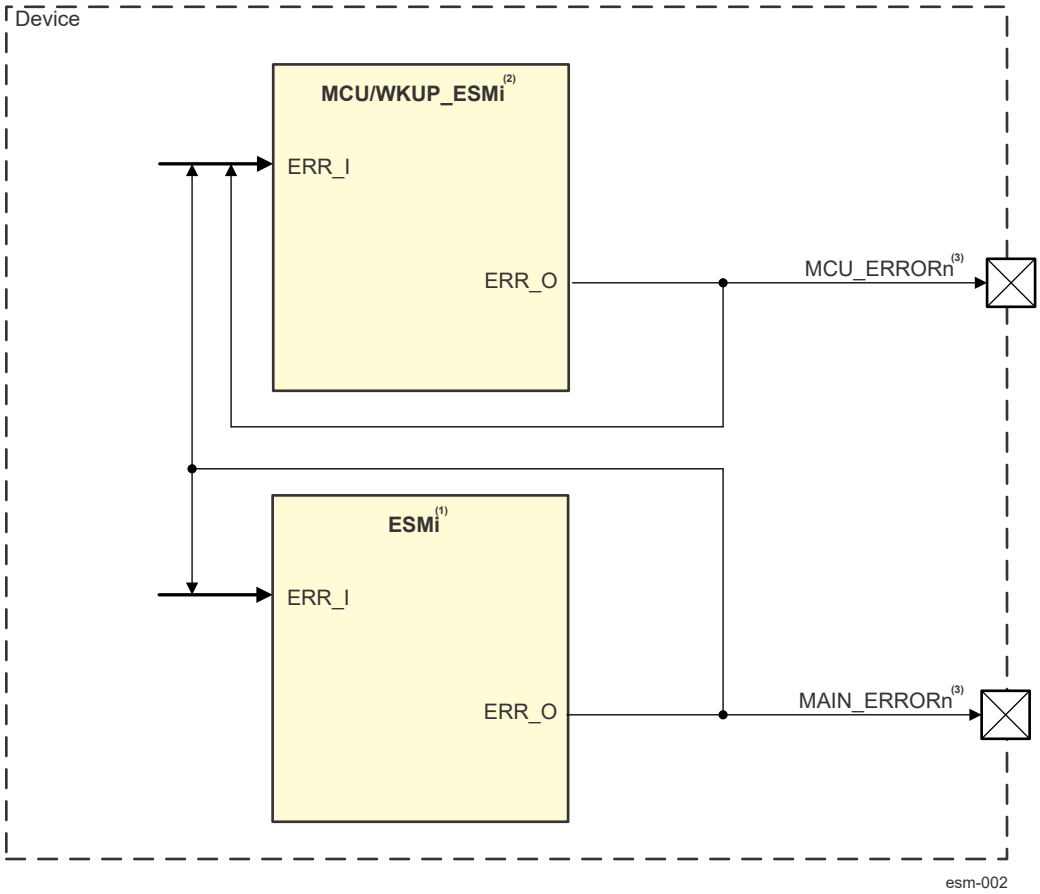
Copyright © 2026 Texas Instruments Incorporated

12.8.2.2 ESM Environment

The MCU_ESM0 and ESM0 modules are hereinafter referred to as ESM module.

This section describes the ESM external connections (environment).

Figure 12-463 shows the ESM pins used for typical connections with external devices.



1. i represents a valid instance of ESM in a domain. See the device specific datasheet for available domains and ESM instances.
2. i represents a valid instance of ESM in a MCU or WKUP domain. See the device specific datasheet for available domains and ESM instances.
3. Not all pins are valid. See the device datasheet for specifics.

Note

The MAIN_ERRORn pin, if valid, is used for observation only.

Figure 12-463. ESM Modules Environment

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.8.2.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.8.2.4 ESM Functional Description

The Error Signaling Module (ESM) centralizes fault reports. It provides mechanisms to classify errors by severity and to provide programmable error response. The error classification in the ESM is determined by programmed configuration for each individual error input. For each individual error input the configuration can be set to assert an output error pin, or generate an interrupt to a CPU, or both. When an individual error input is configured to generate an interrupt, the configuration will also select whether the interrupt that is generated will be one of high priority or low priority.

By reporting the faults in a central location, the system may determine what caused the fault and what action can be taken. In general, the faults can be split into two categories:

- Corrected faults
- Non-corrected faults

The ESM reports errors in two ways:

- An interrupt to a processor in the device. This allows the device to analyze and try to recover from an error.
- An external ERROR pin. This allows the system outside of the SoC to monitor for potentially fatal errors(errors that the device cannot self-recover from). Moreover, the external I/O (ERROR pin) can operate in level or PWM modes. In level mode, the output will remain asserted (active low) for a minimum period of time. After that period of time, if the error has been cleared by an internal processor, the pin will go inactive (high). If it does not go inactive in that time, then an external agent should intervene, as there may be an unrecoverable error. In PWM mode, the error will cause the output pin to maintain its value for a minimum period of time. After that period of time, if the error has been cleared by an internal processor, the pin will continue the PWM pattern. If it does not go inactive in that time, then an external agent should intervene, as there may be an unrecoverable error.

Both mechanisms can be used at the same time for the same fault, signaling both an interrupt and the external ERROR pin. This allows the device to attempt to recover, but if it fails, then the external system is still alerted. If it succeeds, then it can remove the ERROR pin assertion so that the external system knows that a potentially unsafe condition was avoided.

Lastly, the ESM does not specify any methods of intervention, only the process of alerting internal CPUs and external monitor(s) of an existing error event.

Figure 12-464 shows the ESM module block diagram. Note that not all instances may be pinned out in the device. For more information, see Section 12.8.2.2, *ESM Environment*.

Copyright © 2026 Texas Instruments Incorporated

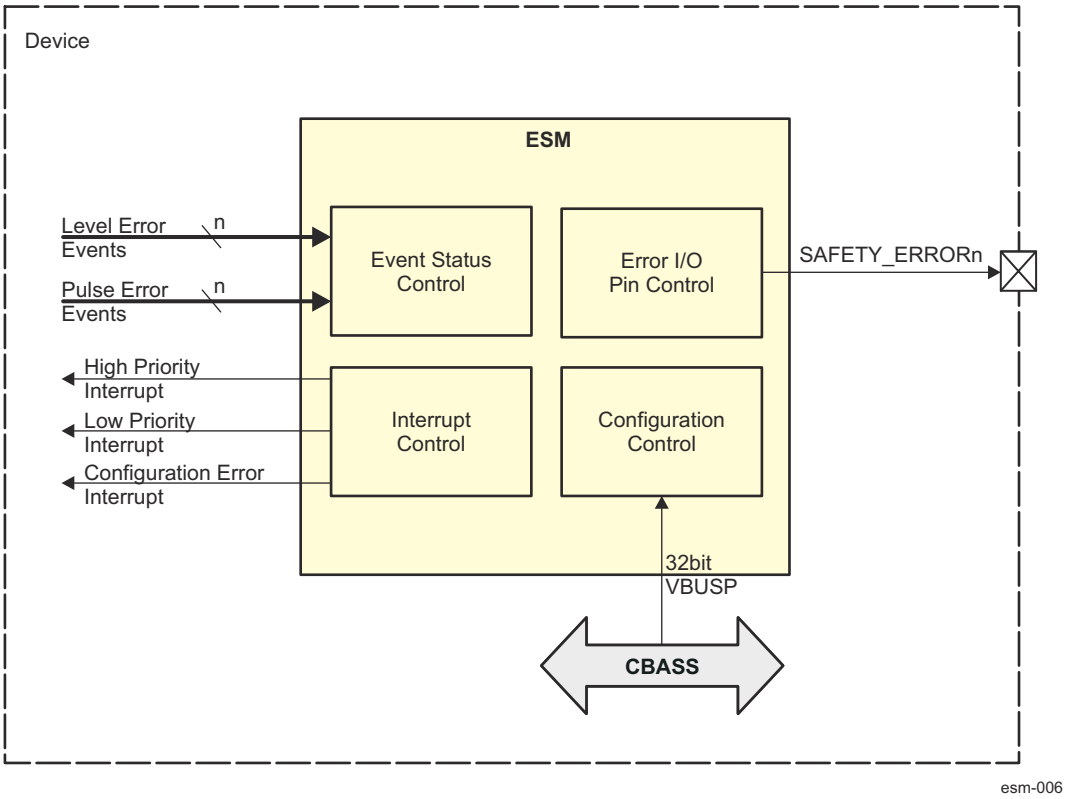


Figure 12-464. ESM Block Diagram

12.8.2.4.1 ESM Interrupt Requests

The ESM module generates three output interrupts to the device interrupt controllers:

- Configuration error interrupt (see Section 12.8.2.4.1.1)
- High priority error interrupt (see Section 12.8.2.4.1.2)
- Low priority error interrupt (see Section 12.8.2.4.1.3)

The error interrupt outputs are provided so that a processor in the device can be signaled to intervene when an error event occurs. Each error event input can be enabled, via software, to cause an error interrupt to occur (via the ESM_INTR_EN_SET_j register). Additionally, each error event input can be programmed to influence either the low priority (default) interrupt or the high priority interrupt (via the ESM_INT_PRIO_j). The low priority interrupt is intended for events that are of interest, but do not require immediate intervention. For example, an indication that there was a single bit error that was corrected may signal a low priority interrupt, so that information can be collected for statistical purposes. A high priority interrupt is intended for events that need immediate attention. For example, an indication that there was an uncorrected two-bit error may be signaled as a high priority interrupt.

12.8.2.4.1.1 ESM Configuration Error Interrupt

The configuration error interrupt (ESM_INT_CFG_LVL_0) indicates that there is an inconsistency in the configuration of one (or more) error group *j* registers (MMRs).

In such inconsistency in the internal copies of any of the MMRs caused by fault associated with error group *j*, the corresponding raw status will be set in the ESM_ERR_RAW register. If the corresponding bit is enabled in the ESM_ERR_EN_SET register, a configuration error interrupt will be triggered.

When a configuration error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM_ERR_STS register to determine which group has a configuration error
2. Write the correct values to the following group registers:

Copyright © 2026 Texas Instruments Incorporated

Note

If there has been a configuration error, the values in the below registers cannot be guaranteed to be correct anymore. Therefore, software should maintain a copy of the correct values prior to an error to ensure that they can be re-programmed with the correct configuration.

a. ESM_INTR_EN_SET_j
b. ESM_INTR_EN_CLR_j
c. ESM_INT_PRIO_j
d. ESM_PIN_EN_SET_j
e. ESM_PIN_EN_CLR_j

3. Service any pending interrupts in steps 4, 5, and 6 below

Note

The raw status of any pending interrupts may be inconsistent. Servicing the interrupt will return it to consistency via the error group ESM_RAW_j register.

4. Write 0x1 to the appropriate bits in the ESM_ERR_STS register. This step will clear the raw status
   a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
   b. If there are no additional errors, the level interrupt will go low
5. Write the end of interrupt vector to the ESM_EOI register
   a. If there are additional configuration error interrupts pending, then a new pulse will be generated and the level interrupt will remain asserted
   b. If there are no additional low priority error interrupts pending, there will be no new pulse.

12.8.2.4.1.2 ESM Low Priority Error Interrupt

Events mapped to the low priority error interrupt (ESM_INT_LOW_LVL_0) are intended to be events of interest that should be addressed eventually, not events that require immediate attention. An example would be an event indicating a corrected error. The system may want to track this for statistical purposes, but it does not require immediate attention.

Any error event can be mapped to the low priority error interrupt. A low priority error interrupt will be generated when an event is enabled to cause an interrupt (via the ESM_INTR_EN_SET_j register) and mapped to the low priority error interrupt (via ESM_INT_PRIO_j register) and the raw status is set (via the ESM_RAW_j register).

When a low priority error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM_LOW_PRI register
   a. If both [31-16]PLS and [15-0]LVL bit fields are equal to 0xFFFF, then interrupt is no longer asserted and the interrupt routine has ended.
   b. If either [31-16]PLS or [15-0]LVL bit fields are not equal to 0xFFFF, software has two options for determining what event to service:
      i. First option: Record the value in [31-16]PLS and [15-0]LVL bit fields. Determine which is higher priority. This is based on the global event map number of the highest priority low priority error event

      Note

      The global event map is neither defined by nor maintain in ESM. The global event map must be defined and maintained by software using software determined memory resources. It is conceivable that the global event map will be predefined and loaded during the execution of a secondary boot loader.

      ii. Second option:

Copyright © 2026 Texas Instruments Incorporated

1. Read the ESM_LOW register to determine which event group(s) have pending low priority error interrupts
2. Read the desired error group *j* ESM_STS_j register
3. Identify which low priority interrupt to service

2. Determine, based on the global event map for the device, where the error event came from
3. Service the error event based on the IP's specification:
   a. The system may take several actions including (but not limited to):
      i. Fixing the error
      ii. Resetting the errored IP peripheral
      iii. Resetting the device
      iv. Communicating outside the device via the error pin for outside intervention

The rest of the steps assume that the error has been handled and the system wants to clear the error event. Clearing the error event depends on whether the event is a level (see Section 12.8.2.4.1.2.1) or pulse (see Section 12.8.2.4.1.2.2) event.

12.8.2.4.1.2.1 ESM Low Priority Error Level Event

When a low priority error level event has to be cleared, the acting processor must perform the following steps:

1. Clear the low priority error level event at the source
2. Write 0x1 to the appropriate bit in the error group *j* of the ESM_STS_j register. This step will clear the raw status
   a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
   b. If there are no error events, the level will de-assert

Note

There is a possible software race condition if software manages to write to the Clear register before the de-asserted level from the source has been synchronized to the ESM clock. If this is an issue, software may perform a read-back at the source IP before writing the clear register to insure order.

3. Write the end of interrupt vector to the ESM_EOI interrupt register
   a. If there are low priority error level events enabled and pending, then a new pulse will be generated
   b. If there are no additional low priority error level events enabled and pending, there will be no new pulse
4. Write a CLEAR (0x5) to the ESM_PIN_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group *j*ESM_PIN_EN_SET_j register), but may be done regardless as an extra CLEAR is not harmful.

12.8.2.4.1.2.2 ESM Low Priority Error Pulse Event

When a low priority error pulse event has to be cleared, the acting processor must perform the following steps:

1. Write 0x1 to the appropriate bit in the error group *j* of the ESM_STS_j register. This will clear the raw status and will de-assert the level interrupt.
2. Write the end of interrupt vector to the ESM_EOI interrupt register
   a. If there are additional low priority error pulse events enabled and pending, then a new pulse will be generated and the level interrupt will remain asserted
   b. If there are no additional low priority error pulse events enabled and pending, there will be no new pulse
3. Clear the error event at the source. The source may generate a new pulse which will show up as a new error event at the ESM

Copyright © 2026 Texas Instruments Incorporated

4. Write a CLEAR (0x5) to the ESM_PIN_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group *j*ESM_PIN_EN_SET_j register), but may be done regardless as an extra CLEAR is not harmful.

12.8.2.4.1.3 ESM High Priority Error Interrupt

Events mapped to the high priority error interrupt are intended to be events that require immediate intervention from the system because a potentially dangerous error has occurred. An example would be an event indicating an uncorrected error. The system will want to diagnose the issue and intervene to ensure there are no violations.

Any error event can be mapped to the high priority error interrupt. A high priority error interrupt will be generated when an event is enabled to cause an interrupt (via the ESM_INTR_EN_SET_j) register and mapped to the high priority interrupt (via the ESM_INT_PRIO_j) register and the raw status is set (via the ESM_RAW_j).

When a high priority error interrupt is received, the acting processor must perform the following steps:

1. Read the ESM_HI_PRI register
   a. If both [31-16] PLS and [15-0] LVL bit fields are equal to 0xFFFF, then interrupt is no longer asserted and the interrupt routine has ended.
   b. If either [31-16] PLS or [15-0] LVL bit fields are not equal to 0xFFFF, software has two options for determining what event to service:
      i. First option: Record the value in [31-16] PLS and [15-0] LVL bit fields. Determine which is higher priority. This is based on the global event map number of the highest priority high priority error event.

Note

The global event map is neither defined by nor maintain in ESM. The global event map must be defined and maintained by software using software determined memory resources. It is conceivable that the global event map will be predefined and loaded during the execution of a secondary boot loader.

      ii. Second option:
         1. Read the ESM_HI register to determine which event group(s) have pending high priority error interrupts
         2. Read the desired error group *j*ESM_STS_j register
         3. Identify which high priority error interrupt to service

2. Determine, based on the global event map for the device, where the error event came from
3. Service the error event based on the IP's specification:
   a. The system may take several actions including (but not limited to):
      i. Fixing the error
      ii. Resetting the errored IP peripheral
      iii. Resetting the device
      iv. Communicating outside the device via the error pin for outside intervention

The rest of the steps assume that the error has been handled and the system wants to clear the error event. Clearing the error event depends on whether the event is a level (see Section 12.8.2.4.1.3.1) or pulse (see Section 12.8.2.4.1.3.2) event.

12.8.2.4.1.3.1 ESM High Priority Error Level Event

When a high priority error level event has to be cleared, the acting processor must perform the following steps:

1. Clear the error event at the source
2. Write 0x1 to the appropriate bit in the error group *j* of the ESM_STS_j register. This step will clear the raw status

Copyright © 2026 Texas Instruments Incorporated

a. If the error event is still asserted (or re-asserted) the raw status will be set back to 0x1
b. If there are no error events, the level will de-assert

Note

There is a possible software race condition if software manages to write to the Clear register before the de-asserted level from the source has been synchronized to the ESM clock. If this is an issue, software may perform a read-back at the source IP before writing the clear register to insure order.

3. Write the end of interrupt vector to the ESM_EOI interrupt register
   a. If there are high priority error level events enabled and pending, then a new pulse will be generated
   b. If there are no additional high priority error level events enabled and pending, there will be no new pulse
4. Write a CLEAR (0x5) to the ESM_PIN_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group *j*ESM_PIN_EN_SET_j register, but may be done regardless as an extra CLEAR is not harmful.

12.8.2.4.1.3.2 ESM High Priority Error Pulse Event

When a high priority error pulse event has to be cleared, the acting processor must perform the following steps:

1. Write 0x1 to the appropriate bit in the error group *j* of the ESM_STS_j register. This will clear the raw status and will de-assert the level interrupt.
2. Write the end of interrupt vector to the ESM_EOI interrupt register
   a. If there are high priority error pulse events enabled and pending, then a new pulse will be generated and the level interrupt will remain asserted
   b. If there are no additional high priority error pulse events enabled and pending, there will be no new pulse
3. Clear the error event at the source. The source may generate a new pulse which will show up as a new error event at the ESM
4. Write a CLEAR (0x5) to the ESM_PIN_CTRL register. This step is optional if the event is not enabled to influence the error pin (error group *j*ESM_PIN_EN_SET_j register), but may be done regardless as an extra CLEAR is not harmful.

12.8.2.4.2 ESM Error Event Inputs

The ESM can have up to 1024 error event inputs, configurable by groups of 32. For the mapping of system interrupt error events to ESM interrupt inputs, see *Interrupt Sources*.

Level error events (active high) are synchronized to the ESM clock. This synchronized value is captured in to a flop.

Pulse error events use rising edge detection. Each pulse error event has 3 redundant inputs. Each input has its own edge detection logic. Multiple transmission protects against Single Event Upsets (SEUs, transient errors) causing a pulse to be lost during transmission and against failure of the edge detection logic. Once an edge has been detected on any of the three inputs, the ESM_RAW_j status is set. Subsequent pulses are likely to come concurrently or quickly enough that software will not have reacted yet. This logic is intentionally biased against false negatives and towards false positives. An SEU that causes an event where none actually occurred will cause software to be called in to action. Software must observe that there is no real error and clear the false status.

12.8.2.4.3 ESM Error Pin Output

The error pin output (ERR_O) is used to signal an external agent that it needs to (or may need to) intervene because of an error. Each error event input can be programmed, via software, to influence the error pin output (via the ESM_PIN_EN_SET_j register). The error pin output is active low or PWM based on the ESM_EN[7-4] PWM_EN field. This bit field should only be modified when the ESM is disabled, based on the ESM_EN register.

Copyright © 2026 Texas Instruments Incorporated

During Power-On Reset (POR), the error pin is active (asserted low) and the device drives this via a weak internal pull-down. The I/O is under the control of the device. When POR is removed from the ESM, it will be driving the error pin so the device can hand over control to the ESM. The user may also add an external pull-down that is only active when the device is in reset.

During a warm reset the state of the error pin is unchanged, that is the error pin logic is only reset by a POR. The device leaves the I/O active during a warm reset.

The ESM has also a software error forcing capability on the error pin. That is, a force error can be set via the ESM_EN[3-0] KEY bit field.

CAUTION

The isolation value for the ERR_O output of ESM is active (0). This is intended and supposed to protect against an accidental transition to a low power state. If the actual low power mode transition is intended, the PMIC should be made aware by software to ignore the ESM error signal. Otherwise device resets will be asserted from companion chip.

Figure 12-465 describes the behavior of the error pin. Not shown is that a reset (Power-On-Reset only) will immediately transition the error pin to the ESM_RESET state and a Global Soft Reset will immediately transition the error pin to the ESM_IDLE state. A pending error interrupt event is any error event with the raw state set and the error pin Influence enabled. There are two types of "clear" events associated with servicing the error pin. The first is to clear the status of the pending event (see Section 12.8.2.4.1 for how to clear level and pulse pending events). The second is the CLEAR event meant to de-assert the error pin.

Copyright © 2026 Texas Instruments Incorporated

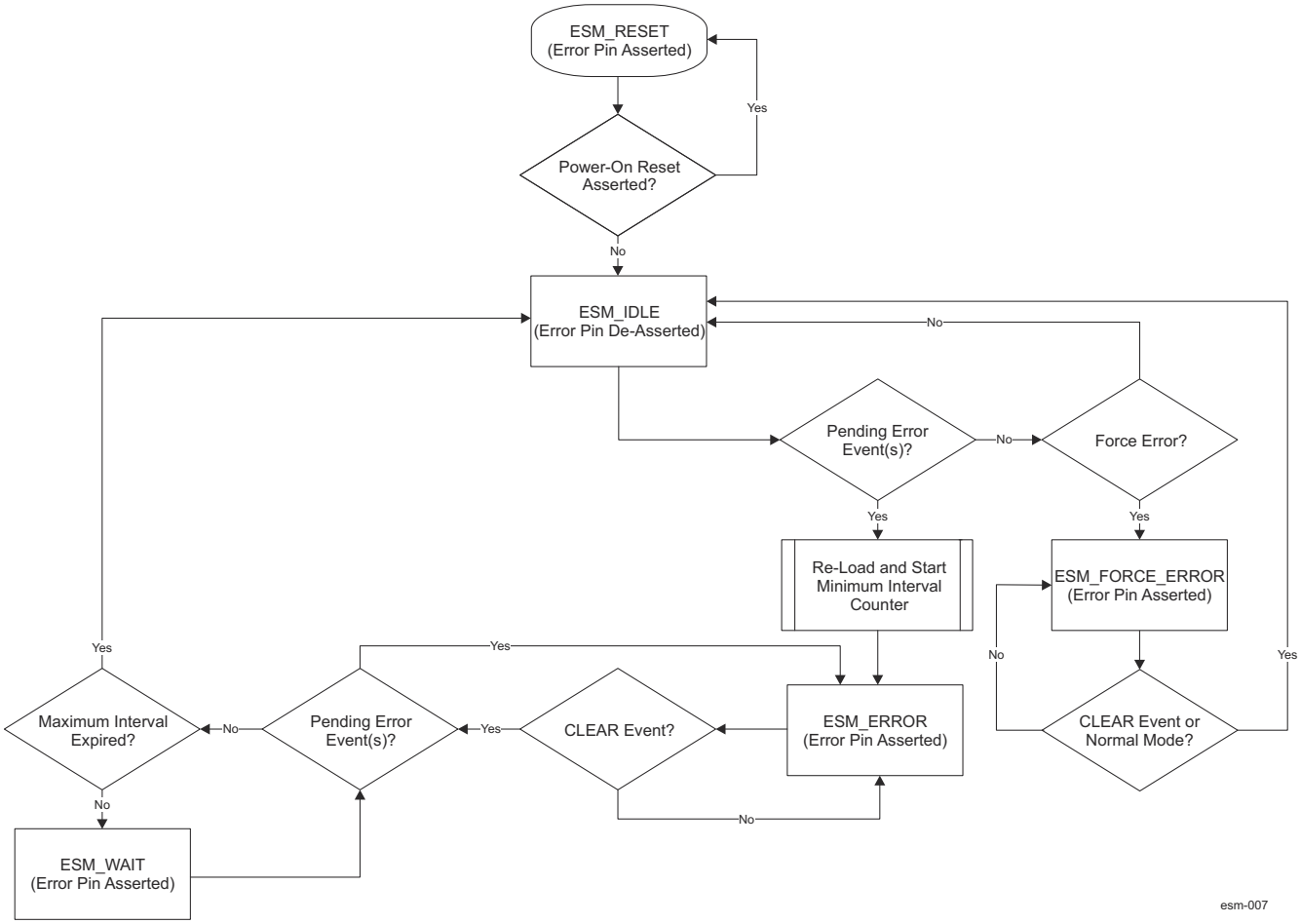


Figure 12-465. ESM Error Pin State Flowchart

If an error event happens that has been programmed to influence the error pin, the error pin will assert (active low) for a minimum time (as programmed by the ESM_PIN_CNTR_PRE register). In order for the error pin to de-assert, the following 3 things must happen:

1. The minimum time interval must expire
2. The event that caused the error pin to assert must be cleared (see Section 12.8.2.4.1)
3. A CLEAR (0x5) must be written to the ESM_PIN_CTRL register.

Note

Step 3 should happen after step 2, but either (or both) of these steps may happen before or after step 1.

Figure 12-466 shows a typical error pin assertion.

Copyright © 2026 Texas Instruments Incorporated

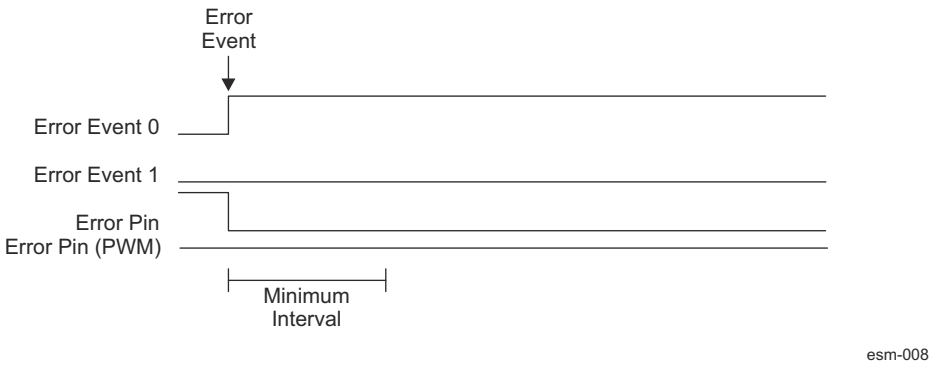


Figure 12-466. ESM Error Pin Assertion

If, during the minimum time, CLEAR is written to the error key, then the error pin will de-assert after the minimum time interval, as shown in Figure 12-467.

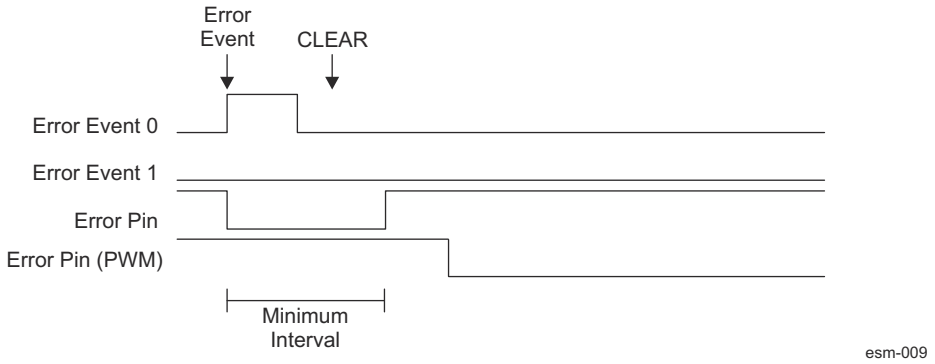


Figure 12-467. ESM Error Pin Assertion with CLEAR during Minimum Interval

If CLEAR is not written till after the minimum time interval, the error pin will de-assert when CLEAR is written. This is regardless of whether the error interrupt event itself is removed before or after the minimum time interval, as shown by the dotted line in Figure 12-468.

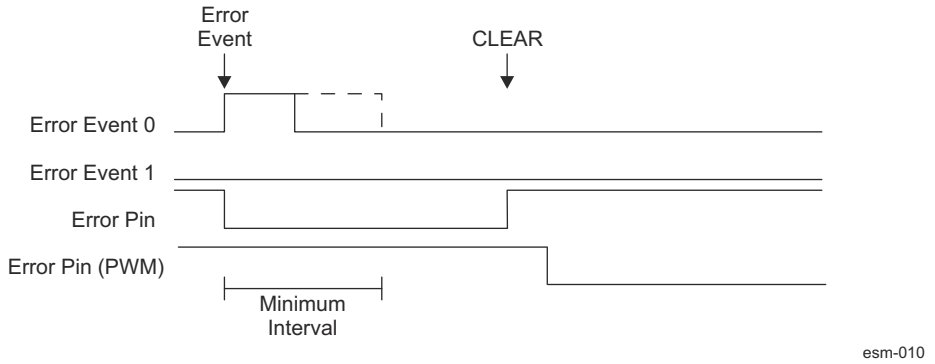


Figure 12-468. ESM Error Pin Asserting with CLEAR after Minimum Interval

When in the ESM_ERROR state and a CLEAR event happen, if there are still pending error events, the ESM stays in the ESM_ERROR state with the error pin asserted. Multiple error events when in the ESM_ERROR state do not reset the minimum time interval counter as shown in Figure 12-469.

Copyright © 2026 Texas Instruments Incorporated

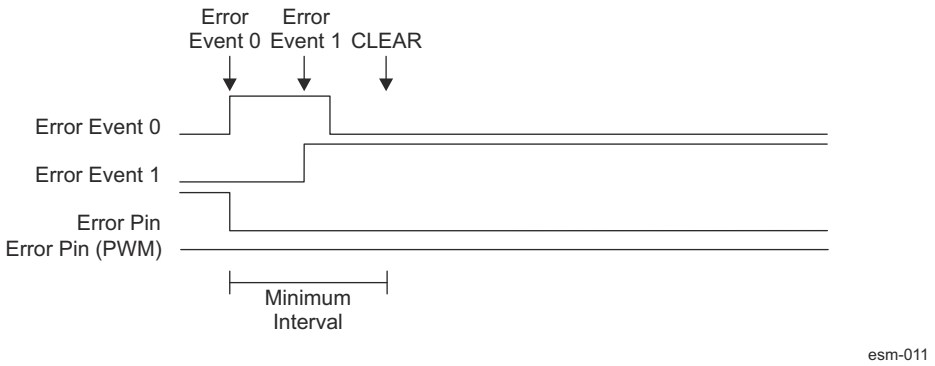


Figure 12-469. ESM Error Pin Asserting with Interval Reset by Additional Error Event(s)

A CLEAR event causes a re-evaluation of whether there are any pending error events. As such, a single CLEAR can be used to clear the error pin after multiple error events. Multiple CLEAR events can occur (such as the one with the dotted arrow shown in Figure 12-470), but are not necessary. No matter how many error events occur nor when (or how many) CLEAR events occur, the error pin will always be asserted for at least the minimum time interval

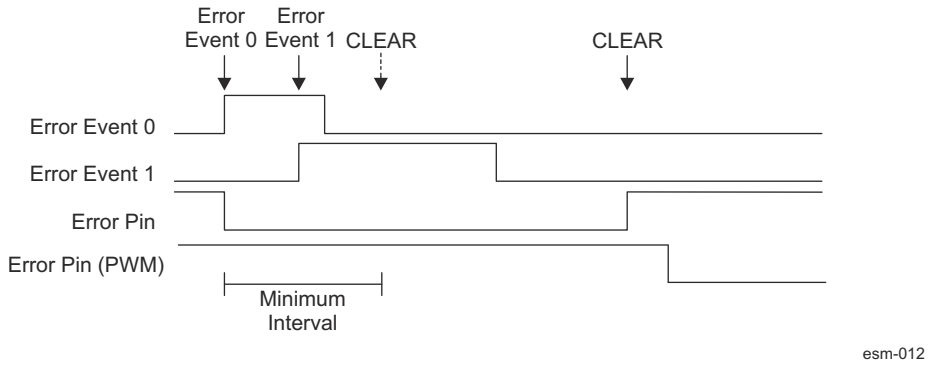


Figure 12-470. ESM Error Pin Asserting with Single CLEAR for Multiple Events

If all error events are cleared and the ESM is in the ESM_WAIT state, waiting for the minimum time interval to expire, and a new error interrupt event occurs, the ESM will go back to the ESM_ERROR state. The minimum time interval will not reset, but a new CLEAR event will be required as shown in Figure 12-471.

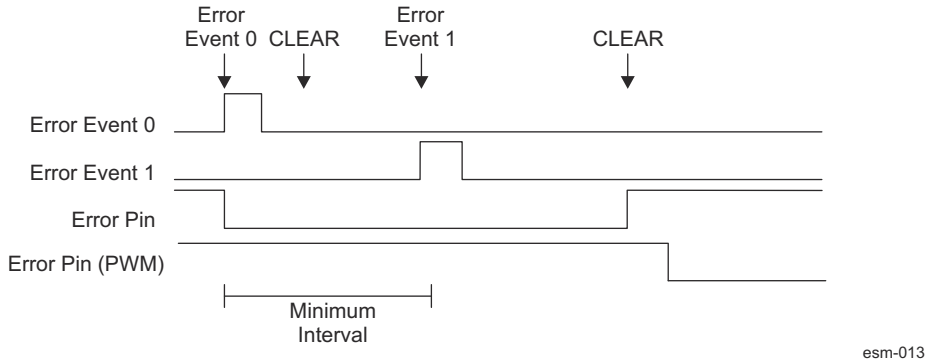


Figure 12-471. ESM Error Pin Asserting with New Error During Minimum Time Interval

Table 12-369 shows some common scenarios of how the error pin as well as the two associated registers (ESM_PIN_CTRL and ESM_PIN_STS) will be set.

Copyright © 2026 Texas Instruments Incorporated

Table 12-369. ESM Error Pin Scenarios

Scenario	Error Pin State Value	ESM_PIN_CTRL[3-0] KEY	ESM_PIN_STS[0] VAL status value	Additional Notes
POR Asserted	0	N/A	N/A	Registers are inaccessible. Device disables the I/O and pulls down internally.
After de-assertion of POR	1	0x0 (Normal Mode)	0x0	-
After de-assertion of Warm Reset (error was not asserted when reset asserted)	1	0x0 (Normal Mode)	0x0	-
After de-assertion of Warm Reset (error was asserted when reset asserted)	0	0x0 (Normal Mode)	0x1	-
Force error pin	0	0xA (Force Error Mode)	0x0	Forcing error on the pin via software.

12.8.2.4.4 PWM Mode

If the error output pin is in PWM mode then when no error is detected it will toggle according to programmable MMR widths for high and low periods. When an error occurs, the error pin stops toggling and remains constant until the error is cleared. An external PMIC that is detecting the PWM toggles can identify the error if the pin stops toggling. The periods should be programmed such that they fit within the expectation of the external PMIC.

12.8.2.4.5 ESM Minimum Time Interval

The minimum time interval is the minimum amount of time that the error pin will be asserted (active low) when an enabled error interrupt event happens. This value is system dependent, but should be enough time so that the external monitoring agent can always see the error pin asserted, but short enough so that if all of the error events are cleared, then the error pin can be de-asserted before the external agent decides to intervene. This is highly dependent on the application and the Fault Tolerant Time Interval.

The Minimum Time Interval counter is clock cycle based, therefore the time of the interval is a combination of the value in the ESM_PIN_CNTR_PRE register and the clock frequency of the ESM. Software must calculate the value accordingly. The Minimum Time Interval should be set according to the needs of the application.

12.8.2.4.6 ESM Protection for Registers

The configuration for each error group *j* of registers are backed up by 3 flops in order to protect against single or double-bit errors. When written, all 3 bits are set to the same value. When read (and for functioning of the internal state machines) the value is the OR of all 3 bits. Whenever any of the bits disagree, the Configuration Error interrupt is asserted (if enabled). The registers covered by this mechanism are:

- ESM_ERR_RAW
- ESM_ERR_STS
- ESM_ERR_EN_SET
- ESM_ERR_EN_CLR
- ESM_RAW_j
- ESM_STS_j
- ESM_INTR_EN_SET_j
- ESM_INTR_EN_CLR_j
- ESM_INT_PRIO_j
- ESM_PIN_EN_SET_j
- ESM_PIN_EN_CLR_j

The error pin control register ESM_PIN_CTRL contains a multi-bit field KEY. The key value ensures normal operation on the error pin and that an error even will be generated if one occurs. Software should periodically read check the KEY bit field value and make sure it is 0x0. If the value is not 0x0, software must re-write it to this key value (unless in test mode forcing an error on the pin) to ensure the normal operation. Table 12-370 lists the KEY values and their respective meaning.

Copyright © 2026 Texas Instruments Incorporated

Table 12-370. ESM Error Pin Control Values

ESM_PIN_CTRL[3-0] KEY	Description
N/A	Registers are inaccessible. Device disables the I/O and pulls down internally.
0x0 (Normal)	Normal operation mode - Error pin will activate when an enabled error event occurs.
0xA (Force Error)	Force error mode - Forces the error pin active. To clear the error pin (return to the ESM_IDLE state) write this field back to normal mode (writing a CLEAR event will also work). Force error mode must be set only while in IDLE. Attempting force error while in another state will have no effect.
0x5 (CLEAR)	CLEAR Event - generates a CLEAR event to the ESM state machine. KEY will return to normal mode (0x0) on the next cycle.
Other Values	All other values - Normal mode. Writing any of these values will have no effect. When reading any of these values indicates that one or more bits have experienced a single event upset, software should write the field back to 0x0. The ESM will continue to operate in normal mode.

The error pin counter pre-load register ESM_PIN_CNTR_PRE should also be read and checked periodically by software.

12.8.2.4.7 ESM Clock Stop

The ESM can only be put in to clock stop if all of the internal state machines are idle and the [3-0] KEY bit field for the global enable command is cleared in the ESM_EN register.

Copyright © 2026 Texas Instruments Incorporated

12.8.3 Memory Cyclic Redundancy Check (MCRC) Controller

This chapter describes the Memory Cyclic Redundancy Check (MCRC) controller in the device.

12.8.3.1 MCRC Overview

VBUSM CRC controller is a module which is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of a memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into MCRC Controller. The responsibility of MCRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a pre-determined good signature value. MCRC controller provides four channels to perform CRC calculation on multiple memories in parallel and can be used on any memory system.

12.8.3.1.1 MCRC Features

MCRC has the following features:

- Four channels to perform background signature verification on any memory subsystem
- Data compression on 8-, 16-, 32-, and 64-bit data size
- Maximum-length PSA (Parallel Signature Analysis) register constructed based on 64-bit primitive polynomial
- Each channel has a CRC Value Register which contains the pre-determined CRC value
- Use timed base event trigger from timer to initiate DMA data transfer
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression
- Three modes of operation:
  - Auto
  - Semi-CPU
  - Full-CPU
- For each channel, CRC can be performed either by MCRC Controller or by CPU
- Automatically performs signature verification without CPU intervention in AUTO mode
- Generates interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself
- Generates CRC fail interrupt in AUTO mode if signature verification fails
- Generates Timeout interrupt if CRC is not performed within the time limit
- Generates DMA request per channel to initiate CRC value transfer
- An 128-byte block burst address for the PSA register to DMA without constant mode bus attribute

12.8.3.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

Copyright © 2026 Texas Instruments Incorporated

12.8.3.2 MCRC Functional Description

12.8.3.2.1 MCRC Block Diagram

Figure 12-472 shows the MCRC internal blocks.

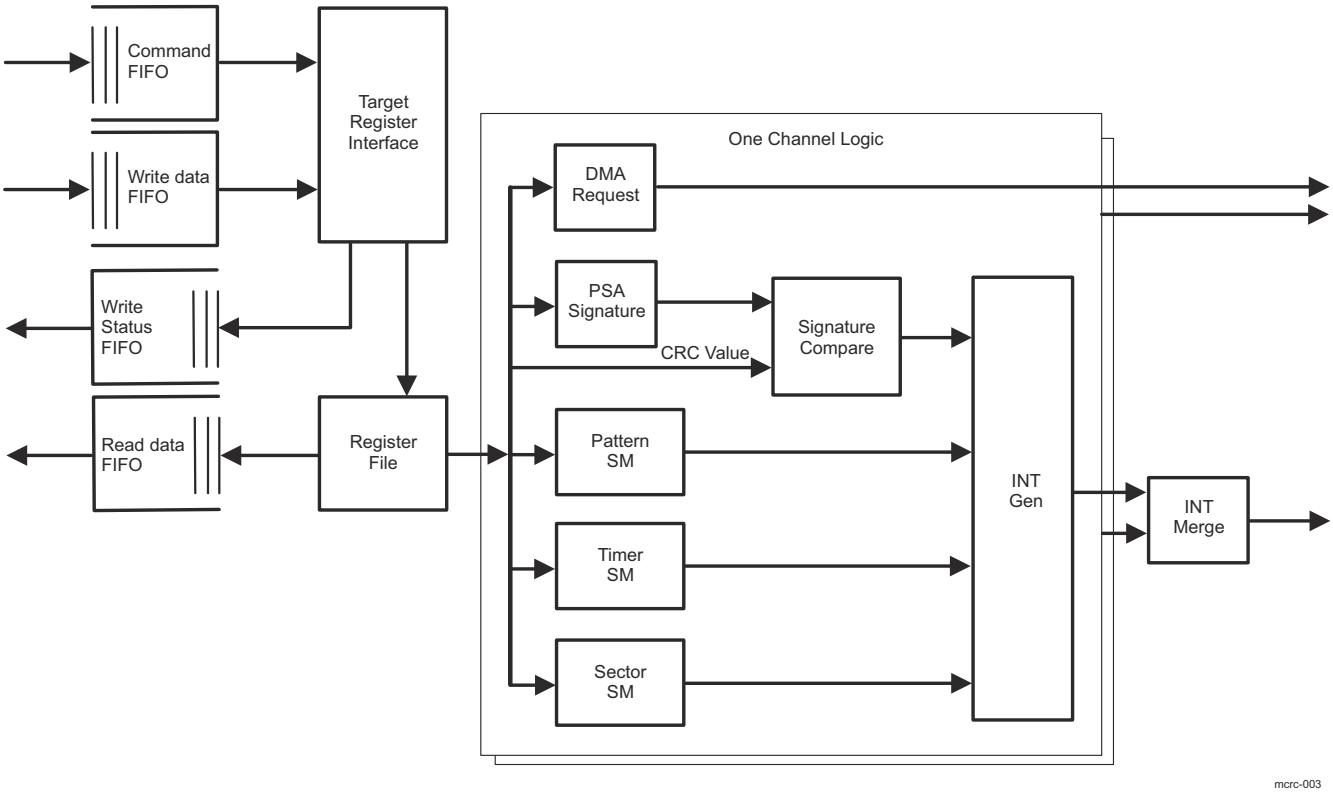


Figure 12-472. MCRC Block Diagram

- **Command FIFO**: The Command FIFO pipelines the commands to the target register interface. The Command and Write FIFOs allow the data to be coincident for processing. If there is no space for writes in the Write Status FIFO or no space in the Read Data FIFO for reads, the command processing will be halted until there is space in the appropriate FIFO. This FIFO is 4 elements deep.
- **Write FIFO**: The Write FIFO pipelines the write data to the target register interface. The Command and Write FIFOs allow the data to be coincident for processing. If there is no space for writes in the Write Status FIFO or no space in the Read Data FIFO for reads, the command processing will be halted until there is space in the appropriate FIFO. This FIFO is 2 elements deep.
- **Write Status FIFO**: The Write Status FIFO pipelines the write status back to the VBUSM. A write status will be issued on the final data phase of a write command. This FIFO is 2 elements deep.
- **Read Data FIFO**: The Read Data FIFO pipelines the read data back to the VBUSM. This FIFO is 3 elements deep.
- **Target Register Interface**: The Target Register Interface directs the written data to the register file.
- **PSA Signature**: The PSA Signature creates the signature of the data written. This data will then be compared to the CRC Value or read by software to determine goodness.
- **Pattern State Machine**: The Pattern State Machine determines when a block of data has been serviced.
- **Timer State Machine**: The Timer State Machine determines when overrun and under-run events are detected.
- **Sector State Machine**: The Sector State Machine determines when a sector error should be captured so the software can determine the errant block of data.
- **Signature Compare**: The Signature Compare block compares the current signature to the CRC Value register and sends the result to the Interrupt Generation block.

Copyright © 2026 Texas Instruments Incorporated

12.8.3.2.2 MCRC General Operation

There are four channels in MCRC controller and for each channel there is a PSA (Parallel Signature Analysis) Signature Register (MCRC64_0_PSA_SIGREGL1-4) and a CRC (Cyclic Redundancy Check) Value Register (MCRC64_0_CRC_REGL1-4).

A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be a 8-, 16-, 32-, or 64-bit data. MCRC module performs the signature calculation and compares the signature to a pre-determined value. The PSA Signature Register compresses an incoming data pattern into a signature when it is written. When one sector of data patterns are written into PSA Signature Register, a final signature corresponding to the sector is obtained. CRC Value Register stores the pre-determined signature corresponding to one sector of data patterns. The calculated signature and the pre-determined signature are then compared to each other for signature verification. To minimize CPU's involvement, data patterns transfer can be carried out at the background of CPU using DMA controller. DMA is setup to transfer data from memory of which the contents to be verified to the memory mapped PSA Signature Register. When DMA transfers data to the memory mapped PSA Signature Register, a signature is generated.

A programmable 20-bit data pattern counter is used for each channel to define the number of data patterns to calculate for each sector. Signature verification can be performed automatically by MCRC controller in AUTO mode or by CPU itself in Semi-CPU or Full-CPU mode. In AUTO mode, a self-sustained CRC signature calculation can be achieved without any CPU intervention.

12.8.3.2.3 MCRC Modes of Operation

MCRC Controller can operate in AUTO, Semi-CPU, and Full-CPU modes.

12.8.3.2.3.1 AUTO Mode

In AUTO mode, MCRC Controller in conjunction with DMA controller can perform CRC without CPU intervention. A sustained transfer of data to both the PSA Signature Register (MCRC64_0_PSA_SIGREGL1-4) and a CRC (Cyclic Redundancy Check) Value Register (MCRC64_0_CRC_REGL1-4) are performed in the background of CPU. When a mismatch is detected, an interrupt is generated to the CPU. A 16-bit, current sector ID register (MCRC64_0_CRC_CURSEC_REG1-4) is provided to identify which sector causes a CRC failure.

12.8.3.2.3.2 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register (MCRC64_0_PSA_SIGREGL1-4). Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register (MCRC64_0_PSA_SECSIGREGL1-4) and compare it to a pre-determined CRC value.

12.8.3.2.3.3 Full-CPU Mode

In Full-CPU mode, the CPU does the data patterns transfer and signature verification all by itself. When CPU has enough throughput, it can perform data patterns transfer by reading data from the memory system to the PSA Signature Register (MCRC64_0_PSA_SIGREGL1). After certain number of data patterns are compressed, the CPU can read from the PSA Signature Register and compare the calculated signature to the pre-determined CRC signature value. In Full-CPU mode, neither interrupt nor DMA request is generated. All counters are also disabled.

12.8.3.2.4 PSA Signature Register

The 64-bit PSA Signature Register (MCRC64_0_PSA_SIGREGL1-4 and MCRC64_0_PSA_SIGREGH1-4) is based on the primitive polynomial found in (EQ 1) to produce the maximum length LFSR (Linear Feedback Shift Register).

Copyright © 2026 Texas Instruments Incorporated

$$f(x) = x^{64} + x^4 + x^3 + x + 1 \quad (41)$$

A. More details of the 64-bit primitive polynomial can be found at W. Stahnke, Primitive binary polynomials, Math. Comp. 27 (1973), 977--980.

There is one PSA Signature Register per CRC channel. PSA Signature Register can be both read and written. When it is written, it can either compress the data or just capture the data depending on the state of CHi_MODE bits (where i = 1 to 4). If CHi_MODE=0x0 (Data Capture), a seed value can be planted in the PSA Signature Register without compression. Other modes other than Data Capture will result with the data compressed by PSA Signature Register when it is written. Each channel can be planted with different seed value before compression starts. When PSA Signature Register is read, it gives the calculated signature.

MCRC Controller should be used in conjunction with the on-chip DMA controller to produce optimal system performance. The incoming data pattern to PSA Signature Register is typically initiated by the DMA controller. When DMA is properly setup, it would read data from the pre-determined memory system and write them to the memory mapped PSA Signature Register. Each time PSA Signature Register is written a signature is generated. CPU itself can also perform data transfer by reading from the memory system and perform write operation to PSA Signature Register if CPU has enough throughput to handle data patterns transfer.

After a system reset and when AUTO mode is enabled, MCRC Controller automatically generates a DMA request to request the pre-determined CRC value corresponding to the first sector of memory to be checked.

In AUTO mode, when one sector of data patterns is compressed, the signature stored at the PSA Signature Register is first copied to the PSA Sector Signature Register (MCRC64_0_PSA_SECSIGREGL1-4) and PSA Signature Register is then cleared out to all zeros. An automatic signature verification is then performed by comparing the signature stored at the PSA Sector Signature Register to the CRC Value Register (MCRC64_0_CRC_REGL1-4). After the comparison the MCRC Controller can generate a DMA request. Upon receiving the DMA request the DMA controller will update the CRC Value Register by transferring the next pre-determined signature value associated with the next sector of memory system. If the signature verification fails then MCRC Controller can generate a CRC fail interrupt.

In Full-CPU mode, no DMA request and interrupt are generated at all. The number of data patterns to be compressed is determined by CPU itself. Full-CPU mode is useful when DMA controller is not available to perform background data patterns transfer. Software can periodically generate a software interrupt to CPU and use CPU to accomplish data transfer and signature verification.

MCRC Controller supports double word, word, half word and byte access to the PSA Signature Register. During a non-doubleword write access, all unwritten byte lanes are padded with zeros before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register is always in 64 bits because a compressed value is always expressed in 64 bits.

There is a software reset per channel for PSA Signature Register. When set, the PSA Signature Register is reset to all zeros.

PSA Signature Register is reset to zero under the following conditions:

- System reset
- PSA Software reset
- One sector of data patterns are compressed

12.8.3.2.5 PSA Sector Signature Register

After one sector of data is compressed, the final resulting signature calculated by PSA Signature Register is transferred to the 64-bit PSA Sector Signature Register (MCRC64_0_PSA_SECSIGREGL1-4 and MCRC64_0_PSA_SECSIGREGH1-4). PSA Signature Register is a read-only register. During Semi-CPU mode, the host CPU must read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

Copyright © 2026 Texas Instruments Incorporated

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enabled to generate when overrun condition occurs.

12.8.3.2.6 CRC Value Register

Associated with each channel there is a 64-bit CRC Value Register (MCRC64_0_CRC_REGL1-4 and MCRC64_0_CRC_REGH1-4).

The CRC Value Register stores the pre-determined CRC value. After one sector of data patterns is compressed by PSA Signature Register, MCRC Controller can automatically compare the resulting signature stored at the PSA Sector Signature Register with the pre-determined value stored at the CRC Value Register if AUTO mode is enabled. If the signature verification fails, MCRC Controller can be enabled to generate an CRC fail interrupt. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. CPU must not read from the CRC Value Register during Semi-CPU or Full-CPU mode because the CRC Value Register is not updated during these two modes.

In AUTO mode, for first sector's signature, DMA request is generated when mode is programmed to AUTO. For subsequent sectors, DMA request is generated after each sector is compressed. Responding to the DMA request, DMA controller reloads the CRC Value Register for the next sector of memory system to be checked. The user software needs to configure the DMA to ensure that the DMA first writes to the MCRC64_0_CRC_REGL1 followed by the MCRC64_0_CRC_REGH1 register in during the AUTO Mode.

When CRC Value Register is updated with a new CRC value, an internal flag is set to indicate that CRC Value Register contains the most current value. This flag is cleared when CRC comparison is performed. Each time at the end of the final data pattern compression of a sector, MCRC Controller first checks to see if the corresponding CRC Value Register has the most current CRC value stored in it by polling the flag. If the flag is set then the CRC comparison can be performed. If the flag is not set then it means the CRC Value Register contains stale information. A CRC underrun interrupt is generated. When an underrun condition is detected, signature verification is not performed.

MCRC Controller supports double word, word, half word and byte access to the CRC Value Register. As noted before comparison between PSA Sector Signature Register and CRC Value Register during AUTO mode is carried out in 64 bits.

12.8.3.2.7 Raw Data Register

The raw or un-compressed data written to the PSA Signature Register is also saved in the 64-bit Raw Data Register (MCRC64_0_RAW_DATAREGL1-4 and MCRC64_0_RAW_DATAREGH1-4). This register is read only.

12.8.3.2.8 Example DMA Controller Setup

DMA controller needs to be setup properly in either AUTO or Semi-CPU mode as DMA controller is used to transfer data patterns. Hardware or a combination of hardware and software DMA triggering are supported.

12.8.3.2.8.1 AUTO Mode Using Hardware Timer Trigger

There are two DMA channels associated with each CRC channel when in AUTO mode. One DMA channel is setup to transfer data patterns from the source memory to the PSA Signature Register (MCRC64_0_PSA_SIGREGL1-4). The second DMA channel is setup to transfer the pre-determined signature to the CRC Value Register (MCRC64_0_CRC_REGL1-4). The trigger source for the first DMA channel can be either by hardware or by software. As illustrated in Figure 12-473, a timer can be used to trigger a DMA request to initiate transfer from the source memory system to PSA Signature Register. In AUTO mode, MCRC Controller

Copyright © 2026 Texas Instruments Incorporated

also generates DMA request after one sector of data patterns is compressed to initiate transfer of the next CRC value corresponding to the next sector of memory. Thus a new CRC value is always updated in the CRC Value Register (MCRC64_0_CRC_REGL1-4)by DMA synchronized to each sector of memory.

A block of system memory is usually divided into many sectors. All sectors are the same size.
The sector size is programmed in the MCRC64_0_CRC_PCOUNT_REG1-4 and the number of sectors in one block is programmed in the MCRC64_0_CRC_SCOUNT_REG1-4 of the respective channel. MCRC64_0_CRC_PCOUNT_REG1-4 multiplies MCRC64_0_CRC_SCOUNT_REG1-4 and multiplies transfer size of each data pattern should give the total block size in number of bytes.

The total size of the memory system to be examined is also programmed in the respective transfer count register inside DMA module. The DMA transfer count register is divided into two parts. They are element count and frame count. Note that a hardware DMA request can be programmed to trigger either one frame or one entire block transfer. In Figure 12-473, a hardware DMA request from a timer is used as a trigger source to initiate DMA transfer. If all four CRC channels are active in AUTO mode then a total of four DMA requests would be generated by MCRC Controller.

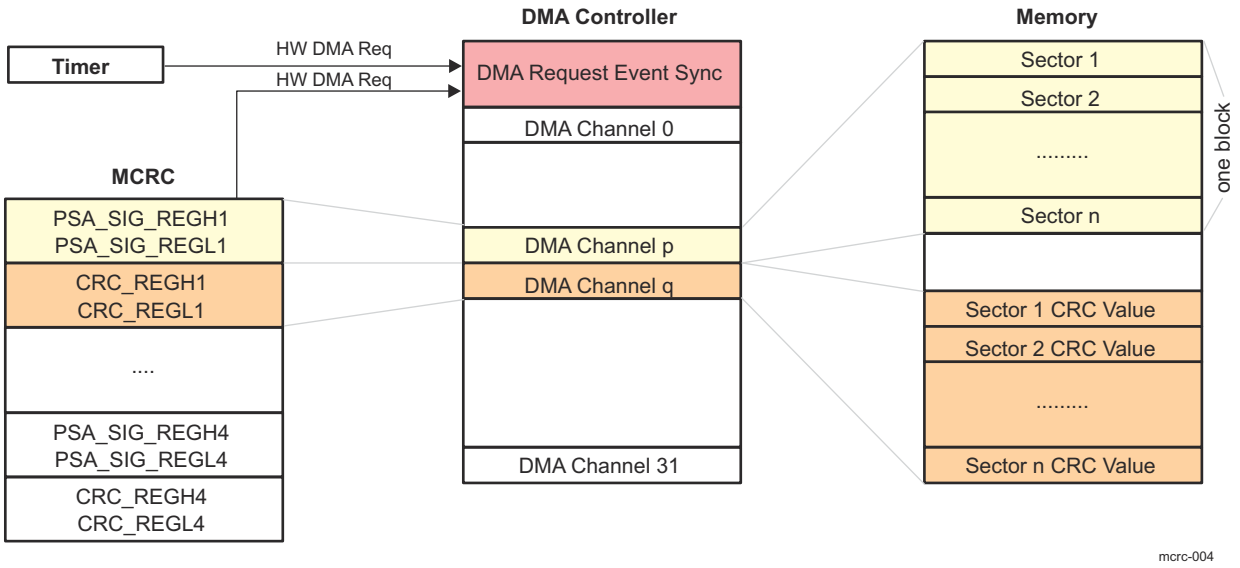


Figure 12-473. AUTO Mode Using Hardware Timer Trigger

12.8.3.2.8.2 AUTO Mode Using Software Trigger

The data patterns transfer can also be initiated by software. CPU can generate a software DMA request to activate the DMA channel to transfer data patterns from source memory system to the PSA Signature Register. To generate a software DMA request CPU needs to set the corresponding DMA channel in the DMA software trigger register. Note that just one software DMA request from CPU is enough to complete the entire data patterns transfer for all sectors. Please see Figure 12-474 for illustration.

Copyright © 2026 Texas Instruments Incorporated

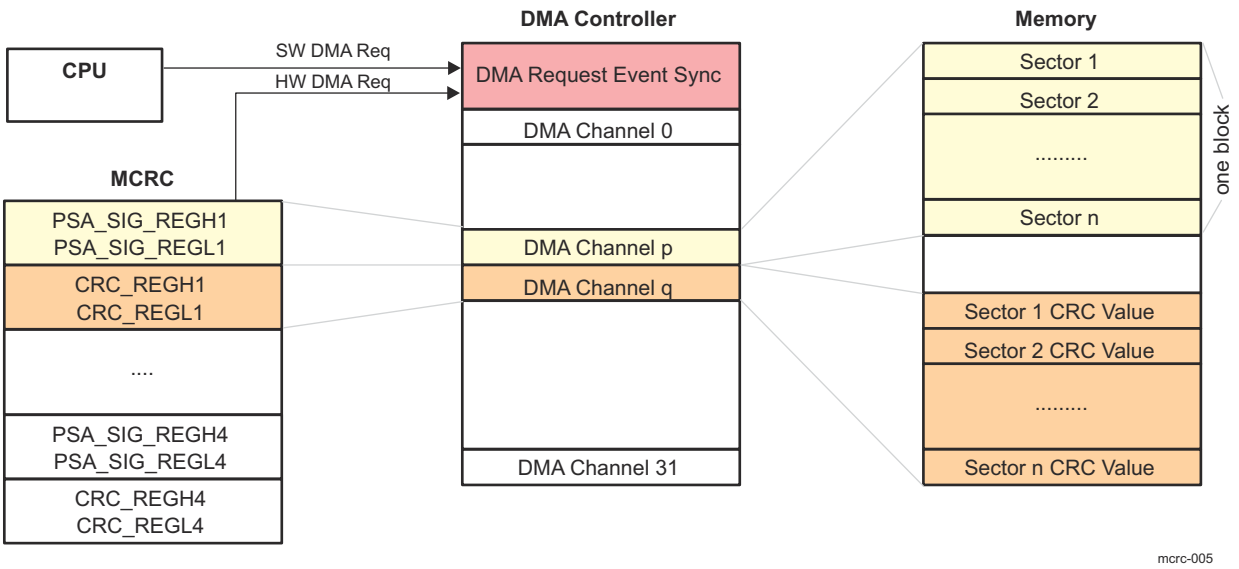


Figure 12-474. AUTO Mode With Software CPU Trigger

12.8.3.2.8.3 Semi-CPU Mode Using Hardware Timer Trigger

During Semi-CPU mode, no DMA request is generated by CRC controller. Therefore, no DMA channel is allocated to update CRC Value Register. CPU should not read from CRC Value Register in semi-CPU mode as it contains stale value. Note that no signature verification is performed at all during this mode. Similar to AUTO mode, either by hardware or by software DMA request can be used as a trigger for data patterns transfer. Figure 12-475 illustrates the DMA setup using semi-CPU mode with hardware timer trigger.

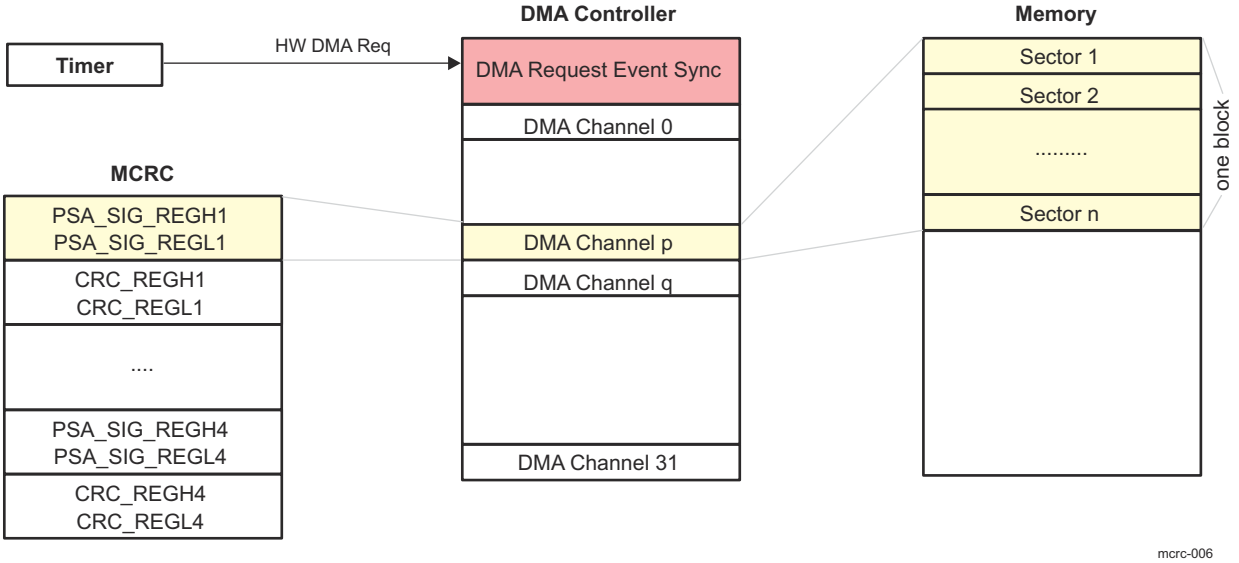


Figure 12-475. Semi-CPU Mode With Hardware Timer Trigger

Table 12-371. DMA Request and Counter Logic Operation According to CRC Mode

CRC Mode	DMA Request	Pattern Counter	Sector Counter	Timeout Counter
AUTO	Active	Active	Active	Active
Semi-CPU	Inactive	Active	Active	Active
Full CPU	Inactive	Inactive	Inactive	Inactive

Copyright © 2026 Texas Instruments Incorporated

12.8.3.2.9 Pattern Count Register

There is a 20-bit data pattern counter for every CRC channel. The data pattern counter is a down counter and can be pre-loaded with a programmable value stored in the Pattern Count Register (MCRC64_0_CRC_PCOUNT_REG1-4). When the data pattern counter reaches zero, a compression complete interrupt is generated in Semi-CPU mode and an automatic signature verification is performed in AUTO mode. In AUTO only, DMA request is generated to trigger the DMA controller to update the CRC Value Register.

Note

The data pattern count must be divisible by the total transfer count as programmed in DMA controller. The total transfer count is the product of element count and frame count.

12.8.3.2.10 Sector Count Register/Current Sector Register

Each channel contains a 16-bit sector counter. The sector count register stores the number of sectors. Sector counter is a free running counter and is incremented by one each time when one sector of data patterns is compressed. When the signature verification fails, the current value stored in the sector counter is saved into current sector register (MCRC64_0_CRC_CURSEC_REG1-4). If signature verification fails, CPU can read from the current sector register to identify the sector which causes the CRC mismatch. To aid and facilitate the CPU in determining the cause of a CRC failure it is advisable to use the following equation during CRC and DMA setup.

CRC Pattern Count × CRC Sector Count = DMA Element Count × DMA Frame Count (42)

The current sector register is frozen from being updated until both the current sector register is read and CRC fail (CHi_CRC_FAIL) status bit is cleared by CPU. If CPU does not respond to the CRC failure in a timely manner before another sector produces a signature verification failure, the current sector register is not updated with the new sector number. An overrun interrupt is generate instead. If current sector register is already frozen with an erroneous sector and emulation is entered with SUSPEND signal goes to high then the register still remains frozen even it is read.

In Semi-CPU mode, the current sector register is used to indicate the sector for which the compression complete has last happened.

Current sector register is reset when the PSA software reset is enabled.

Note

Both data pattern count and sector count registers must be greater than or equal to one for the counters to count. After reset, pattern count and sector count registers default to zero and the associated counters are inactive.

12.8.3.2.11 Interrupts

CRC generate several types of interrupts per channel. Associated with each interrupt there is a interrupt enable bit (see MCRC64_0_CRC_INTS). No interrupt is generated in Full-CPU mode.

- Compression complete interrupt
- CRC fail interrupt
- Overrun interrupt
- Underrun interrupt
- Timeout interrupt

Table 12-372. Interrupt Conditions Per CRC Mode

CRC Mode	Compression Complete	CRC Fail	Overrun	Underrun	Timeout
AUTO	No	Yes	Yes	Yes	Yes
Semi-CPU	Yes	No	Yes	No	Yes

Copyright © 2026 Texas Instruments Incorporated

Table 12-372. Interrupt Conditions Per CRC Mode (continued)

CRC Mode	Compression Complete	CRC Fail	Overrun	Underrun	Timeout
Full-CPU	No	No	No	No	No

12.8.3.2.11.1 Overrun Interrupt

Overrun Interrupt is generated in either AUTO or Semi-CPU mode. During AUTO mode, if a CRC fail is detected then the current sector number is recorded in the current sector register (MCRC64_0_CRC_CURSEC_REG1-4). If CRC fail status bit is not cleared and current sector register is not read by the host CPU before another CRC fail is detected for another sector then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time the signature is copied into the PSA Sector Signature Register (MCRC64_0_PSA_SECSIGREGL1-4). If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value then an overrun interrupt is generated.

12.8.3.2.11.2 Timeout Interrupt

To ensure that the memory system is examined within a pre-defined time frame and no loss of incoming data there is a 24-bit timeout counter per CRC channel. The timeout counter can be pre-loaded with two different pre-load values, watchdog timeout pre-load value (MCRC64_0_CRC_WDTOPLD1-4) and block complete timeout pre-load value (MCRC64_0_CRC_BCTOPLD1-4). The timeout counter is clocked by a prescaler clock which is permanently running at division 64 of FICLK clock.

Watchdog timeout pre-load register (MCRC64_0_CRC_WDTOPLD1-4) is used to check if DMA does supply a block of data responding to a request in a given time frame. Block complete timeout pre-load register (MCRC64_0_CRC_BCTOPLD1-4) is used to check if one complete block of data patterns are compressed within a specific time frame. The timeout counter is first pre-loaded with MCRC64_0_CRC_WDTOPLD1-4 after either AUTO or Semi-CPU mode is selected and starts to down count. If the timeout counter expires before DMA transfers any data pattern to PSA Signature Register then a timeout interrupt is generated. An incoming data pattern before the timeout counter expires will automatically pre-load the timeout counter with MCRC64_0_CRC_BCTOPLD1-4 the block complete timeout pre-load value.

Block complete timeout pre-load value is used to check it one block of data patterns are compressed within a given time limit. If the timeout counter pre-loaded with MCRC64_0_CRC_BCTOPLD1-4 value expires before one block of data patterns are compressed a timeout interrupt is generated. When one block (pattern count × sector count) of data patterns are compressed before the counter has expired, the counter is pre-loaded with MCRC64_0_CRC_WDTOPLD1-4 value again. If the timeout counter is pre-loaded with zero then the counter is disable and no timeout interrupt is generated.

12.8.3.2.11.3 Underrun Interrupt

Underrun interrupt only occurs in AUTO mode. The interrupt is generated when the CRC Value Register is not updated with the corresponding signature when the data pattern counter finishes counting. During AUTO mode, MCRC Controller generates DMA request to update CRC Value Register in synchronization to the corresponding sector of the memory. Signature verification is also performed if underrun condition is detected. And CRC fail interrupt is generated at the same time as the underrun interrupt.

12.8.3.2.11.4 Compression Complete Interrupt

Compression complete interrupt is generated in Semi-CPU mode only. When the data pattern counter reaches zero, the compression complete flag is set and the interrupt is generated

12.8.3.2.11.5 Interrupt Offset Register

MCRC Controller only generates one interrupt request to interrupt manager. An interrupt offset register (MCRC64_0_CRC_INT_OFFSET_REG) is provided to indicate the source of the pending interrupt with highest priority. Table 12-373 shows the offset interrupt vector address of each interrupt in an ascending order of priority.

Copyright © 2026 Texas Instruments Incorporated

Table 12-373. Interrupt Offset Mapping

Interrupt Condition	Offset Value
Phantom	0x0
Ch1 CRC Fail	0x1
Ch2 CRC Fail	0x2
Ch3 CRC Fail	0x3
Ch4 CRC Fail	0x4
reserved	0x5-0x8
Ch1 Compression Complete	0x9
Ch2 Compression Complete	0xA
Ch3 Compression Complete	0xB
Ch4 Compression Complete	0xC
reserved	0xD-0x10
Ch1 Overrun	0x11
Ch2 Overrun	0x12
Ch3 Overrun	0x13
Ch4 Overrun	0x14
reserved	0x15-0x18
Ch1 Underrun	0x19
Ch2 Underrun	0x1A
Ch3 Underrun	0x1B
Ch4 Underrun	0x1C
reserved	0x1D-0x20
Ch1 Timeout	0x21
Ch2 Timeout	0x22
Ch3 Timeout	0x23
Ch4 Timeout	0x24

12.8.3.2.11.6 Error Handling

When an interrupt is generated, host CPU must take appropriate actions to identify the source of error and restart the respective channel in DMA and MCRC module. To restart a CRC channel user must perform the following steps in the ISR:

1. Write to software reset bit in MCRC64_0_CRC_CTRL0 register to reset the respective PSA Signature Register
2. Reset the CHi_MODE bits to 0 in MCRC64_0_CRC_CTRL2 register as Data capture mode
3. Set the CHi_MODE bits in MCRC64_0_CRC_CTRL2 register to desired new mode again
4. Release software reset

The host CPU must use byte write to restart each individual channel.

12.8.3.2.12 Power Down Mode

MCRC module can be put into power down mode when the power down control bit MCRC64_0_CRC_CTRL1[0] PWDN is set. The module wakes up when the PWDN bit is cleared. When MCRC controller is in power down mode, no data tracing alone will happen.

12.8.3.2.13 Emulation

A read access from a register in functional mode can sometimes trigger a certain internal event to follow. For example, reading interrupt offset register triggers an event to clear the corresponding interrupt status flag. During emulation when SUSPEND signal is high, a read access from any register should only return the register

Copyright © 2026 Texas Instruments Incorporated

contents to the bus and should not trigger or mask any event as it would have in functional mode. This is to prevent debugger from reading the interrupt offset register, that is, during refreshing screen and cause the corresponding interrupt status flag to get cleared. Timeout counters are stopped to generate timeout interrupts in emulation mode. No VBUSM bus error will be generated if reading from the unimplemented locations. .

CEMUDBG is the VBUSM suspend signal which need not explicitly indicate that whether CPU is in suspend mode or not. In data trace mode, a separate suspend signal CPU_EMUSP, is used to indicate MCRC controller that the CPU is in suspend mode or not.

Copyright © 2026 Texas Instruments Incorporated

12.8.3.3 MCRC Programming Examples

12.8.3.3.1 Example: Auto Mode Using Time Based Event Triggering

A large memory area with 2 Mbyte (256 k × 32-bit [doubleword]) is to be checked in the background of CPU. CRC is to be performed every 1 Kbyte (128 doubleword). Therefore there will be 2048 pre-recorded CRC values. For illustration purpose, we map MCRC64_0_CRC_REGL1 register to DMA channel 1 and MCRC64_0_PSA_SIGREGL1 register to DMA channel 2. Let's assume all DMA transfers are carried out in 64-bit transfer size.

12.8.3.3.1.1 DMA Setup

- Setup DMA channel 1 with the starting address from which the pre-determined CRC values are stored. Setup the destination address to the MCRC64_0_CRC_REGL1. Put the source address at post increment addressing mode and put the destination address at constant addressing mode. Use hardware DMA request for channel 1 to trigger a frame transfer.
- Setup DMA channel 2 with the source address from which the contents of memory to be verified. Setup the destination address to the MCRC64_0_PSA_SIGREGL1. Program the element transfer count to 128 and the frame transfer count to 2048. Program the read and write element size to 64 bits. Put the source address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request for channel 2 to trigger an entire block transfer.

12.8.3.3.1.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time based DMA request.

- Setup Timer to generate DMA request associated with DMA channel 2. For example, software can setup the timer to generate a DMA request every 10 ms.

12.8.3.3.1.3 CRC Setup

- Program the pattern count MCRC64_0_CRC_PCOUNT_REG1 to 128
- Program the sector count MCRC64_0_CRC_SCOUNT_REG1 to 2048
- For example, we want the entire 2 Mbytes to be compressed within 5 ms. We can program the block complete timeout pre-load (MCRC64_0_CRC_BCTOPLD1-4) value to 15625 (5 ms / (1 FICLK period × 64)) if CRC is operating at 200 MHz.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected MCRC Controller automatically generates a DMA request on channel 1. Around the same time the timer module also generates a DMA request on DMA channel 2. When the first incoming data pattern arrives at the MCRC64_0_PSA_SIGREGL1/H1, the MCRC Controller will compress it. After some time, the DMA controller would update the CMCRC_CRC_REGL1/H1 with a pre-determined value matching the calculated signature for the first sector of 128 64-bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the MCRC64_0_PSA_SECSIGREGL1/H1 does not match the MCRC64_0_CRC_REGL1/H1 Register. MCRC Controller generates a DMA request on DMA channel 1 when one sector of data patterns are compressed. This routine will continue until the entire 2 Mbytes are consumed. If the timeout counter reached zero before the entire 2 Mbytes are compressed, a timeout interrupt is generated. After 2Mbytes are transferred, the DMA can generate an interrupt to CPU. The entire operation will continue again when DMA responds to the DMA request from both the timer and MCRC Controller. The CRC is performed totally without any CPU intervention.

12.8.3.3.2 Example: Auto Mode Without Using Time Based Triggering

A small but highly secured memory area with 1 Kbytes is to be checked in the background of CPU. CRC is to be performed every 1 Kbytes. Therefore, there is only one pre-recorded CRC value. For illustration purpose, we map channel 1 MCRC64_0_CRC_REGL1/H1 to DMA channel 1 and channel 1 MCRC64_0_PSA_SIGREGL1/H1 to DMA channel 2. Assume all transfers carried out by DMA are in 64-bit transfer size.

Copyright © 2026 Texas Instruments Incorporated

12.8.3.3.2.1 DMA Setup

- Setup DMA channel 1 with the source address from which the pre-determined CRC value is stored. Setup the destination address to the MCRC64_0_CRC_REGL1 Register. Put the source address at constant addressing mode and put the destination address at constant addressing mode. Use hardware DMA request for channel 1.
- Setup DMA channel 2 with the source address from which the memory area to be verified. Setup the destination address to the MCRC64_0_PSA_SIGREGL1/H1. Program the element transfer count to 128 and the frame transfer count to 1. Put the source address at post increment addressing mode and put the destination address at constant address mode. Generate a software DMA request on channel 2 after CRC has completed its setup. Enable autoinitiation for DMA channel 2.

12.8.3.3.2.2 CRC Setup

- Program the MCRC64_0_CRC_PCOUNT_REG1 to 128
- Program the MCRC64_0_CRC_SCOUNT_REG1 to 1
- Leaving the timeout count MCRC64_0_CRC_BCTOPLD1 register with the reset value of zero means no timeout interrupt is generated
- Enable AUTO mode and all interrupts

After AUTO mode is selected the MCRC Controller automatically generates a DMA request on channel 1. At the same time the CPU generates a software DMA request on DMA channel 2. When the first incoming data pattern arrives at the MCRC64_0_PSA_SIGREGL1/H1, the MCRC Controller will compress it. After some time, the DMA controller would update the MCRC64_0_CRC_REGL1/H1 with a pre-determined value matching the calculated signature for the first sector of 128 64-bit data patterns. After one sector of data patterns are compressed, the MCRC Controller generate a CRC fail interrupt if signature stored at the MCRC64_0_PSA_SECSIGREGL1/H1 does not match the MCRC64_0_CRC_REGL1/H1. MCRC Controller generate a DMA request on DMA channel 1 again after one sector is compressed. After 1 Kbytes are transferred, the DMA can generate an interrupt to CPU. Responding to the DMA interrupt CPU can restart the CRC routine by generating a software DMA request onto channel 2 again.

12.8.3.3.3 Example: Semi-CPU Mode

If DMA controller is available in a system the CRC module can also operate in semi-CPU mode. This means that CPU can still make use of the DMA to perform data patterns transfer to CRC controller in the background. The difference between semi-CPU mode and AUTO mode is that CRC controller does not automatically perform the signature verification. CRC controllers generates a compression complete interrupt to CPU when the one sector of data patterns are compressed. CPU needs to perform the signature verification itself.

A memory area with 2 Mbytes is to be verified with the help of the CPU. CRC operation is to be performed every 1 Kbyte. Since there are 2 Mbytes (256 K doublewords) of memory to be check and we want to perform a CRC every 1 Kbytes (128 doublewords) and therefore there must be 2048 pre-recorded CRC values. In Semi-CPU mode, the MCRC64_0_CRC_REGL1/H1 is not updated and contains indeterminate data.

12.8.3.3.3.1 DMA Setup

- Setup DMA channel 1 with the source address from which the memory area to be verified are mapped. Setup the destination address to the MCRC64_0_PSA_SIGREGL1/H1. Put the starting address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request to trigger an entire block transfer for channel 1. Disable autoinitiation for DMA channel 1.

12.8.3.3.3.2 Timer Setup

The timer can be any general purpose timer which is capable of generating a time-based DMA request.

- Setup Timer to generate DMA request associated with DMA channel 1. For example, software can setup the timer to generate a DMA request every 10 ms.

12.8.3.3.3.3 CRC Setup

- Program the MCRC64_0_CRC_PCOUNT_REG1 to 128
- Program the MCRC64_0_CRC_SCOUNT_REG1 to 2048

Copyright © 2026 Texas Instruments Incorporated

- For example, we want the entire 2 Mbytes to be compressed within 5 ms. We can program the block complete timeout pre-load value MCRC64_0_CRC_BCTOPLD1 to 15625 (5 ms / (1 FICLK period × 64)) if CRC is operating at 200 MHz
- Enable AUTO mode and all interrupts.

The timer module first generates a DMA request on DMA channel 1 when it is enabled. When the first incoming data pattern arrives at the MCRC64_0_PSA_SIGREGL1/H1, the CRC controller will compress it. After one sector of data patterns are compressed, the CRC controller generate a compression complete interrupt. Upon responding to the interrupt the CPU would read from the MCRC64_0_PSA_SECSIGREGL1/H1. It is up to the CPU on how to deal with the PSA value just read. It can compare it to a known signature value or it can write it to another memory location to build a signature file or even transfer the signature out of the device via SCI or SPI. This routine will continue until the entire 2 Mbytes are consumed. The latency of the interrupt response from CPU can cause overrun condition. If CPU does not read from MCRC64_0_PSA_SECSIGREGL1 before the PSA value is overridden with the signature of the next sector of memory, an overrun interrupt will be generated by CRC controller.

12.8.3.3.4 Example: Full-CPU Mode

In a system without the availability of DMA controller, the CRC routine can be operated by CPU provided the CPU has enough throughput. CPU needs to read from the memory area from which CRC is to be performed.

A memory area with 2 Mbytes is to be checked with the help of the CPU. CRC operation is to be performed every 1 Kbyte. In CPU mode, the MCRC64_0_CRC_REGL1/H1 is not updated and contains indeterminate data.

12.8.3.3.4.1 CRC Setup

- All control registers can be left in their reset state. Only enable Full-CPU mode.

CPU itself reads from the memory and write the data to the MCRC64_0_PSA_SIGREGL1/H1 inside MCRC Controller. When the first incoming data pattern arrives at the MCRC64_0_PSA_SIGREGL1/H1, the MCRC Controller will compress it. After n data patterns are compressed, CPU can read from the MCRC64_0_PSA_SIGREGL1/H1. It is up to the CPU on how to deal with the PSA signature value just read. It can compare it to a known signature value stored at another memory location.

Copyright © 2026 Texas Instruments Incorporated

12.8.4 ECC Aggregator

This section describes the common ECC aggregator functionality.

12.8.4.1 ECC Aggregator Overview

To increase functional and system reliability the memories (for example, FIFOs, queues, SRAMs and others) in many device modules and subsystems are protected by error correcting code (ECC). This is accomplished through an ECC aggregator and ECC wrapper. The ECC aggregator is connected to these memories (hereinafter ECC RAMs) and involved in the ECC process. Each memory is surrounded by an ECC wrapper which performs the ECC detection and correction. The wrapper communicates via serial interface with the aggregator which has memory mapped configuration interface.

The ECC aggregator is also connected to interconnect ECC components that protect the command, address and data buses of the system interconnect. ECC is calculated for the data bus and parity and redundancy for the command and address buses. Each interconnect ECC component has the same serial interface for communication with the aggregator as the ECC wrapper. An ECC aggregator may be connected to both endpoints - the ECC wrapper and interconnect ECC component.

The ECC aggregator, ECC wrapper and interconnect ECC component are considered as single entity and are hereinafter referred to as ECC aggregator unless otherwise explicitly specified.

See *Module Integration* for device modules and subsystems which have ECC aggregator.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.1.1 ECC Aggregator Features

The ECC aggregator has the following features:

- Reduces memory software errors via single error correction (SEC) and double error detection (DED)
- Provides a mechanism to control and monitor the ECC protected memories in a module or subsystem
- SEC and DED over the system interconnect data bus and parity and redundancy for the system interconnect command and address buses
- Generates an interrupt for correctable error
- Generates an interrupt for non-correctable error
- Supports inject only mode for diagnostic purposes
- Supports software readable status for single and double-bit ECC errors and associated information such as row address where error has occurred and data bits that have been flipped
- Supports up to 256 ECC endpoints. An ECC endpoint can be either ECC RAM or interconnect ECC component.
- Detects single bit error via parity checking on:
  - Memory mapped configuration interface FIFO
  - Serial interface FIFO
  - Serial interface transaction
- Single bit error detection via parity checking results in a non-correctable error interrupt
- Supports timeout mechanism on transactions over the ECC serial interface. Timeout occurrence results in a non-correctable error interrupt.
- Certain control bits have redundancy and if a bit flips an interrupt is generated

12.8.4.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Note

Some features may not be available. See *Module Integration* for more information.

12.8.4.2 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.3 ECC Aggregator Functional Description

This section describes the architecture and functional details of the ECC aggregator.

12.8.4.3.1 ECC Aggregator Block Diagram

Figure 12-476 shows the ECC aggregator block diagram.

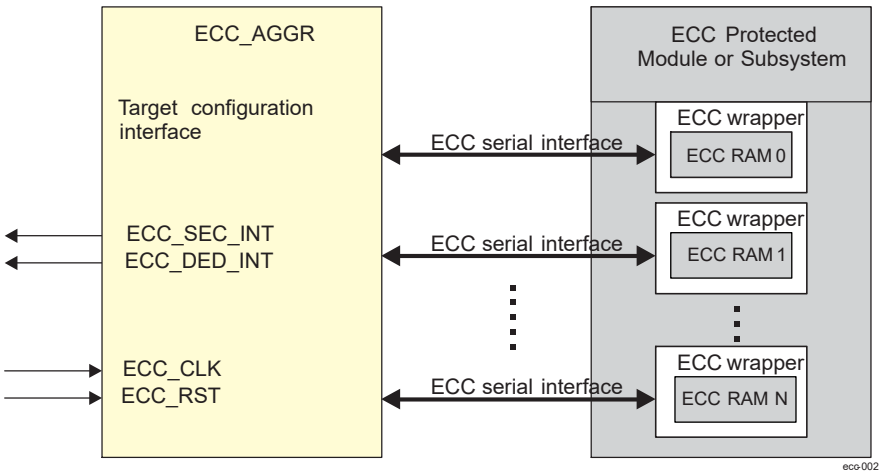


Figure 12-476. ECC Aggregator Block Diagram

The ECC aggregator is connected to one or more ECC endpoints each of which has assigned a unique ID used when the endpoint is accessed for status information or configuration. The ECC aggregator provides software access to all ECC related registers through its memory mapped target configuration interface while the serial interface is used to communicate with the ECC endpoints. Upon detection of single or double-bit error the corresponding interrupt line is asserted.

12.8.4.3.2 ECC Aggregator Register Groups

The ECC aggregator has ECC control, status and interrupt registers for each ECC endpoint in a module or subsystem. These registers are memory mapped and occupy 1 KB address space although part of it may contain reserved locations. The registers are split in the following types:

- **Global registers**. They are common to all ECC endpoints associated with the ECC aggregator and include the ECC_VECTOR and ECC_REV registers. Each ECC endpoint has assigned a unique ID. When this ID is written to the ECC_VECTOR[10-0] ECC_VECTOR field the corresponding endpoint is selected either for control or for status reading.
- **ECC control and status registers**. These registers are specific to each ECC endpoint and reside in the range from address offset 0x10 to 0x28, if the endpoint is ECC RAM or from 0x10 to 0x24, if the endpoint is interconnect ECC component. They are memory mapped but are accessed through the ECC serial interface. They are also selected by the ECC endpoint ID written to the ECC_VECTOR[10-0] ECC_VECTOR field. Because of latency on the serial interface the ECC control and status registers are read by performing special sequence as described in Section 12.8.4.3.3. These registers have also different functionality for both types of endpoints - ECC RAM and interconnect ECC component.
- **Interrupt registers**. They include interrupt status, interrupt enable, interrupt disable, and EOI registers. For more information, see *Interrupts*.

Note

For description of each register, see *ECC_AGGR Registers*.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.3.3 Read Access to the ECC Control and Status Registers

Read accesses to the ECC control and status registers for each ECC endpoint represent read operations over the ECC serial interface and are triggered by performing the following sequence:

1. Software writes the following in the ECC_VECTOR register:
   - The ECC endpoint ID in the ECC_VECTOR[10-0] ECC_VECTOR field to select particular ECC endpoint.
   - The register read address in the ECC_VECTOR[23-16] RD_SVBUS_ADDRESS field to select which register has to be read through the ECC serial interface.
   - A value of 0x1 in the ECC_VECTOR[15] RD_SVBUS bit to trigger read operation through the ECC serial interface.
2. Software polls the ECC_VECTOR[24] RD_SVBUS_DONE bit to check if it is 0x1. This indicates that the read operation on the ECC serial interface has completed.
3. Software reads the data from the register previously selected by the ECC_VECTOR[23-16] RD_SVBUS_ADDRESS field.

The following is an example for serial read operation:

1. Write 0x0010 8005 to the ECC_VECTOR register. This sends read request to the ECC_WRAP_REV or ECC_CBASS_REV register (address = 0x10) associated with ECC endpoint with ID = 5.
2. Poll the ECC_VECTOR[24] RD_SVBUS_DONE bit until value of 0x1 is read.
3. Read the ECC_WRAP_REV or ECC_CBASS_REV register to get its value.

12.8.4.3.4 Serial Write Operation

Write operations over the ECC serial interface are performed as follows:

1. Software specifies the ECC endpoint ID in the ECC_VECTOR[10-0] ECC_VECTOR field. The ECC_VECTOR[23-16] RD_SVBUS_ADDRESS field is a don't care but the ECC_VECTOR[15] RD_SVBUS bit must be set to 0x0.
2. Software performs regular write operation to the desired address. If the ECC endpoint ID has already been specified, step 1 can be skipped. Unlike serial read operations it is not necessary to always specify the endpoint ID before performing serial write operation.

The following is an example for serial write operation:

1. Write 0x0000 0008 to the ECC_VECTOR register.
2. Write 0x0000 000F to the ECC_CTRL register. This sends write request with data 0x0000 000F to the ECC_CTRL register associated with ECC RAM with ID = 8.

12.8.4.3.5 Interrupts

The ECC aggregator generates the following interrupts:

- Correctable interrupt (ECC_SEC_INT) where hardware can correct the error but notifies the system in case of SEC.
- Non-correctable interrupt (ECC_DED_INT) where hardware cannot correct the error in cases of DED, parity check, redundancy check or timeout occurrence.

The following is the sequence for servicing interrupts:

- Software enables the interrupts for an ECC endpoint by writing 0x1 to the corresponding bit of the following interrupt enable registers:
  - ECC_SEC_ENABLE_SET_REG0 through ECC_SEC_ENABLE_SET_REG7 for the correctable interrupt
  - ECC_DED_ENABLE_SET_REG0 through ECC_DED_ENABLE_SET_REG7 for the non-correctable interrupt
- On receiving an interrupt, software checks which ECC endpoint has caused the error by reading the following interrupt status registers:
  - ECC_SEC_STATUS_REG0 through ECC_SEC_STATUS_REG7 for the correctable interrupt
  - ECC_DED_STATUS_REG0 through ECC_DED_STATUS_REG7 for the non-correctable interrupt

Copyright © 2026 Texas Instruments Incorporated

- Software performs serial read operations as described in Section 12.8.4.3.3 to read the following status registers that contain details about the error:
  - If the endpoint is ECC RAM:
    - ECC_ERR_STAT1
    - ECC_ERR_STAT2
    - ECC_ERR_STAT3
  - If the endpoint is interconnect ECC component:
    - ECC_CBASS_ERR_STAT1
    - ECC_CBASS_ERR_STAT2
- After the interrupt has been serviced, depending on the error type, software should clear the corresponding status bits in the ECC_ERR_STAT1 and ECC_ERR_STAT3 registers or in the ECC_CBASS_ERR_STAT1 register. Software has to poll these registers to guarantee that status bits are cleared as there is no other indication for write completion over the ECC serial interface.
  The value of the *_PEND_CLR fields in the ECC_CBASS_ERR_STAT1 register must be read and then written back to decrement the count of each field back to 0x0. A further error capture into the ECC_CBASS_ERR_STAT1 register does not occur unless all its fields are 0x0. The decrement value should not be larger than the read value. If a field in the ECC_CBASS_ERR_STAT1 register should not be modified, write a value of 0x0 to that field.
- Software writes 0x1 to the corresponding end of interrupt register to clear the interrupt:
  - ECC_SEC_EOI_REG for the correctable interrupt
  - ECC_DED_EOI_REG for the non-correctable interrupt

12.8.4.3.6 Inject Only Mode

There are modules that already perform the ECC generation and checking as part of their data path. In this case, the ECC wrapper may be configured in inject only mode, if needed. In this mode the ECC wrapper does not perform ECC detection and correction. The inject only mode allows users to inject single or double-bit errors so that the module logic can be tested for diagnostic purposes.

Note

There is no software control to enable inject only mode. It is configured via tie-off value. Inject only and ECC modes are mutually exclusive.

The interconnect ECC component also supports error injection mode. There is error injection logic for testing of the error checking logic (checkers). The injection logic can be configured to inject either single or double bit error and what data pattern to be used for injection (ECC_CBASS_CTRL[11-8] ECC_PATTERN). The ECC_CBASS_ERR_CTRL1 and ECC_CBASS_ERR_CTRL2 registers should be written first to setup the injection. Then, either the ECC_CBASS_CTRL[3] FORCE_SE or the ECC_CBASS_CTRL[4] FORCE_DE bit must be set to 0x1 to start the injection. Both bits must not be set at the same time. If the injection should continue in incrementing mode, then the ECC_CBASS_CTRL[5] FORCE_N_BIT bit should be set to 0x1. Once the FORCE_N_BIT is set, then each successive injection can simply write the ECC_CBASS_CTRL register to set the FORCE_SE or FORCE_DE again. Reading 0x0 from either the FORCE_SE or the FORCE_DE bit indicates that the injection has completed, as these bits automatically clear when the checker indicates that it has performed the injection. The time for an injection to complete is not guaranteed, so some delay is needed between successive injections.

12.8.4.3.7 ECC Error Injection Sequence

The ECC aggregator is a critical component for enhancing system reliability by protecting memories in various device modules and subsystems. The ECC Aggregator is connected to these memory and interconnect components which have the ECC, to provide access to control and monitor the ECC protected memories in a module or subsystem.

The ECC aggregator supports error injection to aid in diagnostic purposes and to evaluate robustness. The Error Injection feature supported by the ECC aggregator allows users to deliberately introduce errors into their system

Copyright © 2026 Texas Instruments Incorporated

to test the resilience and effectiveness of error correction mechanisms. This feature helps in evaluating the robustness of the ECC aggregator by simulating real-world error scenarios, enabling users to identify potential vulnerabilities and improve their error handling strategies.

This feature includes configurable parameters such as ECC aggregator type, subtype, error type, and injection location, allowing users to customize their testing scenarios based on their specific requirements.

12.8.4.3.7.1 Types of Error Injection

ECC is a mechanism for providing increased protection against soft errors by allowing single bit errors to be detected and corrected and double bit errors to be detected. The ECC Control Register can be used to inject single bit or double bit errors. For more information on this register, refer to Section 12.8.4.3.2. These errors are then reported through the Error Signal Module (ESM).

- Single Error Correction (SEC): In the event of single bit errors, the ECC aggregator has the ability to both detect and correct these errors, ensuring the integrity of the data. Single bit error injection is supported for all endpoints and checkers. Single bit error correction is not supported for parity and redundancy type of checkers.
- Double Error Detection (DED): For double bit errors, the ECC aggregator can detect these errors. However, it does not have the capability to correct them. Instead, it signals the presence of these errors to the Central Processing Unit (CPU) via ESM interrupts. This is supported for all RAM wrapper types and interconnect EDC checker type of endpoints. Parity and Redundant checker types do not support double bit ECC error injection.

The ECC aggregator plays a crucial role in maintaining data accuracy by correcting single bit errors and detecting double bit errors, although it cannot correct the latter. This process helps in enhancing the overall system reliability. Each ECC aggregator connects to two kinds of ESM events – Correctable error events (for SEC events) and Un-correctable error events (for DED events and parity and redundancy checker types). ESM is used for notification of events from endpoints which are not Inject-Only. Inject-Only endpoints notify errors through IP-specific mechanisms. Sometimes a separate ESM event (not via ECC aggregator) is available. Please refer to the individual IP sections for more details on the ESM events generated.

The ECC aggregator provides support for injecting single and double bit errors. In the case of RAM wrapper type endpoints, errors can be injected to a specific row or can be requested to be injected on the next row access (n_row). Errors can be requested to be injected once or on repeated reads. For Interconnect endpoint types, single and double bit errors (depending on the checker type) can be injected.

12.8.4.3.7.1.1 ECC Wrapper Type Endpoints

Each memory is surrounded by an ECC wrapper which performs the ECC detection and correction. The wrapper communicates via serial interface with the aggregator which has memory a mapped configuration interface. For RAM wrapper types, after injecting the error, the RAM location where the error was injected needs to be accessed in order to trigger the error interrupt. If the “force_n_row” option is selected during error injection, then any RAM access will trigger the interrupt and the specific row is not needed.

12.8.4.3.7.1.2 EDC Interconnect Type Endpoints

The ECC aggregator is also connected to interconnect ECC components that protect the command, address and data buses of the system interconnect. ECC is calculated for the data bus and parity and redundancy for the command and address buses. For the interconnect types, the type of checker must be verified to determine if they are parity or redundant types.

12.8.4.3.7.1.3 Inject Only ECC Endpoints

Some modules may already do the generation of and checking of ECC as part of their data path. In this case, the ECC wrapper does not have to perform ECC detection and correction and is configured in inject only mode. The wrapper will not add extra bits to the ram, generate, nor check ECC. It will, however, allow the user to inject single or double-bit errors so that the IP’s logic can be tested for diagnostic purposes. Detection and correction occurs in the IP itself. In some cases, the error may also be routed to ESM, but not to the same ESM event as the ECC aggregator. For more information on these memories, refer to Section 12.8.4.3.6.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.3.7.1.4 Full Functionality Error Capture ECC Endpoints

These ECC aggregators possess the ability to perform Error Injection alongside Error Capture. The wrapper can be utilized to introduce either single or double-bit errors, while also having the capability to detect and correct errors.

For more information regarding the ECC endpoint types, refer to the Appendix spreadsheet.

12.8.4.3.7.2 Types of ECC Checkers

There are various checkers for bus signals, MMRs, or internal registers in the interconnect.

12.8.4.3.7.2.1 Parity Type Checkers

The most common checker type is parity. The parity type refers to checkers where a parity bit is added to cover the entire signal. The parity bit is the ~^signal, where the signal width can be any number of bits.

12.8.4.3.7.2.2 Redundant Type Checkers

The redundant type refers to checkers for critical signals (like request). The redundant signal is the opposite of the signal it is protecting.

12.8.4.3.7.2.3 Error Detect and Correct (EDC) Type Checkers

The EDC type refers to SECDED error-correcting code that can correct single bit error, and detect a two bit error.

For more information regarding the ECC checker type for each aggregator, refer to the Appendix spreadsheet.

12.8.4.3.7.3 Recommendations for Testing Error Injection

The following are the general recommendations on when the error injection on ECC aggregators must be performed:

- The ECC error injection and monitoring of the error response should be run at ‘startup’ where possible.
- An alternative recommended approach is to run ECC error injection at ‘shutdown’.

When running of ECC error injection during ‘runtime’, the system integrator must be aware of the below:

For interconnect endpoints:

- The ECC SEC error injection runs during idle cycles, so can be run during ‘runtime’ for checkers of all types.
  - Parity and redundant bit error injections are non-destructive and can be done at runtime.
- The ECC DED error injection runs during idle cycles, so can be run during ‘runtime’.
  - Only single bit error injections are to be done on parity/redundant checker types. This is because these protect odd number of injections. A 2 bit injection on parity will not cause a failure.

For RAM memories:

- ECC self-test for SEC on RAM memories, can be run at runtime:
  - The ECC self-test for SEC injects a single-bit error that is corrected in-line.
- ECC DED error injection during ‘runtime’ for RAM memories, should only be done for memories with direct read access
  - Two examples of direct read access memories are MSMC SRAM and MCU SRAM
  - Double-bit error injection on all other embedded SRAMS could possibly result in the downstream IP reading bad data if run during ‘runtime’
- CC error injection on inject-only memories, should be run at ‘startup’, they are not recommended to be run during ‘runtime’. For more information on these memories, refer to Section 12.8.4.3.6.

12.8.4.3.7.4 Error Injection Programming Sequence

12.8.4.3.7.4.1 ESM Initialization

The ESM instance must beinitializedwith the following sequence. This initialization will allow the application to specify for each event whether the interrupt is enabled or disabled, the priority of the event, and whether the nErrorPin assertion is enabled or disabled for the event. This needs to be done only once. All the required ESM events can be initialized at the same time.

Copyright © 2026 Texas Instruments Incorporated

For more information on the ESM, refer to Error Signaling Module (ESM)..

To initialize the ESM module for a specified instance:

1. Clear all raw status and enable bits by setting ESM_SFT_RST Register bits [3:0] to 0Fh, to reset the ESM
2. Write 0x1 to the appropriate bits in the ESM_ERR_STS register. This step will clear the configuration interrupt raw status
3. Write 0x1 to the appropriate bits in the ESM_ERR_EN_SET register, to n enable the configuration error interrupt as described in ESM Configuration Error Interrupt..
4. Write 0x1 to the appropriate bits in the error group j of the ESM_STS_j register. This step will clear the level interrupt raw status.
5. Write 0x1 to the appropriate bits in the error group j of the ESM_INTR_EN_SET_j register, to enable the level error interrupt.
6. Write 0x1 to the appropriate bits in the error group j of the ESM_INT_PRIO_j register, to indicate which interrupt the corresponding event influences (if enabled) for event group j.
7. If required, write 0x1 to the appropriate bits in the error group j of the ESM_PIN_EN_SET_j register, to set error output corresponding to the desired events.
8. Enable the global interrupts by setting ESM_EN Register bits [3:0] to 0Fh

12.8.4.3.7.4.2 ECC Initialization

1. Setting the timeout bit is required to enable notification for ECC serial interface timeout errors. The parity bit must be set to enable notification for parity errors. To enable the timeout (bit 1) and parity (bit 0) bits, write 0x1 to the ECC_AGGR_ENABLE_SET register.
2. Write 0x1 to the ECC_SEC_ENABLE_SET_REGx (to enable correctable SEC error interrupts) and ECC_DED_ENABLE_SET_REGx (to enable uncorrectable DED error interrupts) registers for required endpoints.

The following register configurations are per endpoint and done through the SVBUS programming interface. Refer to Section 12.8.4.3.4 for more information.

1. Write 0x1 to the ECC_CHECK field of the ECC_CBASS_CTRL register to enable ECC check for interconnect endpoints.
2. Write 0x1 to the ENABLE_RMW, ECC_CHECK and ECC_ENABLE bit fields of the ECC_CTRL register to enable ECC check for wrapper type endpoints.

12.8.4.3.7.4.3 Error Injection

For ECC Wrapper Type Endpoints:

1. Write to the ECC_ERR_CTRL1 Register to set the row address where FORCE_SEC or FORCE_DED needs to be applied. This is ignored if FORCE_N_ROW is set.
2. Configure the ECC_RAM_CTRL bits[6:3] register based on the type of error injection.
3. Perform a read back of the ECC_RAM_CTRL register to confirm error injection. The read value may not be same as what is written as some fields in the register are not writable or can self-clear.
4. For RAM wrapper types, after injecting the error, the RAM needs to be accessed in the row where the error was injected (if FORCE_N_ROW is used, any row can be accessed) in order to trigger the error interrupt.
5. For Inject-only Wrapper endpoints, only error injection is possible via the ECC aggregator, the detection and correction is be done by the IP itself.

Field	Name	Type	Reset	Description
0	ecc_enable	r/w	1	Enable ECC generation. **Reset to 1 with inject only configuration, but any write to this register byte clears this bit to 0. Resets to 1 in all other configurations.**

Copyright © 2026 Texas Instruments Incorporated

Field	Name	Type	Reset	Description
1	ecc_check	r/w	1/0	Enable ECC check. ECC is completely bypassed if both ecc_enable and ecc_check are '0'. **Reset to 0 with inject only configuration, and not writable. Reset to 1 in all other configurations.**
2	enable_rmw	r/w	1/0	Enable read-modify-write on partial word writes. **If disabled ecc detection and correction will no longer work and if re-enabled the ram contents must all be rewritten to correct ecc codes. Reset to 0 with inject only configuration, and not writable. Reset to 1 in all other configurations.**
3	force_sec	r/w	0	Force single-bit error. Cleared the cycle following the error if error_once is asserted. for write through mode this applies to writes as well as reads.
4	force_ded	r/w	0	Force double-bit error. Cleared the cycle following the error if error_once is asserted. for write through mode this applies to writes as well as reads.
5	force_n_row	r/w	0	Force single/double-bit error on the next RAM access. for write through mode this applies to writes as well as reads.
6	error_once	r/w	0	If this bit is set, the force_sec/force_ded will inject an error to the specified row only once. The force_sec bit will be cleared the cycle after the error is generated. For double-bit errors, the force_ded bit will be cleared the cycle following the double-bit error. In either case force_n_row will be cleared as well. Any subsequent reads will not force an error.
7	check_parity	r/w	1	Enables parity checking on internal data
8	Check svbus timeout	r/w	1	Enable svbus timeout mechanism
9-31	Reserved			

For ECC Interconnect Type Endpoints:

1. Clear any pending error injection by writing 0x0 to the bit fields of the ECC_CBASS_CTRL register
2. Configure the ECC_CBASS_ERR_CTRL1 and ECC_CBASS_ERR_CTRL2 registers based on the error group and type of error injection.
3. Write to the ECC_CBASS_CTRL register to inject the error.
4. For interconnect types, the error injection happens on idle cycles, so no additional access of the memory is required.

12.8.4.3.7.4.4 Error Handling

1. When an error event occurs, the user can query and collect the information about the endpoint and offset information for the location relative to the start of the associated RAM.
2. To clear the ECC error event, clear the pending status of the ECC error events by configuring the ECC_RAM_ERR_STAT1 (for Wrapper type endpoints) and SDL_EDC_CTL_ERR_STATUS1(for Interconnect type endpoints) register.
3. The ECC_SEC_EOI_REG register is used to re-trigger the pulse interrupt signal to ensure that any nested interrupt events are serviced. The software interrupt handler must write to the ECC_SEC_EOI_REG register at the end of the current interrupt processing routine, so that new events can re-trigger the pulse interrupt signal again. For level interrupt signals the ECC_SEC_EOI_REG register is not functional and must not be used.

Copyright © 2026 Texas Instruments Incorporated

4. Clear the ESM error interrupt using the ESM registers.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4 ECC Aggregator Registers

12.8.4.4.1 ECC_AGGR Registers

Table 12-374 lists the memory-mapped registers for an ECC Aggregator (ECC_AGGR).

Table 12-374. ECC Wrapper Registers

Offset	Register Name	Description
ECC Wrapper Registers		
10h	ECC_WRAP_REV	ECC Wrapper Revision Register
14h	ECC_CTRL	ECC RAM Control Register
18h	ECC_ERR_CTRL1	ECC RAM Error Control 1 Register
1Ch	ECC_ERR_CTRL2	ECC RAM Error Control 2 Register
20h	ECC_ERR_STAT1	ECC RAM Error Status 1 Register
24h	ECC_ERR_STAT2	ECC RAM Error Status 2 Register
28h	ECC_ERR_STAT3	ECC RAM Error Status 3 Register
Interconnect ECC Component Registers		
10h	ECC_CBASS_REV	Interconnect ECC Component Revision Register
14h	ECC_CBASS_CTRL	Interconnect ECC Component Control Register
18h	ECC_CBASS_ERR_CTRL1	Interconnect ECC Component Error Control 1 Register
1Ch	ECC_CBASS_ERR_CTRL2	Interconnect ECC Component Error Control 2 Register
20h	ECC_CBASS_ERR_STAT1	Interconnect ECC Component Error Status 1 Register
24h	ECC_CBASS_ERR_STAT2	Interconnect ECC Component Error Status 2 Register

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.2 ECC_WRAP_REV Register (Offset = 10h)

Return to Summary Table.

ECC Wrapper Revision Register

Revision parameters.

Figure 12-477. ECC_WRAP_REV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV															
R-66A49A02h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV															
R-66A49A02h															

Table 12-375. ECC_WRAP_REV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R	66A49A02h	TI internal data.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.3 ECC_CTRL Register (Offset = 14h)

Return to Summary Table.

ECC RAM Control Register

The Global control register controls ECC control bits for the selected ECC RAM

Figure 12-478. ECC_CTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							CHECK_SVBUS_TIMEOUT
R-0h							R/W-1h
7	6	5	4	3	2	1	0
CHECK_PARITY	ERROR_ONCE	FORCE_N_ROW	FORCE_DED	FORCE_SEC	ENABLE_RMW	ECC_CHECK	ECC_ENABLE
R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-1h	R/W-1h

Table 12-376. ECC_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED			Reserved
8	CHECK_SVBUS_TIMEOUT	R/W	1h	Enable ECC serial interface timeout mechanism 0h - Timeout mechanism disabled 1h - Timeout mechanism enabled
7	CHECK_PARITY	R/W	1h	Enables parity checking on internal data 0h - Parity checking disabled 1h - Parity checking enabled
6	ERROR_ONCE	R/W	0h	Force error only once. If this bit is set to 1h, the FORCE_SEC/FORCE_DED injects an error to the specified row only once. The FORCE_SEC bit is cleared the cycle after the error is generated. For double-bit errors, the FORCE_DED bit is cleared the cycle following the double-bit error. Any subsequent reads do not force an error.
5	FORCE_N_ROW	R/W	0h	Force error on any RAM read Force single or double-bit error on the next RAM access. For write through mode this applies to writes as well as reads.
4	FORCE_DED	R/W	0h	Force double-bit error. Cleared the cycle following the error if ERROR_ONCE is 1h. For write through mode this applies to writes as well as reads.
3	FORCE_SEC	R/W	0h	Force single-bit error. Cleared the cycle following the error if ERROR_ONCE is 1h. For write through mode this applies to writes as well as reads.

Copyright © 2026 Texas Instruments Incorporated

Table 12-376. ECC_CTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	ENABLE_RMW	R/W	1h	Enable read-modify-write on partial word writes. 0h - Read-modify-write disabled 1h - Read-modify-write enabled **Note** If disabled, ECC detection and correction does no longer work and if re-enabled the RAM contents must all be rewritten to correct ECC codes. The reset value of this bit is 0h in inject only mode and 1h in ECC mode.
1	ECC_CHECK	R/W	1h	Enable ECC check. 0h - ECC check disabled 1h - ECC check enabled **Note** ECC is completely bypassed if both ECC_ENABLE and ECC_CHECK are 0h. The reset value of this bit is 0h in inject only mode and 1h in ECC mode.
0	ECC_ENABLE	R/W	1h	Enable ECC generation. 0h - ECC generation disabled 1h - ECC generation enabled

Note

All registers in this region have redundancy and if a bit flips and there is an inconsistency an interrupt will be generated and the effective value will be the reset value.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.4 ECC_ERR_CTRL1 Register (Offset = 18h)

Return to Summary Table.

Contains row number and data bit that needs to be flipped to force single-bit error

This register contains ECC error control bits for the selected ECC RAM

Figure 12-479. ECC_ERR_CTRL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC_ROW															
R/W-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_ROW															
R/W-0h															

Table 12-377. ECC_ERR_CTRL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ECC_ROW	R/W	0h	Row address where single or double-bit error needs to be applied. This is ignored if ECC_CTRL[5] FORCE_N_ROW bit is set to 1h.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.5 ECC_ERR_CTRL2 Register (Offset = 1Ch)

Return to Summary Table.

Contains data bit that needs to be flipped for forcing double-bit error

This register contains ECC error control bits for the selected ECC RAM

Figure 12-480. ECC_ERR_CTRL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC_BIT2															
R/W-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_BIT1															
R/W-0h															

Table 12-378. ECC_ERR_CTRL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	ECC_BIT2	R/W	0h	Data bit that needs to be flipped if double-bit error has to be forced. The ECC_CTRL[4] FORCE_DED bit must be set to 1h for these values to take affect.
15:0	ECC_BIT1	R/W	0h	Data bit that needs to be flipped if single-bit error has to be forced. The ECC_CTRL[3] FORCE_SEC bit must be set to 1h for these values to take affect.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.6 ECC_ERR_STAT1 Register (Offset = 20h)

Return to Summary Table.

Contains single-bit/double-bit error status and row corresponding to the error status.

This register contains ECC status bits for the selected ECC RAM

Figure 12-481. ECC_ERR_STAT1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC_BIT1															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR_CTRL_REG_ERROR	CLR_PARITY_ERROR		CLR_ECC_OTHER	CLR_ECC_DED		CLR_ECC_SEC		CTRL_REG_ERROR	PARITY_ERROR		ECC_OTHER	ECC_DED		ECC_SEC	
R/W-0h	R/W-0h		R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h		R/W-0h	R/W-0h		R/W-0h	

Table 12-379. ECC_ERR_STAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	ECC_BIT1	R	0h	Indicates the bit position in the RAM data that is in error. For example, a value of 1h indicates that bit [1] in the RAM data is in error. This is valid only for single-bit errors. **Note** Not used in inject only mode. Always read as 0h.
15	CLR_CTRL_REG_ERROR	R/W	0h	Clear the CTR_REG_ERR bit. A write of 1h clears this bit and the CTR_REG_ERR bit, but if the redundancy protected bits in the ECC_CTRL register have not been written to a known state to correct the error, this flag is immediately set again.
14:13	CLR_PARITY_ERROR	R/W	0h	A write of a non-zero value to this field decrements the CLR_ECC_DED and ECC_DED fields by that value. If the value written is less than the current one, the non-correctable interrupt (ECC_DED_INT) stays asserted. If the value to decrement is more than the current value, the result is 0. 0h – No parity errors have occurred 1h – 1 parity has occurred 2h – 2 parity have occurred 3h – 3 or more parity have occurred
12	CLR_ECC_OTHER	R/W	0h	Clear other error status. 1h indicates a successive single-bit error. Writing 1h clears the status bit.

Copyright © 2026 Texas Instruments Incorporated

Table 12-379. ECC_ERR_STAT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11:10	CLR_ECC_DED	R/W	0h	A write of a non-zero value to this field decrements it and the ECC_DED field by that value. If the value written is less than the current one, the non-correctable interrupt (ECC_DED_INT) stays asserted. If the value to decrement is more than the current value, the result is 0. 0h – No DED errors have occurred 1h – 1 DED has occurred 2h – 2 DED have occurred 3h – 3 or more DED have occurred **Note** Not used in inject only mode. Always read as 0h.
9:8	CLR_ECC_SEC	R/W	0h	A write of a non-zero value to this field decrements it and the ECC_SEC field by that value. If the value written is less than the current one, the correctable interrupt (ECC_SEC_INT) stays asserted. If the value to decrement is more than the current value, the result is 0. 0h – No SEC errors have occurred 1h – 1 SEC has occurred 2h – 2 SEC have occurred 3h – 3 or more SEC have occurred **Note** Not used in inject only mode. Always read as 0h.
7	CTRL_REG_ERROR	R/W	0h	Indicates that a redundancy protected bit in the ECC_CTRL register has been flipped. This means that the redundancy logic have detected a state where not all values are the same and has defaulted to the reset state. Software needs to re-write these registers to a known state. A write of 1h sets this bit. 0h - Bit not flipped 1h - Bit flipped
6:5	PARITY_ERROR	R/W	0h	2-bit saturating counter for the number of parity errors that have occurred since last cleared. This is also a status set register and a non-zero value sets the level interrupt. Software can also write a value to the CLR_PARITY_ERR field to decrement this counter. 0h - No parity errors have occurred 1h - 1 parity error has occurred 2h - 2 parity errors have occurred 3h - 3 or more parity errors have occurred
4	ECC_OTHER	R/W	0h	1h - Indicates that successive single-bit errors have occurred while a write-back is still pending. Software can also write 1h to set the pending status and write 1h to the corresponding clear bit to clear the status. **Note** Not used in inject only mode. Always read as 0h.

Copyright © 2026 Texas Instruments Incorporated

Table 12-379. ECC_ERR_STAT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3:2	ECC_DED	R/W	0h	2-bit saturating counter for the number of double-bit errors that have occurred since last cleared. This is also a status set register and a non-zero value sets the level interrupt. Software can also write a value to the CLR_ECC_SEC field to decrement this counter. 0h - No double-bit errors have occurred 1h - 1 double-bit has error occurred 2h - 2 double-bit have errors occurred 3h - 3 or more double-bit errors have occurred **Note** Not used in inject only mode. Always read as 0h.
1:0	ECC_SEC	R/W	0h	2-bit saturating counter for the number of single-bit errors that have occurred since last cleared. This is also a status set register and a non-zero value sets the level interrupt. Software can also write a value to the CLR_ECC_SEC field to decrement this counter. 0h - No single-bit errors have occurred 1h - 1 single-bit has occurred 2h - 2 single-bit have occurred 3h - 3 or more single-bit have occurred **Note** Not used in inject only mode. Always read as 0h.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.7 ECC_ERR_STAT2 Register (Offset = 24h)

Return to Summary Table.

Contains data bit(s) corresponding to the single/double-bit error status.

This register contains ECC status bits for the selected ECC RAM

Figure 12-482. ECC_ERR_STAT2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC_ROW															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_ROW															
R-0h															

Table 12-380. ECC_ERR_STAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ECC_ROW	R	0h	Row address where the single or double-bit error has occurred. **Note** Not used in inject only mode. Always read as 0h.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.8 ECC_ERR_STAT3 Register (Offset = 28h)

Return to Summary Table.

Contains error status information

This register contains ECC status bits for the selected ECC RAM

Figure 12-483. ECC_ERR_STAT3 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						CLR_SVBUS_TIMEOUT	RESERVED							SVBUS_TIMEOUT	WB_PEND
R-0h						R-0h	R-0h							R/W-0h	R-0h

Table 12-381. ECC_ERR_STAT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reserved
9	CLR_SVBUS_TIMEOUT	R/W	0h	Clear ECC serial interface timeout error status 0h - No effect 1h - Clears this bit and the SVBUS_TIMEOUT_ERR bit
8:2	RESERVED	R	0h	Reserved
1	SVBUS_TIMEOUT	R/W	0h	ECC serial interface timeout error. Write a 1h to set the flag 0h - No timeout error 1h - Timeout error
0	WB_PEND	R	0h	Delayed write-back pending status. 0h - An ECC data correction write-back is not pending 1h - An ECC data correction write-back is pending **Note** Not used in inject only mode. Always read as 0h.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.9 ECC_CBASS_REV Register (Offset = 10h)

Return to Summary Table.

Revision Register

The Revision Register contains the major and minor revisions.

Figure 12-484. ECC_CBASS_REV Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV															
R-Xh															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV															
R-Xh															

Table 12-382. ECC_CBASS_REV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R	Xh	TI Internal data

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.10 ECC_CBASS_CTRL Register (Offset = 14h)

Return to Summary Table.

Control Register

The Control Register allows control over the checkers and injection.

Figure 12-485. ECC_CBASS_CTRL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				ECC_PATTERN				RESERVED		FORCE_N_BIT	FORCE_DE	FORCE_SE	RESERVED	ECC_CHECK	RESERVED
R/W-0h										R/W-0h	R/W-0h	R/W-0h		R/W-1/0	

Table 12-383. ECC_CBASS_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED		0h	Reserved
11:8	ECC_PATTERN	R/W	0h	The data pattern to use for injection. 0h = 0s 1h = Fs 2h = As 3h = 5s
7:6	RESERVED	R	0h	Always read as 0.
5	FORCE_N_BIT	R/W	0h	Update injection fields after the injection to setup for the next incremental injection. 0h = keep current settings after injection. 1h = increment to next bit or group after injection.
4	FORCE_DE	R/W	0h	Inject a double bit error when set. Will be automatically cleared when injection has completed.
3	FORCE_SE	R/W	0h	Inject a single bit error when set. Will be automatically cleared when injection has completed.
2	RESERVED	R/W	0h	Reserved
1	ECC_CHECK	R/W	1h	Enable checkers. 0h = Disabled 1h = Enabled
0	RESERVED	R	0h	Reserved.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.11 ECC_CBASS_ERR_CTRL1 Register (Offset = 18h)

Return to Summary Table.

Error 1 Register

The Error 1 Register allows setting the injection data.

Figure 12-486. ECC_CBASS_ERR_CTRL1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED							ECC_BIT1								
R/W-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_GRP															
R/W-0h															

Table 12-384. ECC_CBASS_ERR_CTRL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reserved
24:16	ECC_BIT1	R/W	0h	First bit to inject an error.
15:0	ECC_GRP	R/W	0h	Group of checker to inject.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.12 ECC_CBASS_ERR_CTRL2 Register (Offset = 1Ch)

Return to Summary Table.

Error 2 Register

The Error 2 Register allows setting the injection data.

Figure 12-487. ECC_CBASS_ERR_CTRL2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							ECC_BIT2								
							R/W-0h								

Table 12-385. ECC_CBASS_ERR_CTRL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED			Reserved
8:0	ECC_BIT2	R/W	0h	Second bit to inject an error. Only valid if ECC_CBASS_CTRL[4] FORCE_DE is set.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.13 ECC_CBASS_ERR_STAT1 Register (Offset = 20h)

Return to Summary Table.

Error Status 1 Register

The Error Status 1 Register allows reading the captured error data.

Figure 12-488. ECC_CBASS_ERR_STAT1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC_GRP															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INJ_UNC_PEND_CLR		INJ_COR_PEND_CLR		UNC_PEND_CLR		COR_PEND_CLR		INJ_UNC_PEND		INJ_COR_PEND		UNC_PEND		COR_PEND	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

Table 12-386. ECC_CBASS_ERR_STAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	ECC_GRP	R	0h	Specific checker that reported the error. The exact mapping to signals is IP dependent.
15:14	INJ_UNC_PEND_CLR	R/W	0h	Number of injected uncorrected pending interrupts (same value as UNC_PEND). Writing decrements INJ_UNC_PEND by that value.
13:12	INJ_COR_PEND_CLR	R/W	0h	Number of injected corrected pending interrupts (same value as COR_PEND). Writing decrements INJ_COR_PEND by that value.
11:10	UNC_PEND_CLR	R/W	0h	Number of uncorrected pending interrupts (same value as UNC_PEND). Writing decrements UNC_PEND by that value.
9:8	COR_PEND_CLK	R/W	0h	Number of corrected pending interrupts (same value as COR_PEND). Writing decrements COR_PEND by that value.
7:6	INJ_UNC_PEND	R/W	0h	Number of injected uncorrected pending interrupts. Writing increments by that value.
5:4	INJ_COR_PEND	R/W	0h	Number of injected corrected pending interrupts. Writing increments by that value.
3:2	UNC_PEND	R/W	0h	Number of uncorrected pending interrupts. Writing increments by that value.
1:0	COR_PEND	R/W	0h	Number of corrected pending interrupts. Writing increments by that value.

Copyright © 2026 Texas Instruments Incorporated

12.8.4.4.14 ECC_CBASS_ERR_STAT2 Register (Offset = 24h)

Return to Summary Table.

Error Status 2 Register

The Error Status 2 Register allows reading the captured error data.

Figure 12-489. ECC_CBASS_ERR_STAT2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERR_TYPE															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_BIT															
R-0h															

Table 12-387. ECC_CBASS_ERR_STAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	ERR_TYPE	R	0h	This field is not supported and can read any value and be ignored.
15:0	ERR_BIT	R	0h	Bit that caused the error. Always valid for EDC single bit corrected errors or redundant errors. Will identify parity segment number (but not the exact bit) with a parity error. Is not valid for EDC double bit errors.

12.8.5 ECC Aggregator Configurations

Table 12-388. Properties of ECC Aggregator Instance COMPUTE_CLUSTER0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_A53_DUAL_U_IDATA_SPRAM_BANK0_LO_ECC_SVBUS	0	ECC Wrapper	Inject only	Yes	42	10 KB
CPU0_A53_DUAL_U_IDATA_SPRAM_BANK0_HI_ECC_SVBUS	1	ECC Wrapper	Inject only	Yes	42	10 KB
CPU0_A53_DUAL_U_IDATA_SPRAM_BANK1_LO_ECC_SVBUS	2	ECC Wrapper	Inject only	Yes	42	10 KB
CPU0_A53_DUAL_U_IDATA_SPRAM_BANK1_HI_ECC_SVBUS	3	ECC Wrapper	Inject only	Yes	42	10 KB
CPU0_A53_DUAL_U_ITAG_SPRAM_RAM0_ECC_SVBUS	4	ECC Wrapper	Inject only	Yes	32	1 KB
CPU0_A53_DUAL_U_ITAG_SPRAM_RAM1_ECC_SVBUS	5	ECC Wrapper	Inject only	Yes	32	1 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK0_ECC_SVBUS	6	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK1_ECC_SVBUS	7	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK2_ECC_SVBUS	8	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK3_ECC_SVBUS	9	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK4_ECC_SVBUS	10	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK5_ECC_SVBUS	11	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK6_ECC_SVBUS	12	ECC Wrapper	Inject only	Yes	39	5 KB
CPU0_A53_DUAL_U_DDATA_SPRAM_BANK7_ECC_SVBUS	13	ECC Wrapper	Inject only	Yes	39	5 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-388. Properties of ECC Aggregator Instance COMPUTE_CLUSTER0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_A53_DUAL_U_DTAG_SPRAM_BANK0_ECC_SVBUS	14	ECC Wrapper	Inject only	Yes	31	496 B
CPU0_A53_DUAL_U_DTAG_SPRAM_BANK1_ECC_SVBUS	15	ECC Wrapper	Inject only	Yes	31	496 B
CPU0_A53_DUAL_U_DTAG_SPRAM_BANK2_ECC_SVBUS	16	ECC Wrapper	Inject only	Yes	31	496 B
CPU0_A53_DUAL_U_DTAG_SPRAM_BANK3_ECC_SVBUS	17	ECC Wrapper	Inject only	Yes	31	496 B
CPU0_A53_DUAL_U_DDIRTY_SPRAM_ECC_SVBUS	18	ECC Wrapper	Inject only	Yes	12	384 B
CPU0_A53_DUAL_U_TLB_SPRAM_BANK0_ECC_SVBUS	19	ECC Wrapper	Inject only	Yes	117	3 KB
CPU0_A53_DUAL_U_TLB_SPRAM_BANK1_ECC_SVBUS	20	ECC Wrapper	Inject only	Yes	117	3 KB
CPU0_A53_DUAL_U_TLB_SPRAM_BANK2_ECC_SVBUS	21	ECC Wrapper	Inject only	Yes	117	3 KB
CPU0_A53_DUAL_U_TLB_SPRAM_BANK3_ECC_SVBUS	22	ECC Wrapper	Inject only	Yes	117	3 KB
CPU0_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY0_ECC_SVBUS	23	ECC Wrapper	Inject only	Yes	38	608 B
CPU0_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY1_ECC_SVBUS	24	ECC Wrapper	Inject only	Yes	38	608 B
CPU0_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY2_ECC_SVBUS	25	ECC Wrapper	Inject only	Yes	38	608 B
CPU0_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY3_ECC_SVBUS	26	ECC Wrapper	Inject only	Yes	38	608 B
CPU1_A53_DUAL_U_IDATA_SPRAM_BANK0_LO_ECC_SVBUS	0	ECC Wrapper	Inject only	Yes	42	10 KB
CPU1_A53_DUAL_U_IDATA_SPRAM_BANK0_HI_ECC_SVBUS	1	ECC Wrapper	Inject only	Yes	42	10 KB
CPU1_A53_DUAL_U_IDATA_SPRAM_BANK1_LO_ECC_SVBUS	2	ECC Wrapper	Inject only	Yes	42	10 KB
CPU1_A53_DUAL_U_IDATA_SPRAM_BANK1_HI_ECC_SVBUS	3	ECC Wrapper	Inject only	Yes	42	10 KB
CPU1_A53_DUAL_U_ITAG_SPRAM_RAM0_ECC_SVBUS	4	ECC Wrapper	Inject only	Yes	32	1 KB
CPU1_A53_DUAL_U_ITAG_SPRAM_RAM1_ECC_SVBUS	5	ECC Wrapper	Inject only	Yes	32	1 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK0_ECC_SVBUS	6	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK1_ECC_SVBUS	7	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK2_ECC_SVBUS	8	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK3_ECC_SVBUS	9	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK4_ECC_SVBUS	10	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK5_ECC_SVBUS	11	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK6_ECC_SVBUS	12	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DDATA_SPRAM_BANK7_ECC_SVBUS	13	ECC Wrapper	Inject only	Yes	39	5 KB
CPU1_A53_DUAL_U_DTAG_SPRAM_BANK0_ECC_SVBUS	14	ECC Wrapper	Inject only	Yes	31	496 B
CPU1_A53_DUAL_U_DTAG_SPRAM_BANK1_ECC_SVBUS	15	ECC Wrapper	Inject only	Yes	31	496 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-388. Properties of ECC Aggregator Instance COMPUTE_CLUSTER0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU1_A53_DUAL_U_DTAG_SPRAM_BANK2_ECC_SVBUS	16	ECC Wrapper	Inject only	Yes	31	496 B
CPU1_A53_DUAL_U_DTAG_SPRAM_BANK3_ECC_SVBUS	17	ECC Wrapper	Inject only	Yes	31	496 B
CPU1_A53_DUAL_U_DDIRTY_SPRAM_ECC_SVBUS	18	ECC Wrapper	Inject only	Yes	12	384 B
CPU1_A53_DUAL_U_TLB_SPRAM_BANK0_ECC_SVBUS	19	ECC Wrapper	Inject only	Yes	117	3 KB
CPU1_A53_DUAL_U_TLB_SPRAM_BANK1_ECC_SVBUS	20	ECC Wrapper	Inject only	Yes	117	3 KB
CPU1_A53_DUAL_U_TLB_SPRAM_BANK2_ECC_SVBUS	21	ECC Wrapper	Inject only	Yes	117	3 KB
CPU1_A53_DUAL_U_TLB_SPRAM_BANK3_ECC_SVBUS	22	ECC Wrapper	Inject only	Yes	117	3 KB
CPU1_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY0_ECC_SVBUS	23	ECC Wrapper	Inject only	Yes	38	608 B
CPU1_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY1_ECC_SVBUS	24	ECC Wrapper	Inject only	Yes	38	608 B
CPU1_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY2_ECC_SVBUS	25	ECC Wrapper	Inject only	Yes	38	608 B
CPU1_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY3_ECC_SVBUS	26	ECC Wrapper	Inject only	Yes	38	608 B
CPU2_A53_DUAL_U_IDATA_SPRAM_BANK0_LO_ECC_SVBUS	0	ECC Wrapper	Inject only	Yes	42	10 KB
CPU2_A53_DUAL_U_IDATA_SPRAM_BANK0_HI_ECC_SVBUS	1	ECC Wrapper	Inject only	Yes	42	10 KB
CPU2_A53_DUAL_U_IDATA_SPRAM_BANK1_LO_ECC_SVBUS	2	ECC Wrapper	Inject only	Yes	42	10 KB
CPU2_A53_DUAL_U_IDATA_SPRAM_BANK1_HI_ECC_SVBUS	3	ECC Wrapper	Inject only	Yes	42	10 KB
CPU2_A53_DUAL_U_ITAG_SPRAM_RAM0_ECC_SVBUS	4	ECC Wrapper	Inject only	Yes	32	1 KB
CPU2_A53_DUAL_U_ITAG_SPRAM_RAM1_ECC_SVBUS	5	ECC Wrapper	Inject only	Yes	32	1 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK0_ECC_SVBUS	6	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK1_ECC_SVBUS	7	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK2_ECC_SVBUS	8	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK3_ECC_SVBUS	9	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK4_ECC_SVBUS	10	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK5_ECC_SVBUS	11	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK6_ECC_SVBUS	12	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DDATA_SPRAM_BANK7_ECC_SVBUS	13	ECC Wrapper	Inject only	Yes	39	5 KB
CPU2_A53_DUAL_U_DTAG_SPRAM_BANK0_ECC_SVBUS	14	ECC Wrapper	Inject only	Yes	31	496 B
CPU2_A53_DUAL_U_DTAG_SPRAM_BANK1_ECC_SVBUS	15	ECC Wrapper	Inject only	Yes	31	496 B
CPU2_A53_DUAL_U_DTAG_SPRAM_BANK2_ECC_SVBUS	16	ECC Wrapper	Inject only	Yes	31	496 B
CPU2_A53_DUAL_U_DTAG_SPRAM_BANK3_ECC_SVBUS	17	ECC Wrapper	Inject only	Yes	31	496 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-388. Properties of ECC Aggregator Instance COMPUTE_CLUSTER0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU2_A53_DUAL_U_DDIRTY_SPRAM_ECC_SVBUS	18	ECC Wrapper	Inject only	Yes	12	384 B
CPU2_A53_DUAL_U_TLB_SPRAM_BANK0_ECC_SVBUS	19	ECC Wrapper	Inject only	Yes	117	3 KB
CPU2_A53_DUAL_U_TLB_SPRAM_BANK1_ECC_SVBUS	20	ECC Wrapper	Inject only	Yes	117	3 KB
CPU2_A53_DUAL_U_TLB_SPRAM_BANK2_ECC_SVBUS	21	ECC Wrapper	Inject only	Yes	117	3 KB
CPU2_A53_DUAL_U_TLB_SPRAM_BANK3_ECC_SVBUS	22	ECC Wrapper	Inject only	Yes	117	3 KB
CPU2_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY0_ECC_SVBUS	23	ECC Wrapper	Inject only	Yes	38	608 B
CPU2_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY1_ECC_SVBUS	24	ECC Wrapper	Inject only	Yes	38	608 B
CPU2_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY2_ECC_SVBUS	25	ECC Wrapper	Inject only	Yes	38	608 B
CPU2_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY3_ECC_SVBUS	26	ECC Wrapper	Inject only	Yes	38	608 B
CPU3_A53_DUAL_U_IDATA_SPRAM_BANK0_LO_ECC_SVBUS	0	ECC Wrapper	Inject only	Yes	42	10 KB
CPU3_A53_DUAL_U_IDATA_SPRAM_BANK0_HI_ECC_SVBUS	1	ECC Wrapper	Inject only	Yes	42	10 KB
CPU3_A53_DUAL_U_IDATA_SPRAM_BANK1_LO_ECC_SVBUS	2	ECC Wrapper	Inject only	Yes	42	10 KB
CPU3_A53_DUAL_U_IDATA_SPRAM_BANK1_HI_ECC_SVBUS	3	ECC Wrapper	Inject only	Yes	42	10 KB
CPU3_A53_DUAL_U_ITAG_SPRAM_RAM0_ECC_SVBUS	4	ECC Wrapper	Inject only	Yes	32	1 KB
CPU3_A53_DUAL_U_ITAG_SPRAM_RAM1_ECC_SVBUS	5	ECC Wrapper	Inject only	Yes	32	1 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK0_ECC_SVBUS	6	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK1_ECC_SVBUS	7	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK2_ECC_SVBUS	8	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK3_ECC_SVBUS	9	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK4_ECC_SVBUS	10	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK5_ECC_SVBUS	11	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK6_ECC_SVBUS	12	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DDATA_SPRAM_BANK7_ECC_SVBUS	13	ECC Wrapper	Inject only	Yes	39	5 KB
CPU3_A53_DUAL_U_DTAG_SPRAM_BANK0_ECC_SVBUS	14	ECC Wrapper	Inject only	Yes	31	496 B
CPU3_A53_DUAL_U_DTAG_SPRAM_BANK1_ECC_SVBUS	15	ECC Wrapper	Inject only	Yes	31	496 B
CPU3_A53_DUAL_U_DTAG_SPRAM_BANK2_ECC_SVBUS	16	ECC Wrapper	Inject only	Yes	31	496 B
CPU3_A53_DUAL_U_DTAG_SPRAM_BANK3_ECC_SVBUS	17	ECC Wrapper	Inject only	Yes	31	496 B
CPU3_A53_DUAL_U_DDIRTY_SPRAM_ECC_SVBUS	18	ECC Wrapper	Inject only	Yes	12	384 B
CPU3_A53_DUAL_U_TLB_SPRAM_BANK0_ECC_SVBUS	19	ECC Wrapper	Inject only	Yes	117	3 KB
CPU3_A53_DUAL_U_TLB_SPRAM_BANK1_ECC_SVBUS	20	ECC Wrapper	Inject only	Yes	117	3 KB
CPU3_A53_DUAL_U_TLB_SPRAM_BANK2_ECC_SVBUS	21	ECC Wrapper	Inject only	Yes	117	3 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-388. Properties of ECC Aggregator Instance COMPUTE_CLUSTER0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU3_A53_DUAL_U_TLB_SPRAM_BANK3_ECC_SVBUS	22	ECC Wrapper	Inject only	Yes	117	3 KB
CPU3_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY0_ECC_SVBUS	23	ECC Wrapper	Inject only	Yes	38	608 B
CPU3_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY1_ECC_SVBUS	24	ECC Wrapper	Inject only	Yes	38	608 B
CPU3_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY2_ECC_SVBUS	25	ECC Wrapper	Inject only	Yes	38	608 B
CPU3_A53_DUAL_U_L1D_TAGRAM_SPRAM_WAY3_ECC_SVBUS	26	ECC Wrapper	Inject only	Yes	38	608 B
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY0_ECC_SVBUS	0	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY1_ECC_SVBUS	1	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY2_ECC_SVBUS	2	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY3_ECC_SVBUS	3	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY4_ECC_SVBUS	4	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY5_ECC_SVBUS	5	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY6_ECC_SVBUS	6	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY7_ECC_SVBUS	7	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY8_ECC_SVBUS	8	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY9_ECC_SVBUS	9	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY10_ECC_SVBUS	10	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY11_ECC_SVBUS	11	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY12_ECC_SVBUS	12	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY13_ECC_SVBUS	13	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY14_ECC_SVBUS	14	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_TAGRAM_SPRAM_WAY15_ECC_SVBUS	15	ECC Wrapper	Inject only	Yes	38	2 KB
A53_DUAL_U_L2_DATARAM_SPRAM_0_ECC_SVBUS	16	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_1_ECC_SVBUS	17	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_2_ECC_SVBUS	18	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_3_ECC_SVBUS	19	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_4_ECC_SVBUS	20	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_5_ECC_SVBUS	21	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_6_ECC_SVBUS	22	ECC Wrapper	Inject only	Yes	72	72 KB
A53_DUAL_U_L2_DATARAM_SPRAM_7_ECC_SVBUS	23	ECC Wrapper	Inject only	Yes	72	72 KB

Table 12-389. Properties of ECC Aggregator Instance CSI_RX_IF0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RX_SHIM_DMA_PSIL_FIFO	1	ECC Wrapper	Inject with error capture	Yes	143	36 KB
CSI_RX_IF_RAM_WRAPPER0_FIFO	2	ECC Wrapper	Inject only	Yes	45	11 KB
CSI_RX_IF_RAM_WRAPPER1_FIFO	3	ECC Wrapper	Inject only	Yes	45	6 KB
CSI_RX_IF_RAM_WRAPPER2_FIFO	4	ECC Wrapper	Inject only	Yes	44	352 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-389. Properties of ECC Aggregator Instance CSI_RX_IF0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RAM_WRAPPER3_FIFO	5	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_VP0_VP_FIFO	6	ECC Wrapper	Inject with error capture	Yes	36	9 KB
CSI_RX_IF_VP1_VP_FIFO	7	ECC Wrapper	Inject with error capture	Yes	36	9 KB

Table 12-390. Properties of ECC Aggregator Instance CSI_RX_IF0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CSI_RX_IF_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	2

Table 12-391. EDC checkers information for ECC Aggregator Instance CSI_RX_IF0

Protected Interconnect	Group ID	Width	Checker Type
CSI_RX_IF_EDC_CTRL_0	0	32	Parity
CSI_RX_IF_EDC_CTRL_0	1	32	Parity

Table 12-392. Properties of ECC Aggregator Instance CSI_RX_IF1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RX_SHIM_DMA_PSIL_FIFO	1	ECC Wrapper	Inject with error capture	Yes	143	36 KB
CSI_RX_IF_RAM_WRAPPER0_FIFO	2	ECC Wrapper	Inject only	Yes	45	11 KB
CSI_RX_IF_RAM_WRAPPER1_FIFO	3	ECC Wrapper	Inject only	Yes	45	6 KB
CSI_RX_IF_RAM_WRAPPER2_FIFO	4	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_RAM_WRAPPER3_FIFO	5	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_VP0_VP_FIFO	6	ECC Wrapper	Inject with error capture	Yes	36	9 KB
CSI_RX_IF_VP1_VP_FIFO	7	ECC Wrapper	Inject with error capture	Yes	36	9 KB

Table 12-393. Properties of ECC Aggregator Instance CSI_RX_IF1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CSI_RX_IF_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	2

Table 12-394. EDC checkers information for ECC Aggregator Instance CSI_RX_IF1

Protected Interconnect	Group ID	Width	Checker Type
CSI_RX_IF_EDC_CTRL_0	0	32	Parity
CSI_RX_IF_EDC_CTRL_0	1	32	Parity

Table 12-395. Properties of ECC Aggregator Instance CSI_RX_IF2

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RX_SHIM_DMA_PSIL_FIFO	1	ECC Wrapper	Inject with error capture	Yes	143	36 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-395. Properties of ECC Aggregator Instance CSI_RX_IF2 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RAM_WRAPPER0_FIFO	2	ECC Wrapper	Inject only	Yes	45	11 KB
CSI_RX_IF_RAM_WRAPPER1_FIFO	3	ECC Wrapper	Inject only	Yes	45	6 KB
CSI_RX_IF_RAM_WRAPPER2_FIFO	4	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_RAM_WRAPPER3_FIFO	5	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_VP0_VP_FIFO	6	ECC Wrapper	Inject with error capture	Yes	36	9 KB
CSI_RX_IF_VP1_VP_FIFO	7	ECC Wrapper	Inject with error capture	Yes	36	9 KB

Table 12-396. Properties of ECC Aggregator Instance CSI_RX_IF2

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CSI_RX_IF_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	2

Table 12-397. EDC checkers information for ECC Aggregator Instance CSI_RX_IF2

Protected Interconnect	Group ID	Width	Checker Type
CSI_RX_IF_EDC_CTRL_0	0	32	Parity
CSI_RX_IF_EDC_CTRL_0	1	32	Parity

Table 12-398. Properties of ECC Aggregator Instance CSI_RX_IF3

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_RX_IF_RX_SHIM_DMA_PSIL_FIFO	1	ECC Wrapper	Inject with error capture	Yes	143	36 KB
CSI_RX_IF_RAM_WRAPPER0_FIFO	2	ECC Wrapper	Inject only	Yes	45	11 KB
CSI_RX_IF_RAM_WRAPPER1_FIFO	3	ECC Wrapper	Inject only	Yes	45	6 KB
CSI_RX_IF_RAM_WRAPPER2_FIFO	4	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_RAM_WRAPPER3_FIFO	5	ECC Wrapper	Inject only	Yes	44	352 B
CSI_RX_IF_VP0_VP_FIFO	6	ECC Wrapper	Inject with error capture	Yes	36	9 KB
CSI_RX_IF_VP1_VP_FIFO	7	ECC Wrapper	Inject with error capture	Yes	36	9 KB

Table 12-399. Properties of ECC Aggregator Instance CSI_RX_IF3

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CSI_RX_IF_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	2

Table 12-400. EDC checkers information for ECC Aggregator Instance CSI_RX_IF3

Protected Interconnect	Group ID	Width	Checker Type
CSI_RX_IF_EDC_CTRL_0	0	32	Parity
CSI_RX_IF_EDC_CTRL_0	1	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-401. Properties of ECC Aggregator Instance CSI_TX_IF0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CSI_TX_IF_V2_TX_SHIM_DMA_PSIL_FIFO	1	ECC Wrapper	Inject with error capture	Yes	143	36 KB
CSI_TX_IF_V2_TX_SHIM_KSDMA_PSIL_ENDPT_IPCFIFO_F0_TPRAM_256X167_SBW_SR	2	ECC Wrapper	Inject with error capture	Yes	167	5 KB
CSI_TX_IF_V2_RAM_WRAPPER0_FIFO	0	ECC Wrapper	Inject only	Yes	44	22 KB
CSI_TX_IF_V2_RAM_WRAPPER1_FIFO	1	ECC Wrapper	Inject only	Yes	44	6 KB

Table 12-402. Properties of ECC Aggregator Instance CSI_TX_IF0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CSI_TX_IF_V2_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	4

Table 12-403. EDC checkers information for ECC Aggregator Instance CSI_TX_IF0

Protected Interconnect	Group ID	Width	Checker Type
CSI_TX_IF_V2_EDC_CTRL_0	0	32	Parity
CSI_TX_IF_V2_EDC_CTRL_0	1	32	Parity
CSI_TX_IF_V2_EDC_CTRL_0	2	32	Parity
CSI_TX_IF_V2_EDC_CTRL_0	3	32	Parity

Table 12-404. Properties of ECC Aggregator Instance DSS_DSI0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
K3_DSS_DSI_TOP_DSI_EDC_CTRL_SYS_EDC_CTRL_0	0	EDC Interconnect	Inject with error capture	Yes	2

Table 12-405. EDC checkers information for ECC Aggregator Instance DSS_DSI0

Protected Interconnect	Group ID	Width	Checker Type
K3_DSS_DSI_DSI_TOP_DSI_EDC_CTRL_SYS_EDC_CTRL_0	0	32	Parity
K3_DSS_DSI_DSI_TOP_DSI_EDC_CTRL_SYS_EDC_CTRL_0	1	32	Parity

Table 12-406. Properties of ECC Aggregator Instance FSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
OSPI_OSPI_WRAP_SRAM	0	ECC Wrapper	Inject with error capture	Yes	32	1 KB

Table 12-407. Properties of ECC Aggregator Instance GICSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
ICB_RAMECC	0	ECC Wrapper	Inject only	No	16	640 B
ITE_RAMECC	1	ECC Wrapper	Inject only	No	58	464 B
LPI_RAMECC	2	ECC Wrapper	Inject only	No	26	208 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-408. Properties of ECC Aggregator Instance MCAN0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MCANSS_MSGMEM_WRAP_MSGMEM_ECC	0	ECC Wrapper	Inject with error capture	No	32	17 KB

Table 12-409. Properties of ECC Aggregator Instance MCAN0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CTRL_EDC_VBUSS	1	EDC Interconnect	Inject with error capture	Yes	1

Table 12-410. EDC checkers information for ECC Aggregator Instance MCAN0

Protected Interconnect	Group ID	Width	Checker Type
CTRL_EDC_VBUSS	0	32	Parity

Table 12-411. Properties of ECC Aggregator Instance MCAN1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MCANSS_MSGMEM_WRAP_MSGMEM_ECC	0	ECC Wrapper	Inject with error capture	No	32	17 KB

Table 12-412. Properties of ECC Aggregator Instance MCAN1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CTRL_EDC_VBUSS	1	EDC Interconnect	Inject with error capture	Yes	1

Table 12-413. EDC checkers information for ECC Aggregator Instance MCAN1

Protected Interconnect	Group ID	Width	Checker Type
CTRL_EDC_VBUSS	0	32	Parity

Table 12-414. Properties of ECC Aggregator Instance MCU_ECC_AGGR1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_RMST_SRC_BUSECC	0	EDC Interconnect	Inject with error capture	Yes	69
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_SLV_DST_BUSECC	1	EDC Interconnect	Inject with error capture	Yes	65
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_WMST_SRC_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	69
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_P2P_CPU0_CFG_SLV_DST_BUSECC	3	EDC Interconnect	Inject with error capture	Yes	4
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_P2P_CPU0_PMST_SRC_BUSECC	4	EDC Interconnect	Inject with error capture	Yes	15
ISAM62A_MCU_PULSAR_UL_BR_IECC_AGGR_CFG_DST_BUSECC	5	EDC Interconnect	Inject with error capture	Yes	4
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	6	EDC Interconnect	Inject with error capture	Yes	6

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	0	24	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	2	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	3	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	4	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	5	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	7	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	9	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	12	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	13	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	14	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	15	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	16	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	17	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	19	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	20	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	21	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	22	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	23	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	24	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	25	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	26	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	27	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	28	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	30	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	31	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	32	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	33	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	34	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	35	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	36	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	37	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	38	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	39	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	40	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	41	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	42	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	43	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	44	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	45	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	46	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	47	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	48	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	49	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	50	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	51	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	52	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	53	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	54	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	55	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	56	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	57	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	58	7	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	59	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	60	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	61	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	62	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	63	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	64	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	65	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	66	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	67	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_SRC_BUSECC	68	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	2	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	3	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	4	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	5	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	6	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	7	23	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	8	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	12	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	13	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	14	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	15	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	16	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	17	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	19	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	20	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	21	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	22	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	23	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	24	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	25	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	26	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	27	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	28	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	30	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	31	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	32	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	33	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	34	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	35	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	36	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	37	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	38	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	39	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	40	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	41	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	42	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	43	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	44	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	45	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	46	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	47	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	48	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	49	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	50	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	51	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	52	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	53	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	54	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	55	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	56	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	57	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	58	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	59	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	60	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	61	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	62	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	63	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_DST_BUSECC	64	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	0	24	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	2	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	3	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	4	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	5	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	7	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	9	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	12	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	13	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	14	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	15	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	16	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	17	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	19	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	20	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	21	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	22	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	23	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	24	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	25	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	26	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	27	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	28	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	30	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	31	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	32	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	33	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	34	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	35	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	36	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	37	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	38	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	39	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	40	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	41	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	42	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	43	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	44	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	45	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	46	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	47	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	48	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	49	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	50	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	51	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	52	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	53	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	54	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	55	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	56	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	57	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	58	7	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	59	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	60	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	61	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	62	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	63	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	64	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	65	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	66	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	67	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_SRC_BUSECC	68	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_DST_BUSECC	0	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_DST_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_DST_BUSECC	2	11	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_DST_BUSECC	3	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	2	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	3	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	4	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	5	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	7	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	11	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	12	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	13	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_SRC_BUSECC	14	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_DST_BUSECC	0	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_DST_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_DST_BUSECC	2	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-415. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_DST_BUSECC	3	3	Parity
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	0	1	Redundant
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	1	32	EDC
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	2	1	Parity
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	3	10	Parity
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	4	4	Parity
SAM62A_MCU_PULSAR_UL_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-416. Properties of ECC Aggregator Instance MCU_MCAN0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MCANSS_MSGMEM_WRAP_MSGMEM_ECC	0	ECC Wrapper	Inject with error capture	No	32	17 KB

Table 12-417. Properties of ECC Aggregator Instance MCU_MCAN0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CTRL_EDC_VBUSS	1	EDC Interconnect	Inject with error capture	Yes	1

Table 12-418. EDC checkers information for ECC Aggregator Instance MCU_MCAN0

Protected Interconnect	Group ID	Width	Checker Type
CTRL_EDC_VBUSS	0	32	Parity

Table 12-419. Properties of ECC Aggregator Instance MCU_MCAN1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MCANSS_MSGMEM_WRAP_MSGMEM_ECC	0	ECC Wrapper	Inject with error capture	No	32	17 KB

Table 12-420. Properties of ECC Aggregator Instance MCU_MCAN1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CTRL_EDC_VBUSS	1	EDC Interconnect	Inject with error capture	Yes	1

Table 12-421. EDC checkers information for ECC Aggregator Instance MCU_MCAN1

Protected Interconnect	Group ID	Width	Checker Type
CTRL_EDC_VBUSS	0	32	Parity

Table 12-422. Properties of ECC Aggregator Instance MCU_MSRAM_256K0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MSRAM32KX64E_MSRAM0_ECC0	0	ECC Wrapper	Inject with error capture	No	64	256 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-423. Properties of ECC Aggregator Instance MCU_MSRAM_256K0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	1	EDC Interconnect	Inject with error capture	Yes	65
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	2	EDC Interconnect	Inject with error capture	Yes	6

Table 12-424. EDC checkers information for ECC Aggregator Instance MCU_MSRAM_256K0

Protected Interconnect	Group ID	Width	Checker Type
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	0	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	1	12	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	2	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	3	10	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	4	18	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	5	2	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	6	3	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	7	12	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	8	4	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	9	3	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	10	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	11	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	12	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	13	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	14	10	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	15	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	16	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	17	4	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	18	8	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	19	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	20	32	EDC
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	21	32	EDC
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	22	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	23	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	24	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	25	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	26	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	27	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	28	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	29	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	30	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	31	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	32	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	33	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	34	20	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	35	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	36	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	37	64	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-424. EDC checkers information for ECC Aggregator Instance MCU_MSRAM_256K0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	38	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	39	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	40	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	41	8	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	42	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	43	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	44	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	45	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	46	56	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	47	22	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	48	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	49	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	50	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	51	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	52	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	53	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	54	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	55	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	56	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	57	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	58	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	59	30	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	60	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	61	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	62	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	63	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	64	36	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	0	1	Redundant
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	1	32	EDC
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	2	1	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	3	10	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	4	4	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-425. Properties of ECC Aggregator Instance MCU_MSRAM_256K1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
MSRAM32KX64E_MSRAM0_ECC0	0	ECC Wrapper	Inject with error capture	No	64	256 KB

Table 12-426. Properties of ECC Aggregator Instance MCU_MSRAM_256K1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	1	EDC Interconnect	Inject with error capture	Yes	65

Copyright © 2026 Texas Instruments Incorporated

Table 12-426. Properties of ECC Aggregator Instance MCU_MSRAM_256K1 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	2	EDC Interconnect	Inject with error capture	Yes	6

Table 12-427. EDC checkers information for ECC Aggregator Instance MCU_MSRAM_256K1

Protected Interconnect	Group ID	Width	Checker Type
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	0	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	1	12	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	2	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	3	10	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	4	18	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	5	2	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	6	3	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	7	12	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	8	4	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	9	3	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	10	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	11	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	12	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	13	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	14	10	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	15	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	16	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	17	4	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	18	8	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	19	1	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	20	32	EDC
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	21	32	EDC
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	22	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	23	1	Redundant
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	24	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	25	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	26	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	27	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	28	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	29	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	30	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	31	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	32	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	33	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	34	20	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	35	58	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	36	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	37	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	38	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	39	64	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-427. EDC checkers information for ECC Aggregator Instance MCU_MSRAM_256K1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	40	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	41	8	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	42	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	43	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	44	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	45	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	46	56	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	47	22	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	48	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	49	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	50	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	51	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	52	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	53	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	54	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	55	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	56	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	57	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	58	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	59	30	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	60	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	61	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	62	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	63	64	Parity
MSRAM32KX64E_MSRAM0_EDC_CTRL_0	64	36	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	0	1	Redundant
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	1	32	EDC
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	2	1	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	3	10	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	4	4	Parity
MSRAM32KX64E_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-428. Properties of ECC Aggregator Instance MCU_R5FSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_ITAG_RAM0	0	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM1	1	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM2	2	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM3	3	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_IDATA_BANK0	4	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK1	5	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK2	6	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK3	7	ECC Wrapper	Inject only	No	72	9 KB
CPU0_DTAG_RAM0	8	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM1	9	ECC Wrapper	Inject only	Yes	28	896 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-428. Properties of ECC Aggregator Instance MCU_R5FSS0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_DTAG_RAM2	10	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM3	11	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDIRTY_RAM	12	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDATA_RAM0	13	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM1	14	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM2	15	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM3	16	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM4	17	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM5	18	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM6	19	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM7	20	ECC Wrapper	Inject only	No	39	5 KB
PULSAR_ULS_ATCM0_BANK0	21	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_ULS_ATCM0_BANK1	22	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_ULS_B0TCM0_BANK0	23	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_ULS_B0TCM0_BANK1	24	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_ULS_B1TCM0_BANK0	25	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_ULS_B1TCM0_BANK1	26	ECC Wrapper	Inject only	No	39	10 KB
CPU0_KS_VIM_RAMECC	27	ECC Wrapper	Inject with error capture	Yes	30	960 B
CPU0_AXI2VBUSM_MEM_MST_RAMECC	29	ECC Wrapper	Inject with error capture	Yes	66	528 B

Table 12-429. Properties of ECC Aggregator Instance MCU_R5FSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
CPU0_VBUSM2AXI_EDC	28	EDC Interconnect	Inject with error capture	Yes	36
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	30	EDC Interconnect	Inject with error capture	Yes	37
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	31	EDC Interconnect	Inject with error capture	Yes	15
SCRP_EDC	32	EDC Interconnect	Inject with error capture	Yes	26
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	33	EDC Interconnect	Inject with error capture	Yes	77
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	34	EDC Interconnect	Inject with error capture	Yes	6

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0

Protected Interconnect	Group ID	Width	Checker Type
CPU0_VBUSM2AXI_EDC	0	1	Redundant
CPU0_VBUSM2AXI_EDC	1	12	Parity
CPU0_VBUSM2AXI_EDC	2	4	Parity
CPU0_VBUSM2AXI_EDC	3	12	Parity
CPU0_VBUSM2AXI_EDC	4	23	Parity
CPU0_VBUSM2AXI_EDC	5	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
CPU0_VBUSM2AXI_EDC	6	10	Parity
CPU0_VBUSM2AXI_EDC	7	2	Parity
CPU0_VBUSM2AXI_EDC	8	3	Parity
CPU0_VBUSM2AXI_EDC	9	1	Parity
CPU0_VBUSM2AXI_EDC	10	2	Parity
CPU0_VBUSM2AXI_EDC	11	2	Parity
CPU0_VBUSM2AXI_EDC	12	1	Parity
CPU0_VBUSM2AXI_EDC	13	1	Parity
CPU0_VBUSM2AXI_EDC	14	1	Parity
CPU0_VBUSM2AXI_EDC	15	3	Parity
CPU0_VBUSM2AXI_EDC	16	4	Parity
CPU0_VBUSM2AXI_EDC	17	2	Parity
CPU0_VBUSM2AXI_EDC	18	2	Parity
CPU0_VBUSM2AXI_EDC	19	2	Parity
CPU0_VBUSM2AXI_EDC	20	1	Redundant
CPU0_VBUSM2AXI_EDC	21	32	EDC
CPU0_VBUSM2AXI_EDC	22	32	EDC
CPU0_VBUSM2AXI_EDC	23	8	Parity
CPU0_VBUSM2AXI_EDC	24	1	Redundant
CPU0_VBUSM2AXI_EDC	25	1	Redundant
CPU0_VBUSM2AXI_EDC	26	10	Parity
CPU0_VBUSM2AXI_EDC	27	64	Parity
CPU0_VBUSM2AXI_EDC	28	63	Parity
CPU0_VBUSM2AXI_EDC	29	17	Parity
CPU0_VBUSM2AXI_EDC	30	10	Parity
CPU0_VBUSM2AXI_EDC	31	7	Parity
CPU0_VBUSM2AXI_EDC	32	8	Parity
CPU0_VBUSM2AXI_EDC	33	8	Parity
CPU0_VBUSM2AXI_EDC	34	64	Parity
CPU0_VBUSM2AXI_EDC	35	56	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	0	1	Redundant
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	1	12	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	2	1	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	3	3	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	4	10	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	5	4	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	6	32	EDC
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	7	32	EDC
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	8	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	9	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	10	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	11	48	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	12	32	EDC
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	13	32	EDC
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	14	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	15	32	EDC
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	16	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	17	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	18	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	19	56	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	20	2	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	21	2	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	22	1	Redundant
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	23	1	Redundant
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	24	3	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	25	10	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	26	12	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	27	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	28	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	29	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	30	38	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	31	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	32	17	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	33	8	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	34	12	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	35	64	Parity
PULSAR_ULS_MEM_MST0_EDC_CTRL_0	36	20	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	0	1	Redundant
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	1	1	Redundant
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	2	3	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	3	32	EDC
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	4	32	EDC
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	5	4	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	6	1	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	7	1	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	8	1	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	9	2	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	10	14	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	11	1	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	12	36	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	13	10	Parity
PULSAR_ULS_PULSAR_AHB2VBUSP_CPU0_EDC_CTRL_0	14	4	Parity
SCRP_EDC	0	11	Parity
SCRP_EDC	1	1	Redundant
SCRP_EDC	2	32	EDC
SCRP_EDC	3	1	Redundant
SCRP_EDC	4	8	Parity
SCRP_EDC	5	4	Redundant
SCRP_EDC	6	3	Parity
SCRP_EDC	7	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
SCRP_EDC	8	4	Redundant
SCRP_EDC	9	12	EDC
SCRP_EDC	10	2	Parity
SCRP_EDC	11	32	Parity
SCRP_EDC	12	4	Parity
SCRP_EDC	13	3	Parity
SCRP_EDC	14	12	Parity
SCRP_EDC	15	4	Parity
SCRP_EDC	16	1	Redundant
SCRP_EDC	17	1	Redundant
SCRP_EDC	18	32	EDC
SCRP_EDC	19	1	Parity
SCRP_EDC	20	32	Parity
SCRP_EDC	21	4	Parity
SCRP_EDC	22	3	Parity
SCRP_EDC	23	12	Parity
SCRP_EDC	24	4	Parity
SCRP_EDC	25	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	0	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	1	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	2	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	3	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	4	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	5	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	6	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	7	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	8	2	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	9	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	10	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	11	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	12	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	13	10	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	14	12	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	15	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	16	4	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	17	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	18	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	19	12	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	20	4	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	21	12	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	22	4	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	23	10	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	24	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	25	12	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	26	4	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	27	12	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	28	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	29	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	30	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	31	4	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	32	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	33	5	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	34	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	35	5	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	36	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	37	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	38	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	39	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	40	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	41	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	42	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	43	5	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	44	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	45	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	46	8	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	47	7	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	48	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	49	4	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	50	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	51	26	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	52	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	53	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	54	26	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	55	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	56	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	57	26	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	58	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	59	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	60	3	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	61	26	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	62	32	EDC
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	63	3	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	64	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	65	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	66	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	67	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-430. EDC checkers information for ECC Aggregator Instance MCU_R5FSS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	68	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	69	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	70	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	71	32	EDC
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	72	1	Redundant
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	73	1	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	74	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	75	10	Parity
PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_PULSAR_ULS_CPU0_CFG_SCRP_P_SCR1_SCR_EDC_CTRL_BUSECC	76	1	Parity
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	0	1	Redundant
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	1	32	EDC
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	2	1	Parity
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	3	10	Parity
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	4	4	Parity
PULSAR_ULS_CPU0_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-431. Properties of ECC Aggregator Instance MMCSD0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
EMMC8SS_16FFC_SDHC_WRAP_TXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB
EMMC8SS_16FFC_SDHC_WRAP_RXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB

Table 12-432. Properties of ECC Aggregator Instance MMCSD1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
EMMCSD4SS_SDHC_WRAP_RXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB
EMMCSD4SS_SDHC_WRAP_TXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB

Table 12-433. Properties of ECC Aggregator Instance MMCSD2

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
EMMCSD4SS_SDHC_WRAP_RXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB
EMMCSD4SS_SDHC_WRAP_TXMEM	0	ECC Wrapper	Inject with error capture	Yes	64	4 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-434. Properties of ECC Aggregator Instance PCIE0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_AXIMFIFO	0	ECC Wrapper	Inject only	Yes	81	324 B
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_AXISFIFO	1	ECC Wrapper	Inject only	Yes	97	388 B
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_DIBRAM	2	ECC Wrapper	Inject only	Yes	81	1 KB
PCIE_G2X1_64_CORE_AXI2VBUSM_MST_KSBUS_AXI2VBUSM_RDATA_BUFFER	3	ECC Wrapper	Inject with error capture	Yes	66	8 KB
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_PNPFIFO	0	ECC Wrapper	Inject only	Yes	45	13 KB
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_RXCPLFIFO	1	ECC Wrapper	Inject only	Yes	45	2 KB
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_RPLYBUF	2	ECC Wrapper	Inject only	Yes	45	2 KB
PCIE_G2X1_64_CORE_DBN_WRAP_RAMS_AXISRODR	3	ECC Wrapper	Inject only	Yes	44	1 KB

Table 12-435. Properties of ECC Aggregator Instance PDMA1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
SAM62_PDMA_UART_PDMA_CORE_TF0_F0_TPRAM_28X128_SBW_SR	0	ECC Wrapper	Inject with error capture	Yes	128	448 B
SAM62_PDMA_UART_PDMA_CORE_TF0_F1_TPRAM_28X128_SBW_SR	1	ECC Wrapper	Inject with error capture	Yes	128	448 B
SAM62_PDMA_UART_PDMA_CORE_RF0_F0_TPRAM_28X144_SBW_SR	2	ECC Wrapper	Inject with error capture	Yes	144	504 B
SAM62_PDMA_UART_PDMA_CORE_RF0_F1_TPRAM_28X144_SBW_SR	3	ECC Wrapper	Inject with error capture	Yes	144	504 B

Table 12-436. Properties of ECC Aggregator Instance PSRAMECC0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
PSRAM256X32E_PSRAM0_ECC	0	ECC Wrapper	Inject with error capture	No	32	1 KB

Table 12-437. Properties of ECC Aggregator Instance PSRAMECC1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
PSRAM256X32E_PSRAM0_ECC	0	ECC Wrapper	Inject with error capture	No	32	1 KB

Table 12-438. Properties of ECC Aggregator Instance R5FSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_ITAG_RAM0	0	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM1	1	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM2	2	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM3	3	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_IDATA_BANK0	4	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK1	5	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK2	6	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK3	7	ECC Wrapper	Inject only	No	72	9 KB
CPU0_DTAG_RAM0	8	ECC Wrapper	Inject only	Yes	28	896 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-438. Properties of ECC Aggregator Instance R5FSS0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_DTAG_RAM1	9	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM2	10	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM3	11	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDIRTY_RAM	12	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDATA_RAM0	13	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM1	14	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM2	15	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM3	16	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM4	17	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM5	18	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM6	19	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM7	20	ECC Wrapper	Inject only	No	39	5 KB
PULSAR_UL_ATCM0_BANK0	21	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_UL_ATCM0_BANK1	22	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_UL_B0TCM0_BANK0	23	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B0TCM0_BANK1	24	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B1TCM0_BANK0	25	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B1TCM0_BANK1	26	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_PULSAR_KS_VIM_COMMON_CORE0_RAM	27	ECC Wrapper	Inject with error capture	Yes	30	960 B
PULSAR_UL_MEM_MST0_KSBUS_AXI2VBUSM_RDATA_BUFFER	28	ECC Wrapper	Inject with error capture	Yes	66	528 B

Table 12-439. Properties of ECC Aggregator Instance SA3_SS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
DMSS_HSM_PKTDMA_CFG_CONFIG	1	ECC Wrapper	Inject with error capture	Yes	108	432 B
DMSS_HSM_PKTDMA_CFG_STATE	2	ECC Wrapper	Inject with error capture	Yes	256	192 B
DMSS_HSM_PKTDMA_TPCFIFO_F0	3	ECC Wrapper	Inject with error capture	Yes	141	212 B
DMSS_HSM_PKTDMA_TPCFIFO_F1	4	ECC Wrapper	Inject with error capture	Yes	141	212 B
DMSS_HSM_PKTDMA_RPCFIFO_F0	5	ECC Wrapper	Inject with error capture	Yes	128	384 B
DMSS_HSM_PKTDMA_RPCFIFO_F1	6	ECC Wrapper	Inject with error capture	Yes	128	384 B
DMSS_HSM_PKTDMA_RPCFIFO_WC	7	ECC Wrapper	Inject with error capture	Yes	22	132 B
DMSS_HSM_PKTDMA_STATS_STST0	8	ECC Wrapper	Inject with error capture	Yes	96	24 B
DMSS_HSM_PKTDMA_STATS_STSR0	9	ECC Wrapper	Inject with error capture	Yes	128	64 B
DMSS_HSM_PKTDMA_RINGOCC_CNTR	10	ECC Wrapper	Inject with error capture	Yes	18	144 B
DMSS_HSM_INTAGGR_STATREG_SR_SPRAM_8X128_SWW_SR	13	ECC Wrapper	Inject with error capture	Yes	128	128 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-439. Properties of ECC Aggregator Instance SA3_SS0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
DMSS_HSM_INTAGGR_COMMON_IM_TPRAM_158X34_SWW_SR	14	ECC Wrapper	Inject with error capture	Yes	34	672 B
DMSS_HSM_IPCSS_RINGACC_STRAM	17	ECC Wrapper	Inject with error capture	Yes	216	162 B
DMSS_HSM_IPCSS_SEC_PROXY_BUF_STRAM	18	ECC Wrapper	Inject with error capture	Yes	91	182 B
DMSS_HSM_IPCSS_SEC_PROXY_BUFRAM	19	ECC Wrapper	Inject with error capture	Yes	64	64 B
DMSS_HSM_IPCSS_MSRAM_ECC0	24	ECC Wrapper	Inject with error capture	Yes	64	4 KB
SA3_UL_CM_PKTRAM0_ECC	0	ECC Wrapper	Inject with error capture	Yes	64	512 B
SA3_UL_CM_PKTRAM1_ECC	1	ECC Wrapper	Inject with error capture	Yes	64	512 B
SA3_UL_CM_PKA_PROG_RAM_ECC	2	ECC Wrapper	Inject with error capture	Yes	24	15 KB
SA3_UL_CM_ENCR_CTXRAM_BANK01_ECC	3	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_ENCR_CTXRAM_BANK23_ECC	4	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_ENCR_CTXRAM_BANK4_ECC	5	ECC Wrapper	Inject with error capture	Yes	256	128 B
SA3_UL_CM_AUTH_CTXRAM_BANK01_ECC	6	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_AUTH_CTXRAM_BANK23_ECC	7	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_AUTH_CTXRAM_BANK45_ECC	8	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_AUTH_CTXRAM_BANK67_ECC	9	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_AUTH_CTXRAM_BANK89_ECC	10	ECC Wrapper	Inject with error capture	Yes	256	256 B
SA3_UL_CM_AUTH_CTXRAM_BANK10_ECC	11	ECC Wrapper	Inject with error capture	Yes	256	128 B

Table 12-440. Properties of ECC Aggregator Instance SA3_SS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
DMSS_HSM_ECCAGGR_EDC_CTRL	0	EDC Interconnect	Inject with error capture	Yes	6
DMSS_HSM_PKTDMA_EDC_CTRL_0	11	EDC Interconnect	Inject with error capture	Yes	72
DMSS_HSM_INTAGGR_EDC_CTRL_0	12	EDC Interconnect	Inject with error capture	Yes	18
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	15	EDC Interconnect	Inject with error capture	Yes	256
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	16	EDC Interconnect	Inject with error capture	Yes	57
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	20	EDC Interconnect	Inject with error capture	Yes	67
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	21	EDC Interconnect	Inject with error capture	Yes	72

Copyright © 2026 Texas Instruments Incorporated

Table 12-440. Properties of ECC Aggregator Instance SA3_SS0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
DMSS_HSM_IPCSS_CBASS_SCR_EDC_CTRL_0	22	EDC Interconnect	Inject with error capture	Yes	132
DMSS_HSM_IPCSS_CBASS_VD2GCLK_EDC_CTRL_0	23	EDC Interconnect	Inject with error capture	Yes	3
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	25	EDC Interconnect	Inject with error capture	Yes	63
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	26	EDC Interconnect	Inject with error capture	Yes	33
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	27	EDC Interconnect	Inject with error capture	Yes	33
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	28	EDC Interconnect	Inject with error capture	Yes	30
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	29	EDC Interconnect	Inject with error capture	Yes	30
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	30	EDC Interconnect	Inject with error capture	Yes	101
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	31	EDC Interconnect	Inject with error capture	Yes	101
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	32	EDC Interconnect	Inject with error capture	Yes	35
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	33	EDC Interconnect	Inject with error capture	Yes	35
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	34	EDC Interconnect	Inject with error capture	Yes	35
DMSS_HSM_PSILSS_L2P_INTAGGR_EVT_EDC_CTRL_0	35	EDC Interconnect	Inject with error capture	Yes	2
DMSS_HSM_PSILSS_L2P_INTAGGR_CEVT_EDC_CTRL_0	36	EDC Interconnect	Inject with error capture	Yes	2
DMSS_HSM_PSILSS_L2P_INTAGGR_MEVT_IN_EDC_CTRL_0	37	EDC Interconnect	Inject with error capture	Yes	2
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	38	EDC Interconnect	Inject with error capture	Yes	35
DMSS_HSM_PSILSS_CFG_EDC_CTRL_0	39	EDC Interconnect	Inject with error capture	Yes	3
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	40	EDC Interconnect	Inject with error capture	Yes	106
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	41	EDC Interconnect	Inject with error capture	Yes	103
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	42	EDC Interconnect	Inject with error capture	Yes	131
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	43	EDC Interconnect	Inject with error capture	Yes	10
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	44	EDC Interconnect	Inject with error capture	Yes	7
DMSS_HSM_CFG_CBASS_SCR_EDC_CTRL_0	45	EDC Interconnect	Inject with error capture	Yes	91
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	46	EDC Interconnect	Inject with error capture	Yes	62

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_ECCAGGR_EDC_CTRL	0	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_ECCAGGR_EDC_CTRL	1	32	EDC
DMSS_HSM_ECCAGGR_EDC_CTRL	2	1	Parity
DMSS_HSM_ECCAGGR_EDC_CTRL	3	10	Parity
DMSS_HSM_ECCAGGR_EDC_CTRL	4	4	Parity
DMSS_HSM_ECCAGGR_EDC_CTRL	5	3	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	0	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	1	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	2	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	3	18	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	4	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	5	6	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	6	6	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	7	18	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	8	18	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	9	18	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	10	18	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	11	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	12	16	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	13	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	14	16	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	15	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	16	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	17	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	18	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	19	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	20	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	21	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	22	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	23	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	24	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	25	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	26	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	27	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	28	10	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	29	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	30	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	31	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	32	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	33	5	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	34	3	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	35	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	36	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	37	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	38	12	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	39	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PKTDMA_EDC_CTRL_0	40	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	41	14	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	42	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	43	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	44	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	45	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	46	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	47	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	48	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	49	4	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	50	32	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	51	10	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	52	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	53	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	54	3	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	55	4	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	56	2	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	57	3	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	58	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	59	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	60	8	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	61	32	EDC
DMSS_HSM_PKTDMA_EDC_CTRL_0	62	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	63	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	64	10	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	65	10	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	66	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	67	1	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	68	3	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	69	4	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	70	2	Parity
DMSS_HSM_PKTDMA_EDC_CTRL_0	71	3	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	0	3	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	1	2	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	2	58	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	3	58	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	4	64	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	5	50	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	6	58	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	7	58	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	8	56	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	9	58	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	10	64	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	11	50	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	12	64	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_INTAGGR_EDC_CTRL_0	13	12	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	14	2	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	15	2	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	16	8	Parity
DMSS_HSM_INTAGGR_EDC_CTRL_0	17	28	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	2	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	3	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	4	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	5	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	6	48	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	7	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	8	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	9	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	10	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	11	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	12	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	13	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	14	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	15	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	16	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	17	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	18	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	19	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	20	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	21	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	24	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	25	3	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	26	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	27	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	28	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	29	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	30	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	31	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	32	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	33	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	34	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	35	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	36	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	37	32	EDC
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	38	32	EDC
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	39	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	40	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	41	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	42	22	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	44	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	45	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	46	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	47	1	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	48	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	49	3	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	50	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	51	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	52	32	EDC
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	53	32	EDC
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	54	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	55	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	56	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	57	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	58	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	59	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	60	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	61	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	62	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	63	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	64	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	65	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	66	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	67	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	68	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	69	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	70	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	71	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	72	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	73	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	74	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	75	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	76	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	77	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	78	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	79	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	80	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	81	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	82	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	83	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	84	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	85	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	86	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	87	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	88	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	89	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	90	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	91	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	92	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	93	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	94	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	95	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	96	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	97	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	98	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	99	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	100	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	101	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	102	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	103	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	104	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	105	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	106	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	107	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	108	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	109	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	110	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	111	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	112	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	113	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	114	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	115	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	116	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	117	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	118	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	119	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	120	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	121	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	122	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	123	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	124	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	125	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	126	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	127	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	128	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	129	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	130	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	131	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	132	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	133	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	134	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	135	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	136	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	137	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	138	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	139	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	140	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	141	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	142	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	143	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	144	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	145	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	146	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	147	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	148	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	149	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	150	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	151	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	152	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	153	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	154	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	155	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	156	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	157	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	158	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	159	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	160	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	161	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	162	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	163	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	164	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	165	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	166	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	167	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	168	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	169	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	170	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	171	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	172	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	173	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	174	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	175	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	176	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	177	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	178	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	179	9	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	180	32	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	181	2	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	182	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	183	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	184	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	185	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	186	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	187	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	188	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	189	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	190	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	191	8	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	192	1	Redundant
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	193	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	194	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	195	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	196	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	197	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	198	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	199	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	200	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	201	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	202	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	203	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	204	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	205	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	206	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	207	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	208	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	209	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	210	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	211	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	212	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	213	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	214	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	215	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	216	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	217	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	218	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	219	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	220	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	221	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	222	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	223	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	224	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	225	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	226	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	227	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	228	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	229	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	230	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	231	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	232	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	233	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	234	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	235	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	236	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	237	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	238	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	239	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	240	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	241	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	242	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	243	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	244	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	245	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	246	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	247	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	248	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	249	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	250	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	251	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	252	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	253	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	254	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_0	255	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	0	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	1	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	2	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	3	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	4	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	5	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	6	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	7	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	8	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	9	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	10	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	11	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	12	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	13	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	14	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	15	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	16	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	17	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	18	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	19	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	20	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	21	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	22	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	23	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	24	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	25	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	26	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	27	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	28	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	29	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	30	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	31	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	32	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	33	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	34	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	35	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	36	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	37	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	38	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	39	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	40	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	41	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	42	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	43	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	44	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	45	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	46	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	47	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	48	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	49	10	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	50	4	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	51	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	52	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	53	16	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	54	16	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	55	12	Parity
DMSS_HSM_IPCSS_RINGACC_EDC_CTRL_1	56	8	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	1	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	2	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	3	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	4	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	5	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	6	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	7	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	8	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	9	2	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	11	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	12	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	13	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	14	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	15	14	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	16	14	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	17	8	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	18	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	19	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	20	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	21	4	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	22	48	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	23	48	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	24	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	25	4	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	26	48	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	27	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	28	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	29	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	30	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	31	12	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	32	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	33	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	34	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	35	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	36	8	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	37	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	38	3	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	39	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	40	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	41	48	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	42	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	43	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	44	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	45	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	46	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	47	4	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	48	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	49	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	50	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	51	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	52	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	53	4	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	54	3	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	55	32	EDC
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	56	32	EDC
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	57	1	Redundant
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	58	10	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	59	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	60	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	61	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	62	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	63	1	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	64	16	Parity
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	65	32	EDC
DMSS_HSM_IPCSS_SEC_PROXY_EDC_CTRL_0	66	32	EDC
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	0	24	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	1	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	2	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	3	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	4	23	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	5	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	6	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	7	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	8	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	9	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	10	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	11	9	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	12	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	13	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	14	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	15	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	16	23	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	17	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	18	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	19	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	20	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	21	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	22	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	23	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	24	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	25	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	26	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	27	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	28	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	29	9	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	30	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	31	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	32	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	33	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	34	4	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	35	9	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	36	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	37	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	38	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	39	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	40	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	41	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	43	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	44	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	45	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	46	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	47	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	48	10	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	49	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	50	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	51	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	52	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	53	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	54	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	55	2	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	56	2	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	57	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	58	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	59	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	60	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	61	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	62	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	63	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	64	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	65	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	66	8	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	67	2	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	68	2	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	69	3	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	70	1	Parity
DMSS_HSM_IPCSS_CBASS_IPCSS_VBM_DST_M2M_BRIDGE_SRC_EDC_CTRL_0	71	8	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	1	23	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	2	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	3	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	4	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	5	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	6	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	7	2	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	8	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	9	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	12	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	13	23	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	14	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	15	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	16	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	17	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	18	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	19	2	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	20	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	21	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	24	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	25	23	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	26	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	27	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	28	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	29	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	30	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	31	2	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	33	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	34	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	35	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	36	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	37	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	38	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	39	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	40	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	41	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	42	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	43	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	44	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	45	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	46	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	47	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	48	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	49	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	50	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	51	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	52	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	53	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	54	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	55	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	56	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	57	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	58	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	59	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	60	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	61	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	62	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	63	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	64	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	65	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	66	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	67	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	68	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	69	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	70	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	71	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	72	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	73	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	74	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	75	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	76	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	77	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	78	12	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	79	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	80	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	81	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	82	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	83	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	84	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	85	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	86	7	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	87	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	88	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	89	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	90	7	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	91	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	92	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	93	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	94	7	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	95	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	96	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	97	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	98	19	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	99	26	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	100	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	101	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	102	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	103	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	104	26	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	105	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	106	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	107	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	108	19	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	109	26	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	110	5	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	111	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	112	3	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	113	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	114	26	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	115	6	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	116	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	117	4	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	118	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	119	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	120	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	121	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	122	1	Redundant
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	123	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	124	32	EDC
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	125	32	EDC
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	126	1	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	127	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	128	23	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	129	10	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	130	2	Parity
DMSS_HSM_IPCSS_CBASS_SCR_SCR_EDC_CTRL_0	131	3	Parity
DMSS_HSM_IPCSS_CBASS_VD2GCLK_EDC_CTRL_0	0	48	Parity
DMSS_HSM_IPCSS_CBASS_VD2GCLK_EDC_CTRL_0	1	48	Parity
DMSS_HSM_IPCSS_CBASS_VD2GCLK_EDC_CTRL_0	2	48	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	1	12	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	2	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	3	10	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	4	12	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	5	2	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	6	3	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	7	12	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	8	4	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	9	3	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	10	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	11	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	12	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	13	1	Redundant
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	14	10	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	15	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	16	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	17	4	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	18	8	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	19	1	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	20	32	EDC
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	21	32	EDC
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	24	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	25	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	26	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	27	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	28	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	29	38	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	30	54	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	31	46	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	32	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	33	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	34	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	35	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	36	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	37	64	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	38	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	39	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	40	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	41	34	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	42	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	43	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	44	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	45	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	46	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	47	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	48	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	49	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	50	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	51	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	52	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	53	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	54	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	55	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	56	34	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	57	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	58	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	59	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	60	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	61	64	Parity
DMSS_HSM_IPCSS_MSRAM_EDC_CTRL_0	62	36	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	1	16	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	3	8	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	4	5	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	7	4	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	8	32	EDC
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	9	32	EDC
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	10	32	EDC
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	11	32	EDC
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	16	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	17	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	18	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	19	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	20	3	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	22	8	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	23	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	24	12	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	25	1	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	26	1	Redundant
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	27	1	Redundant
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	28	1	Redundant
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	29	1	Redundant
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	30	16	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	31	16	Parity
DMSS_HSM_PSILSS_SAUL0_PSIL_SAFEG_EDC_CTRL_0	32	8	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	1	16	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	3	8	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	4	5	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	7	4	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	8	32	EDC
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	9	32	EDC
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	10	32	EDC
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	11	32	EDC
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	16	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	17	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	18	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	19	4	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	20	3	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	22	8	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	23	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	24	12	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	25	1	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	26	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	27	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	28	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	29	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	30	16	Parity
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	31	16	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PKTDMA_STRM_SAFEG_EDC_CTRL_0	32	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	1	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	3	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	4	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	6	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	8	32	EDC
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	10	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	16	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	17	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	18	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	19	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	20	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	21	12	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	22	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	24	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	25	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	26	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	27	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	28	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_SAFEG_EDC_CTRL_0	29	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	1	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	3	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	4	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	6	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	8	32	EDC
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	10	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	13	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	16	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	17	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	18	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	19	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	20	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	21	12	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	22	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	24	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	25	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	26	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	27	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	28	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_SAFEG_EDC_CTRL_0	29	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	3	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	8	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	9	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	12	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	13	7	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	14	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	15	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	16	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	17	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	18	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	20	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	21	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	22	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	23	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	24	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	25	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	26	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	27	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	28	7	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	29	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	30	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	31	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	32	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	33	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	34	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	35	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	36	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	37	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	38	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	39	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	40	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	41	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	44	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	45	16	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	46	1	Redundant
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	47	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	48	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	49	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	50	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	51	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	52	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	53	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	54	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	55	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	56	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	57	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	58	7	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	59	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	60	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	61	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	62	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	63	7	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	64	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	65	7	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	66	5	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	67	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	68	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	69	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	70	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	71	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	72	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	73	2	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	74	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	75	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	76	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	77	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	78	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	79	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	80	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	81	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	82	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	83	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	84	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	85	4	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	86	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	87	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	88	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	89	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	90	8	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	91	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	92	2	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	93	1	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	94	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	95	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	96	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	97	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	98	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	99	3	Parity
DMSS_HSM_PSILSS_PKTDMA_CFGSTRM_BRIDGE_EDC_CTRL_0	100	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	3	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	8	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	9	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	12	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	13	7	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	14	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	15	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	16	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	17	2	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	18	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	20	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	21	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	22	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	23	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	24	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	25	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	26	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	27	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	28	7	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	29	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	30	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	31	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	32	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	33	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	34	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	35	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	36	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	37	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	38	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	39	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	40	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	41	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	44	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	45	16	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	46	1	Redundant
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	47	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	48	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	49	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	50	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	51	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	52	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	53	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	54	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	55	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	56	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	57	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	58	7	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	59	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	60	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	61	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	62	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	63	7	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	64	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	65	7	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	66	5	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	67	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	68	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	69	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	70	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	71	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	72	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	73	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	74	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	75	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	76	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	77	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	78	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	79	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	80	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	81	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	82	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	83	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	84	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	85	4	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	86	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	87	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	88	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	89	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	90	8	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	91	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	92	2	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	93	1	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	94	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	95	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	96	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	97	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	98	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	99	3	Parity
DMSS_HSM_PSILSS_PSILCFG_CFGSTRM_BRIDGE_EDC_CTRL_0	100	3	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	0	16	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	1	8	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	3	5	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	6	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	8	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	10	10	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	13	4	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	16	32	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	17	32	EDC
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	18	48	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	20	2	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	22	2	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	23	5	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	24	4	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	25	12	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	26	16	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	28	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	29	12	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	30	1	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	31	3	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	32	16	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	33	16	Parity
DMSS_HSM_PSILSS_L2P_SAUL0_PSIL_EDC_CTRL_0	34	8	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	0	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	1	8	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	3	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	8	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	10	10	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	13	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	16	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	17	32	EDC
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	18	48	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	20	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	22	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	23	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	24	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	25	12	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	26	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	28	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	29	12	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	30	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	31	3	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	32	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	33	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_STRM_EDC_CTRL_0	34	8	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	0	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	1	8	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	3	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	8	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	10	10	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	13	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	16	32	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	17	32	EDC
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	18	48	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	20	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	22	2	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	23	5	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	24	4	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	25	12	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	26	16	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	28	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	29	12	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	30	1	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	31	3	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	32	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	33	16	Parity
DMSS_HSM_PSILSS_L2P_PKTDMA_CFGSTRM_EDC_CTRL_0	34	8	Parity
DMSS_HSM_PSILSS_L2P_INTAGGR_EVT_EDC_CTRL_0	0	32	Parity
DMSS_HSM_PSILSS_L2P_INTAGGR_EVT_EDC_CTRL_0	1	32	EDC
DMSS_HSM_PSILSS_L2P_INTAGGR_CEVT_EDC_CTRL_0	0	32	Parity
DMSS_HSM_PSILSS_L2P_INTAGGR_CEVT_EDC_CTRL_0	1	32	EDC
DMSS_HSM_PSILSS_L2P_INTAGGR_MEVT_IN_EDC_CTRL_0	0	32	Parity
DMSS_HSM_PSILSS_L2P_INTAGGR_MEVT_IN_EDC_CTRL_0	1	32	EDC
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	0	16	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	1	8	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	2	16	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	3	5	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	4	1	Redundant
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	5	4	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	6	5	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	7	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	8	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	9	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	10	10	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	11	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	12	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	13	4	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	14	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	15	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	16	32	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	17	32	EDC
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	18	48	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	20	2	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	21	2	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	22	2	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	23	5	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	24	4	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	25	12	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	26	16	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	28	1	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	29	12	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	30	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	31	3	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	32	16	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	33	16	Parity
DMSS_HSM_PSILSS_L2P_PSILCFG_CFGSTRM_EDC_CTRL_0	34	8	Parity
DMSS_HSM_PSILSS_CFG_EDC_CTRL_0	0	12	Parity
DMSS_HSM_PSILSS_CFG_EDC_CTRL_0	1	12	Parity
DMSS_HSM_PSILSS_CFG_EDC_CTRL_0	2	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	1	48	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	2	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	3	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	4	3	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	6	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	7	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	8	2	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	9	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	12	48	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	14	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	15	3	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	16	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	17	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	18	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	20	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	21	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	23	48	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	24	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	25	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	26	3	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	28	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	29	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	30	2	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	31	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	33	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	34	48	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	35	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	36	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	37	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	38	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	39	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	40	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	41	2	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	44	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	45	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	46	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	47	12	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	48	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	49	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	50	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	51	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	52	12	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	53	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	54	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	55	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	56	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	57	12	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	58	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	59	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	60	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	61	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	62	12	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	63	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	64	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	65	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	66	10	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	67	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	68	12	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	69	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	70	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	71	4	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	72	9	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	73	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	74	6	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	75	9	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	76	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	77	6	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	78	9	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	79	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	80	6	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	81	9	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	82	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	83	6	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	84	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	85	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	86	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	87	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	88	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	89	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	90	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	91	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	92	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	93	5	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	94	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	95	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	96	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	97	32	EDC
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	98	32	EDC
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	99	32	EDC
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	100	32	EDC
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	101	1	Redundant
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	102	1	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	103	10	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	104	10	Parity
DMSS_HSM_PSILSS_CBASS_DATA_SCR1_SCR_EDC_CTRL_0	105	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	1	32	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	2	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	3	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	4	3	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	6	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	7	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	8	2	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	9	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	12	32	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	14	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	15	3	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	16	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	17	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	18	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	20	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	21	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	23	32	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	24	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	25	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	26	3	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	28	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	29	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	30	2	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	31	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	33	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	34	32	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	35	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	36	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	37	3	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	38	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	39	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	40	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	41	2	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	44	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	45	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	46	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	47	12	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	48	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	49	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	50	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	51	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	52	12	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	53	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	54	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	55	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	56	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	57	12	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	58	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	59	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	60	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	61	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	62	12	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	63	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	64	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	65	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	66	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	67	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	68	12	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	69	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	70	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	71	4	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	72	9	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	73	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	74	6	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	75	9	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	76	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	77	6	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	78	9	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	79	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	80	6	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	81	9	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	82	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	83	6	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	84	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	85	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	86	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	87	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	88	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	89	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	90	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	91	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	92	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	93	5	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	94	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	95	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	96	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	97	32	EDC
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	98	1	Redundant
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	99	1	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	100	10	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	101	10	Parity
DMSS_HSM_PSILSS_CBASS_RESP_SCR2_SCR_EDC_CTRL_0	102	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	1	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	2	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	3	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	4	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	5	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	6	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	7	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	8	2	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	9	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	12	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	13	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	14	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	15	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	16	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	17	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	18	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	19	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	20	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	21	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	23	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	24	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	25	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	26	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	27	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	28	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	29	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	30	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	31	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	33	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	34	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	35	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	36	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	37	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	38	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	39	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	40	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	41	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	44	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	45	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	46	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	47	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	48	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	49	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	50	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	51	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	52	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	53	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	54	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	55	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	56	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	57	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	58	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	59	3	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	60	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	61	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	62	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	63	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	64	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	65	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	66	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	67	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	68	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	69	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	70	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	71	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	72	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	73	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	74	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	75	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	76	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	77	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	78	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	79	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	80	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	81	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	82	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	83	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	84	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	85	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	86	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	87	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	88	10	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	89	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	90	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	91	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	92	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	93	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	94	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	95	5	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	96	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	97	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	98	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	99	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	100	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	101	5	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	102	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	103	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	104	5	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	105	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	106	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	107	5	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	108	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	109	9	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	110	5	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	111	6	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	112	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	113	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	114	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	115	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	116	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	117	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	118	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	119	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	120	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	121	7	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	122	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	123	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	124	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	125	32	EDC
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	126	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	127	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	128	10	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	129	10	Parity
DMSS_HSM_PSILSS_CBASS_ETL_SCR3_SCR_EDC_CTRL_0	130	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	2	32	EDC
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	3	1	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	4	32	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	5	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	6	2	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	7	12	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	8	4	Parity
DMSS_HSM_PSILSS_CBASS_ETL_VD2GCLK_EDC_CTRL_0	9	12	Parity
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	0	1	Parity
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	2	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	3	1	Redundant
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	4	8	Parity
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	5	8	Parity
DMSS_HSM_CFG_CBASS_DMSS_CFG_P2P_BRIDGE_EDC_CTRL_0	6	8	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	0	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	1	23	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	2	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	3	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	4	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	5	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	6	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	7	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	8	2	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	9	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	10	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	11	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	12	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	13	10	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	14	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	15	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	16	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	17	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	18	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	19	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	20	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	21	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	22	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	23	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	24	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	25	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	26	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	27	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	28	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	29	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	30	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	31	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	33	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	34	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	35	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	36	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	37	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	38	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	39	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	40	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	41	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	42	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	44	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	45	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	46	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	47	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	48	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	49	10	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	50	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	51	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	52	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	53	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	54	4	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	55	12	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	56	8	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	57	9	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	58	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	59	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	60	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	61	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	62	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	63	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	64	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	65	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	66	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	67	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	68	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	69	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	70	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	71	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	72	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	73	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	74	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	75	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	76	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	77	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	78	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	79	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	80	26	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	81	3	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	82	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	83	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	84	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	85	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	86	1	Redundant
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	87	1	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	88	10	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	89	10	Parity
DMSS_HSM_CFG_CBASS_SCR_SCR_EDC_CTRL_0	90	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	0	48	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	1	1	Redundant
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	2	32	EDC
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	3	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	4	17	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	5	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	6	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	7	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	8	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	9	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	10	10	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	11	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	12	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	13	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	14	2	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	15	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	16	1	Redundant
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	17	32	EDC
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	18	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	19	22	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	20	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	21	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	22	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	23	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	24	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	25	10	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	26	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	27	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	28	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	29	2	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	30	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	31	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	32	1	Redundant
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	33	32	EDC
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	34	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	35	17	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	36	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	37	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	38	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	39	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-441. EDC checkers information for ECC Aggregator Instance SA3_SS0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	40	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	41	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	42	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	43	1	Redundant
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	44	32	EDC
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	45	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	46	9	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	47	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	48	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	49	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	50	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	51	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	52	1	Redundant
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	53	32	EDC
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	54	1	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	55	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	56	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	57	3	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	58	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	59	4	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	60	12	Parity
DMSS_HSM_CFG_CBASS_VD2GCLK_EDC_CTRL_0	61	1	Parity

Table 12-442. Properties of ECC Aggregator Instance USB0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
USB2SS_16FFC_USB2SS_CORE_AXI2VBUSM_MST_KSBUS_AXI2VBUSM_RDATA_BUFFER	0	ECC Wrapper	Inject with error capture	Yes	66	4 KB
USB2SS_16FFC_USB2SS_CORE_RAMS_MEM_CTRL_RAM0	1	ECC Wrapper	Inject with error capture	Yes	64	56 KB

Table 12-443. Properties of ECC Aggregator Instance USB1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
USB3P0SS64_16FFC_USB3P0SS64_CORE_USB3P0_KSBUS_AXI2VBUSM_KSBUS_AXI2VBUSM_RDATA_BUFFER	0	ECC Wrapper	Inject with error capture	Yes	66	2 KB

Table 12-444. Properties of ECC Aggregator Instance WKUP_PSRAMECC_8K0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
PSRAM8KX32E_PSRAM0_ECC	0	ECC Wrapper	Inject with error capture	No	32	32 KB

Copyright © 2026 Texas Instruments Incorporated

Table 12-445. Properties of ECC Aggregator Instance WKUP_R5FSS0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
CPU0_ITAG_RAM0	0	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM1	1	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM2	2	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_ITAG_RAM3	3	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_IDATA_BANK0	4	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK1	5	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK2	6	ECC Wrapper	Inject only	No	72	9 KB
CPU0_IDATA_BANK3	7	ECC Wrapper	Inject only	No	72	9 KB
CPU0_DTAG_RAM0	8	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM1	9	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM2	10	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DTAG_RAM3	11	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDIRTY_RAM	12	ECC Wrapper	Inject only	Yes	28	896 B
CPU0_DDATA_RAM0	13	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM1	14	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM2	15	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM3	16	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM4	17	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM5	18	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM6	19	ECC Wrapper	Inject only	No	39	5 KB
CPU0_DDATA_RAM7	20	ECC Wrapper	Inject only	No	39	5 KB
PULSAR_UL_ATCM0_BANK0	21	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_UL_ATCM0_BANK1	22	ECC Wrapper	Inject only	No	39	20 KB
PULSAR_UL_B0TCM0_BANK0	23	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B0TCM0_BANK1	24	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B1TCM0_BANK0	25	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_B1TCM0_BANK1	26	ECC Wrapper	Inject only	No	39	10 KB
PULSAR_UL_PULSAR_KS_VIM_COMMON_CORE0_RAM	27	ECC Wrapper	Inject with error capture	Yes	30	960 B
PULSAR_UL_MEM_MST0_KSBUS_AXI2VBUSM_RDATA_BUFFER	28	ECC Wrapper	Inject with error capture	Yes	66	528 B

Table 12-446. Properties of ECC Aggregator Instance WKUP_VTM0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	0	EDC Interconnect	Inject with error capture	Yes	6
K3VTM_N16FFC_MMR_EDC_CTRL_0	1	EDC Interconnect	Inject with error capture	Yes	171
K3VTM_N16FFC_CFG_CBASS_VBUSP_P2P_BRIDGE_EDC_CTRL_0	2	EDC Interconnect	Inject with error capture	Yes	3
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_EDC_CTRL_0	3	EDC Interconnect	Inject with error capture	Yes	50

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	0	1	Redundant
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	1	32	EDC
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	2	1	Parity
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	3	10	Parity
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	4	4	Parity
K3VTM_N16FFC_ECCAGGR_EDC_CTRL	5	3	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	0	1	Redundant
K3VTM_N16FFC_MMR_EDC_CTRL_0	1	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	2	4	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	3	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	4	4	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	5	32	EDC
K3VTM_N16FFC_MMR_EDC_CTRL_0	6	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	7	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	8	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	9	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	10	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	11	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	12	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	13	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	14	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	15	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	16	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	17	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	18	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	19	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	20	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	21	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	22	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	23	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	24	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	25	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	26	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	27	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	28	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	29	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	30	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	31	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	32	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	33	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	34	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	35	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	36	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	37	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	38	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_MMR_EDC_CTRL_0	39	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	40	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	41	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	42	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	43	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	44	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	45	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	46	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	47	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	48	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	49	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	50	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	51	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	52	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	53	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	54	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	55	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	56	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	57	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	58	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	59	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	60	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	61	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	62	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	63	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	64	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	65	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	66	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	67	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	68	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	69	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	70	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	71	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	72	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	73	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	74	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	75	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	76	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	77	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	78	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	79	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	80	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	81	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	82	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	83	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_MMR_EDC_CTRL_0	84	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	85	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	86	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	87	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	88	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	89	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	90	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	91	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	92	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	93	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	94	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	95	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	96	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	97	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	98	10	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	99	16	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	100	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	101	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	102	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	103	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	104	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	105	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	106	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	107	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	108	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	109	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	110	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	111	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	112	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	113	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	114	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	115	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	116	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	117	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	118	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	119	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	120	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	121	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	122	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	123	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	124	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	125	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	126	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	127	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	128	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_MMR_EDC_CTRL_0	129	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	130	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	131	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	132	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	133	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	134	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	135	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	136	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	137	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	138	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	139	5	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	140	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	141	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	142	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	143	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	144	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	145	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	146	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	147	6	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	148	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	149	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	150	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	151	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	152	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	153	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	154	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	155	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	156	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	157	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	158	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	159	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	160	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	161	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	162	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	163	1	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	164	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	165	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	166	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	167	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	168	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	169	8	Parity
K3VTM_N16FFC_MMR_EDC_CTRL_0	170	8	Parity
K3VTM_N16FFC_CFG_CBASS_VBUSP_P2P_BRIDGE_EDC_CTRL_0	0	1	Redundant
K3VTM_N16FFC_CFG_CBASS_VBUSP_P2P_BRIDGE_EDC_CTRL_0	1	1	Redundant
K3VTM_N16FFC_CFG_CBASS_VBUSP_P2P_BRIDGE_EDC_CTRL_0	2	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	0	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	1	10	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	2	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	3	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	4	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	5	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	6	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	7	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	8	2	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	9	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	10	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	11	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	12	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	13	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	14	12	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	15	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	16	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	17	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	18	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	19	12	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	20	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	21	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	22	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	23	10	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	24	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	25	12	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	26	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	27	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	28	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	29	7	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	30	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	31	4	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	32	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	33	26	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	34	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	35	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	36	26	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	37	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	38	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	39	26	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	40	3	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	41	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	42	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	43	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	44	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-447. EDC checkers information for ECC Aggregator Instance WKUP_VTM0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	45	1	Redundant
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	46	1	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	47	10	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	48	10	Parity
K3VTM_N16FFC_CFG_CBASS_CFG_SCR_SCR_EDC_CTRL_0	49	1	Parity

12.8.6 Interconnect ECC Aggregators

Table 12-448. Properties of ECC Aggregator Instance ECC_AGGR0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
IMAILBOX8_MAIN_0_RAMECC	0	ECC Wrapper	Inject with error capture	Yes	32	2 KB

Table 12-449. Properties of ECC Aggregator Instance ECC_AGGR0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	1	EDC Interconnect	Inject with error capture	Yes	13
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	13
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	3	EDC Interconnect	Inject with error capture	Yes	62
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	4	EDC Interconnect	Inject with error capture	Yes	26
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	5	EDC Interconnect	Inject with error capture	Yes	15
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_DST_BUSECC	6	EDC Interconnect	Inject with error capture	Yes	4
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	7	EDC Interconnect	Inject with error capture	Yes	15
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	8	EDC Interconnect	Inject with error capture	Yes	15
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_DST_BUSECC	9	EDC Interconnect	Inject with error capture	Yes	4
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	10	EDC Interconnect	Inject with error capture	Yes	15
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_DST_BUSECC	11	EDC Interconnect	Inject with error capture	Yes	4

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	0	1	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	1	32	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	2	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	3	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	4	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	5	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	6	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	7	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	8	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	9	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	10	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	11	1	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_HSM_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_HSM_VBUSP_S_BRIDGE_BUSECC	12	1	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	0	1	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	1	32	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	2	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	3	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	4	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	5	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	6	1	Redundant
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	7	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	8	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	9	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	10	8	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	11	1	Parity
AM67_MAIN_CENTRAL_CBASS_ISMS_MAIN_0_TIFS_VBUSP_S_P2P_BRIDGE_ISMS_MAIN_0_TIFS_VBUSP_S_BRIDGE_BUSECC	12	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	0	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	1	32	EDC
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	2	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	3	48	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	4	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	5	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	6	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	7	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	8	10	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	9	5	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	10	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	11	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	12	2	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	13	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	14	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	15	2	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	16	8	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	17	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	18	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	19	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	20	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	21	32	EDC
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	22	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	23	48	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	24	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	25	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	26	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	27	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	28	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	29	5	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	30	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	31	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	32	2	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	33	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	34	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	35	2	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	36	8	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	37	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	38	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	39	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	40	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	41	32	EDC
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	42	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	43	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	44	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	45	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	46	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	47	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	48	10	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	49	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	50	1	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	51	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	52	32	EDC
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	53	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	54	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	55	1	Redundant
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	56	4	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	57	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	58	12	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	59	10	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	60	3	Parity
AM67_MAIN_CENTRAL_CBASS_HSM_CLK_1_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_1_BUSECC	61	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	0	1	Redundant
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	1	1	Redundant
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	2	1	Redundant
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	3	12	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	4	4	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	5	12	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	6	10	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	7	10	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	8	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	9	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	10	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	11	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	12	4	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	13	3	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	14	3	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	15	1	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	16	32	EDC
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	17	32	EDC
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	18	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	19	1	Redundant
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	20	12	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	21	10	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	22	3	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	23	12	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	24	4	Parity
AM67_MAIN_IPCSS_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	25	1	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	1	1	Redundant
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	2	24	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	3	3	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	4	3	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	5	1	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	6	1	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	7	2	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	8	3	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	9	1	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	10	10	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	11	5	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	12	3	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	13	4	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_SRC_BUSECC	14	4	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_DST_BUSECC	2	24	Parity
AM67_MAIN_FW_CBASS_ISMS_MAIN_0_FWMGR_CFG_P2P_BRIDGE_ISMS_MAIN_0_FWMGR_CFG_BRIDGE_DST_BUSECC	3	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	0	1	Redundant
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	1	32	EDC
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	2	1	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	3	24	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	4	4	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	5	3	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	6	1	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	7	1	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	8	10	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	9	2	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	10	3	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	11	1	Parity
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	12	1	Redundant
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	13	32	EDC
AM67_MAIN_FW_CBASS_HSM_CLK_2_CLK_EDC_CTRL_CBASS_INT_HSM_CLK_2_BUSECC	14	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	0	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	1	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	2	23	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	3	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	4	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	5	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	6	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	7	2	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	8	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	9	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	10	10	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	11	5	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	12	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	13	4	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_SRC_BUSECC	14	2	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_DST_BUSECC	0	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_DST_BUSECC	1	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_DST_BUSECC	2	23	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_DMSS_CFG_DST_BUSECC	3	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	0	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-450. EDC checkers information for ECC Aggregator Instance ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	1	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	2	10	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	3	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	4	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	5	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	6	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	7	2	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	8	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	9	1	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	10	10	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	11	5	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	12	3	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	13	4	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_SRC_BUSECC	14	2	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_DST_BUSECC	0	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_DST_BUSECC	1	1	Redundant
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_DST_BUSECC	2	10	Parity
ISAM62A_SEC_BR_MAIN_0_IP2P_SA3_PKTDMA_CRED_DST_BUSECC	3	3	Parity

Table 12-451. Properties of ECC Aggregator Instance MCU_ECC_AGGR0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_WR_RAMECC	7	ECC Wrapper	Inject with error capture	Yes	76	152 B
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_RD_RAMECC	8	ECC Wrapper	Inject with error capture	Yes	76	38 B

Table 12-452. Properties of ECC Aggregator Instance MCU_ECC_AGGR0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_SLV_SRC_BUSECC	0	EDC Interconnect	Inject with error capture	Yes	65
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	1	EDC Interconnect	Inject with error capture	Yes	15
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_P2P_CPU0_PMST_DST_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	4
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_RMST_DST_BUSECC	3	EDC Interconnect	Inject with error capture	Yes	69
ISAM62A_MCU_PULSAR_UL_BR_ICPU0_M2M_CPU0_WMST_DST_BUSECC	4	EDC Interconnect	Inject with error capture	Yes	69
ISAM62A_MCU_PULSAR_UL_BR_IECC_AGGR_CFG_SRC_BUSECC	5	EDC Interconnect	Inject with error capture	Yes	15
ISAM67_DM2MCU_VBUSM_GASKET_MCU_0_EDC_CTRL	6	EDC Interconnect	Inject with error capture	Yes	1
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	9	EDC Interconnect	Inject with error capture	Yes	81
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	10	EDC Interconnect	Inject with error capture	Yes	17

Copyright © 2026 Texas Instruments Incorporated

Table 12-452. Properties of ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	11	EDC Interconnect	Inject with error capture	Yes	5
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	12	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	13	EDC Interconnect	Inject with error capture	Yes	11
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	14	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	15	EDC Interconnect	Inject with error capture	Yes	14
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	16	EDC Interconnect	Inject with error capture	Yes	13
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	17	EDC Interconnect	Inject with error capture	Yes	13
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	18	EDC Interconnect	Inject with error capture	Yes	13
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	19	EDC Interconnect	Inject with error capture	Yes	13
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	20	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	21	EDC Interconnect	Inject with error capture	Yes	14
AM67_MCU_CBASS_ISAM67_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	22	EDC Interconnect	Inject with error capture	Yes	13
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	23	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	24	EDC Interconnect	Inject with error capture	Yes	7
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	25	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	26	EDC Interconnect	Inject with error capture	Yes	7
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	27	EDC Interconnect	Inject with error capture	Yes	256
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	28	EDC Interconnect	Inject with error capture	Yes	216

Copyright © 2026 Texas Instruments Incorporated

Table 12-452. Properties of ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	29	EDC Interconnect	Inject with error capture	Yes	256
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	30	EDC Interconnect	Inject with error capture	Yes	134
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	31	EDC Interconnect	Inject with error capture	Yes	256
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	32	EDC Interconnect	Inject with error capture	Yes	151
AM67_MCU_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	33	EDC Interconnect	Inject with error capture	Yes	8
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	34	EDC Interconnect	Inject with error capture	Yes	19
AM67_MCU_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	35	EDC Interconnect	Inject with error capture	Yes	42
AM67_MCU_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	36	EDC Interconnect	Inject with error capture	Yes	5
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	37	EDC Interconnect	Inject with error capture	Yes	19
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	38	EDC Interconnect	Inject with error capture	Yes	42
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	39	EDC Interconnect	Inject with error capture	Yes	81
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	40	EDC Interconnect	Inject with error capture	Yes	135
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	41	EDC Interconnect	Inject with error capture	Yes	101
SAM67_MCU_ECC_AGGR_EDC_CTRL	42	EDC Interconnect	Inject with error capture	Yes	6

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	0	24	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	2	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	3	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	4	23	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	5	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	7	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	9	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	12	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	13	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	14	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	15	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	16	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	17	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	19	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	20	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	21	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	22	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	23	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	24	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	25	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	26	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	27	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	28	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	30	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	31	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	32	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	33	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	34	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	35	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	36	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	37	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	38	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	39	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	40	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	41	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	42	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	43	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	44	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	45	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	46	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	47	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	48	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	49	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	50	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	51	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	52	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	53	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	54	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	55	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	56	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	57	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	58	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	59	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	60	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	61	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	62	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	63	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_SLV_SRC_BUSECC	64	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	2	11	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	3	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	4	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	5	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	7	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	11	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	12	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	13	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_CFG_SLV_SRC_BUSECC	14	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_DST_BUSECC	0	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_DST_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_DST_BUSECC	2	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_P2P_CPU0_PMST_DST_BUSECC	3	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	2	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	3	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	4	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	5	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	6	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	7	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	8	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	12	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	13	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	14	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	15	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	16	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	17	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	19	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	20	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	21	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	22	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	23	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	24	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	25	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	26	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	27	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	28	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	30	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	31	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	32	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	33	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	34	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	35	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	36	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	37	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	38	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	39	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	40	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	41	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	42	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	43	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	44	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	45	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	46	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	47	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	48	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	49	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	50	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	51	7	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	52	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	53	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	54	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	55	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	56	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	57	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	58	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	59	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	60	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	61	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	62	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	63	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	64	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	65	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	66	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	67	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_RMST_DST_BUSECC	68	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	1	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	2	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	3	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	4	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	5	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	6	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	7	36	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	8	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	11	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	12	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	13	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	14	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	15	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	16	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	17	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	18	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	19	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	20	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	21	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	22	9	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	23	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	24	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	25	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	26	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	27	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	28	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	29	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	30	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	31	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	32	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	33	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	34	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	35	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	36	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	37	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	38	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	39	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	40	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	41	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	42	2	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	43	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	44	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	45	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	46	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	47	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	48	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	49	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	50	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	51	7	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	52	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	53	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	54	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	55	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	56	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	57	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	58	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	59	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	60	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	61	8	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	62	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	63	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	64	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	65	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	66	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	67	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__ICPU0_M2M_CPU0_WMST_DST_BUSECC	68	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	0	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	1	1	Redundant
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	2	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	3	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	4	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	5	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	6	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	7	2	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	8	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	9	1	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	10	10	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	11	5	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	12	3	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	13	4	Parity
ISAM62A_MCU_PULSAR_UL_BR__IECC_AGGR_CFG_SRC_BUSECC	14	2	Parity
ISAM67_DM2MCU_VBUSM_GASKET_MCU_0_EDC_CTRL	0	48	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	0	1	Redundant
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	1	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	2	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	3	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	4	32	EDC
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	5	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	6	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	7	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	8	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	9	30	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	10	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	11	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	12	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	13	12	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	14	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	15	3	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	16	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	17	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	18	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	19	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	20	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	21	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	22	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	23	12	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	24	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	25	3	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	26	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	27	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	28	3	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	29	3	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	30	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	31	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	32	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	33	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	34	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	35	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	36	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	37	1	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	38	30	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	39	6	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	40	6	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	41	6	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	42	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	43	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	44	5	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	45	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	46	16	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	47	9	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	48	16	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	49	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	50	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	51	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	52	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	53	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	54	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	55	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	56	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	57	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	58	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	59	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	60	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	61	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	62	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	63	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	64	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	65	20	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	66	20	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	67	10	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	68	3	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	69	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	70	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	71	5	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	72	4	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	73	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	74	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	75	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	76	2	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	77	55	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	78	55	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	79	8	Parity
ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_EDC_CTRL	80	8	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	4	36	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	6	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_SRC_BUSECC	16	2	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	3	36	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_P2P_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRP_32B_CLK4_L0_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	0	32	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	1	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	4	36	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	10	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_SRC_BUSECC	16	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	3	1	Redundant
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	4	36	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	5	3	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	6	32	EDC
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	7	32	EDC
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	8	8	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	9	8	Parity
AM67_MCU_CBASS_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_P2M_BRIDGE_BR_SCRP_32B_CLK2_TO_SCRM_64B_CLK1_L0_BRIDGE_DST_BUSECC	10	14	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	0	32	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	4	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	5	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	6	36	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	7	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	8	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	13	4	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_SRC_BUSECC	16	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	3	4	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	5	10	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	6	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	7	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	8	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	9	1	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	10	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	11	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	12	3	Parity
AM67_MCU_CBASS_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_M2P_BRIDGE_BR_SCRM_64B_CLK1_TO_SCRP_32B_CLK2_L0_BRIDGE_DST_BUSECC	13	4	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	1	32	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	3	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	4	1	Redundant
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	5	1	Redundant
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	6	1	Redundant
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	7	8	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	8	8	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	9	8	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	10	8	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_P2P_BRIDGE_IPULSAR_ULS_MCU_0_CPU0_CFG_SLV_BRIDGE_BUSECC	12	1	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	1	32	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	4	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	5	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	6	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	7	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	8	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	9	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	10	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_1_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_1_CFG_BRIDGE_BUSECC	12	1	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	1	32	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	2	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	4	1	Redundant
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	5	1	Redundant
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	6	1	Redundant
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	7	8	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	8	8	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	9	8	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	10	8	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM62A_MCU_PULSAR_UL_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	12	1	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	1	32	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	4	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	5	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	6	1	Redundant
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	7	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	8	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	9	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	10	8	Parity
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	11	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_IMSRAM32KX64E_MCU_0_CFG_P2P_BRIDGE_IMSRAM32KX64E_MCU_0_CFG_BRIDGE_BUSECC	12	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	0	32	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	4	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	5	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	6	36	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	7	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	8	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	13	4	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	14	2	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_SRC_BUSECC	16	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	0	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	3	4	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	5	10	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	6	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	7	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	8	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	9	1	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	10	2	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	11	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	12	3	Parity
AM67_MCU_CBASS_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_MCU_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_DST_BUSECC	13	4	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	1	32	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	4	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	5	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	6	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	7	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	8	8	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	9	8	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	10	8	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_P2P_BRIDGE_ISAM67_MCU_MCU_ECC_AGGR_MCU_0_CFG_BRIDGE_BUSECC	12	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	4	10	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_SRC_BUSECC	16	2	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	3	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	5	1	Parity
AM67_MCU_CBASS_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_P2P_BRIDGE_ISAM67_MCU2DM_VBUSM_GASKET_MCU_1_CFG_BRIDGE_DST_BUSECC	6	1	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	4	10	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_SRC_BUSECC	16	2	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	3	10	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	5	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_ERR_SLV_BRIDGE_DST_BUSECC	6	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	0	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	1	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	2	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	3	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	4	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	5	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	6	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	7	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	8	2	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	9	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	10	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	11	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	12	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	13	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	14	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	15	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	16	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	17	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	18	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	19	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	20	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	21	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	22	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	23	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	24	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	25	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	26	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	27	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	28	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	29	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	30	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	31	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	32	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	33	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	34	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	35	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	36	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	37	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	38	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	39	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	40	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	41	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	42	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	43	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	44	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	45	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	46	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	47	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	48	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	49	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	50	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	51	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	52	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	53	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	54	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	55	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	56	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	57	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	58	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	59	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	60	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	61	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	62	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	63	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	64	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	65	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	66	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	67	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	68	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	69	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	70	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	71	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	72	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	73	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	74	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	75	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	76	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	77	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	78	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	79	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	80	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	81	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	82	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	83	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	84	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	85	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	86	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	87	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	88	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	89	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	90	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	91	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	92	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	93	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	94	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	95	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	96	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	97	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	98	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	99	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	100	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	101	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	102	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	103	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	104	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	105	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	106	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	107	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	108	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	109	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	110	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	111	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	112	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	113	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	114	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	115	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	116	13	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	117	18	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	118	14	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	119	15	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	120	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	121	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	122	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	123	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	124	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	125	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	126	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	127	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	128	11	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	129	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	130	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	131	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	132	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	133	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	134	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	135	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	136	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	137	11	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	138	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	139	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	140	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	141	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	142	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	143	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	144	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	145	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	146	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	147	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	148	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	149	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	150	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	151	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	152	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	153	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	154	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	155	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	156	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	157	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	158	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	159	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	160	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	161	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	162	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	163	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	164	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	165	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	166	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	167	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	168	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	169	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	170	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	171	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	172	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	173	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	174	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	175	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	176	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	177	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	178	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	179	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	180	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	181	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	182	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	183	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	184	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	185	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	186	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	187	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	188	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	189	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	190	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	191	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	192	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	193	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	194	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	195	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	196	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	197	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	198	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	199	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	200	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	201	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	202	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	203	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	204	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	205	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	206	11	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	207	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	208	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	209	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	210	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	211	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	212	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	213	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	214	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	215	11	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	216	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	217	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	218	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	219	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	220	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	221	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	222	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	223	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	224	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	225	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	226	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	227	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	228	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	229	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	230	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	231	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	232	11	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	233	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	234	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	235	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	236	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	237	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	238	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	239	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	240	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	241	11	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	242	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	243	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	244	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	245	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	246	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	247	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	248	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	249	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	250	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	251	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	252	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	253	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	254	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_0	255	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	0	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	1	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	2	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	3	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	4	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	5	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	6	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	7	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	8	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	9	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	10	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	11	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	12	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	13	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	14	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	15	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	16	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	17	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	18	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	19	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	20	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	21	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	22	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	23	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	24	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	25	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	26	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	27	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	28	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	29	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	30	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	31	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	32	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	33	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	34	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	35	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	36	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	37	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	38	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	39	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	40	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	41	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	42	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	43	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	44	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	45	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	46	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	47	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	48	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	49	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	50	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	51	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	52	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	53	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	54	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	55	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	56	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	57	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	58	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	59	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	60	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	61	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	62	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	63	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	64	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	65	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	66	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	67	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	68	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	69	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	70	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	71	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	72	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	73	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	74	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	75	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	76	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	77	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	78	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	79	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	80	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	81	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	82	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	83	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	84	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	85	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	86	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	87	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	88	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	89	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	90	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	91	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	92	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	93	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	94	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	95	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	96	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	97	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	98	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	99	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	100	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	101	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	102	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	103	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	104	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	105	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	106	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	107	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	108	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	109	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	110	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	111	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	112	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	113	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	114	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	115	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	116	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	117	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	118	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	119	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	120	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	121	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	122	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	123	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	124	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	125	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	126	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	127	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	128	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	129	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	130	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	131	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	132	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	133	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	134	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	135	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	136	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	137	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	138	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	139	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	140	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	141	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	142	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	143	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	144	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	145	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	146	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	147	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	148	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	149	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	150	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	151	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	152	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	153	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	154	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	155	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	156	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	157	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	158	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	159	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	160	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	161	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	162	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	163	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	164	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	165	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	166	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	167	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	168	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	169	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	170	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	171	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	172	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	173	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	174	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	175	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	176	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	177	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	178	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	179	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	180	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	181	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	182	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	183	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	184	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	185	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	186	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	187	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	188	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	189	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	190	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	191	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	192	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	193	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	194	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	195	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	196	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	197	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	198	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	199	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	200	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	201	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	202	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	203	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	204	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	205	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	206	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	207	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	208	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	209	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	210	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	211	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	212	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	213	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	214	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_AM67_MCU_CBASS_SCRP_32B_CLK4_SCR_EDC_CTRL_BUSECC_1	215	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	0	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	1	36	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	2	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	3	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	4	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	5	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	6	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	7	2	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	8	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	9	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	10	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	11	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	12	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	13	36	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	14	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	15	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	16	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	17	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	18	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	19	2	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	20	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	21	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	22	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	23	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	24	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	25	36	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	26	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	27	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	28	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	29	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	30	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	31	2	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	32	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	33	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	34	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	35	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	36	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	37	36	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	38	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	39	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	40	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	41	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	42	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	43	2	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	44	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	45	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	46	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	47	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	48	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	49	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	50	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	51	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	52	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	53	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	54	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	55	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	56	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	57	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	58	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	59	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	60	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	61	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	62	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	63	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	64	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	65	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	66	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	67	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	68	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	69	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	70	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	71	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	72	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	73	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	74	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	75	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	76	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	77	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	78	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	79	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	80	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	81	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	82	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	83	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	84	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	85	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	86	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	87	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	88	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	89	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	90	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	91	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	92	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	93	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	94	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	95	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	96	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	97	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	98	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	99	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	100	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	101	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	102	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	103	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	104	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	105	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	106	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	107	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	108	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	109	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	110	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	111	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	112	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	113	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	114	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	115	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	116	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	117	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	118	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	119	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	120	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	121	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	122	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	123	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	124	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	125	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	126	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	127	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	128	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	129	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	130	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	131	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	132	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	133	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	134	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	135	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	136	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	137	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	138	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	139	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	140	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	141	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	142	12	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	143	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	144	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	145	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	146	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	147	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	148	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	149	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	150	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	151	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	152	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	153	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	154	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	155	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	156	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	157	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	158	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	159	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	160	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	161	15	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	162	9	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	163	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	164	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	165	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	166	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	167	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	168	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	169	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	170	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	171	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	172	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	173	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	174	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	175	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	176	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	177	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	178	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	179	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	180	15	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	181	9	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	182	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	183	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	184	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	185	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	186	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	187	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	188	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	189	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	190	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	191	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	192	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	193	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	194	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	195	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	196	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	197	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	198	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	199	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	200	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	201	39	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	202	7	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	203	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	204	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	205	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	206	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	207	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	208	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	209	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	210	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	211	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	212	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	213	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	214	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	215	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	216	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	217	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	218	11	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	219	9	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	220	44	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	221	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	222	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	223	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	224	19	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	225	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	226	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	227	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	228	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	229	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	230	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	231	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	232	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	233	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	234	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	235	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	236	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	237	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	238	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	239	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	240	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	241	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	242	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	243	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	244	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	245	29	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	246	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	247	7	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	248	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	249	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	250	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	251	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	252	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	253	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	254	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_0	255	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	0	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	1	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	2	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	3	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	4	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	5	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	6	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	7	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	8	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	9	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	10	29	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	11	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	12	7	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	13	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	14	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	15	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	16	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	17	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	18	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	19	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	20	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	21	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	22	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	23	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	24	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	25	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	26	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	27	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	28	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	29	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	30	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	31	29	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	32	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	33	7	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	34	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	35	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	36	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	37	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	38	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	39	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	40	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	41	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	42	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	43	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	44	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	45	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	46	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	47	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	48	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	49	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	50	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	51	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	52	24	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	53	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	54	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	55	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	56	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	57	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	58	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	59	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	60	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	61	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	62	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	63	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	64	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	65	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	66	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	67	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	68	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	69	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	70	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	71	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	72	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	73	24	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	74	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	75	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	76	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	77	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	78	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	79	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	80	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	81	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	82	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	83	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	84	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	85	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	86	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	87	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	88	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	89	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	90	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	91	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	92	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	93	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	94	24	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	95	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	96	6	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	97	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	98	4	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	99	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	100	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	101	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	102	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	103	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	104	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	105	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	106	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	107	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	108	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	109	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	110	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	111	8	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	112	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	113	3	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	114	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	115	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	116	26	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	117	7	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	118	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	119	5	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	120	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	121	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	122	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	123	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	124	1	Redundant
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	125	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	126	32	EDC
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	127	32	EDC
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	128	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	129	1	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	130	36	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	131	10	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	132	2	Parity
AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_AM67_MCU_CBASS_SCRM_64B_CLK1_SCR_EDC_CTRL_BUSECC_1	133	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	0	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	1	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	2	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	3	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	4	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	5	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	6	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	7	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	8	2	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	9	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	10	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	11	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	12	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	13	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	14	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	15	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	16	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	17	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	18	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	19	2	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	20	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	21	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	22	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	23	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	24	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	25	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	26	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	27	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	28	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	29	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	30	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	31	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	32	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	33	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	34	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	35	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	36	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	37	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	38	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	39	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	40	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	41	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	42	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	43	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	44	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	45	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	46	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	47	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	48	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	49	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	50	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	51	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	52	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	53	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	54	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	55	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	56	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	57	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	58	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	59	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	60	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	61	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	62	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	63	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	64	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	65	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	66	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	67	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	68	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	69	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	70	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	71	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	72	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	73	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	74	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	75	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	76	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	77	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	78	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	79	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	80	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	81	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	82	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	83	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	84	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	85	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	86	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	87	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	88	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	89	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	90	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	91	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	92	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	93	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	94	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	95	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	96	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	97	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	98	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	99	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	100	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	101	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	102	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	103	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	104	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	105	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	106	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	107	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	108	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	109	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	110	11	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	111	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	112	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	113	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	114	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	115	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	116	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	117	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	118	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	119	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	120	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	121	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	122	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	123	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	124	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	125	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	126	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	127	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	128	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	129	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	130	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	131	9	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	132	14	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	133	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	134	11	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	135	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	136	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	137	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	138	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	139	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	140	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	141	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	142	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	143	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	144	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	145	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	146	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	147	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	148	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	149	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	150	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	151	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	152	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	153	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	154	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	155	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	156	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	157	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	158	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	159	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	160	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	161	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	162	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	163	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	164	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	165	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	166	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	167	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	168	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	169	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	170	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	171	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	172	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	173	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	174	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	175	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	176	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	177	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	178	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	179	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	180	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	181	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	182	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	183	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	184	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	185	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	186	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	187	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	188	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	189	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	190	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	191	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	192	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	193	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	194	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	195	16	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	196	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	197	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	198	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	199	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	200	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	201	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	202	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	203	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	204	16	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	205	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	206	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	207	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	208	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	209	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	210	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	211	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	212	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	213	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	214	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	215	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	216	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	217	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	218	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	219	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	220	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	221	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	222	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	223	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	224	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	225	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	226	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	227	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	228	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	229	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	230	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	231	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	232	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	233	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	234	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	235	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	236	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	237	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	238	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	239	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	240	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	241	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	242	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	243	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	244	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	245	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	246	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	247	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	248	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	249	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	250	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	251	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	252	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	253	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	254	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_0	255	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	0	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	1	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	2	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	3	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	4	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	5	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	6	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	7	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	8	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	9	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	10	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	11	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	12	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	13	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	14	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	15	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	16	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	17	9	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	18	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	19	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	20	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	21	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	22	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	23	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	24	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	25	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	26	9	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	27	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	28	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	29	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	30	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	31	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	32	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	33	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	34	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	35	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	36	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	37	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	38	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	39	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	40	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	41	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	42	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	43	16	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	44	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	45	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	46	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	47	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	48	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	49	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	50	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	51	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	52	16	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	53	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	54	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	55	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	56	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	57	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	58	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	59	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	60	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	61	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	62	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	63	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	64	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	65	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	66	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	67	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	68	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	69	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	70	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	71	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	72	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	73	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	74	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	75	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	76	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	77	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	78	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	79	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	80	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	81	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	82	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	83	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	84	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	85	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	86	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	87	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	88	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	89	19	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	90	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	91	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	92	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	93	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	94	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	95	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	96	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	97	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	98	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	99	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	100	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	101	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	102	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	103	36	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	104	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	105	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	106	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	107	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	108	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	109	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	110	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	111	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	112	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	113	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	114	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	115	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	116	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	117	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	118	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	119	12	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	120	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	121	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	122	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	123	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	124	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	125	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	126	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	127	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	128	36	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	129	4	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	130	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	131	5	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	132	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	133	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	134	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	135	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	136	8	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	137	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	138	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	139	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	140	26	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	141	3	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	142	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	143	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	144	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	145	32	EDC
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	146	1	Redundant
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	147	1	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	148	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	149	10	Parity
AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_AM67_MCU_CBASS_SCRP_32B_CLK2_SCR_EDC_CTRL_BUSECC_1	150	1	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	0	32	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	1	32	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	2	1	Redundant
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	3	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	4	4	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	5	12	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	6	4	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_AM67_MCU_CBASS_CBASS_DEFAULT_MMRS_EDC_CTRL_BUSECC	7	32	EDC
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	1	1	Redundant
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	4	12	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	5	3	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	6	3	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	7	1	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	8	1	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	9	2	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	10	3	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	12	10	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	13	5	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	14	3	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	15	4	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	16	2	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	17	12	Parity
AM67_MCU_CBASS_DMSC_SLV_P2P_BRIDGE_DMSC_SLV_BRIDGE_BUSECC	18	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	0	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	1	15	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	2	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	3	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	4	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	5	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	6	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	7	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	8	2	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	9	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	10	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	11	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	12	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	13	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	14	12	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	15	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	16	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	17	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	18	10	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	19	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	20	12	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	21	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	22	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	23	4	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	24	7	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	25	2	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	26	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	27	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	28	26	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	29	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	30	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	31	26	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	32	3	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	33	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	34	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	35	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	36	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	37	1	Redundant
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	38	1	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	39	10	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	40	10	Parity
AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_AM67_MCU_CBASS_CBASS_INT_DMSC_SCR_EDC_CTRL_BUSECC	41	1	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	0	1	Redundant
AM67_MCU_CBASS_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	1	1	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	2	4	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	3	10	Parity
AM67_MCU_CBASS_CBASS_DEFAULT_ERR_AM67_MCU_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	4	32	EDC
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	0	1	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	1	1	Redundant
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	2	1	Redundant
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	3	1	Redundant
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	4	10	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	5	3	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	6	3	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	7	1	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	8	1	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	9	2	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	10	3	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	11	1	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	12	10	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	13	5	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	14	3	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	15	4	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	16	2	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	17	10	Parity
AM67_MCU_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	18	3	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	0	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	1	10	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	2	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	3	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUSECC	4	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	5	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	6	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	7	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	8	2	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	9	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	10	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	11	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	12	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	13	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	14	12	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	15	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	16	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	17	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	18	10	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	19	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	20	12	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	21	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	22	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	23	4	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	24	7	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	25	2	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	26	3	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	27	3	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	28	26	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	29	3	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	30	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	31	26	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	32	3	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	33	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	34	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	35	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	36	32	EDC
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	37	1	Redundant
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	38	1	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	39	10	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	40	10	Parity
AM67_MCU_CBASS_ERR_SCR_AM67_MCU_CBASS_ERR_SCR_EDC_CTRL_BUS ECC	41	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	0	36	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	1	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	2	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	3	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	4	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	5	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	6	29	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	7	36	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	8	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	9	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	10	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	11	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	12	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	13	29	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYS CLK0_1_BUSECC	14	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	15	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	16	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	17	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	18	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	19	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	20	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	21	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	22	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	23	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	24	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	25	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	26	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	27	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	28	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	29	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	30	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	31	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	32	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	33	8	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	34	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	35	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	36	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	37	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	38	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	39	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	40	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	41	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	42	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	43	36	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	44	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	45	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	46	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	47	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	48	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	49	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	50	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	51	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	52	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	53	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	54	8	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	55	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	56	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	57	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	58	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	59	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	60	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	61	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	62	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	63	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	64	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	65	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	66	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	67	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	68	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	69	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	70	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	71	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	72	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	73	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	74	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	75	8	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	76	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	77	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	78	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	79	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_1_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_1_BUSECC	80	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	0	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	1	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	2	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	3	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	4	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	5	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	6	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	7	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	8	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	9	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	10	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	11	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	12	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	13	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	14	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	15	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	16	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	17	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	18	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	19	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	20	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	21	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	22	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	23	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	24	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	25	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	26	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	27	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	28	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	29	11	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	30	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	31	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	32	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	33	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	34	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	35	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	36	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	37	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	38	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	39	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	40	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	41	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	42	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	43	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	44	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	45	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	46	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	47	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	48	11	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	49	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	50	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	51	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	52	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	53	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	54	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	55	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	56	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	57	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	58	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	59	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	60	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	61	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	62	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	63	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	64	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	65	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	66	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	67	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	68	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	69	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	70	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	71	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	72	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	73	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	74	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	75	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	76	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	77	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	78	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	79	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	80	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	81	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	82	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	83	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	84	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	85	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	86	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	87	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	88	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	89	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	90	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	91	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	92	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	93	8	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	94	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	95	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	96	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	97	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	98	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	99	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	100	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	101	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	102	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	103	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	104	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	105	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	106	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	107	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	108	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	109	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	110	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	111	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	112	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	113	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	114	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	115	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	116	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	117	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	118	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	119	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	120	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	121	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	122	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	123	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	124	8	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	125	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	126	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	127	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	128	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	129	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	130	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	131	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	132	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	133	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	134	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	0	36	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	1	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	2	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	3	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	4	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	5	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	6	29	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	7	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	8	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	9	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	10	16	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	11	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	12	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	13	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	14	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	15	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	16	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	17	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	18	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	19	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	20	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	21	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	22	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	23	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	24	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	25	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	26	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	27	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	28	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	29	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	30	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	31	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	32	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	33	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	34	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	35	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	36	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	37	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	38	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	39	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	40	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	41	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	42	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	43	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	44	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	45	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	46	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	47	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	48	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	49	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	50	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	51	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	52	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	53	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	54	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	55	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	56	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	57	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	58	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	59	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	60	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	61	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	62	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	63	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	64	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	65	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	66	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	67	9	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	68	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	69	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	70	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	71	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	72	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	73	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	74	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	75	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	76	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	77	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	78	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	79	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	80	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	81	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	82	1	Redundant
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	83	32	EDC
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	84	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-453. EDC checkers information for ECC Aggregator Instance MCU_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	85	16	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	86	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	87	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	88	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	89	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	90	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	91	10	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	92	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	93	4	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	94	12	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	95	2	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	96	3	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	97	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	98	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	99	1	Parity
AM67_MCU_CBASS_MCU_SYSCLK0_2_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_2_BUSECC	100	1	Redundant
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	0	1	Redundant
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	1	32	EDC
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	2	1	Parity
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	3	10	Parity
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	4	4	Parity
SAM67_MCU_MCU_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-454. Properties of ECC Aggregator Instance WKUP_ECC_AGGR0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	0	EDC Interconnect	Inject with error capture	Yes	78
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	1	EDC Interconnect	Inject with error capture	Yes	81

Copyright © 2026 Texas Instruments Incorporated

Table 12-454. Properties of ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	17
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	3	EDC Interconnect	Inject with error capture	Yes	13
AM67_WKUP_DM_CBASS_ISAM67_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	4	EDC Interconnect	Inject with error capture	Yes	17
AM67_WKUP_DM_CBASS_ISAM67_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	5	EDC Interconnect	Inject with error capture	Yes	7
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	6	EDC Interconnect	Inject with error capture	Yes	70
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	7	EDC Interconnect	Inject with error capture	Yes	68
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	8	EDC Interconnect	Inject with error capture	Yes	78
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	9	EDC Interconnect	Inject with error capture	Yes	33
SAM67_DM_ECC_AGGR_EDC_CTRL	10	EDC Interconnect	Inject with error capture	Yes	6

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	0	24	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	2	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	4	48	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	5	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	6	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	7	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	8	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	9	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	10	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	12	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	13	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	14	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	15	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	16	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	17	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	19	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	20	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	21	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	22	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	23	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	24	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	26	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	27	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	28	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	29	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	30	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	31	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	32	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	33	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	34	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	35	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	36	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	37	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	38	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	39	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	40	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	41	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	42	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	43	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	44	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	45	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	46	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	47	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	48	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	49	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	50	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	51	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	52	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	53	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	54	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	55	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	56	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	57	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	58	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	59	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	60	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	61	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	62	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	63	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	64	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	65	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	66	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	67	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	68	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	69	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	70	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	71	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	72	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	73	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	74	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	75	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	76	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	77	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	0	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	2	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	3	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	4	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	5	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	6	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	7	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	8	48	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	9	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	10	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	11	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	12	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	13	1	Redundant
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	14	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	15	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	16	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	17	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	18	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	19	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	20	10	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	21	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	22	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	23	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	24	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	26	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	27	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	28	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	29	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	30	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	31	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	32	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	33	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	34	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	35	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	36	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	37	9	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	38	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	39	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	40	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	41	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	42	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	43	17	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	44	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	45	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	46	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	47	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	48	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	49	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	50	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	51	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	52	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	53	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	54	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	55	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	56	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	57	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	58	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	59	4	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	60	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	61	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	62	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	63	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	64	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	65	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	66	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	67	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	68	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	69	8	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	70	3	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	71	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	72	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	73	5	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	74	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	75	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	76	1	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	77	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	78	17	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	79	6	Parity
AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_AM67_WKUP_DM_CBASS_ISA3SS_AM62A_MAIN_0_PKTDMA_MEM_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	80	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	0	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	4	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_SRC_BUSECC	16	4	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	0	1	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	1	32	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	4	1	Redundant
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	5	1	Redundant
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	6	1	Redundant
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	7	8	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	8	8	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	9	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	10	8	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_IK3VTM_N16FFC_WKUP_0_VBUSP_P2P_BRIDGE_IK3VTM_N16FFC_WKUP_0_VBUSP_BRIDGE_BUSECC	12	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	1	32	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	4	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_SRC_BUSECC	16	4	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	3	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	4	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	5	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_DM_DM_ECC_AGGR_WKUP_0_CFG_BRIDGE_DST_BUSECC	6	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	0	24	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	2	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	4	36	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	5	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	6	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	7	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	8	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	9	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	10	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	12	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	13	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	14	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	15	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	16	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	17	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	19	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	20	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	21	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	22	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	23	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	24	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	25	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	26	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	27	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	28	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	29	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	30	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	31	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	32	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	33	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	34	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	35	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	36	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	37	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	38	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	39	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	40	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	41	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	42	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	43	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	44	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	45	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	46	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	47	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	48	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	49	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	50	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	51	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	52	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	53	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	54	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	55	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	56	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	57	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	58	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	59	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	60	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	61	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	62	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	63	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	64	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	65	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	66	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	67	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	68	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	69	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	0	24	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	2	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	3	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	4	36	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	5	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	6	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	7	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	8	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	9	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	10	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	12	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	13	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	14	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	15	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	16	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	17	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	19	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	20	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	21	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	22	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	23	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	24	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	26	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	27	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	28	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	29	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	30	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	31	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	32	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	33	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	34	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	35	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	36	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	37	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	38	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	39	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	40	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	41	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	42	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	43	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	44	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	45	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	46	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	47	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	48	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	49	2	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	50	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	51	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	52	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	53	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	54	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	55	2	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	56	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	57	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	58	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	59	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	60	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	61	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	62	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	63	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	64	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	65	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	66	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_SRC_EDC_CTRL_BUSECC	67	5	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	0	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	1	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	2	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	3	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	4	48	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	5	2	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	6	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	7	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	8	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	9	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	10	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	11	2	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	12	8	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	13	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	14	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	15	2	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	16	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	17	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	18	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	19	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	20	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	21	8	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	22	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	23	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	24	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	25	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	26	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	27	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	28	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	29	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	30	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	31	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	32	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	33	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	34	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	35	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	36	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	37	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	38	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	39	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	40	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	41	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	42	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	43	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	44	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	45	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	46	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	47	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	48	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	49	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	50	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	51	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	52	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	53	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	54	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	55	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	56	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	57	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	58	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	59	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	60	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	61	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	62	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	63	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	64	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	65	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	66	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	67	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	68	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	69	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	70	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	71	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	72	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	73	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	74	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	75	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	76	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_1_CLK_EDC_CTRL_CBASS_INT_DM_CLK_1_BUSECC	77	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	0	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	1	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	3	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	4	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	5	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	6	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	7	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	8	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	9	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	10	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	11	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	12	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	13	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	14	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	15	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	16	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	17	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	18	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	19	10	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	20	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	21	1	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	22	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	23	32	EDC
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	24	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	26	1	Redundant
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	27	4	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	28	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	29	12	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	30	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-455. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR0 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	31	3	Parity
AM67_WKUP_DM_CBASS_DM_CLK_4_CLK_EDC_CTRL_CBASS_INT_DM_CLK_4_BUSECC	32	1	Parity
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	0	1	Redundant
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	1	32	EDC
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	2	1	Parity
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	3	10	Parity
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	4	4	Parity
SAM67_DM_DM_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-456. Properties of ECC Aggregator Instance WKUP_ECC_AGGR1

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	0	EDC Interconnect	Inject with error capture	Yes	7
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	1	EDC Interconnect	Inject with error capture	Yes	70
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	69
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	3	EDC Interconnect	Inject with error capture	Yes	17
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	4	EDC Interconnect	Inject with error capture	Yes	7
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	5	EDC Interconnect	Inject with error capture	Yes	11
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	6	EDC Interconnect	Inject with error capture	Yes	6

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	3	10	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	4	3	Parity
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	5	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_P2P_BRIDGE_ISAM67_DM_MCU_ECC_AGGR_WKUP_1_CFG_BRIDGE_DST_BUSECC	6	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	0	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	2	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	3	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	4	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	5	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	6	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	7	36	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	8	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	9	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	10	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	12	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	13	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	14	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	15	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	16	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	17	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	19	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	20	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	21	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	22	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	23	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	24	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	26	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	27	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	28	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	29	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	30	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	31	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	32	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	33	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	34	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	35	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	36	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	37	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	38	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	39	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	40	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	41	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	42	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	43	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	44	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	45	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	46	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	47	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	48	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	49	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	50	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	51	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	52	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	53	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	54	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	55	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	56	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	57	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	58	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	59	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	60	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	61	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	62	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	63	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	64	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	65	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	66	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	67	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	68	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_MCU_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	69	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	0	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	2	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	3	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	4	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	5	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	6	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	7	36	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	8	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	9	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	10	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	11	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	12	1	Redundant
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	13	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	14	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	15	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	16	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	17	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	19	10	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	20	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	21	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	22	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	23	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	24	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	25	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	26	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	27	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	28	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	29	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	30	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	31	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	32	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	33	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	34	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	35	6	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	36	6	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	37	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	38	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	39	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	40	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	41	5	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	42	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	43	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	44	2	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	45	2	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	46	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	47	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	48	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	49	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	50	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	51	2	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	52	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	53	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	54	3	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	55	8	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	56	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	57	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	58	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	59	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	60	8	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	61	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	62	4	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	63	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	64	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	65	1	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	66	9	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	67	5	Parity
AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_AM67_WKUP_DM_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2M_BRIDGE_DST_EDC_CTRL_BUSECC	68	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	0	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	1	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	2	1	Redundant
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	3	1	Redundant
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	4	12	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	5	3	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	6	3	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	7	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	8	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	9	2	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	10	3	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	11	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	12	10	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	13	5	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	14	3	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	15	4	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_SRC_BUSECC	16	2	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	0	1	Redundant
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	1	1	Redundant
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	2	1	Redundant
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	3	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-457. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR1 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	4	3	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	5	1	Parity
AM67_MCU_FW_CBASS_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_P2P_BRIDGE_IAM67_MCU_CBASS_MCU_0_CBASS_DMSC_SLV_BRIDGE_DST_BUSECC	6	1	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	0	1	Redundant
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	1	32	EDC
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	2	1	Redundant
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	3	3	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	4	1	Redundant
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	5	4	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	6	12	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	7	12	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	8	10	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	9	3	Parity
AM67_MCU_FW_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	10	1	Parity
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	0	1	Redundant
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	1	32	EDC
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	2	1	Parity
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	3	10	Parity
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	4	4	Parity
SAM67_DM_MCU_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-458. Properties of ECC Aggregator Instance WKUP_ECC_AGGR2

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	0	EDC Interconnect	Inject with error capture	Yes	256
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	1	EDC Interconnect	Inject with error capture	Yes	238
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	2	EDC Interconnect	Inject with error capture	Yes	53
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	3	EDC Interconnect	Inject with error capture	Yes	256
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	4	EDC Interconnect	Inject with error capture	Yes	256

Copyright © 2026 Texas Instruments Incorporated

Table 12-458. Properties of ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Max Number of Checkers
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	5	EDC Interconnect	Inject with error capture	Yes	172
ISAM67_DM2WS_VBUSM_GASKET_MCU_0_EDC_CTRL	6	EDC Interconnect	Inject with error capture	Yes	1
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	7	EDC Interconnect	Inject with error capture	Yes	31
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	8	EDC Interconnect	Inject with error capture	Yes	13
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	9	EDC Interconnect	Inject with error capture	Yes	13
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	10	EDC Interconnect	Inject with error capture	Yes	256
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	11	EDC Interconnect	Inject with error capture	Yes	154
AM67_WKUP_SAFE_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	12	EDC Interconnect	Inject with error capture	Yes	5
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	13	EDC Interconnect	Inject with error capture	Yes	19
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	14	EDC Interconnect	Inject with error capture	Yes	42
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	15	EDC Interconnect	Inject with error capture	Yes	121
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	16	EDC Interconnect	Inject with error capture	Yes	6

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	0	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	1	28	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	2	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	3	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	4	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	5	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	6	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	7	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	8	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	9	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	10	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	11	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	12	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	13	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	14	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	15	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	16	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	17	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	18	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	19	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	20	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	21	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	22	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	23	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	24	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	25	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	26	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	27	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	28	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	29	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	30	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	31	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	32	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	33	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	34	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	35	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	36	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	37	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	38	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	39	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	40	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	41	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	42	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	43	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	44	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	45	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	46	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	47	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	48	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	49	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	50	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	51	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	52	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	53	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	54	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	55	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	56	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	57	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	58	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	59	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	60	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	61	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	62	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	63	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	64	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	65	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	66	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	67	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	68	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	69	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	70	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	71	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	72	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	73	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	74	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	75	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	76	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	77	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	78	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	79	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	80	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	81	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	82	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	83	14	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	84	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	85	21	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	86	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	87	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	88	30	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	89	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	90	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	91	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	92	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	93	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	94	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	95	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	96	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	97	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	98	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	99	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	100	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	101	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	102	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	103	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	104	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	105	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	106	31	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	107	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	108	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	109	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	110	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	111	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	112	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	113	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	114	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	115	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	116	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	117	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	118	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	119	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	120	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	121	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	122	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	123	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	124	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	125	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	126	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	127	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	128	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	129	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	130	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	131	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	132	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	133	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	134	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	135	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	136	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	137	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	138	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	139	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	140	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	141	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	142	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	143	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	144	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	145	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	146	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	147	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	148	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	149	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	150	6	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	151	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	152	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	153	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	154	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	155	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	156	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	157	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	158	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	159	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	160	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	161	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	162	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	163	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	164	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	165	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	166	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	167	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	168	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	169	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	170	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	171	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	172	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	173	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	174	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	175	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	176	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	177	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	178	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	179	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	180	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	181	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	182	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	183	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	184	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	185	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	186	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	187	5	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	188	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	189	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	190	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	191	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	192	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	193	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	194	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	195	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	196	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	197	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	198	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	199	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	200	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	201	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	202	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	203	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	204	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	205	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	206	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	207	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	208	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	209	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	210	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	211	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	212	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	213	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	214	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	215	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	216	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	217	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	218	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	219	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	220	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	221	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	222	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	223	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	224	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	225	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	226	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	227	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	228	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	229	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	230	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	231	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	232	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	233	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	234	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	235	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	236	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	237	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	238	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	239	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	240	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	241	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	242	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	243	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	244	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	245	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	246	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	247	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	248	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	249	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	250	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	251	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	252	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	253	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	254	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_0	255	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	0	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	1	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	2	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	3	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	4	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	5	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	6	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	7	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	8	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	9	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	10	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	11	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	12	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	13	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	14	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	15	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	16	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	17	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	18	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	19	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	20	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	21	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	22	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	23	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	24	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	25	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	26	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	27	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	28	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	29	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	30	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	31	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	32	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	33	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	34	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	35	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	36	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	37	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	38	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	39	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	40	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	41	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	42	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	43	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	44	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	45	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	46	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	47	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	48	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	49	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	50	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	51	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	52	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	53	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	54	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	55	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	56	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	57	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	58	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	59	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	60	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	61	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	62	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	63	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	64	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	65	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	66	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	67	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	68	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	69	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	70	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	71	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	72	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	73	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	74	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	75	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	76	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	77	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	78	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	79	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	80	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	81	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	82	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	83	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	84	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	85	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	86	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	87	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	88	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	89	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	90	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	91	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	92	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	93	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	94	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	95	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	96	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	97	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	98	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	99	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	100	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	101	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	102	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	103	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	104	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	105	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	106	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	107	12	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	108	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	109	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	110	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	111	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	112	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	113	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	114	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	115	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	116	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	117	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	118	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	119	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	120	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	121	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	122	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	123	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	124	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	125	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	126	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	127	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	128	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	129	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	130	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	131	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	132	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	133	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	134	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	135	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	136	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	137	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	138	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	139	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	140	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	141	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	142	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	143	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	144	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	145	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	146	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	147	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	148	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	149	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	150	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	151	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	152	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	153	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	154	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	155	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	156	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	157	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	158	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	159	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	160	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	161	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	162	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	163	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	164	7	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	165	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	166	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	167	7	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	168	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	169	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	170	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	171	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	172	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	173	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	174	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	175	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	176	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	177	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	178	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	179	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	180	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	181	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	182	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	183	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	184	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	185	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	186	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	187	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	188	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	189	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	190	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	191	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	192	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	193	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	194	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	195	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	196	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	197	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	198	6	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	199	3	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	200	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	201	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	202	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	203	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	204	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	205	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	206	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	207	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	208	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	209	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	210	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	211	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	212	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	213	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	214	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	215	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	216	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	217	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	218	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	219	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	220	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	221	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	222	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	223	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	224	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	225	2	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	226	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	227	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	228	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	229	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	230	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	231	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	232	1	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	233	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	234	8	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	235	4	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	236	32	Parity
AM67_MCU_CTRL_MMR_EDC_CTRL_BUSECC_1	237	32	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	0	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	1	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	2	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	3	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	4	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	5	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	6	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	7	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	8	12	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	9	24	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	10	6	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	11	3	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	12	3	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	13	12	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	14	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	15	3	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	16	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	17	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	18	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	19	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	20	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	21	8	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	22	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	23	5	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	24	4	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	25	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	26	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	27	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	28	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	29	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	30	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	31	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	32	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	33	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	34	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	35	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	36	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	37	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	38	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	39	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	40	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	41	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	42	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	43	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	44	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	45	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	46	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	47	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	48	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	49	7	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	50	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	51	1	Parity
AM67_MCU_PLL_MMR_EDC_CTRL_BUSECC	52	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	0	32	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	1	32	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	2	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	3	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	4	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	5	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	6	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	7	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	8	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	9	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	10	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	11	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	12	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	13	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	14	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	15	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	16	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	17	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	18	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	19	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	20	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	21	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	22	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	23	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	24	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	25	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	26	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	27	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	28	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	29	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	30	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	31	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	32	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	33	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	34	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	35	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	36	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	37	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	38	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	39	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	40	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	41	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	42	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	43	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	44	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	45	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	46	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	47	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	48	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	49	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	50	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	51	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	52	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	53	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	54	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	55	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	56	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	57	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	58	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	59	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	60	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	61	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	62	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	63	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	64	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	65	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	66	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	67	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	68	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	69	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	70	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	71	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	72	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	73	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	74	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	75	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	76	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	77	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	78	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	79	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	80	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	81	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	82	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	83	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	84	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	85	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	86	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	87	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	88	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	89	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	90	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	91	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	92	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	93	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	94	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	95	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	96	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	97	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	98	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	99	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	100	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	101	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	102	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	103	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	104	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	105	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	106	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	107	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	108	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	109	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	110	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	111	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	112	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	113	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	114	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	115	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	116	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	117	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	118	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	119	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	120	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	121	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	122	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	123	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	124	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	125	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	126	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	127	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	128	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	129	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	130	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	131	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	132	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	133	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	134	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	135	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	136	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	137	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	138	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	139	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	140	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	141	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	142	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	143	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	144	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	145	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	146	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	147	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	148	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	149	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	150	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	151	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	152	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	153	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	154	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	155	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	156	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	157	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	158	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	159	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	160	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	161	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	162	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	163	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	164	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	165	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	166	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	167	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	168	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	169	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	170	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	171	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	172	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	173	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	174	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	175	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	176	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	177	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	178	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	179	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	180	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	181	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	182	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	183	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	184	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	185	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	186	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	187	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	188	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	189	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	190	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	191	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	192	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	193	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	194	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	195	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	196	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	197	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	198	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	199	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	200	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	201	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	202	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	203	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	204	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	205	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	206	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	207	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	208	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	209	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	210	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	211	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	212	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	213	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	214	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	215	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	216	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	217	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	218	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	219	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	220	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	221	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	222	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	223	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	224	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	225	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	226	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	227	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	228	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	229	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	230	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	231	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	232	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	233	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	234	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	235	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	236	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	237	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	238	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	239	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	240	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	241	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	242	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	243	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	244	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	245	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	246	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	247	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	248	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	249	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	250	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	251	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	252	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	253	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	254	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_0	255	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	0	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	1	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	2	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	3	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	4	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	5	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	6	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	7	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	8	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	9	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	10	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	11	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	12	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	13	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	14	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	15	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	16	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	17	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	18	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	19	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	20	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	21	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	22	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	23	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	24	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	25	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	26	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	27	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	28	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	29	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	30	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	31	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	32	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	33	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	34	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	35	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	36	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	37	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	38	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	39	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	40	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	41	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	42	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	43	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	44	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	45	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	46	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	47	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	48	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	49	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	50	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	51	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	52	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	53	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	54	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	55	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	56	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	57	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	58	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	59	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	60	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	61	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	62	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	63	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	64	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	65	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	66	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	67	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	68	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	69	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	70	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	71	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	72	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	73	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	74	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	75	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	76	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	77	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	78	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	79	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	80	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	81	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	82	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	83	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	84	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	85	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	86	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	87	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	88	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	89	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	90	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	91	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	92	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	93	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	94	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	95	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	96	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	97	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	98	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	99	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	100	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	101	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	102	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	103	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	104	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	105	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	106	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	107	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	108	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	109	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	110	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	111	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	112	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	113	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	114	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	115	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	116	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	117	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	118	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	119	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	120	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	121	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	122	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	123	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	124	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	125	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	126	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	127	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	128	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	129	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	130	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	131	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	132	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	133	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	134	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	135	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	136	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	137	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	138	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	139	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	140	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	141	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	142	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	143	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	144	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	145	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	146	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	147	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	148	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	149	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	150	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	151	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	152	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	153	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	154	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	155	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	156	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	157	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	158	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	159	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	160	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	161	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	162	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	163	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	164	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	165	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	166	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	167	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	168	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	169	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	170	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	171	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	172	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	173	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	174	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	175	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	176	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	177	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	178	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	179	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	180	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	181	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	182	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	183	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	184	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	185	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	186	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	187	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	188	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	189	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	190	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	191	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	192	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	193	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	194	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	195	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	196	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	197	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	198	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	199	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	200	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	201	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	202	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	203	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	204	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	205	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	206	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	207	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	208	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	209	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	210	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	211	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	212	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	213	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	214	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	215	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	216	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	217	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	218	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	219	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	220	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	221	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	222	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	223	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	224	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	225	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	226	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	227	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	228	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	229	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	230	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	231	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	232	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	233	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	234	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	235	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	236	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	237	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	238	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	239	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	240	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	241	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	242	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	243	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	244	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	245	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	246	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	247	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	248	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	249	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	250	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	251	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	252	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	253	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	254	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_1	255	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	0	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	1	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	2	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	3	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	4	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	5	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	6	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	7	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	8	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	9	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	10	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	11	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	12	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	13	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	14	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	15	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	16	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	17	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	18	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	19	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	20	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	21	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	22	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	23	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	24	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	25	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	26	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	27	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	28	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	29	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	30	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	31	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	32	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	33	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	34	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	35	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	36	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	37	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	38	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	39	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	40	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	41	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	42	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	43	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	44	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	45	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	46	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	47	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	48	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	49	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	50	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	51	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	52	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	53	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	54	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	55	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	56	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	57	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	58	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	59	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	60	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	61	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	62	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	63	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	64	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	65	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	66	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	67	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	68	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	69	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	70	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	71	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	72	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	73	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	74	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	75	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	76	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	77	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	78	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	79	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	80	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	81	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	82	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	83	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	84	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	85	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	86	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	87	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	88	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	89	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	90	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	91	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	92	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	93	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	94	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	95	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	96	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	97	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	98	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	99	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	100	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	101	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	102	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	103	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	104	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	105	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	106	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	107	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	108	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	109	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	110	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	111	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	112	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	113	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	114	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	115	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	116	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	117	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	118	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	119	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	120	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	121	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	122	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	123	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	124	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	125	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	126	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	127	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	128	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	129	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	130	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	131	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	132	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	133	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	134	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	135	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	136	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	137	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	138	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	139	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	140	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	141	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	142	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	143	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	144	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	145	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	146	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	147	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	148	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	149	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	150	4	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	151	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	152	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	153	3	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	154	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	155	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	156	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	157	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	158	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	159	2	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	160	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	161	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	162	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	163	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	164	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	165	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	166	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	167	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	168	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	169	1	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	170	32	Parity
AM67XX_MCU_PADCFG_CTRL_MMR_EDC_CTRL_BUSECC_2	171	32	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
ISAM67_DM2WS_VBUSM_GASKET_MCU_0_EDC_CTRL	0	48	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	0	32	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	1	32	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	2	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	3	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	4	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	5	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	6	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	7	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	8	36	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	9	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	10	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	11	2	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	12	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	13	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	14	10	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	15	4	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	16	2	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	17	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	18	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	19	1	Redundant
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	20	4	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	21	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	22	10	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	23	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	24	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	25	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	26	1	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	27	2	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	28	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	29	3	Parity
AM67_WKUP_SAFE_CBASS_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_M2P_BRIDGE_EXPORT_AM67_WKUP_DM_CBASS_TO_AM67_WKUP_SAFE_CBASS_DATA_L0_BRIDGE_BUSECC	30	4	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	0	1	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	1	32	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	2	1	Redundant
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	3	1	Redundant
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	4	1	Redundant
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	5	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	6	1	Redundant
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	7	8	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	8	8	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	9	8	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	10	8	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	11	1	Parity
AM67_WKUP_SAFE_CBASS_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_P2P_BRIDGE_ISAM67_WKUP_SAFE_ECC_AGGR_WKUP_0_CFG_BRIDGE_BUSECC	12	1	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	0	1	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	1	32	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	2	1	Redundant
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	3	1	Redundant
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	4	1	Redundant
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	5	1	Redundant
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	6	1	Redundant
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	7	8	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	8	8	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	9	8	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	10	8	Parity
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	11	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_P2P_BRIDGE_IAM67_WKUP_SAFE_CBASS_WKUP_0_CBASS_ERR_SLV_BRIDGE_BUSECC	12	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	0	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	1	36	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	2	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	3	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	4	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	5	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	6	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	7	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	8	2	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	9	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	10	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	11	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	12	36	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	13	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	14	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	15	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	16	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	17	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	18	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	19	2	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	20	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	21	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	22	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	23	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	24	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	25	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	26	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	27	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	28	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	29	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	30	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	31	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	32	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	33	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	34	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	35	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	36	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	37	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	38	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	39	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	40	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	41	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	42	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	43	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	44	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	45	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	46	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	47	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	48	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	49	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	50	12	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	51	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	52	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	53	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	54	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	55	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	56	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	57	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	58	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	59	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	60	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	61	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	62	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	63	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	64	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	65	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	66	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	67	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	68	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	69	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	70	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	71	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	72	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	73	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	74	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	75	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	76	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	77	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	78	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	79	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	80	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	81	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	82	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	83	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	84	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	85	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	86	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	87	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	88	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	89	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	90	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	91	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	92	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	93	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	94	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	95	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	96	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	97	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	98	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	99	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	100	36	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	101	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	102	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	103	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	104	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	105	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	106	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	107	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	108	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	109	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	110	15	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	111	11	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	112	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	113	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	114	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	115	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	116	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	117	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	118	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	119	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	120	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	121	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	122	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	123	36	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	124	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	125	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	126	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	127	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	128	1	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	129	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	130	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	131	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	132	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	133	15	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	134	11	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	135	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	136	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	137	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	138	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	139	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	140	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	141	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	142	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	143	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	144	15	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	145	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	146	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	147	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	148	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	149	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	150	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	151	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	152	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	153	15	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	154	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	155	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	156	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	157	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	158	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	159	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	160	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	161	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	162	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	163	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	164	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	165	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	166	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	167	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	168	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	169	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	170	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	171	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	172	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	173	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	174	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	175	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	176	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	177	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	178	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	179	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	180	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	181	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	182	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	183	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	184	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	185	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	186	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	187	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	188	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	189	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	190	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	191	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	192	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	193	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	194	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	195	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	196	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	197	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	198	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	199	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	200	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	201	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	202	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	203	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	204	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	205	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	206	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	207	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	208	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	209	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	210	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	211	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	212	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	213	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	214	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	215	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	216	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	217	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	218	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	219	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	220	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	221	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	222	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	223	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	224	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	225	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	226	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	227	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	228	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	229	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	230	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	231	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	232	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	233	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	234	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	235	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	236	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	237	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	238	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	239	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	240	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	241	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	242	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	243	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	244	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	245	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	246	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	247	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	248	9	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	249	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	250	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	251	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	252	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	253	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	254	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_0	255	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	0	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	1	9	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	2	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	3	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	4	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	5	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	6	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	7	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	8	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	9	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	10	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	11	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	12	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	13	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	14	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	15	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	16	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	17	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	18	9	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	19	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	20	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	21	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	22	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	23	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	24	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	25	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	26	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	27	9	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	28	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	29	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	30	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	31	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	32	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	33	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	34	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	35	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	36	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	37	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	38	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	39	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	40	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	41	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	42	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	43	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	44	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	45	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	46	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	47	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	48	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	49	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	50	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	51	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	52	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	53	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	54	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	55	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	56	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	57	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	58	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	59	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	60	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	61	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	62	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	63	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	64	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	65	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	66	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	67	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	68	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	69	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	70	17	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	71	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	72	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	73	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	74	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	75	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	76	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	77	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	78	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	79	17	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	80	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	81	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	82	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	83	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	84	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	85	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	86	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	87	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	88	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	89	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	90	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	91	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	92	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	93	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	94	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	95	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	96	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	97	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	98	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	99	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	100	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	101	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	102	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	103	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	104	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	105	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	106	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	107	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	108	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	109	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	110	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	111	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	112	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	113	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	114	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	115	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	116	19	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	117	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	118	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	119	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	120	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	121	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	122	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	123	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	124	4	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	125	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	126	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	127	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	128	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	129	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	130	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	131	12	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	132	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	133	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	134	5	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	135	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	136	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	137	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	138	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	139	8	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	140	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	141	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	142	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	143	26	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	144	3	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	145	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	146	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	147	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	148	32	EDC
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	149	1	Redundant
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	150	1	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	151	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	152	10	Parity
AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_AM67_WKUP_SAFE_CBASS_SCRP_SAFE_CLK4_SCR_EDC_CTRL_BUSECC_1	153	1	Parity
AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	0	1	Redundant
AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	1	1	Parity
AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	2	4	Parity
AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	3	10	Parity
AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_AM67_WKUP_SAFE_CBASS_CBASS_DEFAULT_ERR_EDC_CTRL_BUSECC	4	32	EDC

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	0	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	1	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	2	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	3	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	4	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	5	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	6	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	7	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	8	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	9	2	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	10	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	11	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	12	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	13	5	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	14	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	15	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	16	2	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	17	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SLV_P2P_BRIDGE_ERR_SLV_BRIDGE_BUSECC	18	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	0	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	1	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	2	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	3	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	4	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	5	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	6	4	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	7	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	8	2	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	9	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	10	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	11	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	12	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	13	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	14	12	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	15	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	16	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	17	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	18	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	19	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	20	12	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	21	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	22	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	23	4	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	24	7	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	25	2	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	26	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	27	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	28	26	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	29	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	30	3	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	31	26	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	32	3	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	33	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	34	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	35	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	36	32	EDC
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	37	1	Redundant
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	38	1	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	39	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	40	10	Parity
AM67_WKUP_SAFE_CBASS_ERR_SCR_AM67_WKUP_SAFE_CBASS_ERR_SCR_EDC_CTRL_BUSECC	41	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	0	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	1	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	2	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	3	15	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	4	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	5	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	6	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	7	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	8	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	9	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	10	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	11	8	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	12	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	13	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	14	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	15	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	16	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	17	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	18	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	19	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	20	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	21	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	22	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	23	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	24	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	25	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	26	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	27	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	28	8	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	29	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	30	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	31	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	32	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	33	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	34	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	35	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	36	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	37	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	38	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	39	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	40	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	41	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	42	10	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	43	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	44	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	45	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	46	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	47	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	48	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	49	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	50	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	51	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	52	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	53	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	54	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	55	9	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	56	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	57	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	58	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	59	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	60	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	61	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	62	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	63	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	64	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	65	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	66	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	67	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	68	9	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	69	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	70	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	71	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	72	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	73	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	74	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	75	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	76	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	77	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	78	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	79	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	80	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	81	8	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	82	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	83	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	84	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	85	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	86	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	87	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	88	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	89	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	90	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	91	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	92	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	93	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	94	17	Parity

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	95	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	96	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	97	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	98	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	99	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	100	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	101	2	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	102	8	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	103	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	104	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	105	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	106	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	107	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	108	1	Redundant
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	109	32	EDC
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	110	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	111	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	112	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	113	3	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	114	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	115	4	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	116	12	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	117	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	118	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	119	1	Parity
AM67_WKUP_SAFE_CBASS_MCU_SYSCLK0_4_CLK_EDC_CTRL_CBASS_INT_MCU_SYSCLK0_4_BUSECC	120	1	Redundant
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	0	1	Redundant

Copyright © 2026 Texas Instruments Incorporated

Table 12-459. EDC checkers information for ECC Aggregator Instance WKUP_ECC_AGGR2 (continued)

Protected Interconnect	Group ID	Width	Checker Type
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	1	32	EDC
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	2	1	Parity
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	3	10	Parity
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	4	4	Parity
SAM67_WKUP_SAFE_ECC_AGGR_EDC_CTRL	5	3	Parity

Table 12-460. Properties of ECC Aggregator Instance DMASS0_ECC_AGGR_0

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
DMSS_AM67_PKTDMA_CFG_CONFIG	0	ECC Wrapper	Inject with error capture	Yes	96	2 KB
DMSS_AM67_PKTDMA_CFG_STATE	1	ECC Wrapper	Inject with error capture	Yes	220	1 KB
DMSS_AM67_PKTDMA_TPCFIFO_F0	2	ECC Wrapper	Inject with error capture	Yes	141	3 KB
DMSS_AM67_PKTDMA_TPCFIFO_F1	3	ECC Wrapper	Inject with error capture	Yes	141	3 KB
DMSS_AM67_PKTDMA_RPCFIFO_F0	4	ECC Wrapper	Inject with error capture	Yes	128	2 KB
DMSS_AM67_PKTDMA_RPCFIFO_F1	5	ECC Wrapper	Inject with error capture	Yes	128	2 KB
DMSS_AM67_PKTDMA_RPCFIFO_WC	6	ECC Wrapper	Inject with error capture	Yes	22	792 B
DMSS_AM67_PKTDMA_STATS_STST0	7	ECC Wrapper	Inject with error capture	Yes	96	348 B
DMSS_AM67_PKTDMA_STATS_STSR0	8	ECC Wrapper	Inject with error capture	Yes	128	384 B
DMSS_AM67_PKTDMA_RINGOCC_CNTR	9	ECC Wrapper	Inject with error capture	Yes	18	675 B
DMSS_AM67_BCDMA_CFG_CONFIG	10	ECC Wrapper	Inject with error capture	Yes	86	1 KB
DMSS_AM67_BCDMA_CFG_STATE	11	ECC Wrapper	Inject with error capture	Yes	400	6 KB
DMSS_AM67_BCDMA_PCFIFO_DFIFO_F0	12	ECC Wrapper	Inject with error capture	Yes	128	4 KB
DMSS_AM67_BCDMA_PCFIFO_DFIFO_F1	13	ECC Wrapper	Inject with error capture	Yes	128	4 KB
DMSS_AM67_BCDMA_TPCFIFO_F0	14	ECC Wrapper	Inject with error capture	Yes	141	3 KB
DMSS_AM67_BCDMA_TPCFIFO_F1	15	ECC Wrapper	Inject with error capture	Yes	141	3 KB
DMSS_AM67_BCDMA_RPCFIFO_F0	16	ECC Wrapper	Inject with error capture	Yes	128	2 KB
DMSS_AM67_BCDMA_RPCFIFO_F1	17	ECC Wrapper	Inject with error capture	Yes	128	2 KB
DMSS_AM67_BCDMA_RPCFIFO_WC	18	ECC Wrapper	Inject with error capture	Yes	19	713 B
DMSS_AM67_BCDMA_STATS_STST0	19	ECC Wrapper	Inject with error capture	Yes	96	684 B
DMSS_AM67_BCDMA_STATS_STSR0	20	ECC Wrapper	Inject with error capture	Yes	96	684 B

Copyright © 2026 Texas Instruments Incorporated

Table 12-460. Properties of ECC Aggregator Instance DMASS0_ECC_AGGR_0 (continued)

RAM ID Name	RAM ID	ECC Type	Inject Type	Accessible Flag	Row Width	RAM Size
DMSS_AM67_BCDMA_RINGOCC_CNTR	21	ECC Wrapper	Inject with error capture	Yes	18	369 B
DMSS_AM67_INTAGGR_STATREG_SR_SPRAM_184X128_SWW_SR	22	ECC Wrapper	Inject with error capture	Yes	128	3 KB
DMSS_AM67_INTAGGR_COMMON_IM_TPRAM_1531X34_SWW_SR	23	ECC Wrapper	Inject with error capture	Yes	34	6 KB
DMSS_AM67_IPCSS_RINGACC_STRAM	24	ECC Wrapper	Inject with error capture	Yes	216	540 B
DMSS_AM67_IPCSS_SEC_PROXY_BUF_STRAM	25	ECC Wrapper	Inject with error capture	Yes	91	865 B
DMSS_AM67_IPCSS_SEC_PROXY_BUFRAM	26	ECC Wrapper	Inject with error capture	Yes	64	64 B
DMSS_AM67_IPCSS_MSRAM_ECC0	27	ECC Wrapper	Inject with error capture	Yes	64	16 KB

12.9 Display Subsystem and Peripherals

Copyright © 2026 Texas Instruments Incorporated

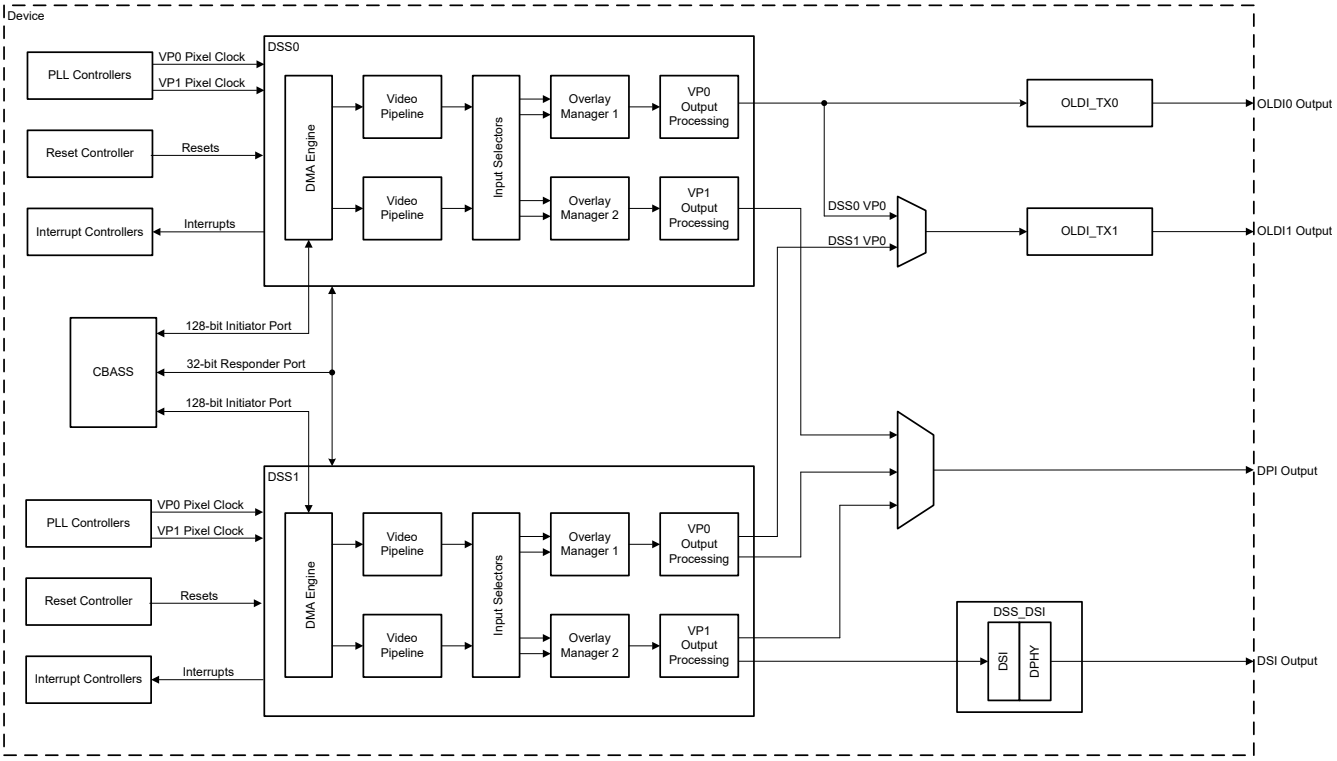
12.9.1 Display Subsystem (DSS)

This section describes the Display Subsystem (DSS) in the device.

12.9.1.1 DSS Overview

The Display Subsystem (DSS) is a flexible, multi-pipeline subsystem that supports high-resolution display outputs. DSS includes input pipelines providing multi-layer blending with transparency to enable on-the-fly composition. Various pixel processing capabilities are supported, such as color space conversion and scaling, among others. DSS includes a DMA engine, which allows direct access to the frame buffer (device system memory). Display outputs can directly drive device pads as a Display Parallel Interface (DPI).

Figure 12-490 is a high-level overview of the display subsystem.



Copyright © 2026 Texas Instruments Incorporated

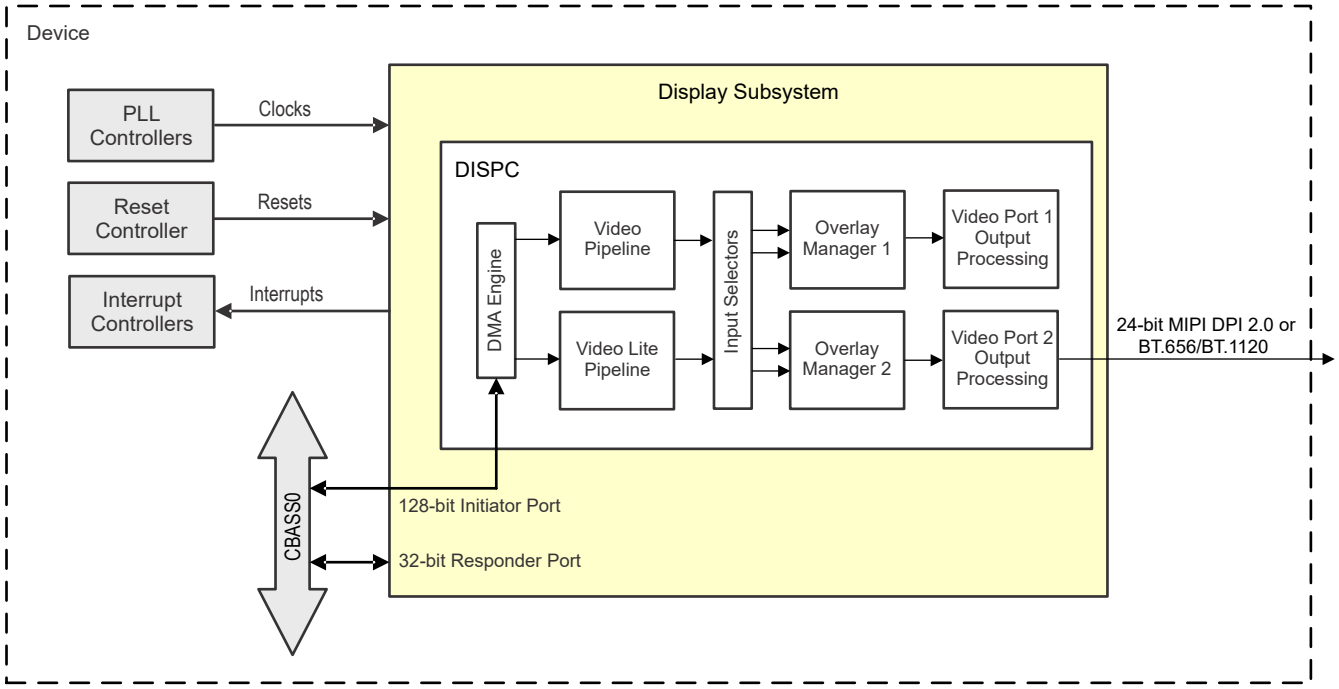


Figure 12-490. DSS Overview

12.9.1.1.1 DSS Features

The Display Subsystem module includes the following features:

- Triple display support over OLDI/LVDS (1x or 2x OLDI-SL, 1x OLDI-DL), DSI or DPI
  - OLDI-SL (Single Link): up to 1920x1080 at 60fps (165MHz Pixel Clock)
  - OLDI-DL (Dual Link): up to 3840x1080 at 60fps (150MHz Pixel Clock)
  - MIPI® DSI: with 4 Lane MIPI® D-PHY supports up to 3840x1080 at 60fps (300MHz Pixel Clock)
  - DPI (24bit RGB parallel interface): up to 1920x1080 at 60fps (165MHz Pixel Clock)
  - Support for RGB/YUV422 modes
  - Support for progressive/interlaced modes
- Four display pipelines with hardware overlap support. A maximum of two display pipelines may be used per display.
- Supports safety features such as freeze frame detection and data correctness check.
- Two input display processing pipelines
  - One video pipeline supporting full RGB and 8/10-bit YUV data formats with 3/5-tap 16-phase scaler capable of 0.25x to 16x resizing.
  - One video_lite pipeline supporting full RGB and 8/10 YUV data formats (no resizing support)
- Two Overlay Managers with multi layer alpha blending
- On-the-fly X/Y-axis flip of the source (Flip/Mirror mode support)
- Safety check (freeze frame detection and data correctness check)

Note

Some features may not be available. See *Module Integration* for more information.

12.9.1.1.2 Unsupported Features

See the *Module Integration* section for information about unsupported features.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.2 DSS Environment

This section describes the interfaces handled by the display subsystem.

DSS supports two types of display interfaces:

- Display parallel interface via DISPC Video Port 2 (VP2) output. For more information, see Section 12.9.1.2.1
- Two low-voltage differential signaling (LVDS) interfaces, each with four data lanes and one clock lane, via Open LDI Transmitters (OLDITX0 and OLDITX1) connected to DISPC Video Port 1 (VP1) output. For more information, see Section 12.9.1.2.2.

For more information on video ports configuration, see Section 12.9.1.4.1.10, *DISPC Video Port Outputs*.

Figure 12-491 is a diagram of the DSS external connections.

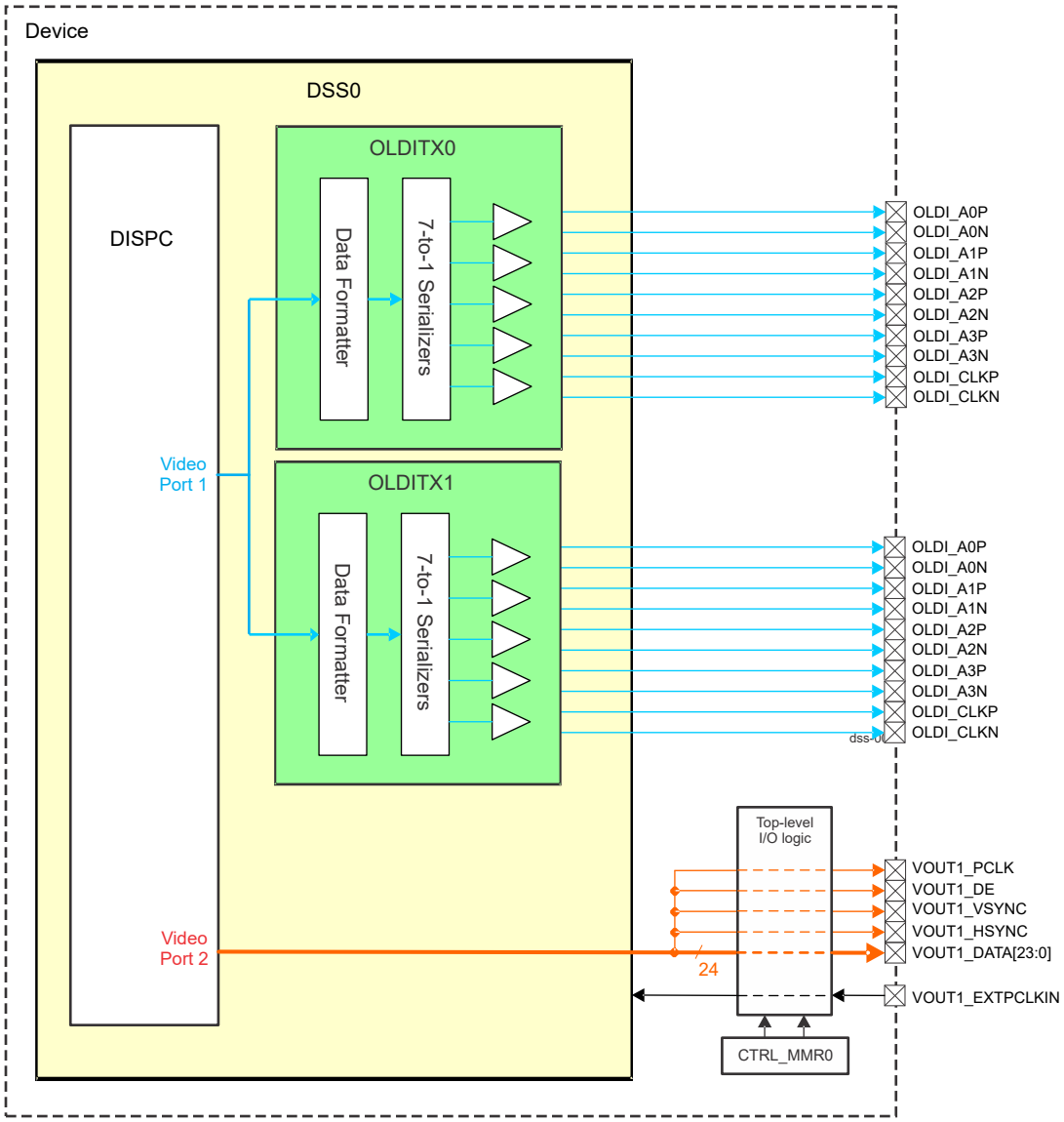


Figure 12-491. DSS Environment

Copyright © 2026 Texas Instruments Incorporated

Note

For more information about device level signals (pull-up/down resistors, buffer type, multiplexing and others), see tables *Pin Attributes* and *Pin Multiplexing* in the device-specific Datasheet.

12.9.1.2.1 DSS Parallel Interface

The DISPC Video Port 2 (VP2) outputs the required data and control signals to device pads to support the following display interface modes:

- Parallel MIPI DPI 2.0 (Digital Pixel Interface): RGB 16/18/24-bit output with separate sync signals.
- BT.656/BT.1120 interface: YUV422 output (8/10-bit modes) with embedded syncs.

Table 12-461 describes the DISPC VP2 output signals.

Table 12-461. DSS Signals for MIPI DPI 2.0 or BT.656/BT.1120

Module Pin	Device Level Signal	Type(1)	Description	Module Pin Reset Value
DSS_DPI2_DATA[23:0]	VOUT1_DATA[23:0]	O	Pixel data output. RGB data for MIPI DPI 2.0 interface. YUV data for BT.656/BT.1120 interfaces.	0
DSS_DPI2_PCLK	VOUT1_PCLK	O	Pixel clock output. The maximum interface frequency is 165MHz.	0
DSS_DPI2_VSYNC	VOUT1_VSYNC	O	Vertical synchronization. The frame synchronization pulse (vsync) toggles after all the lines in a frame are transmitted and a programmable number of line clock cycles has elapsed at the beginning and the end of each frame.	0
DSS_DPI2_HSYNC	VOUT1_HSYNC	O	Horizontal synchronization. The line synchronization pulse (hsync) toggles after all pixels in a line are transmitted and a programmable number of pixel clock wait-states has elapsed at the beginning and the end of each line.	0
DSS_DPI2_DE	VOUT1_DE	O	Pixel data output-enable signal to indicate when data must be latched using the pixel clock.	0

(1) I = Input, O = Output, I/O = Input/Output

Note

The effective output pixel clock rate for interleaved data formats (that is, BT.656 output mode or TDM (Time Division Multiplexed) output mode) will be either 1/2 or 1/3 of the maximum pixel clock rate, respectively, depending on the interleaving ratio.

12.9.1.2.1.1 Pixel Data Formats

This section describes the pixel data bus for RGB formats and shows timing diagrams of transactions and synchronizations.

For the active matrix display type, one pixel per pixel clock is displayed. The diagrams represent the configuration of assertion of the data on the rising edge of the pixel clock. It is possible to program the interface timing to output the data on the falling edge of the pixel clock.

Figure 12-492 through Figure 12-495 show the interface to 12-, 16-, 18-, and 24-bit RGB active matrix displays. Each vertical line represents one output pixel. The width of the data bus can be configured through DSS_VP1_CONTROL[10-8] DATALINES register bitfield.

Copyright © 2026 Texas Instruments Incorporated

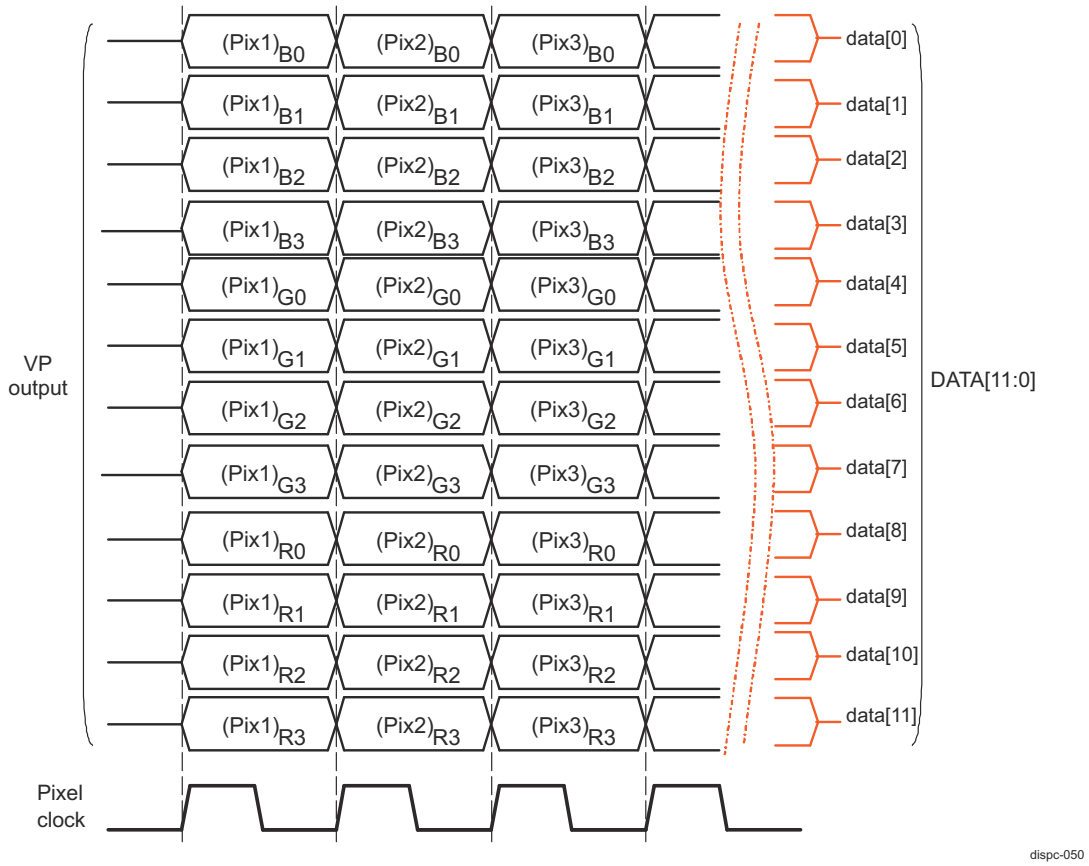


Figure 12-492. DISPC Video Port Pixel Data - 12-bit RGB Active Matrix

Copyright © 2026 Texas Instruments Incorporated

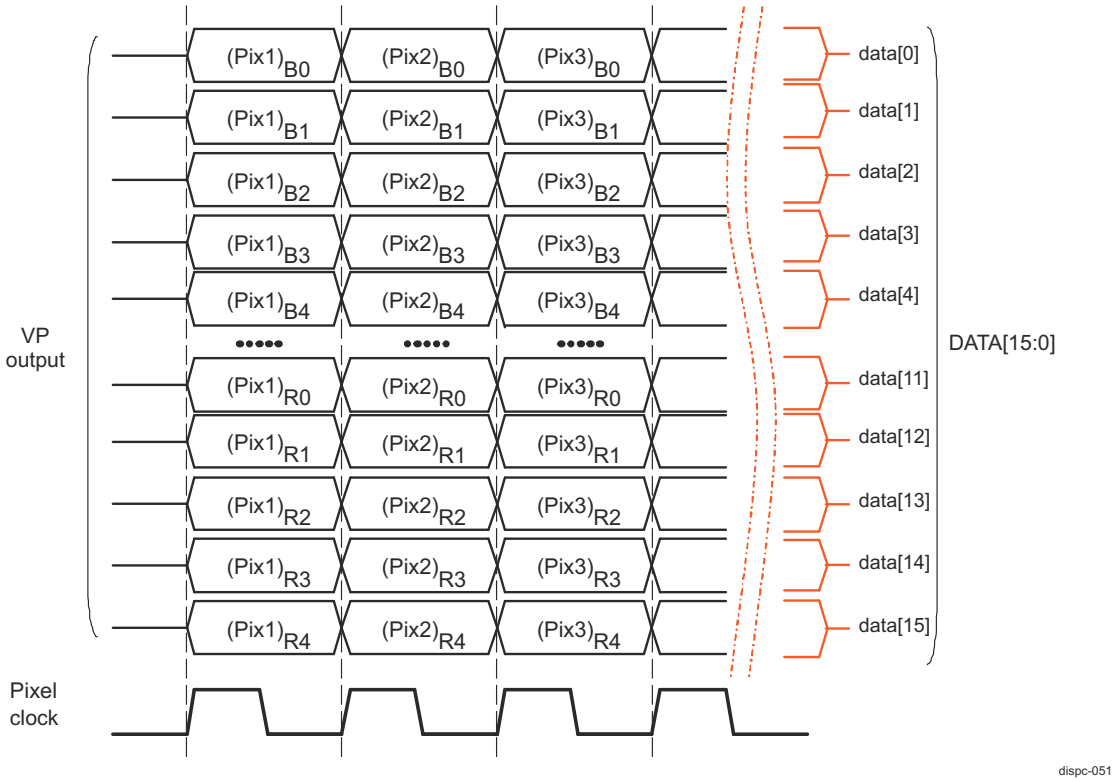


Figure 12-493. DISPC Video Port Pixel Data - 16-bit RGB Active Matrix

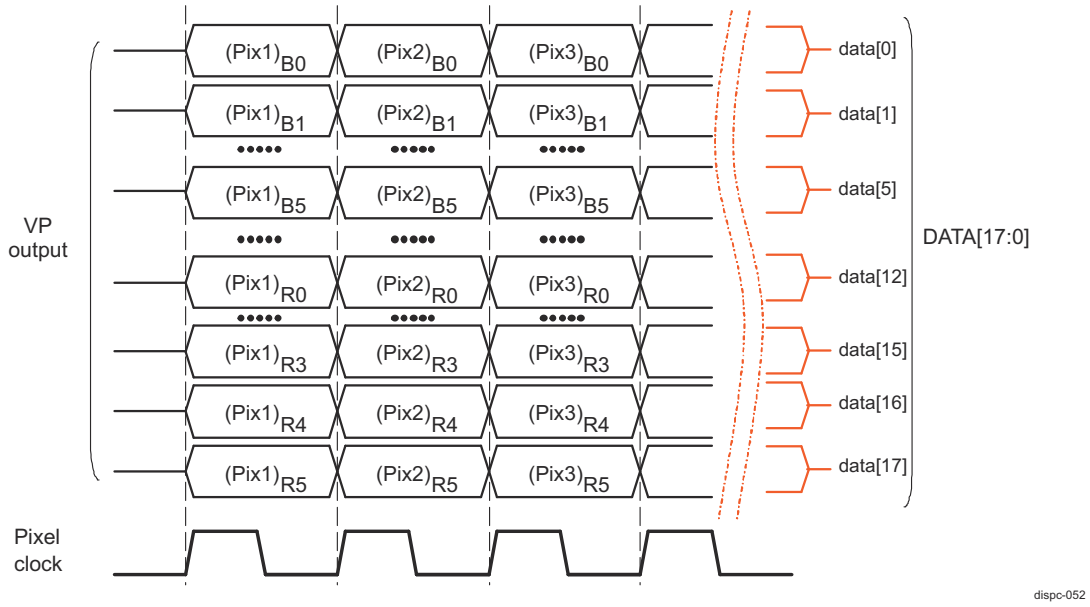


Figure 12-494. DISPC Video Port Pixel Data - 18-bit RGB Active Matrix

Copyright © 2026 Texas Instruments Incorporated

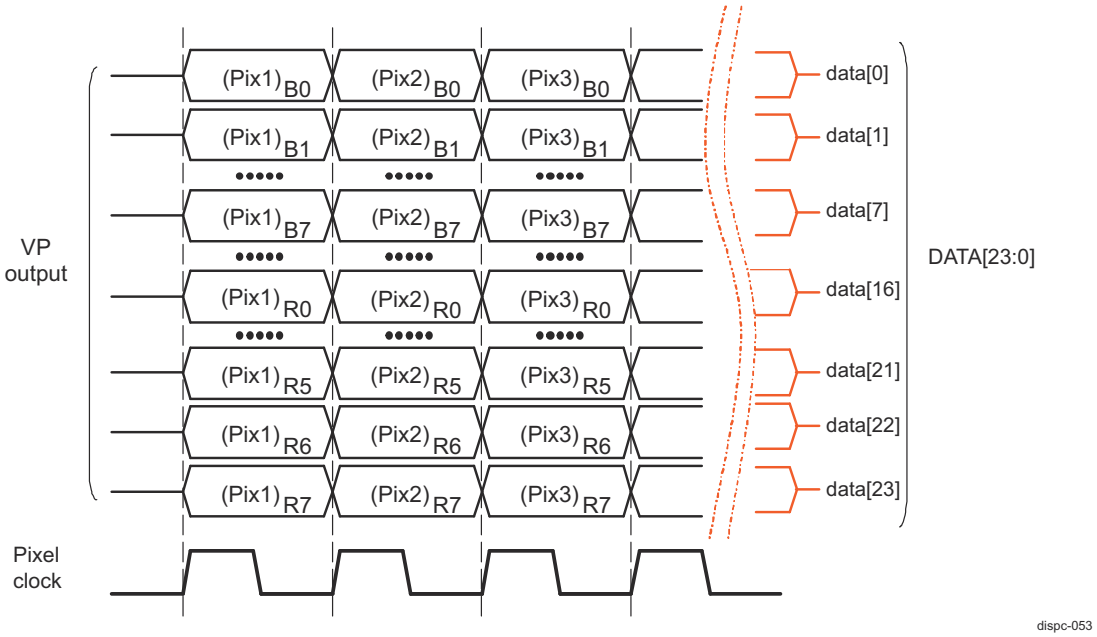


Figure 12-495. DISPC Video Port Pixel Data - 24-bit RGB Active Matrix

12.9.1.2.1.2 Display Timing Diagrams

Figure 12-496 through Figure 12-499 show examples with timing diagrams of synchronization signals and pixel clocks for active matrix panels. The DISPC video ports directly drive these signals, which are related to the programmable fields listed in Table 12-462. For more information, see also Section 12.9.1.4.1.10.7, *DISPC VP Timing Generator and Display Panel Settings*.

Table 12-462. DISPC Video Port Register Fields for Active Matrix Display

Name	Register	Description
PPL	DSS_VP1_SIZE_SCREEN[11-0] PPL value + 1	Pixels per line
LPP	DSS_VP1_SIZE_SCREEN[27-16] LPP value + 1	Lines per panel
HBP	DSS_VP1_TIMING_H[31-20] HBP value + 1	Horizontal back porch
HFP	DSS_VP1_TIMING_H[19-8] HFP value + 1	Horizontal front porch
HSW	DSS_VP1_TIMING_H[7-0] HSW value + 1	Horizontal synchronization pulse width
VBP	DSS_VP1_TIMING_V[31-20] VBP value	Vertical back porch
VFP	DSS_VP1_TIMING_V[19-8] VFP value	Vertical front porch
VSW	DSS_VP1_TIMING_V[7-0] VSW value + 1	Vertical synchronization pulse width
ALIGN	DSS_VP1_POL_FREQ[18] ALIGN	Alignment between HSYNC and VSYNC assertion
ONOFF	DSS_VP1_POL_FREQ[17] ONOFF	HSYNC and VSYNC pixel clock control
RF	DSS_VP1_POL_FREQ[16] RF	HSYNC and VSYNC pixel clock edge control
IEO	DSS_VP1_POL_FREQ[15] IEO	Invert output enable
IPC	DSS_VP1_POL_FREQ[14] IPC	Invert PCLK
IHS	DSS_VP1_POL_FREQ[13] IHS	Invert HSYNC
IVS	DSS_VP1_POL_FREQ[12] IVS	Invert VSYNC

- Active matrix timing configuration 1:
  - DSS_VP1_POL_FREQ[17] ONOFF = 0
  - DSS_VP1_POL_FREQ[16] RF = 0

    The HSYNC and VSYNC signals are driven on the opposite edge of PCLK from the pixel data.
  - DSS_VP1_POL_FREQ[15] IEO = 0

Copyright © 2026 Texas Instruments Incorporated

The DE signal is active high.

– DSS_VP1_POL_FREQ[14] IPC = 0

The pixel data are driven on the rising edge of PCLK.

– DSS_VP1_POL_FREQ[13] IHS = 0

The HSYNC signal is active high.

– DSS_VP1_POL_FREQ[12] IVS = 0

The VSYNC signal is active high.

– DSS_VP1_POL_FREQ[18] ALIGN = 0

The VSYNC and HSYNC assertion is not aligned.

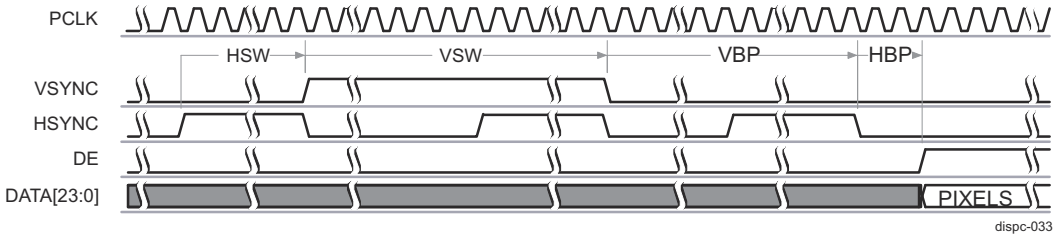


Figure 12-496. DISPC Display Timing Diagram of Configuration 1 (Start of Frame)

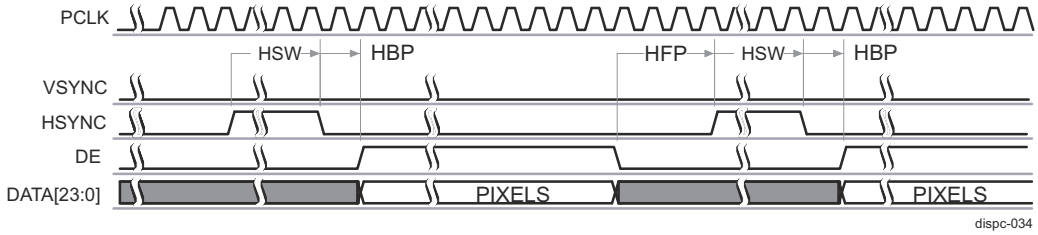


Figure 12-497. DISPC Display Timing Diagram of Configuration 1 (Between Lines)

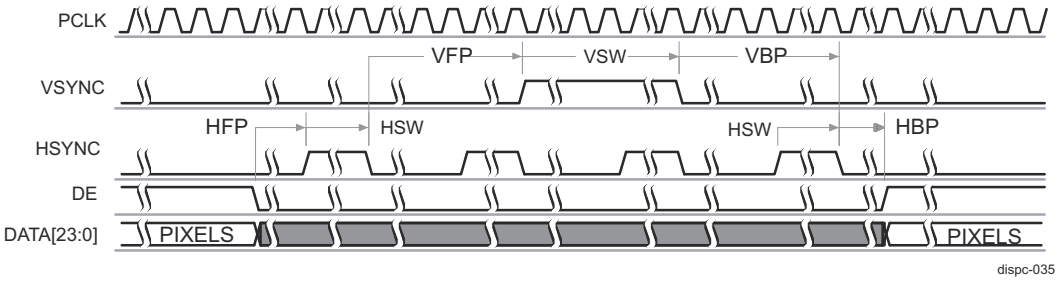


Figure 12-498. DISPC Display Timing Diagram of Configuration 1 (Between Frames)

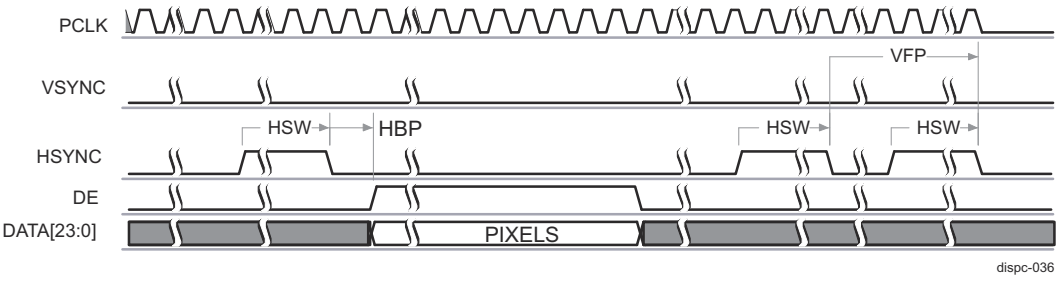


Figure 12-499. DISPC Display Timing Diagram of Configuration 1 (End of Frame)

- Active matrix timing configuration 2:

Copyright © 2026 Texas Instruments Incorporated

– DSS_VP1_POL_FREQ[17] ONOFF = 1
– DSS_VP1_POL_FREQ[16] RF = 1

The HSYNC and VSYNC signals are driven on the rising edge of PCLK.

– DSS_VP1_POL_FREQ[15] IEO = 1

The DE signal is active low.

– DSS_VP1_POL_FREQ[14] IPC = 1

The pixel data is driven on the falling edge of PCLK.

– DSS_VP1_POL_FREQ[13] IHS = 1

The HSYNC signal is active low.

– DSS_VP1_POL_FREQ[12] IVS = 1

The VSYNC signal is active low.

– DSS_VP1_POL_FREQ[18] ALIGN = 0

The VSYNC and HSYNC assertion is not aligned.

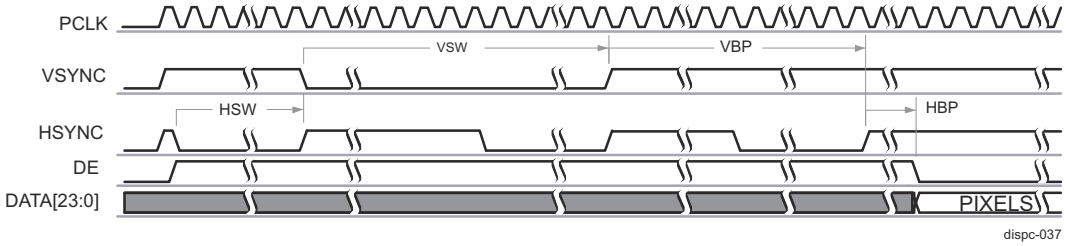


Figure 12-500. DISPC Display Timing Diagram of Configuration 2 (Start of Frame)

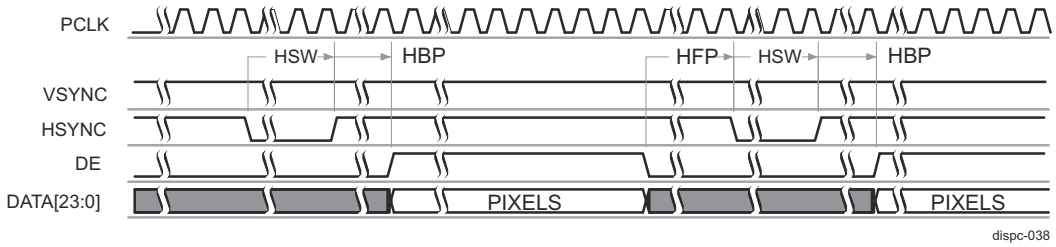


Figure 12-501. DISPC Display Timing Diagram of Configuration 2 (Between Lines)

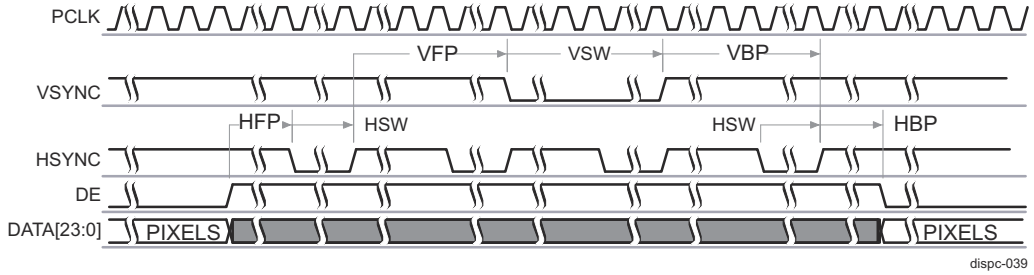


Figure 12-502. DISPC Display Timing Diagram of Configuration 2 (Between Frames)

Copyright © 2026 Texas Instruments Incorporated

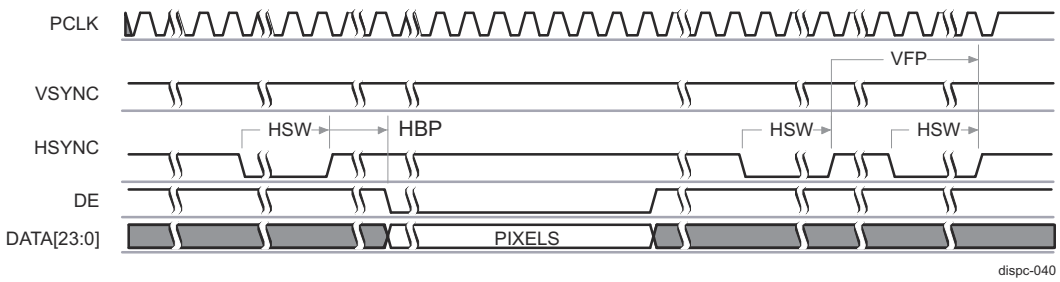


Figure 12-503. DISPC Display Timing Diagram of Configuration 2 (End of Frame)

- Active matrix timing configuration 3:
  - DSS_VP1_POL_FREQ[17] ONOFF = 1
  - DSS_VP1_POL_FREQ[16] RF = 1

    The HSYNC and VSYNC signals are driven on the rising edge of PCLK.
  - DSS_VP1_POL_FREQ[15] IEO = 0

    The DE signal is active high.
  - DSS_VP1_POL_FREQ[14] IPC = 0

    The pixel data are driven on the rising edge of PCLK.
  - DSS_VP1_POL_FREQ[13] IHS = 0

    The HSYNC signal is active high.
  - DSS_VP1_POL_FREQ[12] IVS = 0

    The VSYNC signal is active high.
  - DSS_VP1_POL_FREQ[18] ALIGN = 0

    The VSYNC and HSYNC assertion is not aligned.

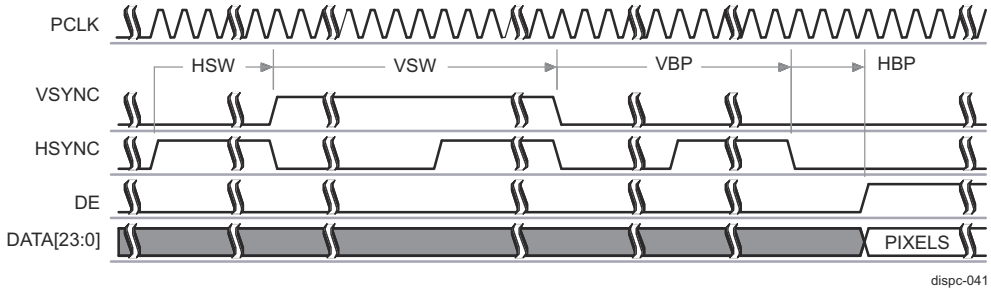


Figure 12-504. DISPC Display Timing Diagram of Configuration 3 (Start of Frame)

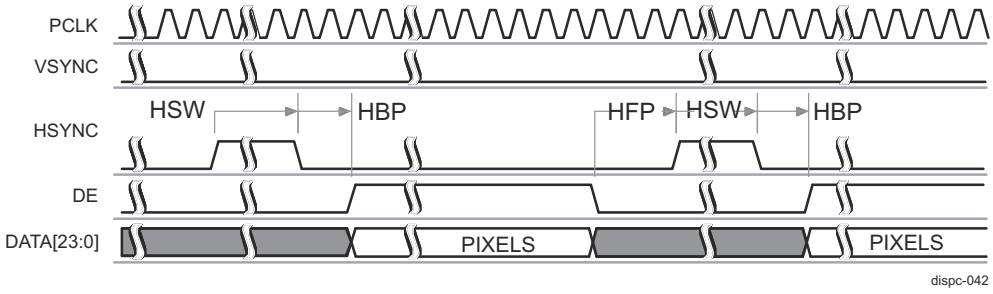


Figure 12-505. DISPC Display Timing Diagram of Configuration 3 (Between Lines)

Copyright © 2026 Texas Instruments Incorporated

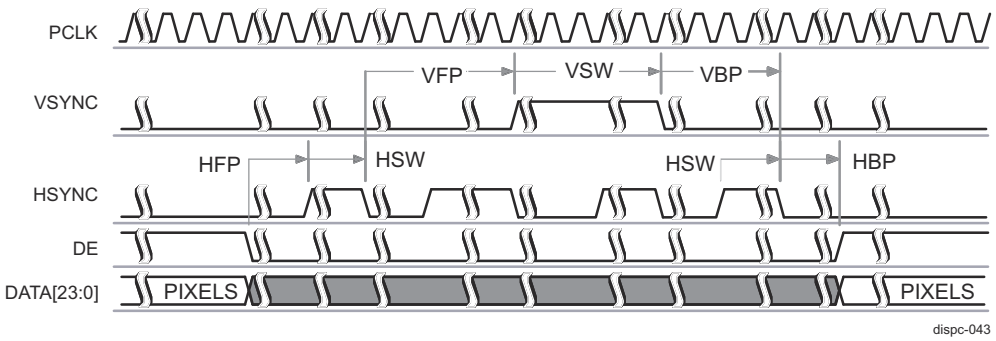


Figure 12-506. DISPC Display Timing Diagram of Configuration 3 (Between Frames)

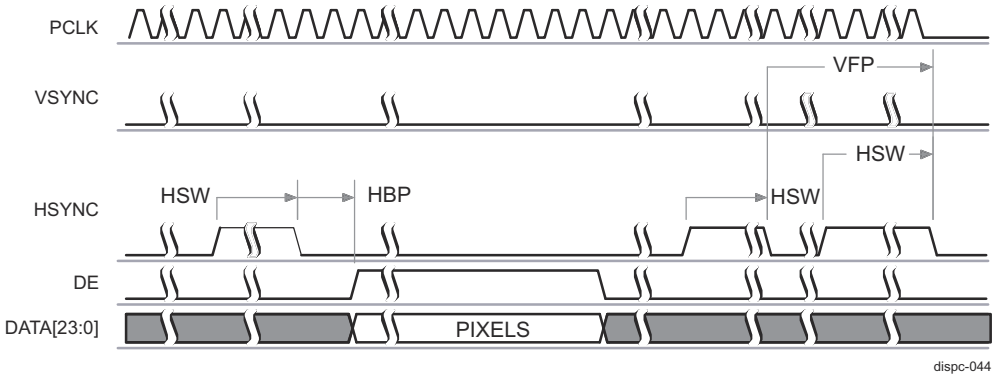


Figure 12-507. DISPC Display Timing Diagram of Configuration 3 (End of Frame)

12.9.1.2.2 DSS LVDS Interface

The LVDS display interface is provided via two single-link OLDITX modules, which receives parallel RGB pixel data and synchronization signals from DSS DISPC Video Port 1 (VP1). The VP1 pixel data format and timings are the same as described for VP2 in Section 12.9.1.2.1.1 and Section 12.9.1.2.1.2, respectively. OLDITX works only with 18-bit or 24-bit RGB input source data.

The OLDITX translates 21-bit or 28-bit wide video and sync data into 3-bit or 4-bit wide serial data, 7-bits deep. The 24-bit serialized RGB pixel data is transmitted on four LVDS data lanes, clocked by one clock lane. OLDITX can also be configured to transmit only 18-bit RGB data (6-bits/pixel color component) on three LVDS lanes, when connected to a low-resolution display panel. In 18-bit configuration, only three LVDS data lanes and one clock lane are active. The fourth data lane is disabled.

Table 12-463 describes the OLDITX output LVDS signals.

Table 12-463. DSS OLDITX Signals for LVDS Interface

Module Pin	Device Level Signal	Type(1)	Description	Module Pin Reset Value
OLDI_DATA0X	OLDI_A0P	O	OLDI differential data lane A0 (+)	-
OLDI_DATA0Y	OLDI_A0N	O	OLDI differential data lane A0 (-)	-
OLDI_DATA1X	OLDI_A1P	O	OLDI differential data lane A1 (+)	-
OLDI_DATA1Y	OLDI_A1N	O	OLDI differential data lane A1 (-)	-
OLDI_DATA2X	OLDI_A2P	O	OLDI differential data lane A2 (+)	-
OLDI_DATA2Y	OLDI_A2N	O	OLDI differential data lane A2 (-)	-
OLDI_DATA3X	OLDI_A3P	O	OLDI differential data lane A3 (+)	-
OLDI_DATA3Y	OLDI_A3N	O	OLDI differential data lane A3 (-)	-
OLDI_CLOCKX	OLDI_CLKP	O	OLDI differential clock lane CLK (+)	-

Copyright © 2026 Texas Instruments Incorporated

Table 12-463. DSS OLDITX Signals for LVDS Interface (continued)

Module Pin	Device Level Signal	Type(1)	Description	Module Pin Reset Value
OLDI_CLOCKY	OLDI_CLKN	O	OLDI differential clock lane CLK (-)	-

(1) I = Input, O = Output, I/O = Input/Output

Table 12-464 shows the supported LVDS pixel components mapping for 18-bit and 24-bit output data modes. The OLDI mapping type is defined in DSS0_DSS_OLDI_CFG[3-1] MAP register field as follows:

- Type A = Single-link 18-bit. MAP = '000'.
- Type B = Single-link 24-bit JEIDA. MAP = '001'.
- Type C = Single-link 24-bit. MAP = '010'.
- Type D = Dual-link 18-bit. MAP = '100'.
- Type E = Dual-link 24-bit JEIDA. MAP = '101'.
- Type F = Dual-link 24-bit. MAP = '110'.

Table 12-464. DSS OLDITX LVDS Data Format

Mode	Single-link	Single-link	Single-link	Dual-link	Dual-link	Dual-link
	18-bit	24-bit JEIDA	24-bit	18-bit	24-bit JEIDA	24-bit
OLDI mapping type	A	B	C	D	E	F
Pixels per LVDS clock cycle	1pixel/1clock	1pixel/1clock	1pixel/1clock	2pixel/1clock	2pixel/1clock	2pixel/1clock
Number of clock pairs	1	1	1	1or2	1or2	1or2
Number of LVDS data pairs	3	4	4	6	8	8
Color depth	6	8	8	6	8	8
A00	R0	R2	R0	OR0	OR2	OR0
A01	R1	R3	R1	OR1	OR3	OR1
A02	R2	R4	R2	OR2	OR4	OR2
A03	R3	R5	R3	OR3	OR5	OR3
A04	R4	R6	R4	OR4	OR6	OR4
A05	R5	R7	R5	OR5	OR7	OR5
A06	G0	G2	G0	OG0	OG2	OG0
A10	G1	G3	G1	OG1	OG3	OG1
A11	G2	G4	G2	OG2	OG4	OG2
A12	G3	G5	G3	OG3	OG5	OG3
A13	G4	G6	G4	OG4	OG6	OG4
A14	G5	G7	G5	OG5	OG7	OG5
A15	B0	B2	B0	OB0	OB2	OB0
A16	B1	B3	B1	OB1	OB3	OB1
A20	B2	B4	B2	OB2	OB4	OB2
A21	B3	B5	B3	OB3	OB5	OB3
A22	B4	B6	B4	OB4	OB6	OB4
A23	B5	B7	B5	OB5	OB7	OB5
A24	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC
A25	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC
A26	DEN	DEN	DEN	DEN	DEN	DEN
A30	See (1)	R0	R6	See (1)	OR0	OR6
A31	---	R1	R7	---	OR1	OR7
A32	---	G0	G6	---	OG0	OG6
A33	---	G1	G7	---	OG1	OG7

Copyright © 2026 Texas Instruments Incorporated

Table 12-464. DSS OLDITX LVDS Data Format (continued)

Mode	Single-link	Single-link	Single-link	Dual-link	Dual-link	Dual-link
A34	---	B0	B6	---	OB0	OB6
A35	---	B1	B7	---	OB1	OB7
A36	---	NA	NA	---	NA	NA
A40	See (2)	See (2)	See (2)	ER0	ER2	ER0
A41	---	---	---	ER1	ER3	ER1
A42	---	---	---	ER2	ER4	ER2
A43	---	---	---	ER3	ER5	ER3
A44	---	---	---	ER4	ER6	ER4
A45	---	---	---	ER5	ER7	ER5
A46	---	---	---	EG0	EG2	EG0
A50	---	---	---	EG1	EG3	EG1
A51	---	---	---	EG2	EG4	EG2
A52	---	---	---	EG3	EG5	EG3
A53	---	---	---	EG4	EG6	EG4
A54	---	---	---	EG5	EG7	EG5
A55	---	---	---	EB0	EB2	EB0
A56	---	---	---	EB1	EB3	EB1
A60	---	---	---	EB2	EB4	EB2
A61	---	---	---	EB3	EB5	EB3
A62	---	---	---	EB4	EB6	EB4
A63	---	---	---	EB5	EB7	EB5
A64	---	---	---	NA	NA	NA
A65	---	---	---	NA	NA	NA
A66	---	---	---	DEN	DEN	DEN
A70	See (1)	---	---	See (1)	ER0	ER6
A71	---	---	---	---	ER1	ER7
A72	---	---	---	---	EG0	EG6
A73	---	---	---	---	EG1	EG7
A74	---	---	---	---	EB0	EB6
A75	---	---	---	---	EB1	EB7
A76	---	---	---	---	NA	NA

(1) Not used - output disabled.
(2) Reserved for other video channel (Single-Link Mode)

Figure 12-508 and Figure 12-509 show the LVDS data output bit format (that is, the pixel bit numbers onto LVDS interface bit numbers). The rising edge of the LVDS clock occurs two LVDS sub-symbols before the current cycle of data. The clock is composed of a 4 LVDS sub-symbol high time and a 3 LVDS sub-symbol low time.

Copyright © 2026 Texas Instruments Incorporated

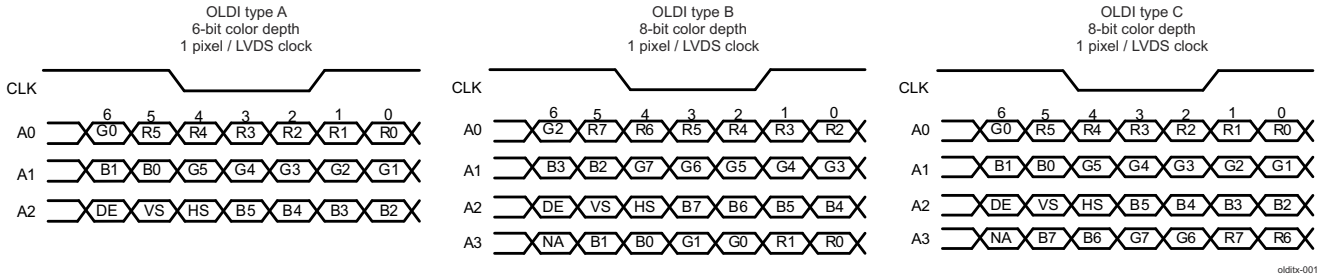


Figure 12-508. DSS OLDITX Pixel Bit Numbers on LVDS Output (Single-Link)

OLDI type D
6-bit color depth
2 pixel / LVDS clock

OLDI type E
8-bit color depth
2 pixel / LVDS clock

OLDI type F
8-bit color depth
2 pixel / LVDS clock

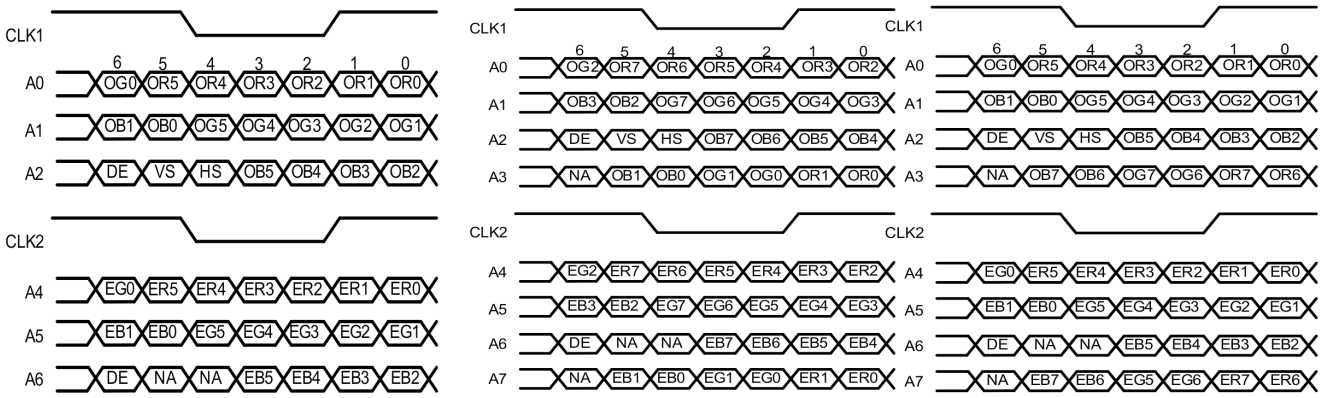


Figure 12-509. DSS OLDITX Pixel Bit Numbers on LVDS Output (Dual-Link)

Figure 12-510 and Figure 12-511 show the OLDITX input data mapped to LVDS output and the serial bit positions.

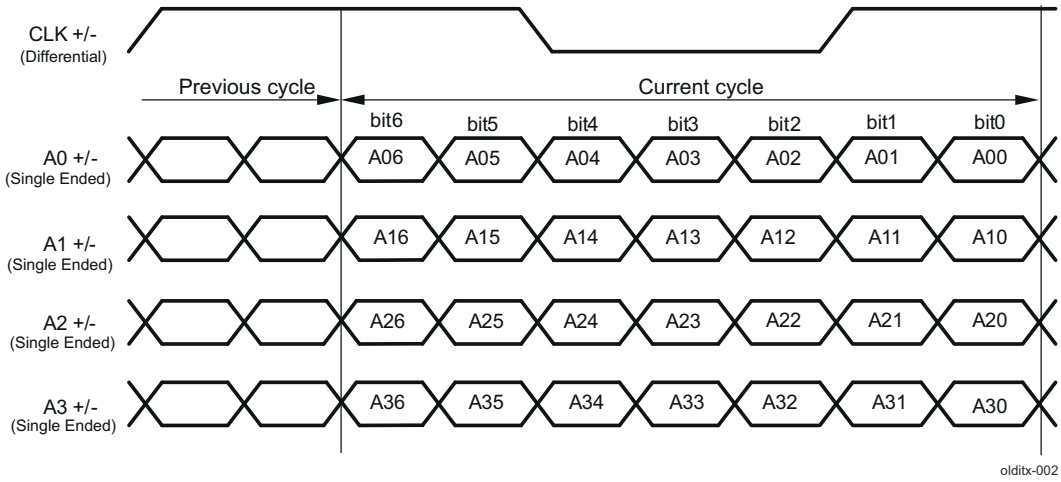


Figure 12-510. DSS OLDITX Inputs Mapped to LVDS Outputs

Copyright © 2026 Texas Instruments Incorporated

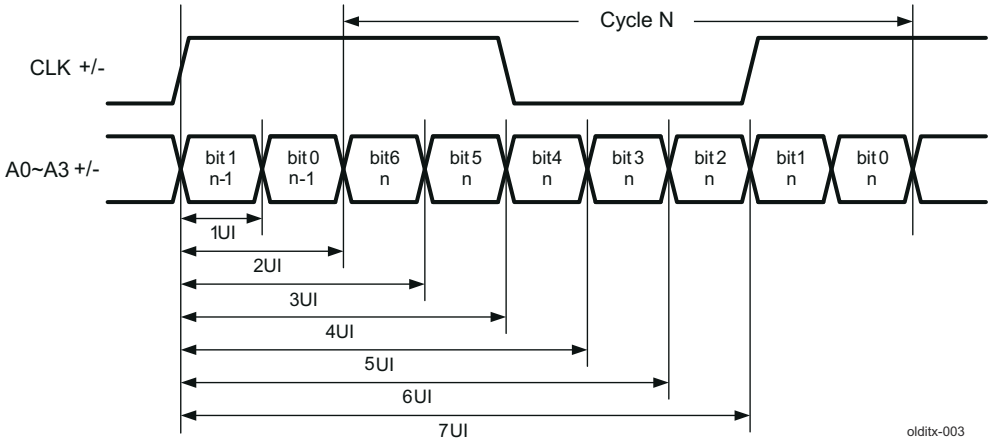


Figure 12-511. DSS OLDITX LVDS Serial Bit Positions

The OLDITX LVDS signal lines support a minimum bit time of 0.866 ns. This corresponds to a maximum pixel clock rate of 165 MHz (for a single link operation). A bit time period (known as TUI or Time Unit Interval) consists of several timing parameters:

- TCCS (Transmitter Channel-to-Channel Skew) - refers to the max skew across the differential data pairs within a link (in a single-link mode).
- SW - setup and hold time (internal data sample window) in the receiver.
- RSKM (Receiver Input Skew Margin) - the total time margin that remains after subtracting the sampling window (SW) size and the transmitter channel-to-channel skew (TCCS) from the time unit interval (TUI).

For more information on the OLDITX LVDS timing parameters, refer to device-specific Datasheet.

12.9.1.3 Integration

See the *Module Integration* section for information about clocks, resets and hardware requests.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4 DSS Functional Description

12.9.1.4.1 DISPC Functional Description

12.9.1.4.1.1 DISPC Overview

Figure 12-512 is a simplified block diagram of DISPC.

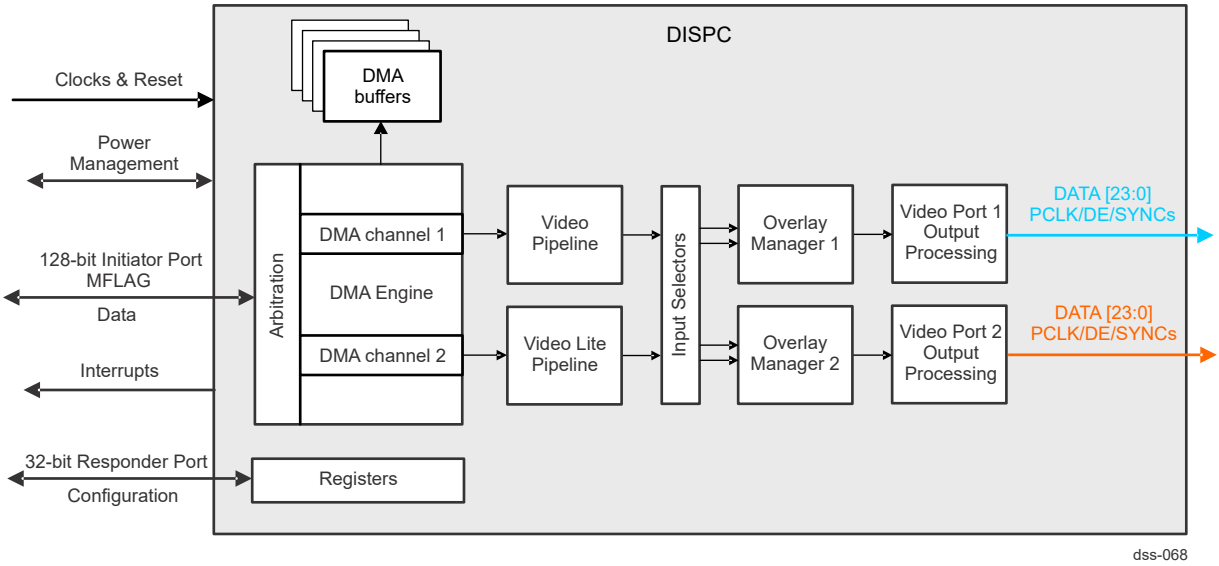


Figure 12-512. DISPC Architecture Overview

The DISPC integrated within DSS is capable of fetching pixel data from the device system memory through its single initiator port, performing various pixel processing, and then providing the processed pixels to an external display panel. The internal DMA engine tightly coupled to the DISPC is used for the pixel data transfer from system memory (frame buffer). The DISPC DMA engine is in charge of scheduling the memory requests. Several processes are configurable in order to manage the video pipeline features (color space conversion, up-sampling, down-sampling) and overlay features. The internal timing generator logic generates the video port output signals based on VESA DMT and CEA-861 standards. The DISPC video port output can be connected to display panels either directly (for MIPI DPI 2.0 or BT.656/BT.1120 support), or through Open LDI transmitter.

Note

DISPC does not support any tiled frame buffer, nor any compressed frame buffer. DISPC has no internal capability to support rotation of the frame buffer.

Note

The display resolution is programmable and can be any width in the range [1:4096] pixels. The following limitations apply, related to the type of display or the processing done:

- Active Matrix screen + dithering forces a width multiple of 2 pixels
- Active Matrix + TDM may force a width multiple of 2 pixels

The display buffers in the system memory must consist of contiguous pixels.

12.9.1.4.1.2 DISPC Clocks

DISPC has one clock domain for its internal logic and separate domains for each video port output.

The DISPC functional clock (DSS_FUNC_CLK) serves as the internal logic clock and also acts as the interface clock for the DISPC initiator and responder ports to system interconnect. There is no internal divisor on this clock.

Copyright © 2026 Texas Instruments Incorporated

The DISPC pixel clocks (DPI_x_IN_CLK) serve as the clocks for the DISPC video port outputs to OLDITX0 and OLDITX1 modules (VP1 pixel clock DPI_0_IN_CLK) or for the parallel display interface (VP2 pixel clock DPI_1_IN_CLK). There are no internal divisors on the pixel clocks.

The frequency of DSS_FUNC_CLK clock must be greater or equal to the DPI_x_IN_CLK clocks, in order to get the DISPC internal logic to function properly. The frequency of the DPI_x_IN_CLK clocks depend on the output display resolution and required frame rate. For the maximum supported frequency ratings, refer to device-specific Datasheet.

All input clocks are asynchronous to each other. They can be generated by different sources.

The DSS_COMMON_DSS_SYSCONFIG[0] AUTOCLKGATING register bit is set by default to allow the auto-gating of the interface and functional clocks. The AUTOCLKGATING bit can be reset to disable the auto-gating of the clocks, if required.

DISPC provides also a clock-gating control on sub-module level, via configuration of the DSS_COMMON_DISPC_CLKGATING_DISABLE register fields.

12.9.1.4.1.3 DISPC Resets

DISPC receives a single hardware reset signal. For more information, see *DSS Integration*.

To perform a software reset on the DISPC, set the DSS_COMMON_DSS_SYSCONFIG[1] SOFTRESET bit to 0x1. The DSS_COMMON_DSS_SYSSTATUS[0] DISPC_FUNC_RESETDONE bit indicates that the software reset is complete (for the DISPC internal logic) when its value is 0x1. When the software reset completes, the DSS_COMMON_DSS_SYSCONFIG[1] SOFTRESET bit is automatically reset. Software must ensure that the software reset completes before performing DISPC operations.

The completion of the software reset for the video ports logic is indicated in the DSS_COMMON_DSS_SYSSTATUS[3-1] DISPC_VP_RESETDONE register bit-field.

12.9.1.4.1.4 DISPC Power Management

DISPC supports a power management protocol with the device Power Sleep Controller (PSC).

The (software) sequence, while performing a *clkstop_req* to DISPC, is as follows:

1. Disable DISPC by programing the [0] ENABLE bit of DSS_VP1_CONTROL register to 0.
2. DISPC hardware completes the output of the current frame. Then hardware sets the DSS_COMMON_DSS_SYSSTATUS[9] DISPC_IDLE_STATUS register bit to 1.
3. Poll the DISPC_IDLE_STATUS bit to ensure that DISPC is in idle mode.
4. PSC initiates *clkstop_req* to DISPC.
5. DISPC hardware acknowledges with *clkstop_ack* immediately.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.5 DISPC Interrupt Requests

DISPC supports two active-high level sensitive interrupt output lines: DSS_INT0 and DSS_INT1. All DISPC sub-module interrupt events are fully mapped to both of these interrupts, as shown in Figure 12-513. There are two sets of IRQ aggregating registers, one set for each interrupt line, which enable fully independent monitoring and control of the interrupt events by two processor hosts at device level.

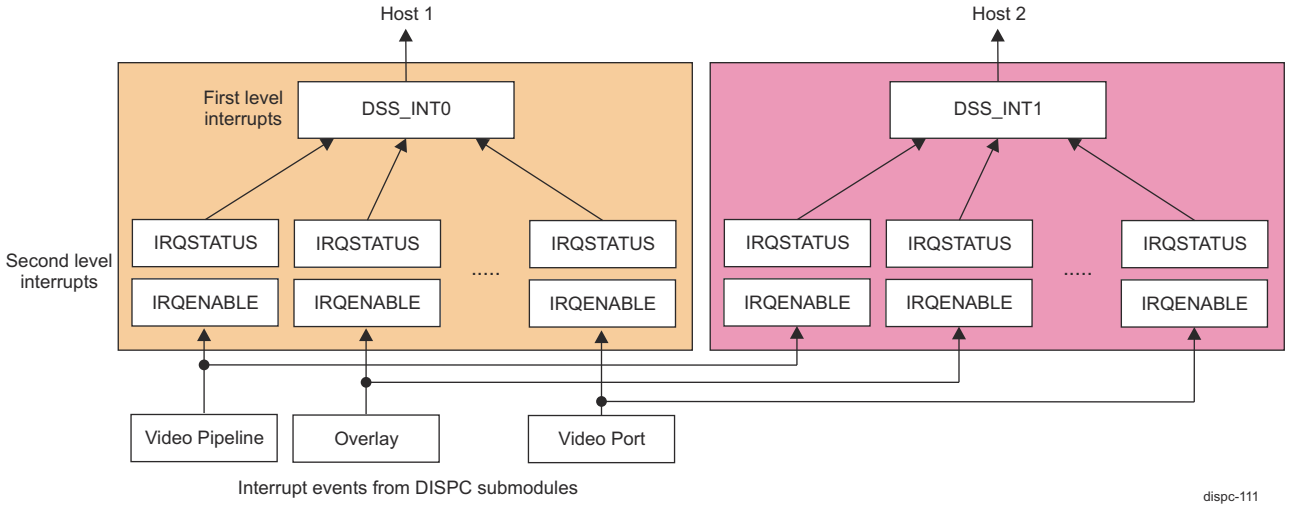


Figure 12-513. DISPC Interrupts Generation

Each of the interrupt signals indicates that one or more interrupt events are detected by the hardware. Each event is independently maskable for each interrupt output.

There are two level of interrupt events. The first level is used to indicate common events and is also source for the second level of interrupts. The second level of interrupt events consists of status and enable interrupt registers for each video pipeline and each video port.

Table 12-465 describes the first level of interrupt events with associated mask and status register fields. Each interrupt event is captured in an interrupt status register. Equivalent IRQSTATUS_RAW registers exist, which are updated even if interrupts are not enabled. This allows software to get access to updated status for all interrupt events.

Copyright © 2026 Texas Instruments Incorporated

Table 12-465. DISPC Interrupts - First Level

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_DISPC_IRQENABLE_SET	DSS_INT0 Interrupt Status DSS_COMMON_DISPC_IRQSTATUS	DSS_INT1 Interrupt Mask DSS_COMMON1_DISPC_IRQENABLE_SET	DSS_INT1 Interrupt Status DSS_COMMON1_DISPC_IRQSTATUS	Description
VID_IRQ	[4] SET_VID_IRQ	[4] VID_IRQ	[4] SET_VID_IRQ	[4] VID_IRQ	At least one event of the VID pipeline interrupt events has occurred. See Table 12-466 for more details.
VIDL_IRQ	[5] SET_VID_IRQ	[5] VID_IRQ	[5] SET_VID_IRQ	[5] VID_IRQ	At least one event of the VIDL1 pipeline interrupt events has occurred. See Table 12-467 for more details.
VP1_IRQ	[0] SET_VP_IRQ	[0] VP_IRQ	[0] SET_VP_IRQ	[0] VP_IRQ	At least one event of the VP1 interrupt events has occurred. See Table 12-468 for more details.
VP2_IRQ	[1] SET_VP_IRQ	[1] VP_IRQ	[1] SET_VP_IRQ	[1] VP_IRQ	At least one event of the VP2 interrupt events has occurred. See Table 12-469 for more details.

Table 12-466 describes the second level of interrupts for VID pipeline with associated mask and status register bits.

Table 12-466. DISPC Interrupts - Second Level - VID Pipeline

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VID_IRQENABLE_0	DSS_INT0 Interrupt Status DSS_COMMON_VID_IRQSTATUS_0	DSS_INT1 Interrupt Mask DSS_COMMON1_VID_IRQENABLE_0	DSS_INT1 Interrupt Status DSS_COMMON1_VID_IRQSTATUS_0	Description
VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	End of the video window: The DMA engine has fetched all the data from memory for the video for the current frame.

Copyright © 2026 Texas Instruments Incorporated

Table 12-466. DISPC Interrupts - Second Level - VID Pipeline (continued)

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VID_IRQENABLE_0	DSS_INT0 Interrupt Status DSS_COMMON_VID_IRQSTATUS_0	DSS_INT1 Interrupt Mask DSS_COMMON1_VID_IRQENABLE_0	DSS_INT1 Interrupt Status DSS_COMMON1_VID_IRQSTATUS_0	Description
VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	Video DMA buffer underflow: The input video DMA buffer goes underflow. This does not necessary means that the buffer is empty (out of order refill), but simply that the required pixel is not in yet.
VIDSAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	Video output MISR signature mismatch OR Video output freeze frame detect. The MISR signature generated does not match the expected signature OR The Video output frame freeze detection has triggered.

Table 12-467 describes the second level of interrupts for VIDL pipeline with associated mask and status register bits.

Table 12-467. DISPC Interrupts - Second Level - VIDL Pipeline

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VID_IRQENABLE_1	DSS_INT0 Interrupt Status DSS_COMMON_VID_IRQSTATUS_1	DSS_INT1 Interrupt Mask DSS_COMMON1_VID_IRQENABLE_1	DSS_INT1 Interrupt Status DSS_COMMON1_VID_IRQSTATUS_1	Description
VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	[1] VIDENDWINDOW_EN	[1] VIDENDWINDOW_IRQ	End of the video window: The DMA engine has fetched all the data from memory for the video for the current frame.

Copyright © 2026 Texas Instruments Incorporated

Table 12-467. DISPC Interrupts - Second Level - VIDL Pipeline (continued)

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VID_IRQENABLE_1	DSS_INT0 Interrupt Status DSS_COMMON_VID_IRQSTATUS_1	DSS_INT1 Interrupt Mask DSS_COMMON1_VID_IRQENABLE_1	DSS_INT1 Interrupt Status DSS_COMMON1_VID_IRQSTATUS_1	Description
VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	[0] VIDBUFFERUNDERFLOW_EN	[0] VIDBUFFERUNDERFLOW_IRQ	Video DMA buffer underflow: The input video DMA buffer goes underflow. This does not necessary means that the buffer is empty (out of order refill), but simply that the required pixel is not in yet.
VIDSAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	[2] SAFETYREGION_EN	[2] SAFETYREGION_IRQ	Video output MISR signature mismatch, or video output freeze frame detect. The MISR signature generated does not match the expected signature, or the video output frame freeze detection has triggered.

Table 12-468 describes the second level of interrupts for VP1 output with associated mask and status register bits.

Table 12-468. DISPC Interrupts - Second Level - VP1 Output

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VP_IRQENABLE_0	DSS_INT0 Interrupt Status DSS_COMMON_VP_IRQSTATUS_0	DSS_INT1 Interrupt Mask DSS_COMMON1_VP_IRQENABLE_0	DSS_INT1 Interrupt Status DSS_COMMON1_VP_IRQSTATUS_0	Description
VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	Sync for VP1 output. Shadow to work copy of registers associated with VP1 has occurred. DSS_VP1_CONTROL[5] GO register bit is cleared.
SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	Security violation for VP1 output. A security violation (for example, a secure video pipeline connected to non-secure VP/OVR) has occurred.

Copyright © 2026 Texas Instruments Incorporated

Table 12-468. DISPC Interrupts - Second Level - VP1 Output (continued)

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VP_IRQENABLE_0	DSS_INT0 Interrupt Status DSS_COMMON_VP_IRQSTATUS_0	DSS_INT1 Interrupt Mask DSS_COMMON1_VP_IRQENABLE_0	DSS_INT1 Interrupt Status DSS_COMMON1_VP_IRQSTATUS_0	Description
FRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	Frame done for VP1 output. After disabling the VP1 output of the DISPC, the interrupt is set when the active frame related to the VP1 has completed.
VSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	VSYNC for VP1 output: VSYNC interrupt for the VP1 has occurred at the end of the frame.
VSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	VSYNC for odd field. VSYNC_ODD interrupt has occurred at the end of the frame (EVSYNC received and the field polarity is odd).
PROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	Programmed line number. The VP1 has reached the user-programmed line number.
SYNCLOST_IRQ	[4] VPSYNCLOST_EN	[4] VPSYNCLOST_IRQ	[4] VPSYNCLOST_EN	[4] VPSYNCLOST_IRQ	Synchronization lost on VP1 output: Occurs when VSYNC width/front or back porches are not wide enough to load the pipeline with data (VP1 output).
VPSAFETYREGION_IRQ	[9-6] SAFETYREGION_EN Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_EN Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	VP1 output MISR signature mismatch, or VP1 output freeze frame detect. The MISR signature generated does not match the expected signature, or the VP1 output frame freeze detection has triggered.

Table 12-469 describes the second level of interrupts for VP2 output with associated mask and status register bits.

Copyright © 2026 Texas Instruments Incorporated

Table 12-469. DISPC Interrupts - Second Level - VP2 Output

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VP_IRQENABLE_1	DSS_INT0 Interrupt Status DSS_COMMON_VP_IRQSTATUS_1	DSS_INT1 Interrupt Mask DSS_COMMON1_VP_IRQENABLE_1	DSS_INT1 Interrupt Status DSS_COMMON1_VP_IRQSTATUS_1	Description
VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	[11] VPSYNC_EN	[11] VPSYNC_IRQ	Sync for VP2 output. Shadow to work copy of registers associated with VP2 has occurred. DSS_VP2_CONTROL[5] GO register bit is cleared.
SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	[10] SECURITYVIOLATION_EN	[10] SECURITYVIOLATION_IRQ	Security violation for VP2 output. A security violation (for example, a secure video pipeline connected to non-secure VP/OVR) has occurred.
FRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	[0] VPFRAMEDONE_EN	[0] VPFRAMEDONE_IRQ	Frame done for VP2 output. After disabling the VP2 output of the DISPC, the interrupt is set when the active frame related to the VP2 has completed.
VSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	[1] VPVSYNC_EN	[1] VPVSYNC_IRQ	VSYNC for VP2 output: VSYNC interrupt for the VP2 has occurred at the end of the frame.
VSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	[2] VPVSYNC_ODD_EN	[2] VPVSYNC_ODD_IRQ	VSYNC for odd field. VSYNC_ODD interrupt has occurred at the end of the frame (EVSYNC received and the field polarity is odd).
PROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	[3] VPPROGRAMMEDLINENUMBER_EN	[3] VPPROGRAMMEDLINENUMBER_IRQ	Programmed line number. The VP2 has reached the user-programmed line number.
SYNCLOST_IRQ	[4] VPSYNCLOST_EN	[4] VPSYNCLOST_IRQ	[4] VPSYNCLOST_EN	[4] VPSYNCLOST_IRQ	Synchronization lost on VP2 output: Occurs when VSYNC width/front or back porches are not wide enough to load the pipeline with data (VP2 output).

Copyright © 2026 Texas Instruments Incorporated

Table 12-469. DISPC Interrupts - Second Level - VP2 Output (continued)

Interrupt Name	DSS_INT0 Interrupt Mask DSS_COMMON_VP_IRQENABLE_1	DSS_INT0 Interrupt Status DSS_COMMON_VP_IRQSTATUS_1	DSS_INT1 Interrupt Mask DSS_COMMON1_VP_IRQENABLE_1	DSS_INT1 Interrupt Status DSS_COMMON1_VP_IRQSTATUS_1	Description
VPSAFETYREGION_IRQ	[9-6] SAFETYREGION_EN Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_EN Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	[9-6] SAFETYREGION_IRQ Bit [9] = Safety Region 3 Bit [8] = Safety Region 2 Bit [7] = Safety Region 1 Bit [6] = Safety Region 0	VP2 output MISR signature mismatch, or VP2 output freeze frame detect. The MISR signature generated does not match the expected signature, or the VP2 output frame freeze detection has triggered.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.6 DISPC DMA Engine

The DISPC DMA engine:

- Requests and supplies data from system memory to the VID and VIDL pipelines through the system interconnect, based on the configuration of the video pipeline settings.
- Is fully programmable and fetches pixel data via 128-bit requests using 1D burst.

Each pipeline has a dedicated DMA buffer and channel with independent settings. If a pipeline is disabled (that is, not used), then its DMA buffer can be assigned to another pipeline by configuring the DSS_COMMON_DISPC_GLOBAL_BUFFER register. For example, unused buffers for VIDL pipeline can be used by VID pipeline.

Each DMA channel supports a total of 8 line buffers, each of which can store 1280 32-bit pixels.

- The size of the DMA buffer associated to the video lite pipeline (VIDL1) is 8x320x128-bit.
- The size of the DMA buffer associated to the video pipeline (VID) is 8x320x128-bit.

The DMA engine fetches encoded pixels from the system memory only when the video layer is enabled (a valid configuration has been programmed for the video layer), that is, the video window is present and the video pipeline is active.

12.9.1.4.1.6.1 DISPC DMA Addressing and Bursts

For each line to be fetched, the DMA engine address generator:

- Calculates the pixel address
- Aligns the address
- Determines byte enable pattern
- Determines 1D burst length

The meaning of the address bits is as follows:

- Bits [31-0]: system (DDR) memory address (pixel address)
- Bits [37-32]: 32 bits + address extension

The address extension bits are defined by the programmable parameter DSS_VID_BA_EXT_0/ DSS_VID_BA_EXT_1 and DSS_VID_BA_UV_EXT_0/DSS_VID_BA_UV_EXT_1 registers, optionally used to extend the BA memory addressing to 48-bit addressed external memory space (that is, to extend DISPC address space into 4GB+ space).

The DDR scan pixel addresses are generated by the DMA engine in order to read data from the system memory. The base address defines the start address of the first pixel, and then the address is incremented based on the number of pixels per line, offset between two consecutive lines and number of lines. The DSS_VID_ROW_INC register allow the access to a frame using 1D bursts (but as a two-dimensional block) by adding a fixed address offset at the end of a line. The ROW_INC can also be used to skip lines from the input frame.

The byte address of each pixel in the frame buffer located in the system memory is determined by:

Pixel address = base_address + x × (bpp/8) + y × (width × (bpp/8) + increment), where:

- "base_address" corresponds to the base address defined by:
  - DSS_VID_BA_0/DSS_VID_BA_1[31-0] BA register bit-fields for all formats, and Y frame buffer in case of YUV-NV12 format. Optional extension bits in DSS_VID_BA_EXT_0/DSS_VID_BA_EXT_1 registers.
  - DSS_VID_BA_UV_0/DSS_VID_BA_UV_1[31-0] BA register bit-fields for UV frame buffer in case of YUV-NV12 format. Optional extension bits in DSS_VID_BA_UV_EXT0/DSS_VID_BA_UV_EXT_1 registers.
- "bpp" corresponds to the number of bits per pixel defined by the DSS_VID_ATTRIBUTES[6-1] FORMAT register bit-field.
- "width" corresponds to the number of pixels per line defined by the DSS_VID_PICTURE_SIZE[11-0] MEMSIZEX + 1 register bit-field.
- "increment" corresponds to the number of bytes to skip between two contiguous lines defined by the DSS_VID_ROW_INC[31-0] ROWINC – 1 register bit-field.
- "x" corresponds to the pixel position on the x-axis.
- "y" corresponds to the pixel position on the y-axis.

Copyright © 2026 Texas Instruments Incorporated

Note

Since the base address is aligned on pixel size boundary the horizontal resolution is one pixel. In case of YUV422 formats, the resolution is 4 bytes (2 pixels). In case of RGB24 packed format the resolution is 4 pixels. In case of Y frame buffer (YUV-NV12 format) the resolution is one byte. The vertical resolution is one line.

In case of YUV422 format, the number of pixels per line shall be a multiple of 2 pixels and the size of a pixel shall be considered as 2 bytes. In case of YUV420-NV12 or YUV420-NV21 format, the Y buffer shall be considered as an 8-bit frame buffer, and the CbCr shall be considered as a 16-bit frame buffer. The pixel size is 1 byte and 2 bytes, respectively for Y and CbCr.

For YUV420-NV12 or YUV420-NV21 format, the pixel values are defined in two separate buffers (Y and UV buffers). The first buffer consists of Y values (8 bits for each Y sample). The second buffer consists of CbCr values (16 bits for each pair of CbCr samples). The base address of the Y and UV buffers are as defined earlier in this section.

In case of interlaced mode, DSS_VID_BA_0 and DSS_VID_BA_UV_0 registers define the base address of the even field, and DSS_VID_BA_1 and DSS_VIDL1_BA_1 registers define the base address of the odd field.

The number of bytes to skip between pixels and between lines are defined using DSS_VID_PIXEL_INC and DSS_VID_ROW_INC registers, respectively. They define the values to be used for the Y buffer. For the CbCr buffer, the DSS_VID_ROW_INC_UV register define the values.

Table 12-470 summarizes the register settings for a simple access of a picture in the system memory.

Table 12-470. DISPC Register Settings for Accessing Image in Internal Memory

Video Pipeline Registers	Value
DSS_VID_BA_0 (1) and DSS_VID_BA_1 (2)	The physical base address (PBA) of image in the memory for all formats and Y buffer.
DSS_VID_BA_EXT_0 and DSS_VID_BA_EXT_1	Address extension bits of PBA for all formats and Y buffer.
DSS_VID_BA_UV_0 (1) and DSS_VID_BA_UV_1 (2)	The physical base address (PBA) of UV buffers image in the memory.
DSS_VID_BA_UV_EXT_0 and DSS_VID_BA_UV_EXT_1	Address extension bits of PBA for UV buffers.
DSS_VID_PIXEL_INC	1 or other in pixel incremental value.
DSS_VID_ROW_INC	1 or other in row incremental value. Used for Y buffer.
DSS_VID_ROW_INC_UV	1 or other in row incremental value. Used for UV buffer.

(1) Base address of even field, in case of interlaced mode.
(2) Base address of odd field, in case of interlaced mode.

An interconnect request (128 bits) corresponds to one or several pixels, depending on the bits per pixel. Therefore, the DMA engine determines the appropriate burst sequence to optimize the fetching of each new line. The DMA engine must prevent a single burst from crossing two lines. The DMA engine supports only 1D burst. 1D burst is used, if the fetch data is linear in memory. The size of the burst can be one of the following values: 1x128-bit, 2x128-bit, 4x128-bit, or 8x128-bit. Because the burst size must be aligned to the burst boundary, in case of misalignment, the DMA engine may issue one or more smaller burst requests.

12.9.1.4.1.6.2 DISPC Read DMA Buffers

When the vertical front porch (VFP) period starts after the last horizontal front porch (HFP) of the last line, the DMA buffers are flushed according to the video port output associated with the particular video pipeline. The DMA engine restarts fetching data from the memory through the DSS Initiator port. Enabling or disabling the DISPC flushes the DMA buffers.

Programmable high and low thresholds, independent for each DMA buffer, are used by the DMA engine to start and stop requesting data through the Initiator port.

Copyright © 2026 Texas Instruments Incorporated

- When low threshold (set in the DSS_VID_BUF_THRESHOLD [15-0] BUFLOWTHRESHOLD register bit-field) is reached, the DMA engine starts a request on the device interconnect to fill the DMA buffer.
- When high threshold (set in the DSS_VID_BUF_THRESHOLD [31-16] BUFHIGHTHRESHOLD register bit-field) is reached, the DMA engine stops requesting encoded pixels.

Note

The configuration of thresholds for optimal performance can be defined using the DSS_VID_BUF_SIZE_STATUS [15-0] BUFSIZE register field value, as follows:

- For high threshold: BUFSIZE (in number of 128-bit words) – 1
- For low threshold: BUFSIZE (in number of 128-bit words) – burst size (in number of 128-bit words)

The following limitations for BUFLOWTHRESHOLD values must be also considered:

- If the scaler in VID pipeline is disabled, BUFLOWTHRESHOLD can be programmed as low as interconnect latency and pixel output rate allow it.
- If the scaler in VID pipeline is enabled, BUFLOWTHRESHOLD must be programmed to guarantee that at least four full lines can be stored.

To avoid underflow at the beginning of a frame and have sufficient encoded pixel data to start some processing, a preloading of the DMA buffer is configurable between a fixed value of bytes and the high threshold value. The preload ensures a minimum number of pixels present in the buffer. When the preload value is reached, the associated channel will start pulling pixels out of the DMA buffer. To enable the preload based on the value entered in the DSS_VID_PRELOAD[11-0] PRELOAD register bit-field, the DSS_VID_ATTRIBUTES[19] BUFPRELOAD register bit must be set to 0x0.

The vertical blanking between two frames must be long enough to allow fetching the number of pixels defined by the DSS_VID_PRELOAD register and preloading the whole video pipeline. If the value set in the preload register is greater than some overflow conditions detected by the hardware, then data will start to be read from the video DMA buffer before the preload value is reached. If SYNCLOST_IRQ event occurs the video buffer needs to be increased (buffer merge). Preload value must be greater or equal to low threshold, and smaller or equal to high threshold value.

Note

When self-refresh mode is selected (which means that the data in the DMA buffer are used for multiple frames) the DMA buffers are not flushed at the end of each frame. Each DMA buffer has an independent control for selecting the self-refresh mode. For more information, see Section 12.9.1.4.1.6.8.2, *DISPC DMA Ultra-Low Power Mode*.

12.9.1.4.1.6.3 DISPC Flip/Mirror Support

DISPC supports on-the-fly source image flip along the x/y-axis for 8-bit/component formats (ARGB or YUV) to create a mirror effect on the source data. The DMA engine reads the source frame from right to left by requesting burst transfers for each line in negative address increments while performing each burst transfer as a linear incremental burst. The read data is repacked and stored in the line buffer incrementally - effectively storing the frame as a flipped image for the processing pipeline. The configuration of the flip/mirror operation is explained in Table 12-471.

Table 12-471. DISPC Flip/Mirror Configuration

Flip Direction	FLIP Bit	Base Address	Byte Increment
Along Y-axis (vertical mirror)	1	Start of window	1
Along X-axis (horizontal mirror)	0	Start of last line of the window	$-(2^{(\text{width of the line in bytes})})$
Along both X-axis and Y-axis (horizontal and vertical mirror)	1	Start of last line of the window	$-(2^{(\text{width of the line in bytes})})$

In Table 12-471:

Copyright © 2026 Texas Instruments Incorporated

- The FLIP bit is located in DSS_VID_ATTRIBUTES register.
- The base address of the video buffer must be configured as explained in Section 12.9.1.4.1.6.1, *DISPC DMA Addressing and Bursts*.
- The number of bytes to increment at the end of the row in the video buffer can be configured through DSS_VID_ROW_INC and DSS_VID_ROW_INC_UV registers. For more information, see Section 12.9.1.4.1.6.1, *DISPC DMA Addressing and Bursts*, and Section 12.9.1.4.1.6.4, *DISPC DMA Predecimation*.

12.9.1.4.1.6.4 DISPC DMA Predecimation

The predecimation process consists of downscaling an image by fetching only the necessary pixels out of the memory. Vertical and horizontal predecimation are possible:

- Vertical predecimation: The picture stored in memory can be predecimated vertically by skipping lines. Burst mode is used to fetch the data when skipping lines. Only the lines that will be used by the DISPC are fetched from memory; the other lines are skipped. The DMA engine sends requests only for the useful lines using 1D burst. The base address indicates the first valid pixel to fetch from memory. The number of lines to skip is set in the DSS_VID_ROW_INC and DSS_VID_ROW_INC_UV registers.
- Horizontal predecimation: When fetching data from memory, it is possible to skip 1 out of 2 pixels, up to 1 of 2047 pixels, by setting the DSS_VID_PIXEL_INC register to the number of pixels to skip (n), multiplied by the size of a pixel (in bytes), +1. The condition to generate a burst is that there is at least one useful pixel per 128-bit OCP request. Therefore, when the pixels are 16/32-bit, the maximum number of pixels that can be skipped is 8/4. If PixelWidthInBytes+BytesToSkip is greater than 16, the programmed burst is changed into single request.

No decimation is supported when the input format is 1, 2, 4, or 8-bit BITMAP.

For RGB and YUV420 data formats, each pixel data container in memory holds 1 pixel. Thus, when configuring the DSS_VID_PIXEL_INC register, the value of n equals the number of pixels to skip:

- For RGB format, one pixel data container = 32 bits = 1 pixel
- For YUV format:
  - One Y pixel data container = 8 bits = 1 pixel
  - One UV pixel data container = 16 bits = 1 pixel

For YUV422 format, each 32-bit pixel data container holds the Luma components for 2 pixels, and the Chrominance component of 1 pixel (see Figure 12-514). Therefore, for the valid values of the PIXELINC bit field in the case of the following YUV422 format, caution must be taken because n equals the number of pixel data containers to skip, and not the number of pixels:

- For n = 1, PIXELINC = 5
- For n = 2, PIXELINC = 9
- For n = 3, PIXELINC = 13
- For n = 4, PIXELINC = 17, etc.

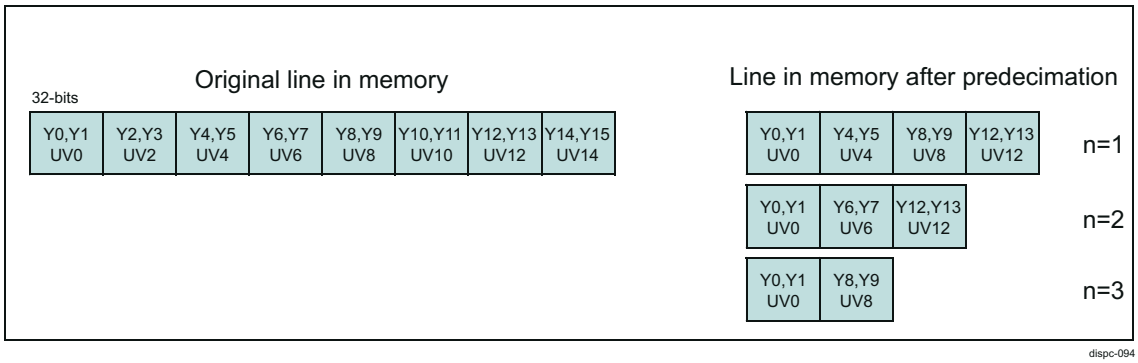


Figure 12-514. DISPC YUV422 Predecimation

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.6.5 DISPC DMA MFLAG Mechanism

The MFLAG mechanism allows a dynamic increase of the priority of DISPC real-time traffic, when required, based on the fullness of the DISPC DMA read buffers.

The MFLAG mechanism is used when fullness of the DMA buffers is critical (close to underflow). The mechanism is implemented for all DMA buffers of the video pipelines.

Programmable buffer thresholds (forming hysteresis) are used to indicate when a local MFLAG signal is generated. The 1-bit MFLAG signal is generated on DISPC Initiator port in order to inform the system that the outstanding requests from DISPC shall be considered with higher priority in order to get faster interconnect responses. The MFLAG signal is asynchronous to any ongoing interconnect transaction.

Each pipeline maintains its own MFLAG bit. The MFLAG bit is asserted depending on the fullness of the DMA buffers associated with the pipeline and depending on the thresholds programmed by software. All MFLAG signals are OR-ed together to generate the single 1-bit MFLAG for the Initiator port connected to the DISPC DMA engine.

The threshold for video pipelines corresponds to the fullness of the associated DMA buffer, and is defined by two threshold parameters:

- HT_MFLAG: High threshold.
  - For read access from video pipelines, when the pipeline buffer reaches the programmed value, the associated local MFLAG signal goes low (deasserted).
  - This threshold can be programmed in the DSS_VID_MFFLAG_THRESHOLD [31-16] HT_MFLAG register field.
- LT_MFLAG: Low threshold.
  - For read access from video pipelines, when the pipeline buffer reaches the programmed value, the associated local MFLAG signal goes high (asserted).
  - This threshold can be programmed in the DSS_VID_MFFLAG_THRESHOLD [15-0] LT_MFLAG register field.

Summary of the MFLAG value, based on DMA read buffer fullness:

- If DMA read buffer fullness < LT_MFLAG, then MFLAG signal = 1
- If LT_MFLAG < DMA read buffer fullness < HT_MFLAG, then MFLAG signal = 1
- If DMA read buffer fullness > HT_MFLAG, then MFLAG signal = 0

By default, the MFLAG mechanism is disabled (DSS_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE[1-0] MFLAG_CTRL register field = 0x0), and the MFLAG signal is low (de-asserted). The arbitration scheme for the pipelines is the same as described in Section 12.9.1.4.1.6.7, *DISPC DMA Arbitration*. That is, round-robin either between high-priority pipelines, or between normal-priority pipelines (if all pipelines are of normal priority).

When the MFLAG_CTRL register field is set to 0x2, the MFLAG mechanism is enabled, and the MFLAG signal is dynamically set to 0 or 1, depending on DMA buffer fullness and programmed threshold levels, as explained previously in this section. In this case, the arbitration scheme for the pipelines is round-robin between those high-priority pipelines, which have asserted their local MFLAG signals. If there are no high-priority pipelines with their local MFLAG signals asserted, then the arbitration scheme is the same as described in Section 12.9.1.4.1.6.7, *DISPC DMA Arbitration*.

The DSS_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE[6] MFLAG_START bit defines the following additional rules for the MFLAG mechanism:

- If the MFLAG_START bit is set to 0x0 (default value), then when the DMA buffer is empty at the beginning of the frame, the MFLAG signal of each pipeline is kept at 0 until PRELOAD is reached (for more information on preloading, see Section 12.9.1.4.1.6.2, *DISPC Read DMA Buffers*). Then, based on the setting of the DSS_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE[1-0] MFLAG_CTRL register field, the MFLAG signal is generated and DISPC internal logic is arbitrating between pipeline requests.
- If the MFLAG_START bit is set to 0x1, then even in the beginning of the frame when the DMA buffer is empty, the configuration of DSS_COMMON_DISPC_GLOBAL_MFLAG_ATTRIBUTE[1-0] MFLAG_CTRL register field determines the generation of the MFLAG signal.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.6.6 DISPC DMA Priority Requests Control

The DSS_COMMON_DSS_CBA_CFG register controls the priority level for DMA requests going out to the memory through the system interconnect.

As explained in Section 12.9.1.4.1.6.5, *DISPC DMA MFLAG Mechanism*, the DISPC Initiator port generates a 1-bit MFLAG output signal to raise the priority of all requests made on that port, if any of its DMA buffers runs critically low (determined by a set of user programmable threshold values for each buffer).

DSS uses the MFLAG signal from DISPC to set a 3-bit priority level output (*Mpriority*) for the Initiator port to either a low or high value (configurable in DSS_COMMON_DSS_CBA_CFG[2-0] PRI_LO and [5-3] PRI_HI register fields with optional values of 0~7) as follows:

- When MFLAG = 0, the PRI_LO register field determines the value of the Mpriority output for the normal transactions.
- When MFLAG = 1, the PRI_HI register field determines the value of the Mpriority output for the high-priority transactions.

This Mpriority output directly drives the respective input of the system interconnect port, which corresponds to the DISPC DMA Initiator port.

CAUTION

Upon a hardware reset, DSS_COMMON_DSS_CBA_CFG[2-0] PRI_LO and [5-3] PRI_HI register fields are set by default to 4 and 1, respectively. Afterwards, these priority level register fields can only be modified by a secure host.

12.9.1.4.1.6.7 DISPC DMA Arbitration

The read requests sent to the system interconnect are pipelined and arbitrated in a round-robin scheme. The default arbitration scheme can be modified by setting the priority attribute of each pipeline as defined in the DSS_VID_ATTRIBUTES[23] ARBITRATION register bit.

By default, all pipelines have the same priority (normal priority), which means all pipeline requests are treated in a round-robin order manner. If one or more pipelines require a higher number of requests going to the system interconnect, its priority can be moved up to high priority. In this case, the high-priority pipeline is granted access before any pipeline in normal priority. If more than one active pipeline is in high priority, then the behavior is the same as all active pipelines in normal priority. Normal active pipelines are not treated until all high active pipelines are finished. The ARBITRATION bit cannot be modified during the entire frame.

12.9.1.4.1.6.8 DISPC DMA Power Modes

12.9.1.4.1.6.8.1 DISPC DMA Low Power Mode

Each DMA buffer can be associated to the pipeline or merged with other DMA buffers. The total number of DMA buffers for each individual pipeline is from 0 (pipeline inactive) to number of pipelines (in that case all the DMA buffers are associated to a single pipeline) supported by the DMA engine.

The user is responsible for configuring correctly the number of DMA buffers to guarantee no underflow. The DMA buffers allocated to each pipeline shall be greater or equal to the minimum required DMA buffer to support the throughput and the system latency.

When the size of the buffer is changed, the thresholds shall be re-programmed by the user to reflect the new DMA buffer configuration. Increasing the size of the buffer used enables to put the interconnect and the system memory in standby for a longer period, therefore, leading to a possible system power reduction.

The low power mode of the DISPC DMA can also be achieved by keeping the thresholds (bit-fields [31-16] BUFHIGHTHRESHOLD and [15-0] BUFLOWTHRESHOLD of DSS_VID_BUF_THRESHOLD register) farther apart. The farther they are there will be longer periods of idling on the DMA fetch interface resulting in lower power. The user needs to ensure a minimum value of BUFLOWTHRESHOLD, so that there is no underflow.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.6.8.2 DISPC DMA Ultra-Low Power Mode

In ultra-low power mode, the system interconnect is used to fill up the DMA buffers to store all the data required to display a full frame. Then, the system interconnect is not used anymore to fetch new pixels for the following frames. The data are fetched once into the DMA buffer and then the following frames re-use the DMA buffer to display on the screen.

The programming of the ultra-low power mode is independent for each pipeline and is achieved via the DSS_VID_ATTRIBUTES[24] SELFREFRESH register bit. One pipeline may have all frame pixels into the DMA buffer and other pipeline may have to refill the DMA buffers along the display scan, because the frame buffer is too big to be stored in the DMA buffer.

Two ultra-low power modes can be entered, manual or automatic mode:

- Manual self-refresh mode: Starting self-refresh mode is done manually by setting the SELFREFRESH bit to 0x1 after capturing a frame in the DMA buffers. Self-refresh mode is stopped by setting the SELFREFRESH bit to 0x0. Software must first disable the SELFREFRESH bit during at least one frame in order to capture the data (by setting the GOBIT register bit of the video port the pipeline is associated with). Once the DSS_VP1_CONTROL[5] GOBIT bit has been set, the software must read the GOBIT bit to ensure that the frame has been loaded into the buffer, then it can set the SELFREFRESH bit to 1. Once SELFREFRESH is enabled, the fetch of data from the system memory is stopped for the following frames. The software needs to reset the SELFREFRESH bit in order to restart fetching data from system memory.
- Automatic self-refresh mode: By setting the DSS_VIDL1_ATTRIBUTES[17] SELFREFRESHAUTO bit to 0x1, the transition from disabled to enabled for self-refresh mode is controlled by hardware. This allows the software to reset the SELFREFRESH bit to "disabled", and then automatically after the fetch of the first frame the hardware switches back SELFREFRESH bit to "enabled". The SELFREFRESH must be disabled during at least one frame in order to capture the data, so software must reset the bit to 0 every time the data in the DMA buffer needs to be updated. The hardware reads the data inside the DMA buffer without accessing the interconnect and system memory during the frame, then modifies the SELFREFRESH bit to reflect the current state of the self-refresh mode by setting the bit to 0x1.

12.9.1.4.1.7 DISPC Pixel Data Formats

The video pipelines support various types of memory formats, as listed in Table 12-472.

For BITMAP formats the nibble mode (pixels in each byte are packed in reverse order) can be enabled by setting the DSS_VID_ATTRIBUTES[10] NIBBLEMODE register bit to 0x1.

The pixel data format can be selected by loading the corresponding value from DSS_VID_ATTRIBUTES [6-1] FORMAT register field.

Table 12-472. DISPC Supported Pixel Data Formats

FORMAT Register Field Value		Pixel Format(4)	Component Bit Depth
Alpha	Alpha-X		
0x00	0x20	ARGB16-4444	4
0x01	0x21	ABGR16-4444	4
0x02	0x22	RGBA16-4444	4
0x03	NA	RGB16-565	5(R,B), 6(G)
0x04	NA	BGR16-565	5(R,B), 6(G)
0x05	0x25	ARGB16-1555	1(A), 5(R,G,B)
0x06	0x26	ABGR16-1555	1(A), 5(R,G,B)
0x07	0x27	ARGB32-8888	8
0x08	0x28	ABGR32-8888	8
0x09	0x29	RGBA32-8888	8
0x0A	0x2A	BGRA32-8888	8
0x0B	NA	RGB24-888	8
0x0C	NA	BGR24-888	8

Copyright © 2026 Texas Instruments Incorporated

Table 12-472. DISPC Supported Pixel Data Formats (continued)

FORMAT Register Field Value		Pixel Format[4]	Component Bit Depth
0x0E	0x2E	ARGB32-2101010	2(A),10(R,G,B)
0x0F	0x2F	ABGR32-2101010	2(A),10(R,G,B)
0x10	0x30	ARGB64-16161616	16
0x11	0x31	RGBA64-16161616	16
0x12	NA	BITMAP1	1
0x13	NA	BITMPA2	2
0x14	NA	BITMAP4	4
0x15	NA	BITMAP8	8
0x16	NA	RGB565A8 [1]	5(R,B), 6(G), separate 8(A)
0x17	NA	BGR565A8 [1]	5(R,B), 6(G), separate 8(A)
Packed	**Planar**	**Pixel Format**	**Component Bit Depth**
0x3E	NA	YUV422-YUV2	8/10/12 [3]
0x3F	NA	YUV422-UYVY	8/10/12 [3]
NA	0x3D	YUV420-NV12	8/10/12 [3]
NA	See [2]	YUV420-NV21	8/10/12

(1) The video pipelines support an optional 8-bit Alpha plane (separate buffer), if the pixel format of the source/destination buffer is either RGB16-565 or BGR16-565.
(2) NV21 formats for YUV420 are indirectly supported through chroma swapping during the color space conversion.
(3) 10bit/12bit versions of these YUV formats are also supported in both packed and unpacked (in 16-bit container) formats. For unpacked formats, both LSB or MSB alignments are supported. These configurations are set using a separate configuration register DSS_VID_ATTRIBUTES). Default is 8-bit packed format support.
(4) All RGB formats with alpha component include both pre/non-pre-multiplied data support. Also, alpha can be disabled to work as X (can be replaced with global alpha value).

Figure 12-515 shows the pixel data memory organization for the bitmap pixel formats.

Copyright © 2026 Texas Instruments Incorporated

BITMAP 1-bpp (0x12)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

BITMAP 2-bpp (0x13)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15		P14		P13		P12		P11		P10		P9		P8		P7		P6		P5		P4		P3		P2		P1		P0	

BITMAP 1-bpp (0x14)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 7								Pixel 6								Pixel 5								Pixel 4							
Pixel 3								Pixel 2								Pixel 1								Pixel 0							
Pixel 6								Pixel 7								Pixel 4								Pixel 5							
Pixel 2								Pixel 3								Pixel 0								Pixel 1							

BITMAP 1-bpp (0x15)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 3																Pixel 2															
Pixel 1																Pixel 0															

For BITMPA P-1/2/4 bpp, the Nibble mode (pixels in each byte are packed in reverse order) is also supported.

BITMAP 1-bpp (0x12) – Nibble Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P24	P25	P26	P27	P28	P29	P30	P31	P16	P17	P18	P19	P20	P21	P22	P23	P8	P9	P10	P11	P12	P13	P14	P15	P0	P1	P2	P3	P4	P5	P6	P7

BITMAP 2-bpp (0x13) – Nibble Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P12		P13		P14		P15		P8		P9		P10		P11		P4		P5		P6		P7		P0		P1		P2		P3	

BITMAP 4-bpp (0x14) – Nibble Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 6								Pixel 7								Pixel 4								Pixel 5							
Pixel 2								Pixel 3								Pixel 0								Pixel 1							

disp-301

Figure 12-515. DISPC Bitmap Pixel Formats

Figure 12-516 and Figure 12-517 show the pixel data memory organization for the RGB 16-bit pixel formats.

Copyright © 2026 Texas Instruments Incorporated

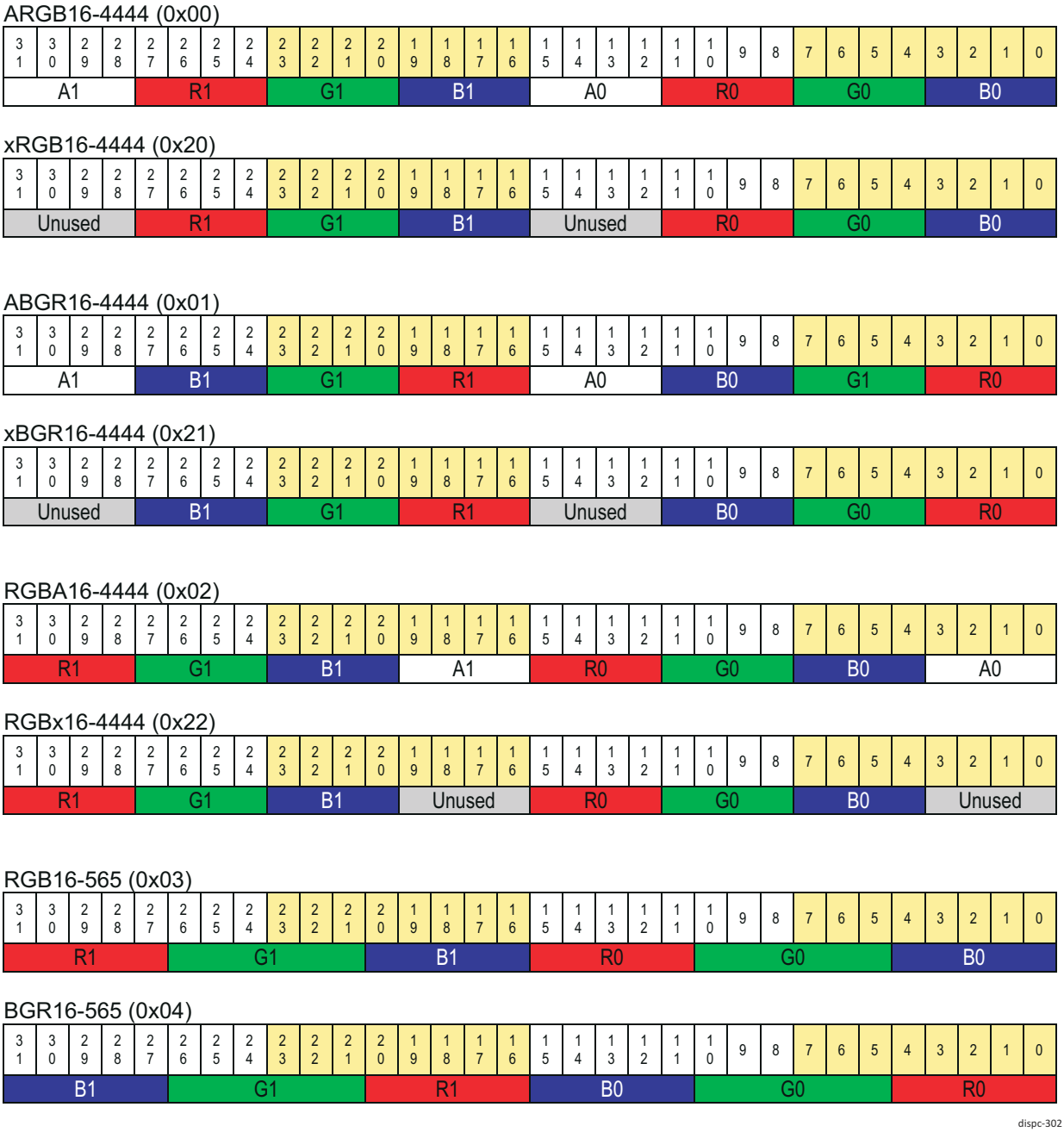


Figure 12-516. DISPC RGB 16-bit Pixel Formats 1

Copyright © 2026 Texas Instruments Incorporated

RGB565A8 (0x16) and BGR565A8 (0x17) are RGB16-565 and BGR16-565, respectively, with a separate Alpha-8bit plane.

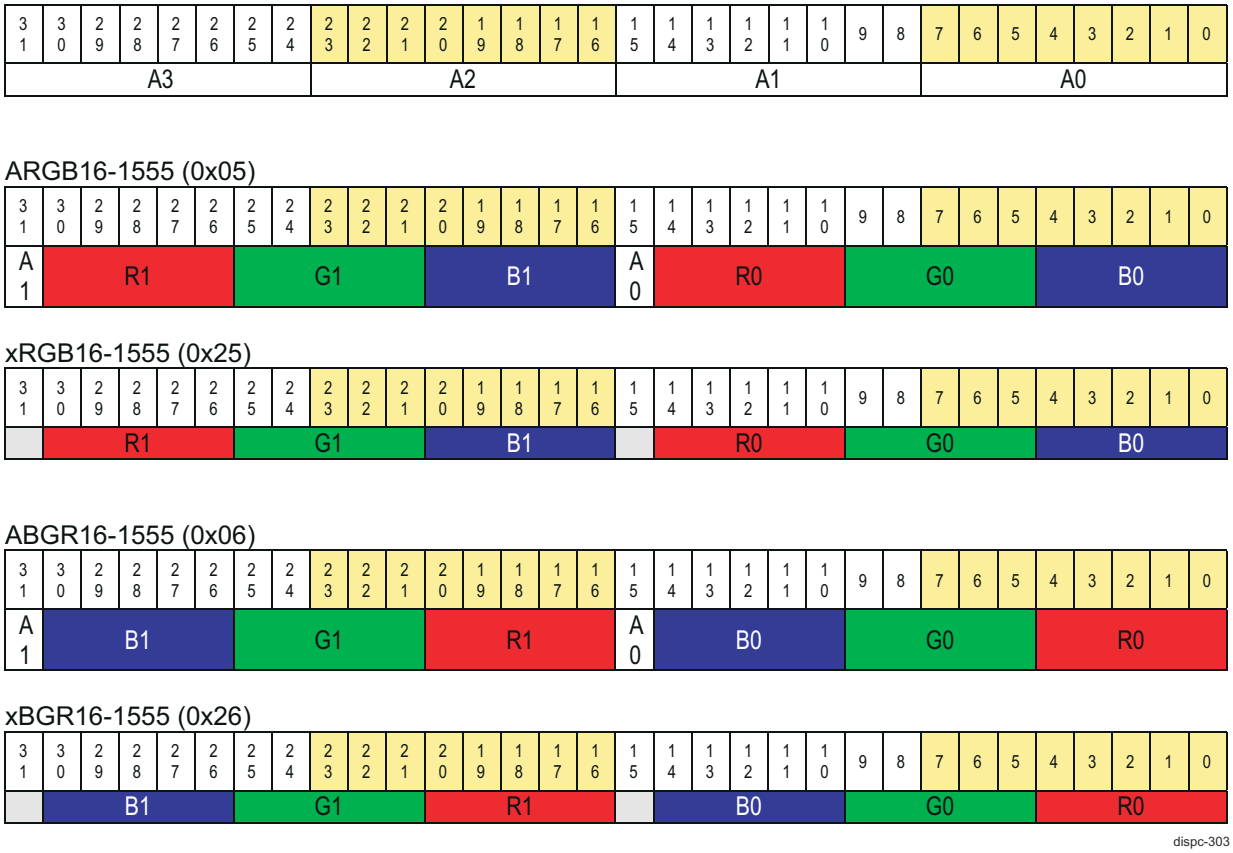


Figure 12-517. DISPC RGB 16-bit Pixel Formats 2

Figure 12-518 shows the pixel data memory organization for the RGB 24-bit pixel formats.

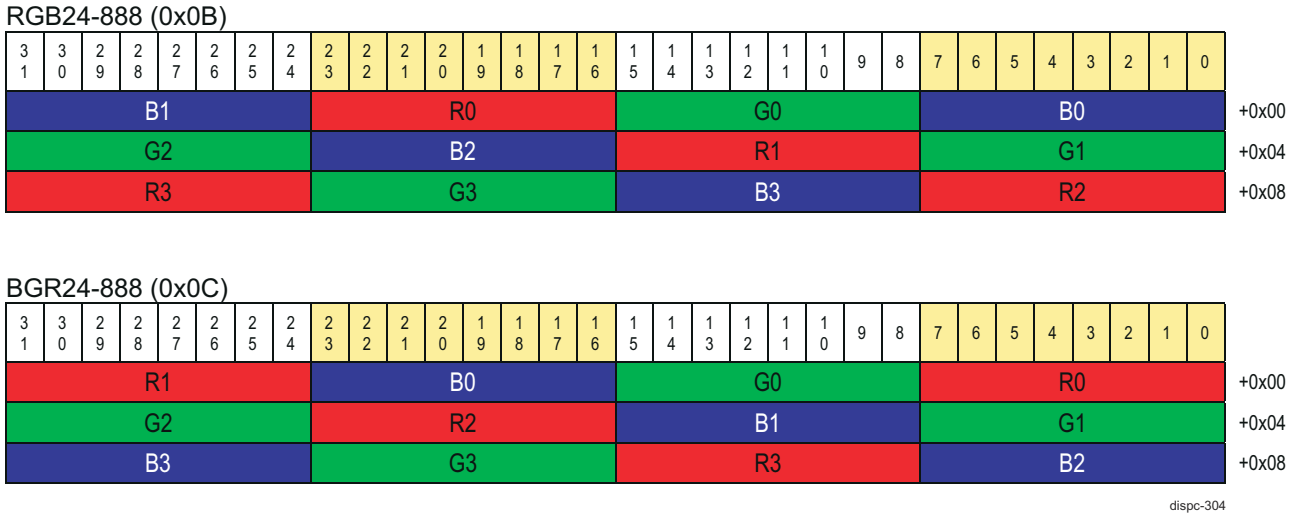


Figure 12-518. DISPC RGB 24-bit Pixel Formats

Figure 12-519 and Figure 12-520 show the pixel data memory organization for the RGB 32-bit pixel formats.

Copyright © 2026 Texas Instruments Incorporated

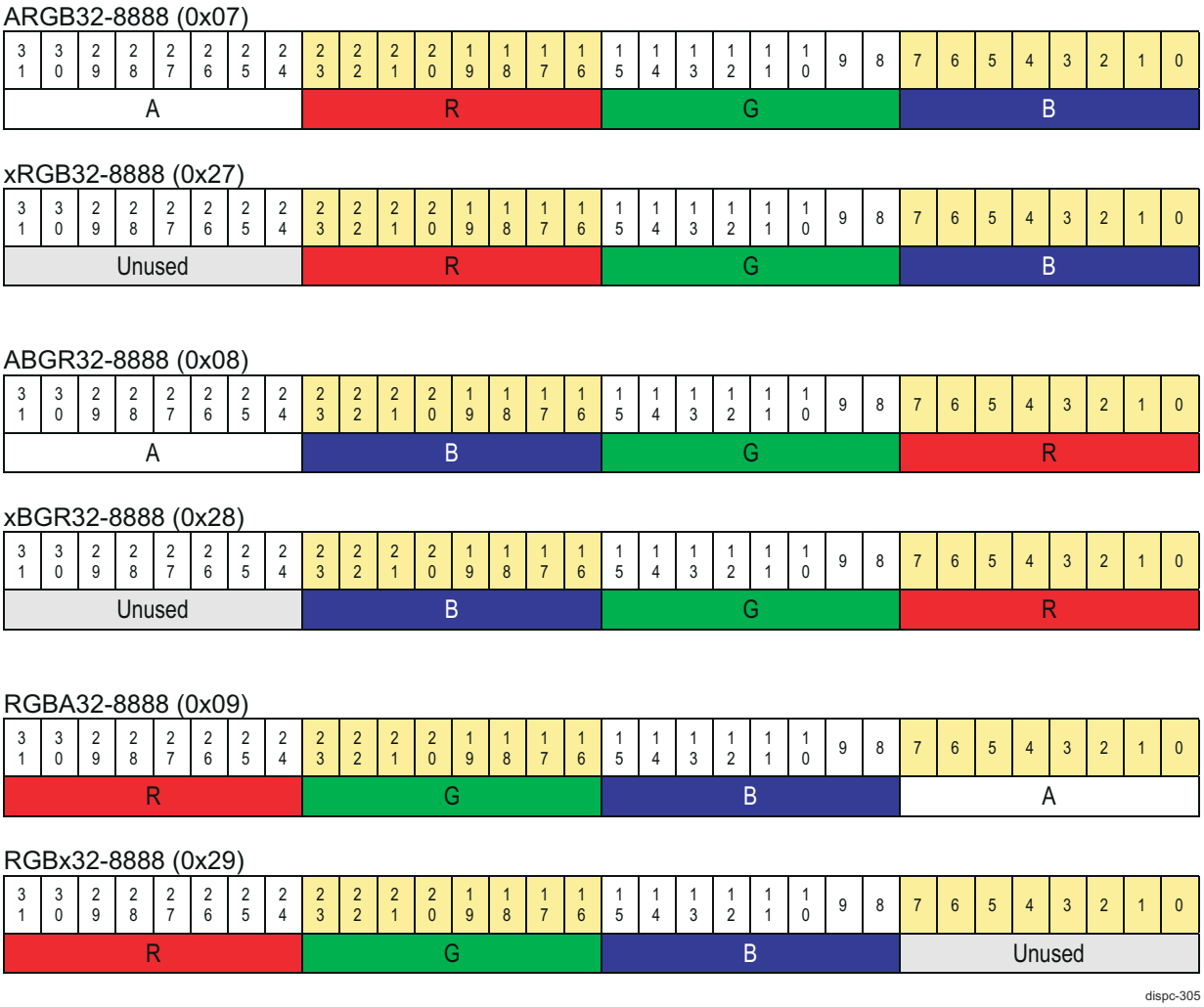


Figure 12-519. DISPC RGB 32-bit Pixel Formats 1

Copyright © 2026 Texas Instruments Incorporated

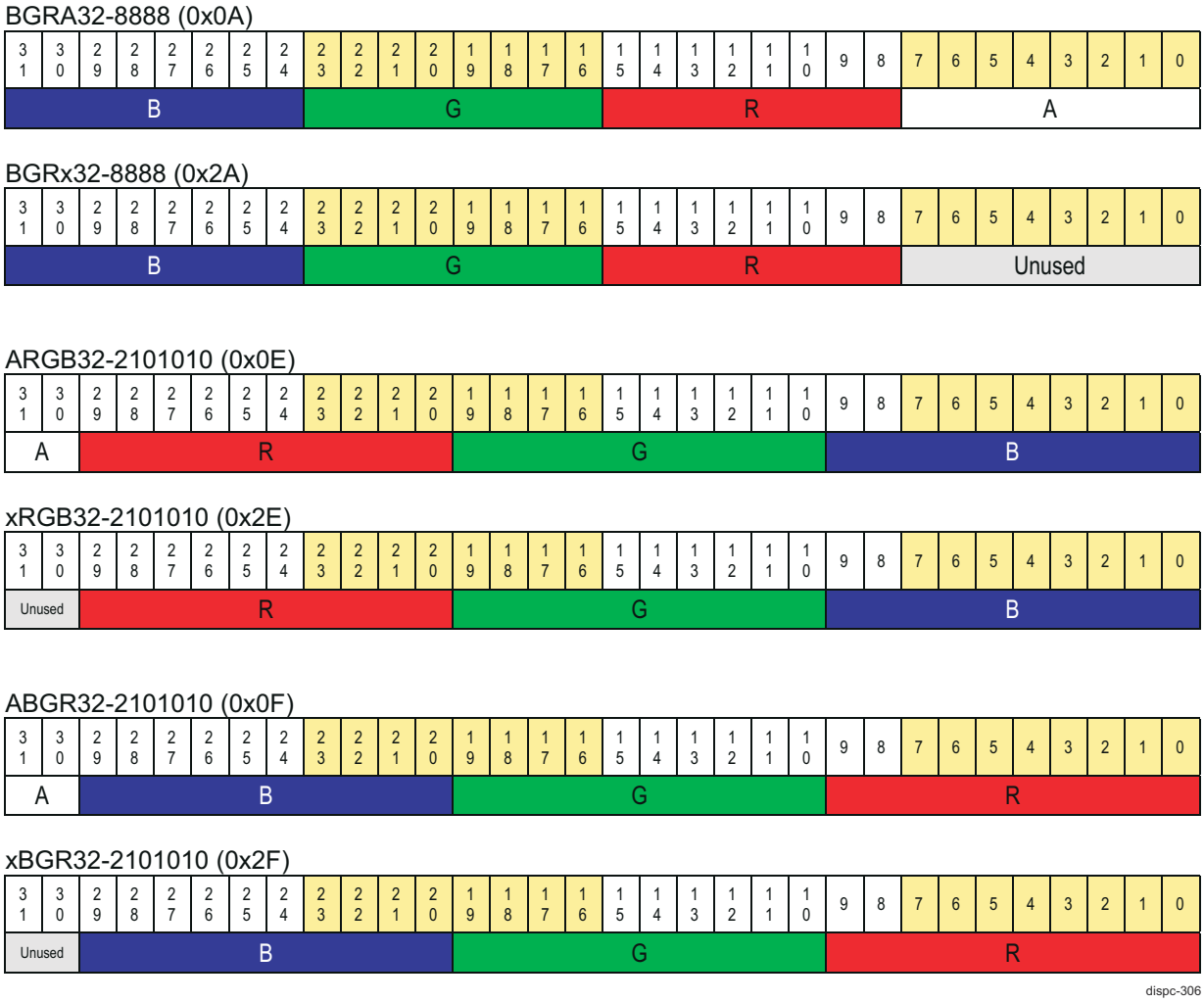


Figure 12-520. DISPC RGB 32-bit Pixel Formats 2

Figure 12-521 shows the pixel data memory organization for the RGB 64-bit pixel formats.

Copyright © 2026 Texas Instruments Incorporated

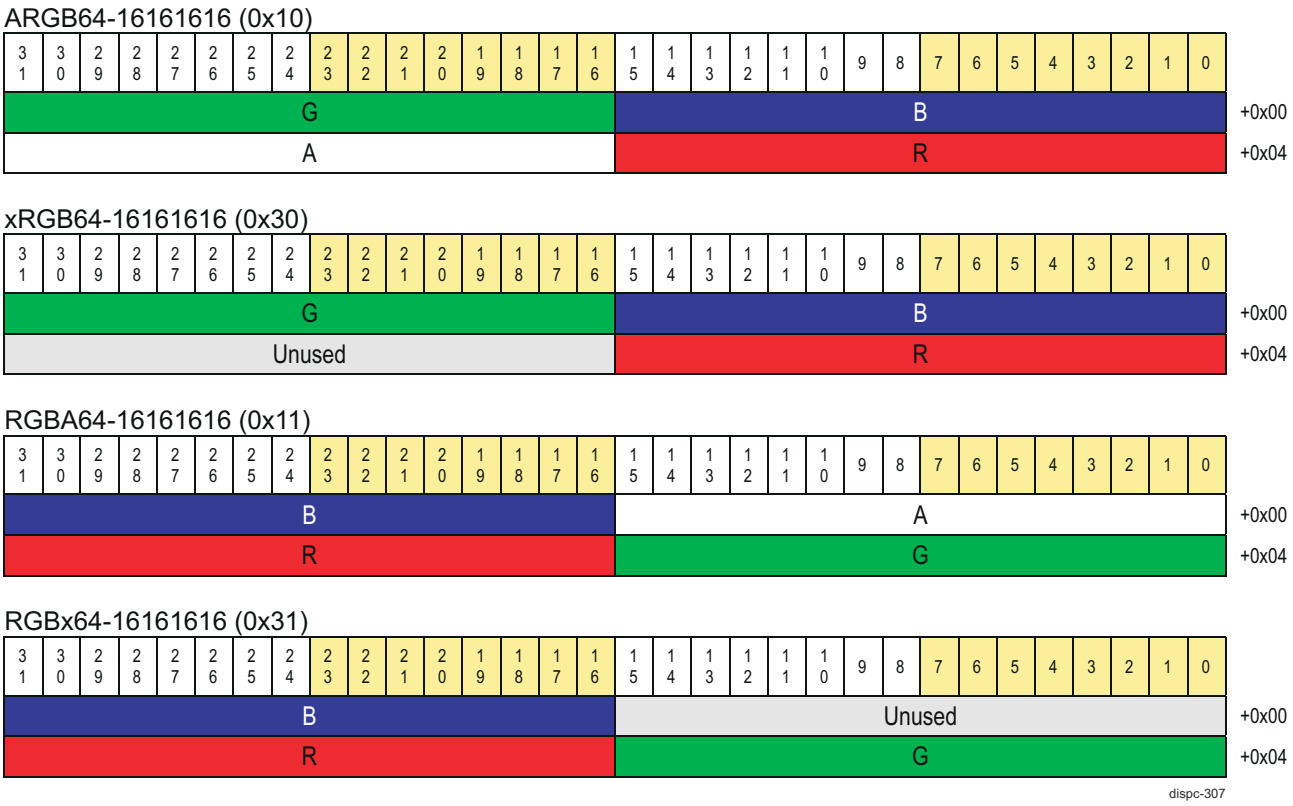


Figure 12-521. DISPC RGB 64-bit Pixel Formats

Figure 12-522 and Figure 12-523 show the pixel data memory organization for the YUV 8-bit pixel formats, together with some specific register settings.

ATTRIBUTES2.YUV_SIZE = 0 (8b)
ATTRIBUTES2.YUV_MODE = 0 (N/A)
ATTRIBUTES2.YUV_ALIGN = 0 (N/A)

YUV422 (1-plane/Packed)

YUV2 4:2:2 (0x3E)

Cr0	Y1	Cb0	Y0	+0x0
Cr1	Y3	Cb1	Y2	+0x4
Cr2	Y5	Cb2	Y4	+0x8
Cr3	Y7	Cb3	Y6	+0xC

UYVY 4:2:2 (0x3F)

Y1	Cr0	Y0	Cb0	+0x0
Y3	Cr1	Y2	Cb1	+0x4
Y5	Cr2	Y4	Cb2	+0x0
Y7	Cr3	Y6	Cb3	+0x4

dispc-308

Figure 12-522. DISPC YUV 8-bit Pixel Formats 1

Copyright © 2026 Texas Instruments Incorporated

YUV420 (2-plane/Planar)

YUV 4:2:0 – NV12 (0x3D)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Y3	Y2	Y1	Y0	+0x0
Y7	Y6	Y5	Y4	+0x4
Y11	Y10	Y9	Y8	+0x8
Y15	Y14	Y13	Y12	+0xC

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Cr1	Cb1	Cr0	Cb0	+0x0
Cr3	Cb3	Cb2	Cb2	+0x4
Cr5	Cb5	Cr4	Cb4	+0x8
Cr7	Cb7	Cb6	Cb6	+0xC

YUV 4:2:0 – NV21

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Y3	Y2	Y1	Y0	+0x0
Y7	Y6	Y5	Y4	+0x4
Y11	Y10	Y9	Y8	+0x8
Y15	Y14	Y13	Y12	+0xC

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0	
Cb1	Cr1	Cb0	Cr0	+0x0
Cb3	Cr3	Cr2	Cr2	+0x4
Cb5	Cr5	Cb4	Cr4	+0x8
Cb7	Cr7	Cr6	Cr6	+0xC

NV21 is not directly supported. It must be specified as NV12 and then the chroma components must be swapped during the YUV to RGB conversion process.

dispc-309

Figure 12-523. DISPC YUV 8-bit Pixel Formats 2

Figure 12-524 shows the pixel data memory organization for the YUV 10-bit pixel formats, together with some specific register settings.

Copyright © 2026 Texas Instruments Incorporated

YUV 10-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 32-bit word with 2 MSB in each 32-bit word not used.
Note: Each new line in memory must start at a 128-bit aligned address for this format

ATTRIBUTES2.YUV_SIZE = 1 (10b)
ATTRIBUTES2.YUV_MODE = 0 (Packed)
ATTRIBUTES2.YUV_ALIGN = 0 (N/A)

YUV422 (1-plane/Packed)

YUV2 4:2:2 – 10bit (0x3E)

31 30	29 28 27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
	Y1		Cb0		Y0	+0x0
	Cb1		Y2		Cr0	+0x4
	Y4		Cr1		Y3	+0x8
	Cr2		Y5		Cb2	+0xC

UYVY 4:2:2 – 10bit (0x3F)

31 30	29 28 27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
	Cr0		Y0		Cb0	+0x0
	Y2		Cb1		Y1	+0x4
	Cb2		Y3		Cr1	+0x8
	Y5		Cr2		Y4	+0xC

YUV420 (2-plane/Planar)

YUV 4:2:0 – NV12 (10-bit) (0x3D)

31 30	29 28 27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
	Y2		Y1		Y0	+0x0
	Y5		Y4		Y3	+0x4
	Y8		Y7		Y6	+0x0
	Y11		Y10		Y9	+0x4

31 30	29 28 27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
	Cb1		Cr0		Cb0	+0x0
	Cr2		Cb2		Cr1	+0x4
	Cb4		Cr3		Cb3	+0x8
	Cr5		Cb5		Cr4	+0xC

dispc-310

Figure 12-524. DISPC YUV 10-bit Pixel Formats

Figure 12-525 and Figure 12-526 show the pixel data memory organization for the YUV 12-bit pixel formats, together with some specific register settings.

Copyright © 2026 Texas Instruments Incorporated

YUV 12-bit formats have the same component packing order as 8-bit formats except that the packing is done across a multiple 64-bit word with 4 MSB in each 64-bit word not used.
Note: Each new line in memory must start at a 128-bit aligned address for this format

ATTRIBUTES2.YUV_SIZE = 2 (12b)
ATTRIBUTES2.YUV_MODE = 0 (Packed)
ATTRIBUTES2.YUV_ALIGN = 0 (N/A)

YUV422 (1-plane)

YUV2 4:2:2 – 12bit (0x3E)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
Y1[7:0]	Cb0	Y0			+0x0
	Y2	Cr0		Y1 [11:8]	+0x4
Cr1[7:0]	Y3	Cb1			+0x8
	Cb2	Y4		Cr1[11:8]	+0xC
Y6[7:0]	Cr2	Y5			+0x10
	Y7	Cb3		Y6[11:8]	+0x14
Cb4[7:0]	Y8	Cr3			+018
	Cr4	Y9		Cb4[11:8]	+1C

UYVY 4:2:2 – 12bit (0x3F)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
Cr0[7:0]	Y0	Cb0			+0x0
	Cb1	Y1		Cr0 [11:8]	+0x4
Y3[7:0]	Cr1	Y2			+0x8
	Y4	Cb2		Y3[11:8]	+0xC
Cb3[7:0]	Y5	Cr2			+0x10
	Cr3	Y6		Cb3[11:8]	+0x14
Y8[7:0]	Cb4	Y7			+018
	Y9	Cr4		Y8[11:8]	+1C

dispc-311

Figure 12-525. DISPC YUV 12-bit Pixel Formats 1

Copyright © 2026 Texas Instruments Incorporated

YUV420 (2-plane)

YUV 4:2:0 – NV12 (12-bit) (0x3D)

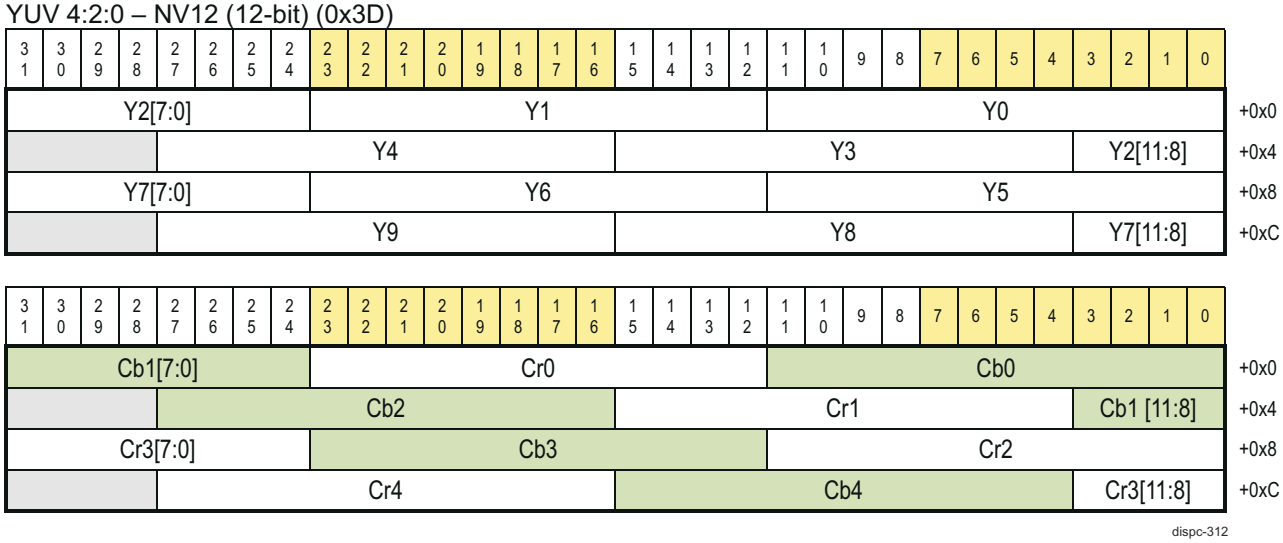


Figure 12-526. DISPC YUV 12-bit Pixel Formats 2

Figure 12-527 through Figure 12-529 show the pixel data memory organization for the YUV 10-bit/12-bit unpacked pixel formats in 16-bit container, together with some specific register settings.

Copyright © 2026 Texas Instruments Incorporated

YUV 10-bit/12-bit unpacked formats have the same component packing order as 8-bit formats except that each component is stored in a 16-bit container (with MSB or LSB bits within the 16-bit container not used depending on the MSB/LSB alignment).

ATTRIBUTES2.YUV_SIZE = 1 or 2 (10b or 12b)
ATTRIBUTES2.YUV_MODE = 1 (Unpacked)
ATTRIBUTES2.YUV_ALIGN = 0 or 1 (LSB or MSB aligned)

YUV422 (1-plane)

10-bit unpacked LSB aligned

YUV2 4:2:2 – 10bit (0x3E)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
unused	Cb0	unused	Y0	+0x0
unused	Cr0	unused	Y1	+0x4

UYVY 4:2:2 – 10bit (0x3F)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
unused	Y0	unused	Cb0	+0x0
unused	Y1	unused	Cr0	+0x4

10-bit unpacked MSB aligned

YUV2 4:2:2 – 10bit (0x3E)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
Cb0	unused	Y0	unused	+0x0
Cr0	unused	Y1	unused	+0x4

UYVY 4:2:2 – 10bit (0x3F)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0	
Y0	unused	Cb0	unused	+0x0
Y1	unused	Cr0	unused	+0x4

dispc-313

Figure 12-527. DISPC YUV 10-bit/12-bit Unpacked Pixel Formats 1

Copyright © 2026 Texas Instruments Incorporated

YUV422 (1-plane)

12-bit unpacked LSB aligned

YUV2 4:2:2 – 12bit (0x3E)

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
unused	Cb0			unused	Y0		+0x0
unused	Cr0			unused	Y1		+0x4

UYVY 4:2:2 – 12bit (0x3F)

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
unused	Y0			unused	Cb0		+0x0
unused	Y1			unused	Cr0		+0x4

12-bit unpacked MSB aligned

YUV2 4:2:2 – 12bit (0x3E)

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
Cb0			unused	Y0		unused	+0x0
Cr0			unused	Y1		unused	

UYVY 4:2:2 – 12bit (0x3F)

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
Y0			unused	Cb0		unused	0x0
Y1			unused	Cr0		unused	

dispc-314

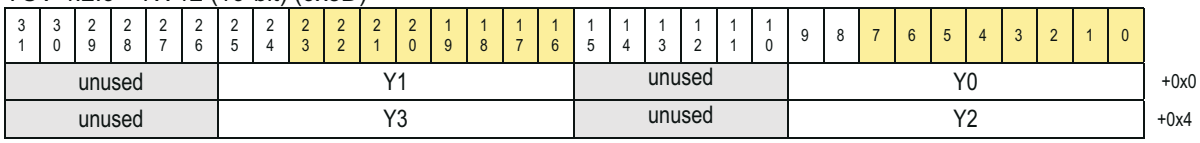
Figure 12-528. DISPC YUV 10-bit/12-bit Unpacked Pixel Formats 2

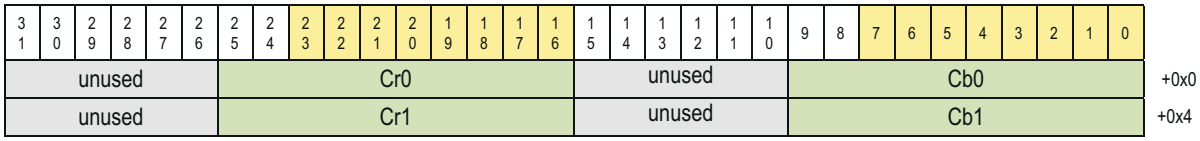
Copyright © 2026 Texas Instruments Incorporated

YUV420 (2-plane)

10-bit unpacked LSB aligned

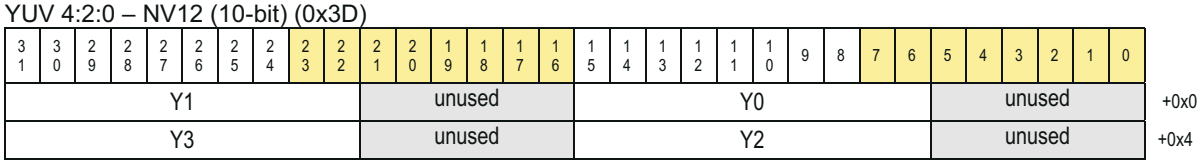
YUV 4:2:0 – NV12 (10-bit) (0x3D)

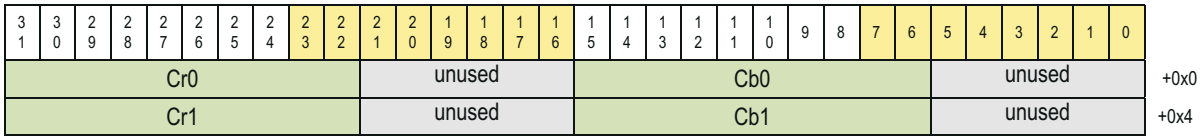




10-bit unpacked MSB aligned

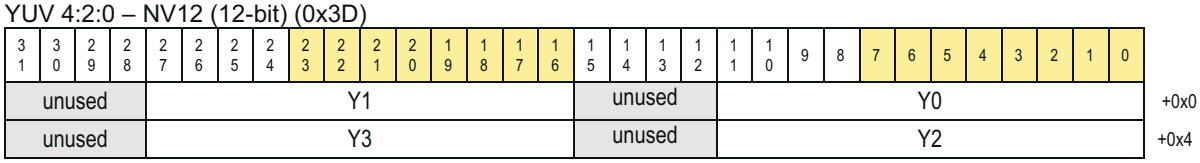
YUV 4:2:0 – NV12 (10-bit) (0x3D)

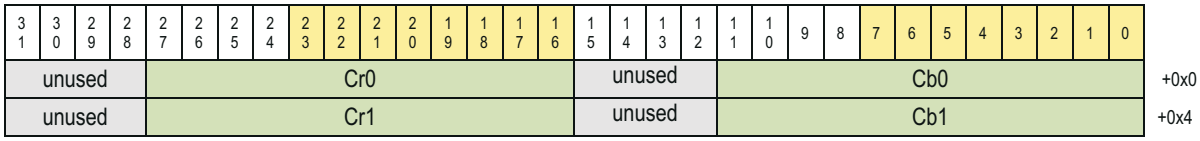




12-bit unpacked LSB aligned

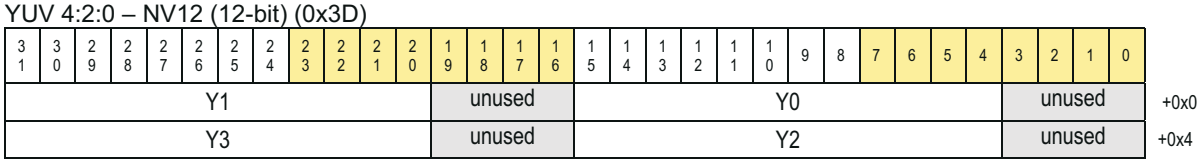
YUV 4:2:0 – NV12 (12-bit) (0x3D)

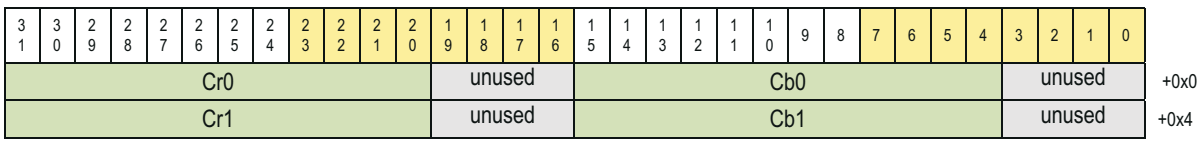




12-bit unpacked MSB aligned

YUV 4:2:0 – NV12 (12-bit) (0x3D)





dispc-315

Figure 12-529. DISPC YUV 10-bit/12-bit Unpacked Pixel Formats 3

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.8 DISPC Video Pipelines

DISPC includes two input pipelines:

- Video pipeline (VID)
- Video lite pipeline (VIDL1)

The video pipeline (VID) consists of:

- VC-1 range mapping unit for YUV422/YUV420 input formats;
- Replication logic for ARGB input formats;
- One 256 x 24-bit entries Color Look-up Table (CLUT);
- Polyphase-filter based resizer unit (scaler) supporting chroma upsampling;
- Color Space Conversion (CSC) unit (YUV to RGB), which can be used also for programmable BCHS (Brightness/Contrast/Hue/Saturation) control;

The video lite pipeline (VIDL1) is identical to the video pipeline (VID), except for:

- Polyphase-filter based resizer (scaler) is not included;
- Chroma upsampling is done using dedicated YUV420-to-422 and YUV422-to-444 upsamplers (instead of using the scaler);

Each pipeline processing block can be independently bypassed.

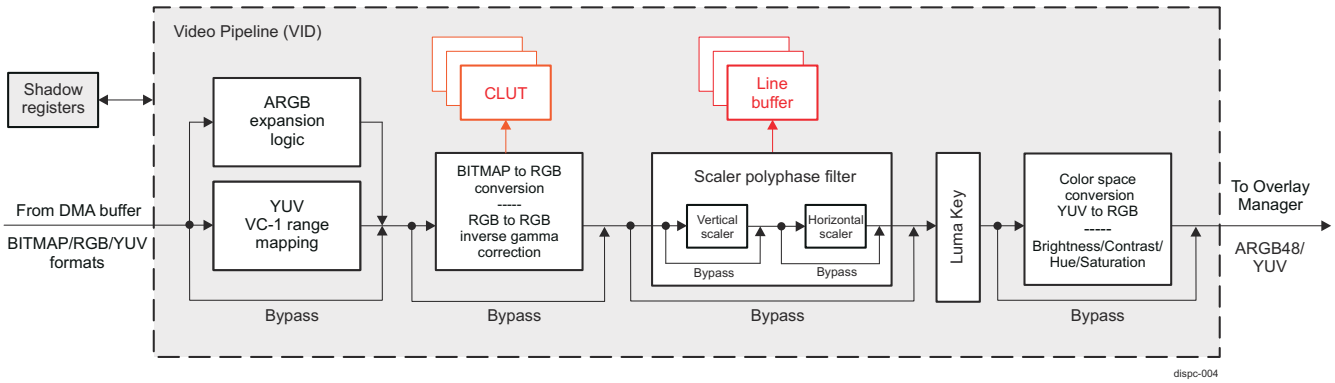


Figure 12-530. DISPC Video Pipeline Configuration

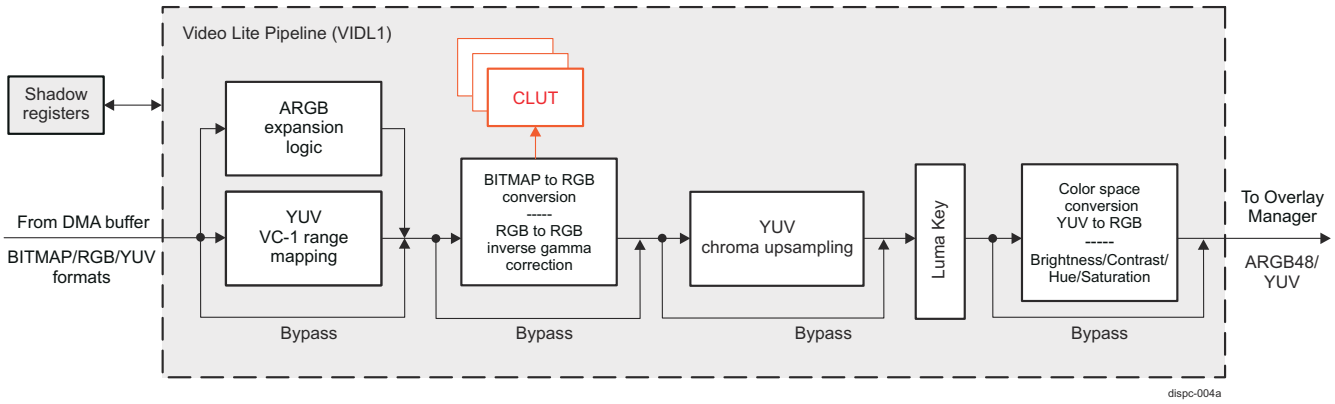


Figure 12-531. DISPC Video Lite Pipeline Configuration

The input of each pipeline is connected to the video DMA buffer controller. Each pipeline pixel output is always connected to the overlay managers (OVR1 and OVR2). Each pipeline configuration supports various BITMAP, RGB (ARGB and RGBA), and YUV formats, as listed in Section 12.9.1.4.1.7, *DISPC Pixel Data Formats*.

The 256-entry CLUT is either used to convert BITMAP (1, 2, 4, or 8-bit indexed formats) into RGB format, or for RGB to RGB inverse gamma correction. For a BITMAP format data, scaling is not supported. Scaling and color

Copyright © 2026 Texas Instruments Incorporated

look-up table features are mutually exclusive. If the color look-up feature is enabled, then video pipeline scaler has to be disabled.

For chroma sub-sampled YUV formats (YUV422 and YUV420-NV12/NV21):

- VID pipeline: the scaler unit upsamples the chrominance data using both vertical and horizontal polyphase filters;
- VIDL1 pipeline: dedicated YUV420 to YUV422, and YUV422 to YUV444 chroma upsamplers are used;

For ARGB source data with less than/equal to 10-bit component data size the replication logic (ARGB expansion) converts the data to ARGB48 by replicating the MSBs into the LSBs:

- When scaling is disabled (or no scaler supported), the resulting ARGB48 data is directly provided to the pipeline output;
- When vertical scaling is engaged, the resulting ARGB48 data is first truncated to ARGB8888, and then converted to ARGB10101010 (by MSBs replication into LSBs), before being fed to the vertical scaler input;
- When vertical scaling is disabled, but horizontal scaling is engaged, the resulting ARGB48 data is directly provided to the horizontal scaler input;

A pipeline can be enabled by setting the DSS_VIDL1_ATTRIBUTES[0] ENABLE register bit. If the video pipeline is disabled, the video window does not exist on the screen and the whole video pipeline and its DMA are inactive. Prior to enabling the video layer a valid configuration has to be set by the user.

Note

The DISPC input pipelines, VID and VID1L, will be commonly referred to as *video pipeline (VID)* in the following sections. Any differences in their functionality will be highlighted.

12.9.1.4.1.8.1 DISPC VID Replication Logic

The replication logic (ARGB expansion) converts ARGB pixel formats into ARGB48 format by replicating the MSBs into the LSBs. The logic is always enabled.

Table 12-473 shows how some of the ARGB formats supported by video pipelines are remapped into ARGB48 by default.

Table 12-473. DISPC VID Replication: ARGB Pixel Formats Remapping into ARGB48-12121212

Formats	A[11:0]	R[11:0]	G[11:0]	B[11:0]
	MSB - LSB	MSB - LSB	MSB - LSB	MSB - LSB
xRGB12-4444	111111111111	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGBx12-4444	111111111111	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGB16-565	111111111111	R[4:0]R[4:0]R[4:3]	G[5:0]G[5:0]	B[4:0]B[4:0]B[4:3]
xRGB16-1555	111111111111	R[4:0]R[4:0]R[4:3]	G[4:0]G[4:0]G[4:3]	B[4:0]B[4:0]B[4:3]
ARGB16-1555	AAAAAAAAAA	R[4:0]R[4:0]R[4:3]	G[4:0]G[4:0]G[4:3]	B[4:0]B[4:0]B[4:3]
ARGB16-4444	A[3:0]A[3:0]A[3:0]	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
RGBA16-4444	A[3:0]A[3:0]A[3:0]	R[3:0]R[3:0]R[3:0]	G[3:0]G[3:0]G[3:0]	B[3:0]B[3:0]B[3:0]
ARGB32-8888	A[7:0]A[7:4]	R[7:0]R[7:4]	G[7:0]G[7:4]	B[7:0]B[7:4]

12.9.1.4.1.8.2 DISPC VID VC-1 Range Mapping Unit

The VC-1 range mapping unit is used when the video frame picture is decoded using a VC-1 codec by the video accelerator. It remaps the Y, Cb, and Cr components to the full range (from the reduced data range) before displaying. The unit is used primarily for YUV4:2:0-NV12 (NV21) pixel format, but also can be applied to YUV4:2:2 pixel formats (YUV2 and UYVY).

The VC-1 range mapping unit is enabled by setting the DSS_VID_ATTRIBUTES2 [0] VC1ENABLE bit to 0x1. The VC-1 range mapping must be enabled only for 8 bits/component YUV input data.

Copyright © 2026 Texas Instruments Incorporated

The DSS_VID_ATTRIBUTES2 [3:1] VC1_RANGE_Y and [6:4] VC1_RANGE_CBCR bit fields are two 3-bit values programmed by the user.

The equations for the mapping process are:

Y_{out} = CLIP((((Y_{int} – 128) × (VC1_RANGE_Y + 9) + 4) / 8) + 128)

C_b = CLIP((((C_b – 128) × (VC1_RANGE_CBCR + 9) + 4) / 8) + 128)

C_r = CLIP((((C_r – 128) × (VC1_RANGE_CBCR + 9) + 4) / 8) + 128)

Note

The input and output pixel values are unsigned (Y, Cr, and Cb).

The function CLIP () clips to 0 or 255 when minimum or maximum, respectively, is reached. Otherwise, the resulting output remains identical.

12.9.1.4.1.8.3 DISPC VID Color Look-Up Table (CLUT)

The video pipeline supports a look up table to perform either of the following operations:

- Conversions of BITMAP formats (1, 2, 4, or 8-bit indexed) into RGB24 format (CLUT mode), or
- Inverse gamma correction on non-linear RGB source data (gamma mode)

The look-up table consists of 3 separate 256 x 8-bit memories and is indexed either by the source BITMAP data or by R/G/B component data. The table is loaded through direct register access by writing to the CLUT registers.

The sequence to load the table is:

1. SW writes (only writes are supported) 32-bit values using single access, or burst access in linear increment burst mode, into DSS_VID_CLUT_0 through DSS_VID_CLUT_15 registers. The LSB 24 bits [23:0] are used for the value, and the MSB 8 bits [31:24] are used for the index into the table (see Figure 12-532).
2. Loop to Step 1, if there is a new access to the CLUT registers. Software can access other registers between two accesses to the CLUT registers.

SW needs to ensure that there is no visible effect when modifying the table, since it is not under hardware control.

The usage of the look-up table in CLUT mode is activated when a BITMAP format is selected in the DSS_VID_ATTRIBUTES[6:1] FORMAT register bit-field.

The usage of the look-up table for RGB inverse gamma correction can be enabled by setting DSS_VID_ATTRIBUTES[30] GAMMAINVERSION register bit.

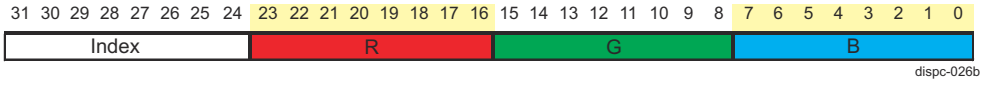


Figure 12-532. DISPC VID CLUT/Gamma Data Memory Organization

Note

Scaling and color look-up table features are mutually exclusive. If color look-up feature is enabled, then video pipeline scaler has to be disabled.

12.9.1.4.1.8.4 DISPC VID Chrominance Resampling

12.9.1.4.1.8.4.1 Chrominance Resampling for VID Pipeline

The chrominance resampling from 8-bpc or 10-bpc to 12-bit color components is always performed using filtering (the scaler unit filter). Chrominance resampling and rescaling can be combined to support native rescaling of YUV formats.

Copyright © 2026 Texas Instruments Incorporated

The usage of the scaler unit for resampling the chrominance of YUV420 and YUV422 is shown in Figure 12-533 and Figure 12-534, respectively. The settings of the scaler unit to perform chrominance resampling are described in *DISPC VID Scaler Unit*.

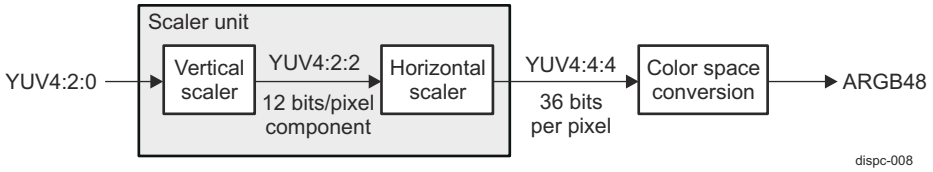


Figure 12-533. DISPC VID YUV420 to ARGB48 Using Scaler Unit for Resampling Chrominance

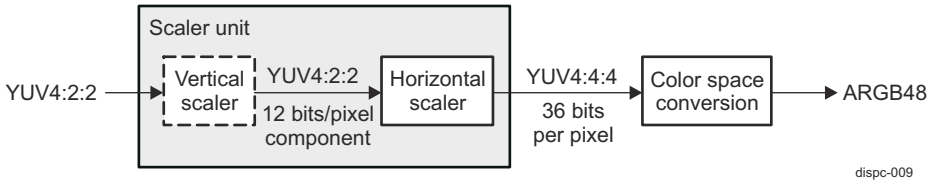


Figure 12-534. DISPC VID YUV422 to ARGB48 Using Scaler Unit for Resampling Chrominance

The vertical scaler is not a mandatory block for resampling the chrominance of YUV422 data, as shown in Figure 12-534.

12.9.1.4.1.8.4.2 Chrominance Resampling for VIDL1 Pipeline

VIDL1 pipeline does not include a scaler unit. Chroma upsampling is done using dedicated YUV420 to YUV422, and YUV422 to YUV444 chroma upsamplers.

VIDL1 pipeline converts YUV420 (chroma) data to YUV422 (chroma) using a simple vertical average filter (average of two adjacent chroma lines to generate a missing line except for the very bottom line which is generated by repeating the previous chroma line). If the video image is a sub-image of a larger video frame data, then the vertical Luma line offset from the top should be an even number.

VIDL1 pipeline converts YUV422 data to YUV444 format using a 4-tap filter (implementing the Catmull-Rom algorithm) to generate missing chroma data as shown below:

Cout[2*i] = Cin[i]

Cout[2*i+1] = CLIP (-1/16*Cin[i-1] + 9/16*Cin[i] + 9/16*Cin[i+1] – 1/16*Cin[i+2])

The missing chroma data is generated as a simple weighted average of the surrounding 4 chroma values, which is equivalent to a 4x1 filter kernel with values: [-1/16, 9/16, 9/16, -1/16]. In this algorithm, edge effects are treated by repeating the first and last pixel.

12.9.1.4.1.8.5 DISPC VID Scaler Unit

Note

Only VID pipeline includes a resizer unit (scaler). VIDL1 pipeline does not include a scaler.

The programmable scaler filter works with all supported video formats, including formats with alpha channel. Alpha channel is scaled with the same parameters as RGB color components. For the YUV formats, Y and Cb/Cr are processed independently. An RGB 64-bit source (16 bits per component) is truncated to 8-bit, if scaler is enabled. Otherwise, it will be truncated to ARGB48 format in the scaler bypass mode. 10-bit/12-bit YUV source is also truncated to 8-bit, since the scaler is enabled to either upsample the chroma component and/or resize the YUV frame data directly.

Copyright © 2026 Texas Instruments Incorporated

The filter is based on a finite impulse response (FIR) filter with 16 phases. The filter is a 5-tap for horizontal filtering, and can be configured for 3 or 5 taps for vertical filtering. The filtering can be used for various processing:

- Up-sampling of the picture
- Down-sampling of the picture
- Anti-aliasing reduction
- Chrominance resampling in case of YUV data formats

The following limitations must be considered:

- The up-sampling ratio is up to x16.
- The down-sampling ratio using 3-tap configuration is down to x0.5 for RGB format.
- The down-sampling ratio using 5-tap configuration is down to x0.25 for RGB format.

Note

The user must set the correct size and position of the original video before resize in order for the up-sampled/down-sampled video to be displayed inside the screen boundaries.

Figure 12-535 shows an example of video up-sampling, with corresponding video window attributes.

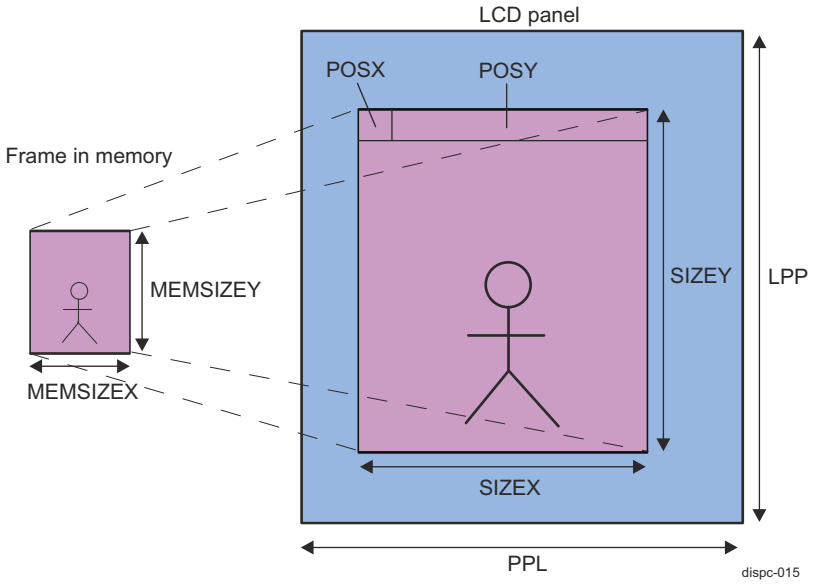


Figure 12-535. DISPC Video Window Attributes

The video window attributes can be configured in the following register bit-fields:

- Source data format (input to the scaler unit) in DSS_VID_ATTRIBUTES [6-1] FORMAT bit-field
- Video picture width in system memory (frame buffer) in DSS_VID_PICTURE_SIZE [11-0] MEMSIZEX bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed except for YUV 422 formats. In case of YUV2422 and UYVY422 formats the width has to be a multiple of 2 pixels.
- Video picture height in system memory (frame buffer) in DSS_VID_PICTURE_SIZE [27-16] MEMSIZEY bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed.
- Video window width at pipeline output (after resizing) in DSS_VID_SIZE [11-0] SIZEX bit-field. The window width is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed up to the screen width minus the horizontal start location of the window on the screen.

Copyright © 2026 Texas Instruments Incorporated

- Video window height at pipeline output (after resizing) in DSS_VID_SIZE [27-16] SIZEY bit-field. The window height is from 1 up to 4096 pixels. All the integer values in the range [1:4096] are allowed up to the screen height minus the vertical start location of the window on the screen.
- Video window overlay X-position in DSS_OVR1_ATTRIBUTES_0 [17-6] POSX bit-field. See Section 12.9.1.4.1.9, *DISPC Overlay Managers*.
- Video window overlay Y-position in DSS_OVR1_ATTRIBUTES_0 [30-19] POSY bit-field. See Section 12.9.1.4.1.9, *DISPC Overlay Managers*.
- For configuration of LPP and PPL video port output display parameters, see Section 12.9.1.4.1.10.7, *DISPC VP Timing Generator and Display Panel Settings*.

For vertical up-sampling and down-sampling in a 3-tap configuration, the equations are:

For RGB formats

$$Aout(n)=\left(\sum_{i=-1}^{i=1}Ci(\Phi)\times Ain(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Rout(n)=\left(\sum_{i=-1}^{i=1}Ci(\Phi)\times Rin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Gout(n)=\left(\sum_{i=-1}^{i=1}Ci(\Phi)\times Gin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Bout(n)=\left(\sum_{i=-1}^{i=1}Ci(\Phi)\times Bin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

For YUV formats

$$Yout(n)=\left(\sum_{i=-1}^{i=1}Cyi(\Phi y)\times Yin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Cbout(n)=\left(\sum_{i=-1}^{i=1}Cci(\Phi c)\times Cbin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Crout(n)=\left(\sum_{i=-1}^{i=1}Cci(\Phi c)\times Crin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

Component	Vertical	Horizontal
Cwidth (Coefficient Width)	10 bits	10 bits
InWidth (Input Width)	10 bits	12 bits
OutWidth (Output Width)	12 bits	12 bits

dispc-013 (43)

For vertical and horizontal up-sampling and down-sampling in a 5-tap configuration, the equations are:

For RGB formats

$$Aout(n)=\left(\sum_{i=-2}^{i=2}Ci(\Phi)\times Ain(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Rout(n)=\left(\sum_{i=-2}^{i=2}Ci(\Phi)\times Rin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Gout(n)=\left(\sum_{i=-2}^{i=2}Ci(\Phi)\times Gin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Bout(n)=\left(\sum_{i=-2}^{i=2}Ci(\Phi)\times Bin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

For YUV formats

$$Yout(n)=\left(\sum_{i=-2}^{i=2}Cyi(\Phi y)\times Yin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Cbout(n)=\left(\sum_{i=-2}^{i=2}Cci(\Phi c)\times Cbin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

$$Crout(n)=\left(\sum_{i=-2}^{i=2}Cci(\Phi c)\times Crin(n+i)\right)>>(Cwidth-(OutWidth-InWidth)-1)$$

Component	Vertical	Horizontal
Cwidth (Coefficient Width)	10 bits	10 bits
InWidth (Input Width)	10 bits	12 bits
OutWidth (Output Width)	12 bits	12 bits

dispc-012 (44)

Note

The pixel (n + 1) is the previous pixel with respect to pixel (n). The line (n + 1) is the previous line with respect to line (n).

The coefficients Ci() depend on the phase between input and output pixels.

The coefficients are different for Y and Cr/Cb filtering because the calculations are independent due to the chrominance resampling for YUV422 and YUV420.

First, the vertical filter is applied to the encoded input pixel data, and then the horizontal filter is applied on the resulting pixel values to generate the output pixel values. The vertical input of the filter consists of:

Copyright © 2026 Texas Instruments Incorporated

- Six lines of 1280 × 32 bits for 5-tap configuration
- Three lines of 2560 x 32 bits for 3-tap configuration

Table 12-474 lists some of the scaler supported configurations.

Table 12-474. DISPC VID Line Buffer Width for Scaler Unit

Pixel Format	Maximum Input Width (Pixels) for 5-tap	Maximum Input Width (Pixels) for 3-tap
32 bits per pixel (ARGB32-8888, ARGB-2101010, etc.)	1280 pixels wide (1)	2560 pixels wide
YUV420, YUV422	2560 pixels wide	4096 pixels wide

(1) For the 5-tap configuration, the 6th video line is used on the output of the horizontal scaler, so that horizontal down-scaling can be supported with a minimum clock ratio equal to 1. The maximum output width is limited to 1280 pixels in order to be able to support clock ratio of 1 due to the 6th video line buffer. The 6th video line is also used in order to start the next output line generation while the video pipeline output is being stalled by the overlay manager (either due to display in blanking period and/or in background pixel period). The 6th line buffer is actually 48-bit wide to allow storage of ARGB48 format pixel data.

At the beginning of frame scaling processing, the first line may be duplicated multiple times depending on the initial vertical phase programmed for the poly-phase filter.

At the end of frame scaling processing the last line is duplicated, if the scaling logic requires loading more lines and the last line has been reached.

Similarly, the first pixel may be duplicated multiple times depending on the initial horizontal phase programmed for the poly-phase filter. The last pixel is duplicated, if the scaling logic requires loading more pixels and the last pixel has been reached.

The programmable coefficients of the polyphase filters are signed 10-bit values (except for the central coefficient, which is unsigned). The vertical video scaler has a 10-bit input and a 12-bit output. The horizontal scaling stage takes the resulting 12-bit input and produces 12-bit output.

Figure 12-536 and Figure 12-537 show the scaler macro-architecture for components A, R, G, and B. Figure 12-538 and Figure 12-539 show the scaler macro-architecture for components Y, Cr, and Cb.

Note

The scaling and CSC clipping is set by the same bit, DSS_VID_ATTRIBUTES[11] FULLRANGE.

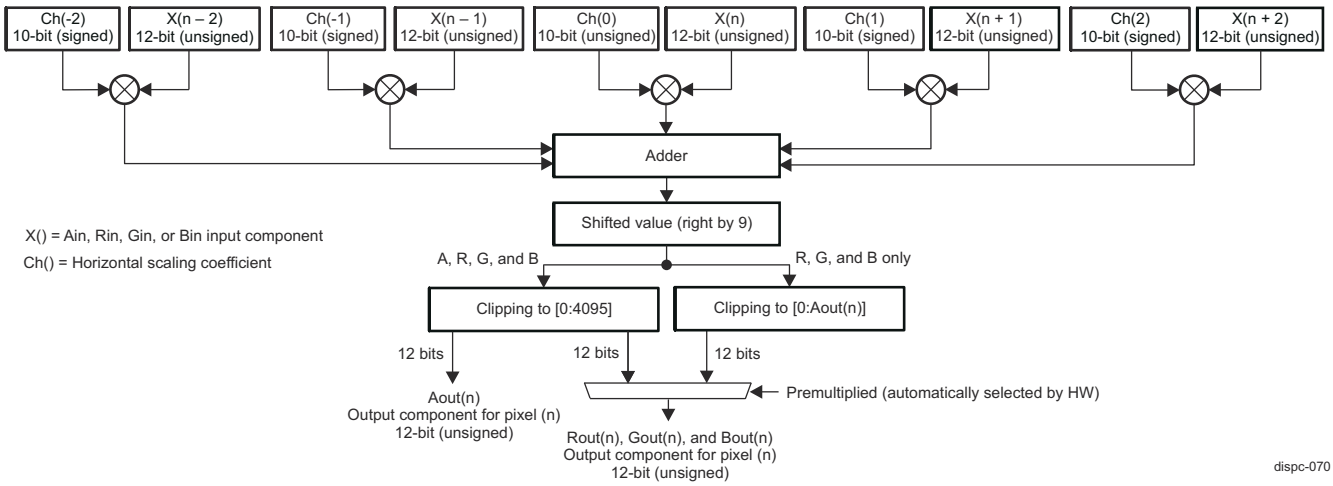


Figure 12-536. DISPC VID Macro-Architecture of the Horizontal Scaling for A, R, G, and B Components (5-tap Restriction)

Copyright © 2026 Texas Instruments Incorporated

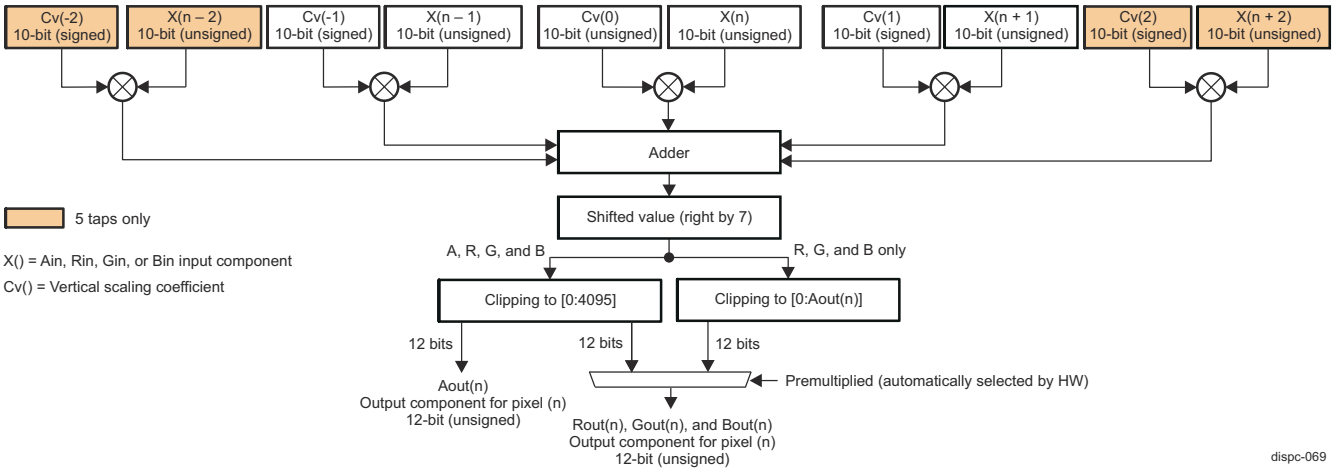


Figure 12-537. DISPC VID Macro-Architecture of the Vertical Scaling for A, R, G, and B Components (5 and 3 taps)

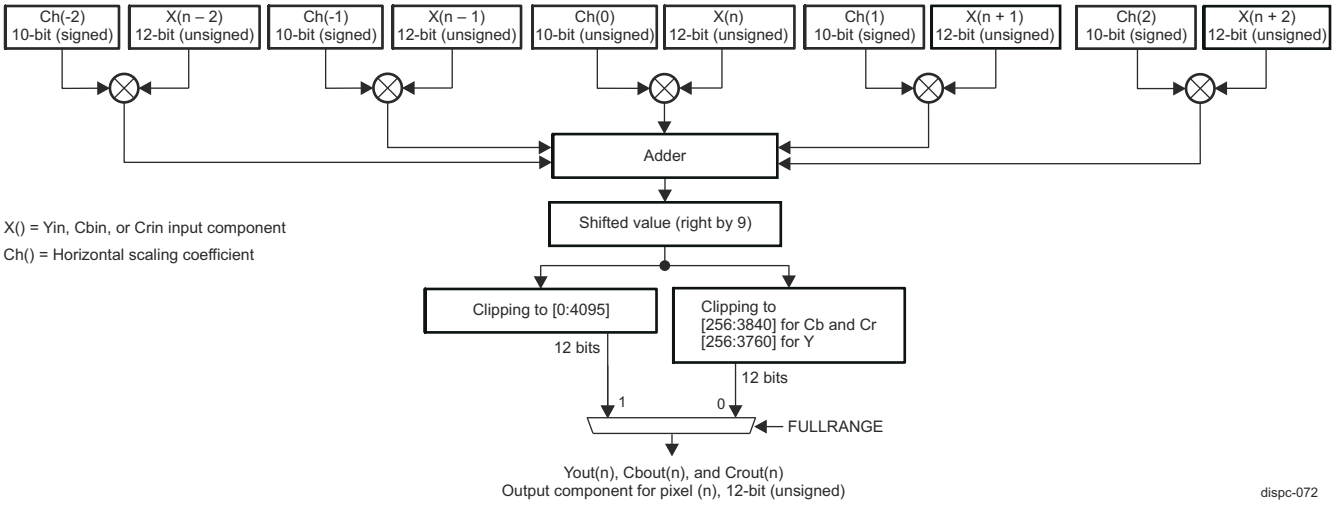


Figure 12-538. DISPC VID Macro-Architecture of the Horizontal Scaling for Y, Cr, and Cb Components (5-tap Restriction)

Copyright © 2026 Texas Instruments Incorporated

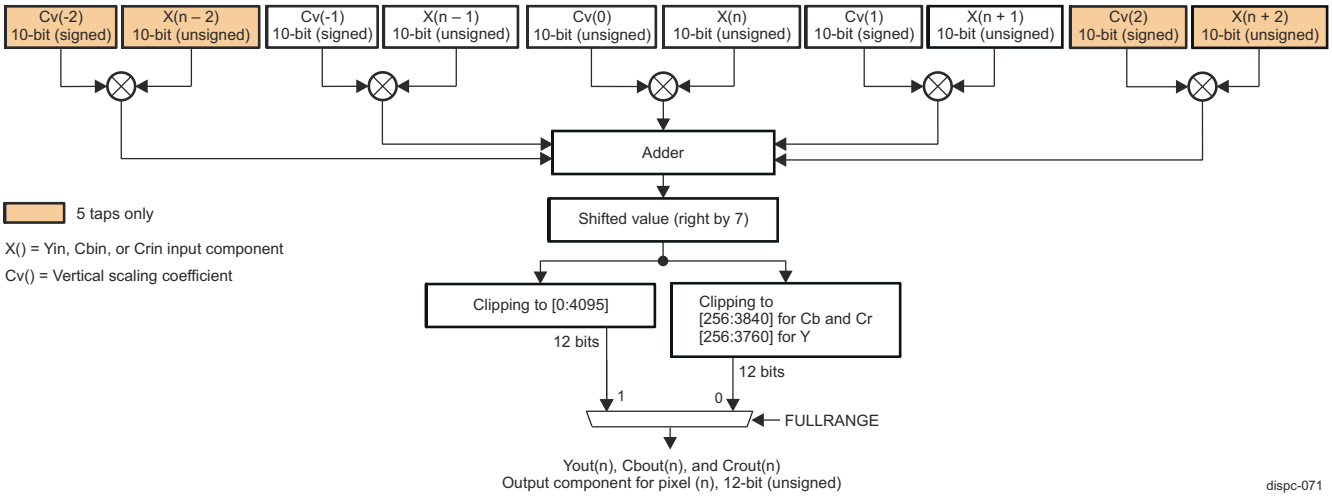


Figure 12-539. DISPC VID Macro-Architecture of the Vertical Scaling for Y, Cr, and Cb Components (5 and 3 taps)

Table 12-475 list the register fields in the function of the coefficients for the VID horizontal scaler in the DSS_VID_FIR_COEF_H0_0 to DSS_VID_FIR_COEF_H0_8, and DSS_VID_FIR_COEF_H12_0 to DSS_VID_FIR_COEF_H12_15 registers.

Table 12-475. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in VID Horizontal Scaler

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
	Signed coefficient [29-20] FIRHC2 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Unsigned central coefficient [9-0] FIRHC0 bitfield	Signed coefficient [19-10] FIRHC1 bitfield	Signed coefficient [29-20] FIRHC2 bitfield
0	DSS_VID_FIR_COEF_H12_0	DSS_VID_FIR_COEF_H12_0	DSS_VID_FIR_COEF_H0_0	DSS_VID_FIR_COEF_H12_0	DSS_VID_FIR_COEF_H12_0
1	DSS_VID_FIR_COEF_H12_1	DSS_VID_FIR_COEF_H12_1	DSS_VID_FIR_COEF_H0_1	DSS_VID_FIR_COEF_H12_15	DSS_VID_FIR_COEF_H12_15
2	DSS_VID_FIR_COEF_H12_2	DSS_VID_FIR_COEF_H12_2	DSS_VID_FIR_COEF_H0_2	DSS_VID_FIR_COEF_H12_14	DSS_VID_FIR_COEF_H12_14
3	DSS_VID_FIR_COEF_H12_3	DSS_VID_FIR_COEF_H12_3	DSS_VID_FIR_COEF_H0_3	DSS_VID_FIR_COEF_H12_13	DSS_VID_FIR_COEF_H12_13
4	DSS_VID_FIR_COEF_H12_4	DSS_VID_FIR_COEF_H12_4	DSS_VID_FIR_COEF_H0_4	DSS_VID_FIR_COEF_H12_12	DSS_VID_FIR_COEF_H12_12
5	DSS_VID_FIR_COEF_H12_5	DSS_VID_FIR_COEF_H12_5	DSS_VID_FIR_COEF_H0_5	DSS_VID_FIR_COEF_H12_11	DSS_VID_FIR_COEF_H12_11
6	DSS_VID_FIR_COEF_H12_6	DSS_VID_FIR_COEF_H12_6	DSS_VID_FIR_COEF_H0_6	DSS_VID_FIR_COEF_H12_10	DSS_VID_FIR_COEF_H12_10
7	DSS_VID_FIR_COEF_H12_7	DSS_VID_FIR_COEF_H12_7	DSS_VID_FIR_COEF_H0_7	DSS_VID_FIR_COEF_H12_9	DSS_VID_FIR_COEF_H12_9
8	DSS_VID_FIR_COEF_H12_8	DSS_VID_FIR_COEF_H12_8	DSS_VID_FIR_COEF_H0_8	DSS_VID_FIR_COEF_H12_8	DSS_VID_FIR_COEF_H12_8
9	DSS_VID_FIR_COEF_H12_9	DSS_VID_FIR_COEF_H12_9	DSS_VID_FIR_COEF_H0_7	DSS_VID_FIR_COEF_H12_7	DSS_VID_FIR_COEF_H12_7
10	DSS_VID_FIR_COEF_H12_10	DSS_VID_FIR_COEF_H12_10	DSS_VID_FIR_COEF_H0_6	DSS_VID_FIR_COEF_H12_6	DSS_VID_FIR_COEF_H12_6
11	DSS_VID_FIR_COEF_H12_11	DSS_VID_FIR_COEF_H12_11	DSS_VID_FIR_COEF_H0_5	DSS_VID_FIR_COEF_H12_5	DSS_VID_FIR_COEF_H12_5
12	DSS_VID_FIR_COEF_H12_12	DSS_VID_FIR_COEF_H12_12	DSS_VID_FIR_COEF_H0_4	DSS_VID_FIR_COEF_H12_4	DSS_VID_FIR_COEF_H12_4

Copyright © 2026 Texas Instruments Incorporated

Table 12-475. DISPC Register Fields Associated to Coefficients for ARGB and Y Configuration in VID Horizontal Scaler (continued)

Phases	Ch(2)	Ch(1)	Ch(0)	Ch(-1)	Ch(-2)
13	DSS_VID_FIR_COEF_H12_13	DSS_VID_FIR_COEF_H12_13	DSS_VID_FIR_COEF_H0_3	DSS_VID_FIR_COEF_H12_3	DSS_VID_FIR_COEF_H12_3
14	DSS_VID_FIR_COEF_H12_14	DSS_VID_FIR_COEF_H12_14	DSS_VID_FIR_COEF_H0_2	DSS_VID_FIR_COEF_H12_2	DSS_VID_FIR_COEF_H12_2
15	DSS_VID_FIR_COEF_H12_15	DSS_VID_FIR_COEF_H12_15	DSS_VID_FIR_COEF_H0_1	DSS_VID_FIR_COEF_H12_1	DSS_VID_FIR_COEF_H12_1

Note

In Table 12-475, the cells without color are duplicated from the grey cells.

Similar table approach applies to the vertical scaler (registers DSS_VID_FIR_COEF_V0_0 to DSS_VID_FIR_COEF_V0_8, and DSS_VID_FIR_COEF_V12_0 to DSS_VID_FIR_COEF_V12_15 are used).

Similar table approach applies to the coefficients for Cb/Cr filtering in case of YUV format (registers DSS_VID_FIR_COEF_H0_C_0 to DSS_VID_FIR_COEF_H0_C_8, and DSS_VID_FIR_COEF_H12_C_0 to DSS_VID_FIR_COEF_H12_C_15, and DSS_VID_FIR_COEF_V0_C_0 to DSS_VID_FIR_COEF_V0_C_8 and DSS_VID_FIR_COEF_V12_C_0 to DSS_VID_FIR_COEF_V12_C_15 are used) .

The VID scaler unit vertical and/or horizontal sampling is selected by configuring the DSS_VID_ATTRIBUTES[8-7] RESIZEENABLE bit field.

Prior to enabling the video up/down-sampling block a valid configuration has to be set by the user. After configuring the required VID registers change the DSS_VP_CONTROL[5] GOBIT register bit of the video port the video pipeline is associated with. The software has to wait before setting the GO bit that the hardware has reset the bit. The software reset is not recommended since the application cannot guarantee to be able to reset it before the H/W.

The following fields define the configuration of the video up-sampling/down-sampling block in the VID pipeline for ARGB and YUV formats:

- Vertical upsampling and downsampling increment value in the [23-0] FIRVINC bit field of DSS_VID_FIRV (for ARGB and Y) and DSS_VID_FIRV2 (for CbCr) registers. The unsigned integer value range is 2^{23}. Software calculates the value using the following equation:

$$FIRVINC = 2^{21} * \left(\frac{MEMSIZEY+1}{SIZEY+1}\right)$$

dispc-066

- Horizontal upsampling and downsampling increment value in the [23-0] FIRHINC bit field of the DSS_VID_FIRH (for ARGB and Y) and DSS_VID_FIRH2 (for CbCr) registers. The unsigned integer value range is [1:16384]. Software calculates the value using the following equation:

$$FIRHINC = 2^{21} * \left(\frac{MEMSIZEH+1}{SIZEH+1}\right)$$

dispc-067

- Vertical up/downsampling accumulator value in the [23-0] VERTICALACCU bit field of the DSS_VID_ACCUV_0 and DSS_VID_ACCUV_1 (for ARGB and Y), and DSS_VID_ACCUV2_0 and DSS_VID_ACCUV2_1 (for CbCr) registers. The accumulator value indicates on which phase the vertical filtering starts. The DSS_VID_ACCUV_0 register is used for progressive output, while for interlace output both DSS_VID_ACCUV_0 and DSS_VID_ACCUV_1 registers are used. The DSS_VID_ACCUV2_0 and DSS_VID_ACCUV2_1 registers are used in the same manner for progressive or interlace output.

Copyright © 2026 Texas Instruments Incorporated

- Vertical upsampling and downsampling line buffer configuration via the DSS_VID_ATTRIBUTES[21] VERTICALTAPS bit: The default value at reset time is 0x0 (3-tap configuration is used). If the bit field is reset, the 3-tap configuration is used.
- Horizontal upsampling and downsampling accumulator value in the [23-0] HORIZONTALACCU bit field of the DSS_VID_ACCUH_0 and DSS_VID_ACCUH_1 (for ARGB and Y), and DSS_VID_ACCUH2_0 and DSS_VID_ACCUH2_1 (for CbCr) registers. The accumulator value indicates on which phase the horizontal filtering starts. The register DSS_VID_ACCUH_0 is used for progressive output, while for interlace output both DSS_VID_ACCUH_0 and DSS_VID_ACCUH_1 registers are used. The DSS_VID_ACCUH2_0 and DSS_VID_ACCUH2_1 are used in the same manner for progressive or interlace output.

Table 12-476 lists the scaler vertical and horizontal accumulator values and phases.

Table 12-476. DISPC VID Scaler Vertical and Horizontal Accumulator Phases

Accumulator Value (MSB bits)	Phases f
0	0
256 or -3840	1
512 or -3584	2
768 or -3328	3
1024 or -3072	4
1280 or -2816	5
1536 or -2560	6
1792 or -2304	7
2048 or -2048	8
2304 or -1792	9
2560 or -1536	10
2816 or -1280	11
3072 or -1024	12
3328 or -768	13
3584 or -512	14
3840 or -256	15

- Vertical upsampling and downsampling central coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling central coefficients are defined in the DSS_VID_FIR_COEF_V0_0 to DSS_VID_FIR_COEF_V0_8 registers. There are 9 registers for the 16 phases with 1 coefficient for each of them. Symmetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).
  - Four CbCr, the vertical upsampling and downsampling central coefficients are set in DSS_VID_FIR_COEF_V0_C_0 to DSS_VID_FIR_COEF_V0_C_8 registers.
- Vertical upsampling and downsampling coefficients:
  - For ARGB and Y, the vertical upsampling and downsampling coefficients are defined in the DSS_VID_FIR_COEF_V12_0 to DSS_VID_FIR_COEF_V12_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the vertical up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the vertical upsampling and downsampling coefficients are set in DSS_VID_FIR_COEF_V12_C_0 to DSS_VID_FIR_COEF_V12_C_15 registers.
- Horizontal upsampling and downsampling central coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling central coefficients are defined in the DSS_VID_FIR_COEF_H0_0 to DSS_VID_FIR_COEF_H0_8 registers. There are 9 registers for the 16 phases with 1 coefficient for each of them. Symmetrical implementation is used, so only 9 coefficients are used. Each register contains one 10-bit unsigned coefficient (the central one).

Copyright © 2026 Texas Instruments Incorporated

  - Four CbCr, the horizontal upsampling and downsampling central coefficients are set in DSS_VID_FIR_COEF_H0_C_0 to DSS_VID_FIR_COEF_H0_C_8 registers.
- Horizontal upsampling and downsampling coefficients:
  - For ARGB and Y, the horizontal upsampling and downsampling coefficients are defined in the DSS_VID_FIR_COEF_H12_0 to DSS_VID_FIR_COEF_H12_15 registers. There are 16 registers for the 16 phases with 2 coefficient for each of them, so a total of 32 programmable coefficients for the horizontal up/down-sampling block. Each register contains two 10-bit signed coefficients.
  - Four CbCr, the horizontal upsampling and downsampling coefficients are set in DSS_VID_FIR_COEF_H12_C_0 to DSS_VID_FIR_COEF_H12_C_15 registers.

The YUV filtering is based on the equations of the ARGB filtering. In addition to the registers used for ARGB filtering configuration, a second set of registers for filtering is used. The first set of registers is used for Y configuration (instead of ARGB configuration) and the second set of registers is used for CbCr filtering configuration. The two sets of registers can be the same when the YUV format is not converted to RGB after filtering. When the RGB conversion is required after filtering, then the chrominance needs to be re-sampled with a different filtering configuration because:

- Chrominance samples are offset to the luminance samples. That is, DSS_VID_FIRH2 and DSS_VID_FIRV2, and DSS_VID_ACCUV2_0/DSS_VID_ACCUV2_1 and DSS_VID_ACCUH2_0/DSS_VID_ACCUH2_1 registers need to be configured differently than DSS_VID_FIRH and DSS_VID_FIRV, and DSS_VID_ACCUH_0/DSS_VID_ACCUH_1 and DSS_VID_ACCUV_0/DSS_VID_ACCUV_1 registers used for the luminance filtering.
- Chrominance is sub-sampled by two horizontally only in case of YUV422, and horizontally and vertically in case of YUV420. The coefficients for the vertical chrominance re-sampling are identical to the coefficients for vertical luminance filtering in case of YUV422. The coefficients for the horizontal and vertical chrominance re-sampling are different than the ones for luminance filtering in case of YUV420.

12.9.1.4.1.8.6 DISPC VID Color Space Conversion YUV to RGB

The Color Space Conversion (CSC) unit converts the video-encoded pixel values from YUV444 format into ARGB48 format (12-bit value per component A, R, G, and B, with A fixed at 0xFFF).

In case of YUV420 or YUV422 formats, a chrominance resampling to YUV444 is performed before converting the YUV into RGB values (see Section 12.9.1.4.1.8.4, *DISPC VID Chrominance Resampling*). The YUV422/ YUV420 to YUV444 chrominance resampling is a pre-processing to the color space conversion.

Figure 12-540 and Figure 12-541 show the 3 × 3 11-bit coefficients used to convert from YUV444 into ARGB48. The value of A component is fixed at 0xFFF in the output. The coefficients are set according to the standard used to encode the pixel data in YUV color space. Table 12-477 summarizes the coefficients with their respective register bit fields.

Table 12-477. DISPC VID CSC - YUV to RGB Register Settings

Coefficients	Register Fields
R_Y	DSS_VID_CSC_COEF0[10-0] C00
R_{Cr}	DSS_VID_CSC_COEF0[26-16] C01
R_{Cb}	DSS_VID_CSC_COEF1[10-0] C02
G_Y	DSS_VID_CSC_COEF1[26-16] C10
G_{Cr}	DSS_VID_CSC_COEF2[10-0] C11
G_{Cb}	DSS_VID_CSC_COEF2[26-16] C12
B_Y	DSS_VID_CSC_COEF3[10-0] C20
B_{Cr}	DSS_VID_CSC_COEF3[26-16] C21
B_{Cb}	DSS_VID_CSC_COEF4[10-0] C22
Y offset	DSS_VID_CSC_COEF5[15-3] PREOFFSET1
Cr offset	DSS_VID_CSC_COEF5[31-19] PREOFFSET2
Cb offset	DSS_VID_CSC_COEF6[15-3] PREOFFSET3

Copyright © 2026 Texas Instruments Incorporated

- Limited video data range conversion

  If the active range for the luminance samples (Y) is [256:3760] and for the chrominance samples (Cb and Cr) is [256:3840], the following equation in Figure 12-540 (based on 11-bit coefficients) must be used to convert the YUV to RGB. To clip the values of R, G, and B output components to the full output data range [0:4095], set the DSS_VID_ATTRIBUTES[11] FULLRANGE bit to 0x1.

$$\begin{bmatrix} R_{OUT} \\ G_{OUT} \\ B_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} R_Y & R_{cr} & R_{cb} \\ G_Y & G_{cr} & G_{cb} \\ B_Y & B_{cr} & B_{cb} \end{bmatrix} * \begin{bmatrix} Y_{IN} - 256 \\ Cr_{IN} - 2048 \\ Cb_{IN} - 2048 \end{bmatrix}$$

dispc-005

Figure 12-540. DISPC VID CSC YCbCr to RGB Equation (Limited Video Range), 12-Bit Outputs

  In Figure 12-540, the offset values (-256, -2048, -2048) are also programmed via corresponding register bit-fields shown in Table 12-477, *DISPC VID CSC - YUV to RGB Register Settings*).

- Full video data range conversion

  If the active range for the luminance samples (Y) and chrominance samples (Cb and Cr) is [0:4095], the following equation in Figure 12-541 (based on 11-bit coefficients) must be used to convert the YUV to RGB. To clip the values of R, G, and B output components to the full output data range [0:4095], set the DSS_VID_ATTRIBUTES[11] FULLRANGE bit to 0x1.

$$\begin{bmatrix} R_{OUT} \\ G_{OUT} \\ B_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} R_Y & R_{cr} & R_{cb} \\ G_Y & G_{cr} & G_{cb} \\ B_Y & B_{cr} & B_{cb} \end{bmatrix} * \begin{bmatrix} Y_{IN} \\ Cr_{IN} - 2048 \\ Cb_{IN} - 2048 \end{bmatrix}$$

dispc-006

Figure 12-541. DISPC VID CSC YCbCr to RGB Equation (Full Video Range), 12-Bit Outputs

  In Figure 12-541, the offset values (0, -2048, -2048) are also programmed via corresponding register bit-fields shown in Table 12-477, *DISPC VID CSC - YUV to RGB Register Settings*).

Note

In case of YUV420-NV21 data, the coefficients for Cr and Cb must be swapped in order to correctly support YUV420-NV21 format.

The scaling and CSC clipping is set by the same bit, DSS_VID_ATTRIBUTES[11] FULLRANGE.

12.9.1.4.1.8.7 DISPC VID Brightness/Contrast/Saturation/Hue Control

The brightness/contrast/saturation controls are implemented in the same Color Space Conversion (CSC) block by making adjustments to the conversion coefficients and input/output offset values as shown in Figure 12-542:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} m*A0 & m*k*B0 & m*k*C0 \\ m*A1 & m*k*B1 & m*k*C1 \\ m*A2 & m*k*B2 & m*k*C2 \end{bmatrix} \begin{bmatrix} Y + Y_offset_adj \\ Cr + Cr_offset \\ Cb + Cb_offset \end{bmatrix} + \begin{bmatrix} R_offset \\ G_offset \\ B_offset \end{bmatrix}$$

dispc-101

Figure 12-542. DISPC VID Brightness/Contrast/Saturation Equation for YUV to RGB

In Figure 12-542 the following terminology is used:

- A/B/C coefficients and the offset parameters are for YUV to RGB conversion. See Table 12-478 below for coefficients and offsets mapping to register fields.

Copyright © 2026 Texas Instruments Incorporated

- The gain term (m) is used for contrast control. The allowed range is: 0 < m <= 2.0 (where m=1 means contrast bypass).
- The gain term (k) is used for saturation control. The allowed range is: 0 < k <= 2.0 (where k=1 means saturation bypass).
- The brightness offset (b) is added to Y_offset_adj. The allowed range (for 8-bit example) is: -128 <= b <= +127 (where b=0 means brightness bypass).

The hue adjustment can also be built into the above equation (Figure 12-542) to comprehend the following Cb/Cr hue angle adjustments:

Cb_hue_adj = (Cb*cosØ + Cr* sinØ)

Cr_hue_adj = (Cr*cosØ - Cb* sinØ), where Ø is the desired hue angle (with Ø=0 meaning hue adjustment bypass)

The final combined matrix equation for controlling brightness/contrast/saturation/hue (BCSH) during YUV to RGB conversion is shown in Figure 12-543

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} m*A0 & m*k*[B0*\cos(Hue)+C0*\sin(Hue)] & m*k*[C0*\cos(Hue)-B0*\sin(Hue)] \\ m*A1 & m*k*[B1*\cos(Hue)+C1*\sin(Hue)] & m*k*[C1*\cos(Hue)-B1*\sin(Hue)] \\ m*A2 & m*k*[B2*\cos(Hue)+C2*\sin(Hue)] & m*k*[C2*\cos(Hue)-B2*\sin(Hue)] \end{bmatrix} \begin{bmatrix} Y+Y_offset_adj \\ Cr+Cr_offset \\ Cb+Cb_offset \end{bmatrix} + \begin{bmatrix} R_offset \\ G_offset \\ B_offset \end{bmatrix}$$

dispc-102

Figure 12-543. DISPC VID Brightness/Contrast/Saturation/Hue Equation for YUV to RGB

Gain and offset terms in the above equation (Figure 12-543) are programmed in the hardware registers as adjusted CSC coefficients (11-bit signed) and offset values (13-bit signed) to control the brightness, contrast, saturation and hue. Mapping of the coefficients and offsets to register fields is provided in Table 12-478. Outputs are clipped to keep the output values in the proper range.

Table 12-478. DISPC VID BCSH Register Settings for YUV to RGB

Coefficients	Register Fields
A0	DSS_VID_CSC_COEF0[10-0] C00
B0	DSS_VID_CSC_COEF0[26-16] C01
C0	DSS_VID_CSC_COEF1[10-0] C02
A1	DSS_VID_CSC_COEF1[26-16] C10
B1	DSS_VID_CSC_COEF2[10-0] C11
C1	DSS_VID_CSC_COEF2[26-16] C12
A2	DSS_VID_CSC_COEF3[10-0] C20
B2	DSS_VID_CSC_COEF3[26-16] C21
C2	DSS_VID_CSC_COEF4[10-0] C22
Y offset	DSS_VID_CSC_COEF5[15-3] PREOFFSET1
Cr offset	DSS_VID_CSC_COEF5[31-19] PREOFFSET2
Cb offset	DSS_VID_CSC_COEF6[15-3] PREOFFSET3
R offset	DSS_VID_CSC_COEF6[31-19] POSTOFFSET1
G offset	DSS_VID_CSC_COEF7[15-3] POSTOFFSET2
B offset	DSS_VID_CSC_COEF7[31-19] POSTOFFSET3

For RGB input formats, the same CSC block can be used to adjust contrast, brightness, saturation, and hue using the matrix equation from Figure 12-544.

Copyright © 2026 Texas Instruments Incorporated

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \begin{bmatrix} m*A0 & m*k*[B0*\cos(Hue)+C0*\sin(Hue)] & m*k*[C0*\cos(Hue)-B0*\sin(Hue)] \\ m*A1 & m*k*[B1*\cos(Hue)+C1*\sin(Hue)] & m*k*[C1*\cos(Hue)-B1*\sin(Hue)] \\ m*A2 & m*k*[B2*\cos(Hue)+C2*\sin(Hue)] & m*k*[C2*\cos(Hue)-B2*\sin(Hue)] \end{bmatrix} \begin{bmatrix} a0 & b0 & c0 \\ a1 & b1 & c1 \\ a2 & b2 & c2 \end{bmatrix} \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} + \begin{bmatrix} R_offset \\ G_offset \\ B_offset \end{bmatrix}$$

Coefficients a/b/c are related to A/B/C (for RGB to YUV and YUV to RGB conversions) as follows:

$$\begin{bmatrix} a0 & b0 & c0 \\ a1 & b1 & c1 \\ a2 & b2 & c2 \end{bmatrix} = \begin{bmatrix} A0 & B0 & C0 \\ A1 & B1 & C1 \\ A2 & B2 & C2 \end{bmatrix} -1$$

dispc-103

Figure 12-544. DISPC VID Brightness/Contrast/Saturation/Hue Equation for RGB Input

In Figure 12-544 the following specifics apply:

- A/B/C coefficients and the RGB offset parameters mapping to register fields is provided in Table 12-478.
- The gain term (m) is used for contrast control. The allowed range is: 0 < m <= 2.0 (where m=1 means contrast bypass).
- The gain term (k) is used for saturation control. The allowed range is: 0 < k <= 2.0 (where k=1 means saturation bypass).
- The brightness is controlled by the RGB offset values.
- The Hue angle is used to adjust hue. Hue Ø=0 means hue adjustment bypass.

Outputs are clipped to keep the output values in the proper range.

12.9.1.4.1.8.8 DISPC VID Luma Key Support

The video pipeline supports luma key transparency for Blue-ray video layer composition. When enabled, Y value of each pixel will be checked against luma key range values. The range values are defined in DSS_VID_LUMAKEY register fields, as follows: [11-0] LUMAKEYMIN < Y < [27-16] LUMAKEYMAX. If this condition is true, then the pixel alpha value will be forced to zero (transparent) to make the pixel transparent during the blending process in the overlay manager.

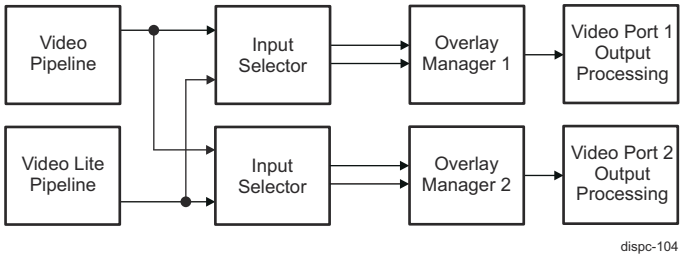
12.9.1.4.1.9 DISPC Overlay Managers

An overlay manager (OVR) is responsible for composting selected input layers together to generate the final display output frame. DISPC implements two identical overlay managers (see *DISPC Architecture Overview*):

- OVR1, with output connected to Video Port (VP1).
- OVR2, with output connected to Video Port (VP2).

Each OVR is preceded by a programmable input pipeline selector which re-maps the outputs of the selected input pipeline according to the overlay manager z-order configuration.

The output of each OVR is connected to the Color Phase Rotation (CPR) unit through a gamma table of the corresponding video port.



dispc-104

Figure 12-545. DISPC Overlay Manager and Input Selector

Note

The DISPC overlay managers, OVR1 and OVR2, will be commonly referred to as *overlay manager (OVR)* in the following sections.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.9.1 DISPC Overlay Input Selector

The overlay input selector before the overlay manager is a n-input to n-output crossbar switch. As such, any input can be connected to any output in the selector. The selection is based on the z-order layer selection in the overlay configuration through the following register fields:

- DSS_OVR1_ATTRIBUTES_0[4-1] CHANNELIN field for layer 0, z-order 0
- DSS_OVR1_ATTRIBUTES_1[4-1] CHANNELIN field for layer 1, z-order 1

The z-order determines the order in which the selected layers are blended together to generate the final output by the overlay manager:

- The lowest enabled order input is the bottom-most layer, just above the background color
- The highest enabled order input is the top-most layer

The OVR input selector also passes the following attributes from the selected input pipeline to the corresponding selector output port:

- Source pipeline size attributes:
  - Number of pixels per line (from DSS_VID_SIZE [11-0] SIZEX register field)
  - Number of lines (from DSS_VID_SIZE [27-16] SIZEY register field)
- Source pipeline global blending level (from DSS_VID_GLOBAL_ALPHA register)

Then, the overlay manager uses the above listed attributes as input layer attributes along with POSX and POSY overlay manager configuration parameters. POSX and POSY can be configured as follows:

- For layer 0, z-order 0, in DSS_OVR1_ATTRIBUTES_0[17-6] POSX and [30-19] POSY register fields
- For layer 1, z-order 1, in DSS_OVR1_ATTRIBUTES_1[17-6] POSX and [30-19] POSY register fields

Note

The output of an input pipeline can be mapped to more than one overlay manager simultaneously, if and only if the following conditions are true:

- All video port timing generators controlling the overlay managers are identically programmed (same frame width/height with same blanking parameters running at same frame rate) and clocked by the same pixel clock source, and ...
- The pipeline output is positioned on each display output at the same frame position.

The software is responsible for managing the pipeline assignment and setting up the video port timing generators properly in order to avoid any resource conflicts that may cause the DSS to lock up (sync loss).

12.9.1.4.1.9.2 DISPC Overlay Mechanism

The overlay mechanism consists of displaying one layer (VID and/or VIDL1 pipeline layers) using:

- A priority rule based on the display Z-order (selected by configuring the overlay input selector, as described in Section 12.9.1.4.1.9.2)
- Transparency color keys configuration (destination and source transparency)
- Alpha blending values (using the alpha component of each pixel or a global alpha value set by the user)
- Size and position attributes of the source window data, as described in Section 12.9.1.4.1.9.2

The overlay manager is configured using the Z-order parameter. The Z-order value defined for each pipeline indicates the visibility order to the window on the screen. If the Z-order value of window A is lower than the Z-order to layer B, then layer A is displayed below layer B. The transparency color keys and the alpha blending factors are then used to blend the layers together (see *Overlay Alpha Blender*, and *Overlay Transparency Color Keys*).

Figure 12-546 gives an example of how input pipeline frame outputs are merged together to generate the final display output.

Copyright © 2026 Texas Instruments Incorporated

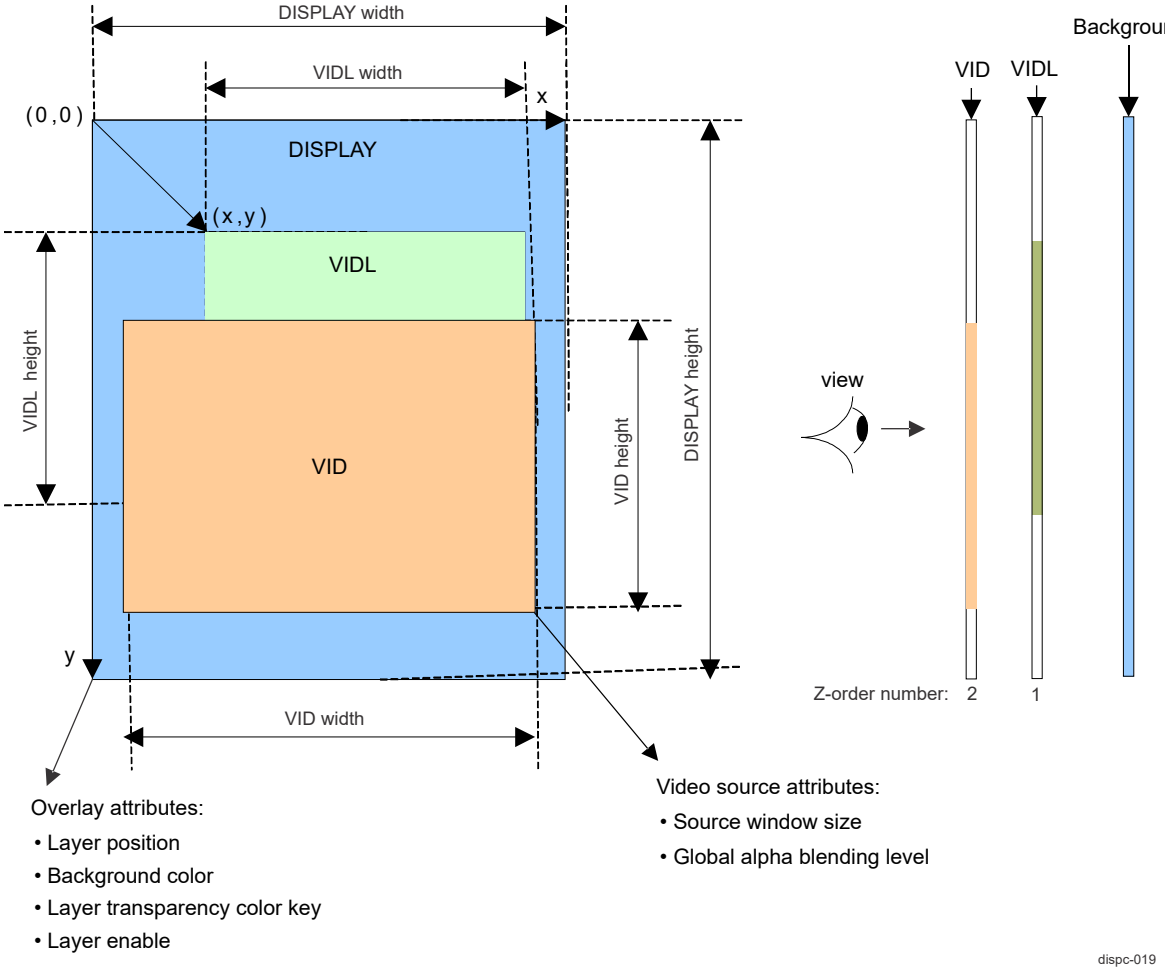


Figure 12-546. DISPC Overlay Example and Display Attributes

As shown in Figure 12-546, each input pipeline generates one rectangular sub-frame output in ARGB48 format. The overlay is responsible for positioning the sub-frame in the display output frame (specified by DSS_VP1_SIZE_SCREEN[11-0] PPL and [27-16] LPP video port register fields), using the overlay window position parameters (specified by DSS_OVR1_ATTRIBUTES_0 DSS_OVR1_ATTRIBUTES_1[17-6] POSX and [30-19] POSY overlay register fields), and the window size parameters (specified by DSS_VID_SIZE[11-0] SIZEX and [27-16] SIZEY pipeline register fields). When the overlay manager does not require the input pipeline data (either because it is currently outputting the background color pixel or it is being stalled by the VP output module), the overlay manager stalls the input pipeline.

When there are no video-encoded pixels at a specific position, the programmable solid background color appears. The solid background color is set in the DSS_OVR1_DEFAULT_COLOR and DSS_OVR1_DEFAULT_COLOR2 registers.

The entire pixels of the video window have to be inside the display screen. Depending on the width of the buffer to be displayed in the video layer and the position, the width must be adjusted by software to limit the right edge of the window inside the display screen. The same is also true for the video layer height in the input pipe line configuration.

12.9.1.4.1.9.2.1 Premultipled Alpha and Alpha Output Generation

The source image in an ARGB format may have its RGB component already premultiplied with the Alpha (AR'G'B') where:

R' = A*R

Copyright © 2026 Texas Instruments Incorporated

G’ = A*G

B’ = A*B

The processing in that case is as follows:

- Color component of premultiplied layer are multiplied with the global alpha if global_A is not equal to zero
- Color component of the composed underlying layer are multiplied with (1-A*Global_A)

Note

L1 (Layer-1) cannot be pre-multiplied if the overlay output is driving the display since the layer (or the background color) defines the bottom most layer. In this case, L1 alpha is 1 always.

12.9.1.4.1.9.2.2 Overlay Alpha Blender

The alpha blending value is defined by:

- The component value A when using an ARGB or RGBA pixel format (alpha in the source pixel data is converted to 8-bit alpha value in the input pipeline logic):
  - For ARGB-1555, the alpha blending is defined using a 1-bit value. It is converted into an 8-bit value by duplicating the 1-bit value (see Table 12-479).
  - For ARGB-4444, the alpha blending is defined using a 4-bit value. It is converted into an 8-bit value by duplicating the 4-bit value (see Table 12-479).
  - If the pixel format contains no alpha blending value, the pixel alpha value is considered to be 0xFF, and if alpha is equal to 0xFF, there is no multiplication.
  - For BITMAP or YUV formats, there is no alpha blending factor associated with each pixel value. Only the global alpha blending factor associated with the window displaying the BITMAP or YUV format is used.
- The global alpha blending value can be set in the DSS_VID_GLOBAL_ALPHA register of the input pipeline, and is updated in synch with the selected output channel.
- The modulated alpha blending value (that is, a total alpha blending value, when a combination of the pixel alpha blending value A and a global alpha blending are present) is determined as: Alpha = (Pixel Alpha × Global Alpha) / 256.

Table 12-479 shows the remapping and the percentage of alpha blending achieved.

Table 12-479. DISPC Overlay Alpha Blending – ARGB

Alpha Blending 1-Bit Value (ARGB16-1555)	Alpha Blending 4-Bit Value (ARGB16-4444)	Alpha Blending 8-Bit Value (Converted Value)	% Blending
0x0	0x0	0x00	100% (transparent)
N/A	0x1	0x11	93.33%
N/A	0x2	0x22	86.6%
N/A	...	...	...
N/A	0xE	0xEE	6.6%
0x1	0xF	0xFF	0% (opaque)

Premultiplied Alpha

The image ARGB may have its RGB component already premultiplied with the alpha (AR'G'B') where:

- R' = A*R
- G' = A*G
- B' = A*B

In that case, the processing is as follows:

- Color components of premultiplied layers are multiplied with the global alpha, if global alpha is not equal to 0.
- Color components of the composed underlying layer are multiplied with (1 - A*global alpha).

The additional premultiplied alpha option is accessible through the DSS_VID_ATTRIBUTES [28] PREMULTIPLYALPHA register bit of the input pipeline. The following settings are available:

Copyright © 2026 Texas Instruments Incorporated

- PREMULTIPLYALPHA bit = 0: Source is not premultiplied with alpha. Full blending is done in the overlay manager.
- PREMULTIPLYALPHA bit = 1: Source is premultiplied with alpha. Partial blending is done.

Note

There is no alpha blending between the background and the next layer (layer-0) above it.

Layer-0 cannot be pre-multiplied, if the overlay output is driving the display, since that layer (or the background color) defines the bottom most layer. In this case, layer-0 alpha is 1 always.

12.9.1.4.1.9.2.3 Overlay Transparency Color Keys

The overlay manager supports two color transparency modes - source and destination. These modes cannot be active at the same time.

The source transparency is tied to the top most layer, so the layer to have this transparency applied to has to be mapped to the highest z-ordered input (layer-1) of the overlay manager. The destination transparency is tied to the bottom most layer, so the layer to have this transparency applied to has to be mapped to the lowest z-ordered input (layer-0) of the overlay manager.

The source and destination transparencies are defined as following:

- Source color transparency: Top layer pixel that meets the source transparency color check is made transparent.
- Destination color transparency: Top layer pixel is made transparent, only if the bottom layer pixel does not meet the destination transparency color check.

The transparency color key is enabled via the DSS_OVR1_CONFIG[10] TCKLCDENABLE register bit.

The minimum transparency color values are specified in the TRANSCOLORKEY field of the DSS_OVR1_TRANS_COLOR_MIN and DSS_OVR1_TRANS_COLOR_MIN2 registers.

The maximum transparency color values are specified in the TRANSCOLORKEY field of the DSS_OVR1_TRANS_COLOR_MAX and DSS_OVR1_TRANS_COLOR_MAX2 registers.

The transparency color key values defined in the registers are compared with the pixel values. If each color component (R, G, and B) is in the range defined by MIN and MAX, then there is a match. If at least one of the color component is not in the range, then there is no match.

- **Source transparency color key**

  The values of the source transparency color key define the range of encoded pixel data considered as a transparent pixel. The encoded pixel values with the source color key value inside the valid range are not visible, and the encoded pixel values of the under layers or solid background color are visible.

  The source transparency color key can be used with YUV and RGB formats (ARGB, RGB, RGBA, xRGB, and RGBx). In that case the A information is ignored for the comparison between the pixel value and the color key value. It is possible to use YUV formats with some care, since the comparison is between the input pixel value of the overlay manager from pipeline and the color key value. The YUV data is converted to RGB format. The user must consider the color space conversion processing in order to define the RGB color key value used for the comparison, in case the original format is YUV.

  The scaler can be enabled as a preprocessor in the video pipelines. The pixel scaling processing can be considered in order to define the color key value to be used after the rescaling for the comparison between the input pixel value to the overlay manager and the color key value.

  The source transparency color key mode is selected by setting the DSS_OVR1_CONFIG[11] TCKLCDSELECTION register bit to 0x1.

  Figure 12-547 shows an example of source transparency color key usage. The pixels with the transparency color key are not displayed. Instead, pixels of the resulting layer underneath are shown.

Copyright © 2026 Texas Instruments Incorporated

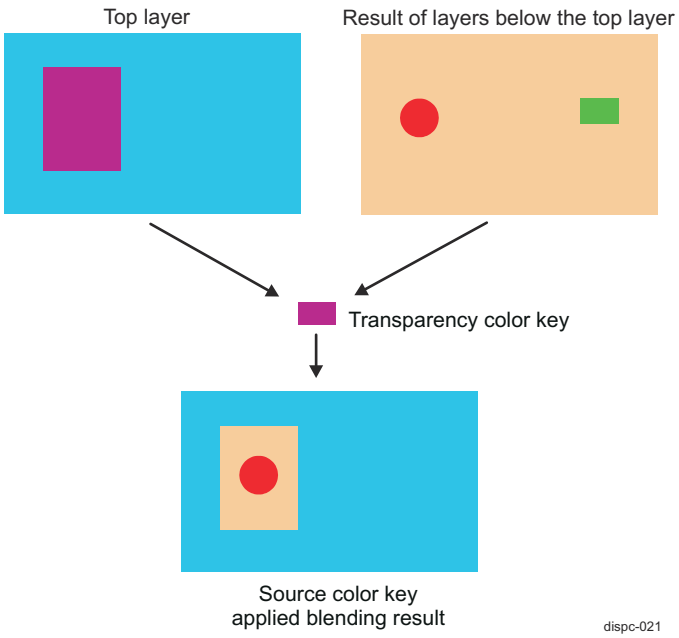


Figure 12-547. DISPC Overlay Source Transparency Color Key Example

- **Destination transparency color key**

  The destination transparency color key values define the range of the encoded pixels in layer-0 (bottom layer), which are not going to be displayed. That is, the encoded pixel values matching the destination color key value range are pixels not visible on the screen. Pixels at the same position in the layers above layer-0 are visible. The destination transparency color key is applicable in the layer in the background only when there is an overlap between the layer in background position and the layer just above it, otherwise, the destination transparency color key is ignored. See *DISPC Overlay Mechanism*, for details on layer position depending on the Z-order parameter.

  The destination transparency color key can be used only with RGB formats (ARGB, RGB, xRGB, RGBA, and RGBx). In that case the A information is ignored for the comparison between the pixel value and the color key value. It is possible to use YUV formats with some care since the comparison is between the input pixel value of the overlay manager from pipeline and the color key value. The YUV data is converted to RGB format. The user must consider the color space conversion processing in order to define the RGB color key value used for the comparison in case the original format is YUV.

  The scaler can be enabled as a preprocessor in the pipelines. The pixel scaling processing must be considered in order to define the color key value to be used after rescaling for the comparison between the input pixel value to the overlay manager and the color key value.

  The source transparency color key mode is selected by setting the DSS_OVR1_CONFIG[11] TCKLCDSELECTION register bit to 0x0.

  Figure 12-548 shows an example of the destination color key usage. The pixels from layer-0 (bottom layer) equal to the transparency color key are not displayed and are replaced by the pixels from layer-1 (top layer). All other layer-0 pixels, different from the transparency color key, are displayed over layer-1.

Copyright © 2026 Texas Instruments Incorporated

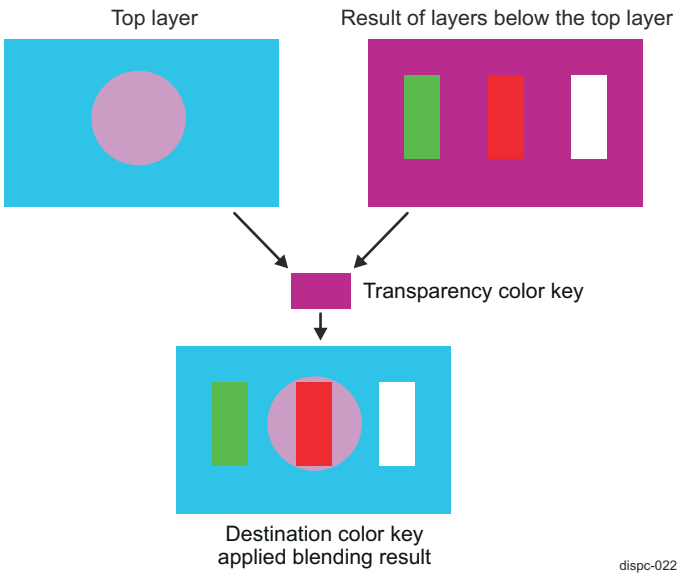


Figure 12-548. DISPC Overlay Destination Transparency Color Key Example

12.9.1.4.1.9.3 Overlay 3D Support

In order to support S3D (Stereoscopic 3D) by combining in hardware left and right frames into the same output frame, the following configurations (defined by HDMI 1.4b standard specification) can be used:

- Separate pipeline mode: Left/Right or Top/Bottom. One layer is used for left/top and another layer is used for right/bottom.

12.9.1.4.1.9.4 Overlay Color Bar Insertion

Each overlay manager supports a simple internal color bar insertion in each display output path to enable testing of display output interface without using the frame buffer data from the memory.

The colors are: White, Yellow, Cyan, Green, Magenta, Red, Blue, Black, as shown in below. These make three primary colors, three secondary colors, white, and black.

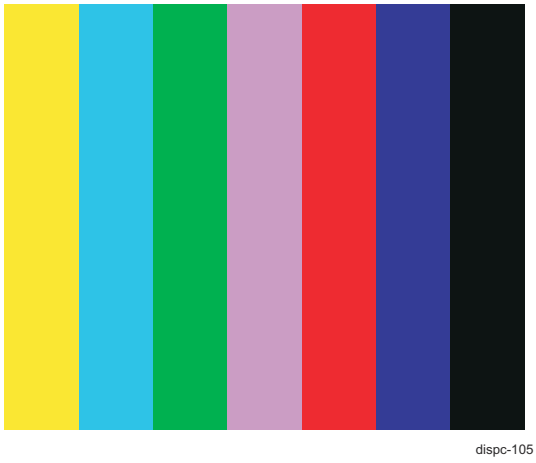


Figure 12-549. DISPC Overlay Internal Color Bar

When the DSS_OVR1_CONFIG[1] COLORBAREN register bit is set to 1, the overlay output data is replaced by the predefined ARGB48 color bar data.

When color bar mode is used, the color space conversion matrix should be appropriately programmed, so that it can convert the full-range RGB to the desired color format.

Copyright © 2026 Texas Instruments Incorporated

Table 12-480. DISPC Overlay Color Bar Table

Color	G	B	R
White	0xFFF	0xFFF	0xFFF
Yellow	0xFFF	0	0xFFF
Cyan	0xFFF	0xFFF	0
Green	0xFFF	0	0
Magenta	0	0xFFF	0xFFF
Red	0	0	0xFFF
Blue	0	0xFFF	0
Black	0	0	0

12.9.1.4.1.10 DISPC Video Port Outputs

DISPC implements two identical Video Port (VP) outputs:

- VP1 processes data received from Overlay Manager 1 (OVR1)
- VP2 processes data received from Overlay Manager 2 (OVR2)

A VP output path consists of several processing blocks (see Figure 12-550):

- Gamma correction unit
- Color phase rotation (CPR) unit, also used for RGB to YUV color space conversion (CSC)
- Active matrix dithering with TDM (multiple cycle output format)
- BT.656/BT.1120
- Timing generator
- Async buffer

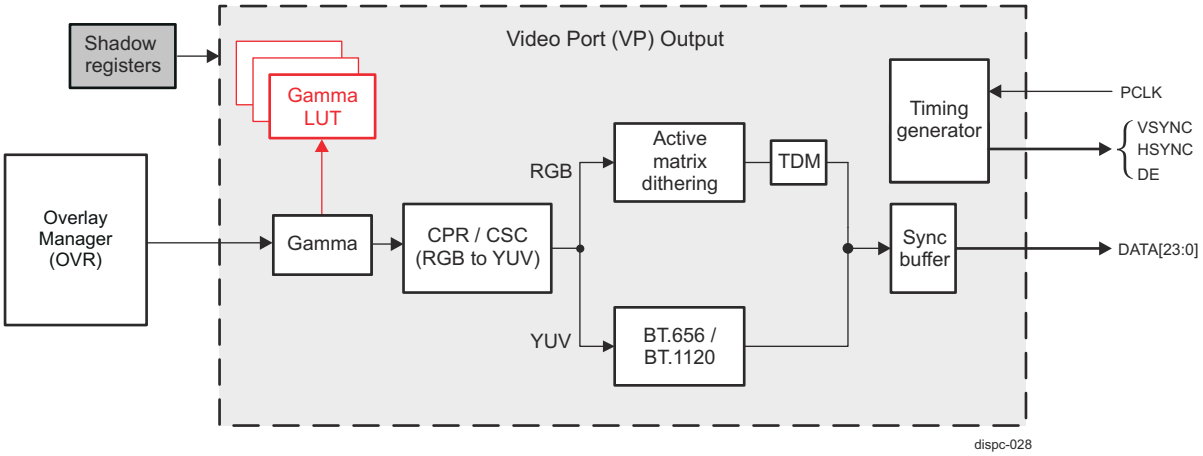


Figure 12-550. DISPC VP Output Architecture

The Color Space Conversion (CSC) processing can be configured to occur either before or after the gamma correction in the VP output module via the DSS_VP1_CONFIG[26] COLORCONVPOS register bit. For RGB to YUV color space conversion, the conversion must be done after the gamma correction. For other RGB output applications, the conversion must be done before the final gamma correction.

Note

The DISPC video port (VP) outputs, VP1 and VP2, will be commonly referred to as *video port (VP)* in the following sections.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.10.1 DISPC VP Gamma Correction Unit

The VP supports a look up table to perform the gamma correction on the display output. The look up table consists of 3 separate 256x8-bit memories, which are indexed by R/G/B component data.

When performing gamma correction, the selected encoded pixel values from the video pipeline path are sent by the overlay manager to the gamma curve table. Each R/G/B component of the encoded pixel value is used as a pointer to index 1 out of 256 x 24-bit gamma curve entries in the table. Each 8-bit component is replaced with the 8-bit table value corresponding to R, G, or B component. The table is loaded by software via the DSS_VP1_GAMMA_TABLE_0 through DSS_VP1_GAMMA_TABLE_15 registers. It is possible to load only part of the table. For each access to the table, the 24-bit value is associated with index in the table by concatenating the 24-bit value (LSB of 32-bit access) and the 8-bit index value (MSB of the 32-bit access).

The sequence to load the gamma table is:

1. SW writes (only writes are supported) 32-bit gamma correction values using single access, or burst access in linear increment burst mode, into the 64-byte region starting at DSS_VP1_GAMMA_TABLE_0 register address. The LSB 24 bits [23:0] are used for the value, and the MSB 8 bits [31:24] are used for the index into the table.
2. Loop to step #1, if there is a new access to the gamma table register. The software can access other registers between two accesses to the gamma table register.

Software needs to ensure that there is no visible effect when modifying the table, since it is not under hardware control.

The usage of the gamma table is activated by setting DSS_VP1_CONFIG[2] GAMMAENABLE register bit.

Figure 12-551 describes the format of one of the gamma curve values in the memory.

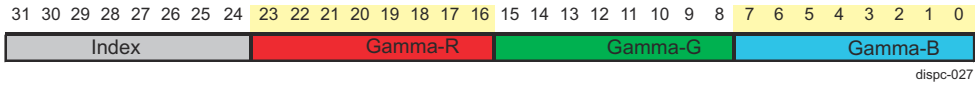


Figure 12-551. DISPC VP Output Gamma Table Write Data Format

12.9.1.4.1.10.2 DISPC VP Color Phase Rotation Unit

If required, a color phase rotation (CPR) processing can be applied on the "gamma" data before the spatial/ temporal dithering.

The CPR can be selected to correct the non-pure white backlight of the display panel by using a programmable matrix to convert the 36-bit RGB pixel value into a new 36-bit RGB pixel value. The matrix is programmed through a set of nine 11-bit signed coefficients. The output of the calculation is clipped to [0:4095]. The CPR is enabled by setting the DSS_VP1_CFG[15] CPR bit to 0x1.

The CPR processing is expressed by the equation shown in Figure 12-552. Table 12-481 lists all coefficients with their respective register bitfields for settings.

$$\begin{bmatrix} R_{out} \\ G_{out} \\ B_{out} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Rr & Rg & Rb \\ Gr & Gg & Gb \\ Br & Bg & Bb \end{bmatrix} * \begin{bmatrix} R_{in} \\ G_{in} \\ B_{in} \end{bmatrix}$$

dispc-023

Figure 12-552. DISPC VP CPR Matrix

Table 12-481. DISPC VP CPR and CSC Coefficients with Associated Register Fields

Register Field	Color Phase Rotation	Color Space Conversion RGB to YUV
DSS_VID_CSC_COEF0[10-0] C00	RR	YR

Copyright © 2026 Texas Instruments Incorporated

Table 12-481. DISPC VP CPR and CSC Coefficients with Associated Register Fields (continued)

Register Field	Color Phase Rotation	Color Space Conversion RGB to YUV
DSS_VID_CSC_COEF0[26-16] C01	RG	YG
DSS_VID_CSC_COEF1[10-0] C02	RB	YB
DSS_VID_CSC_COEF1[26-16] C10	GR	CrR
DSS_VID_CSC_COEF2[10-0] C11	GG	CrG
DSS_VID_CSC_COEF2[26-16] C12	GB	CrB
DSS_VID_CSC_COEF3[10-0] C20	BR	CbR
DSS_VID_CSC_COEF3[26-16] C21	BG	CbG
DSS_VID_CSC_COEF4[10-0] C22	BB	CbB
DSS_VID_CSC_COEF6[31-19] POSTOFFSET1	-	R offset
DSS_VID_CSC_COEF7[15-3] POSTOFFSET2	-	G offset
DSS_VID_CSC_COEF7[31-19] POSTOFFSET3	-	B offset

12.9.1.4.1.10.3 DISPC VP Color Space Conversion - RGB to YUV

The RGB to YUV color space conversion (CSC) in the video port is done by using the same programable logic used for CPR. If CPR is also required, then the process can be combined with the CSC processing by adjusting matrix coefficients. Table 12-481 lists the associated coefficients with their respective register bit-fields. The CSC processing is enabled by setting the DSS_VP1_CONFIG[24] COLORCONVENABLE register bit.

The color space conversion can be selected in order to convert from 36-bit RGB to YUV444. The conversion matrix is programmed through a set of nine 11-bit signed coefficients. The result is clipped to [256:3760] for the luminance and [256:3840] for the chrominance, when a full-range YUV output is not desired (equivalent to clipping to [16..235] and [16..240] in 8-bit video). It his case the DSS_VP1_CONFIG[25] FULLRANGE register bit must be set to 0x0.

$$\begin{bmatrix} Y_{OUT} \\ Cr_{OUT} \\ Cb_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cr_R & Cr_G & Cr_B \\ Cb_R & Cb_G & Cb_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 256 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-098

Figure 12-553. DISPC VP CSC RGB to YUV Matrix (FULLRANGE=0)

If the programmed active range for the luminance samples (Y) and chrominance samples (Cb and Cr) is [0:4095], then the values Y, Cb, and Cr are clipped to the range [0:4095]. The following equation gives the 11-bit coefficients of the RGB to YUV color space conversion, for full-range (when DSS_VP1_CONFIG[25] FULLRANGE = 0x1).

$$\begin{bmatrix} Y_{OUT} \\ Cr_{OUT} \\ Cb_{OUT} \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} Y_R & Y_G & Y_B \\ Cr_R & Cr_G & Cr_B \\ Cb_R & Cb_G & Cb_B \end{bmatrix} * \begin{bmatrix} R_{IN} \\ G_{IN} \\ B_{IN} \end{bmatrix} + \begin{bmatrix} 0 \\ 2048 \\ 2048 \end{bmatrix}$$

dispc-099

Figure 12-554. DISPC VP CSC RGB to YUV Matrix (FULLRANGE=1)

In order to provide YUV422 data to the BT.656 or BT.1120 blocks, the YUV444 format is further converted to YUV422 by sub-sampling the chrominance values. The hardware does this by averaging two-by-two the

Copyright © 2026 Texas Instruments Incorporated

chrominance samples. The conversion from YUV444 to YUV422 is performed when the color space conversion in the VP is enabled.

12.9.1.4.1.10.4 DISPC VP BT.656 and BT.1120 Modes

Each VP output can be configured in BT.656 or BT.1120 mode. The following standards are not supported in BT.656 mode:

- BT.470 (Support for conventional Analog TV Systems)
- BT.803 (The avoidance of interference generated by digital television studio equipment)
- BT.1364 (Format of ancillary data signals carried in digital component studio interfaces)

Unsupported formats when BT.1120 mode is used:

- BT.1364 (Support for Ancillary Data during blanking period)

BT.656/BT.1120 modes use embedded EAV/SAV syncs.

Enabling BT.656 or BT.1120 format for a VP output is done by setting DSS_VP1_CONFIG[20] BT656ENABLE or [21] BT1120ENABLE register bit, respectively

Note

It is not possible to enable BT.656 and BT.1120 mode simultaneously on the same output.

Figure 12-555 shows signal mapping on video port DATA[23:0] output data bus. Bits 9 to 0 are dedicated for BT.656 mode (10-bit). In BT.656 mode however, for compatibility with existing 8-bit interfaces, the two LSBs are ignored and only bits [9-2] are effectively used.

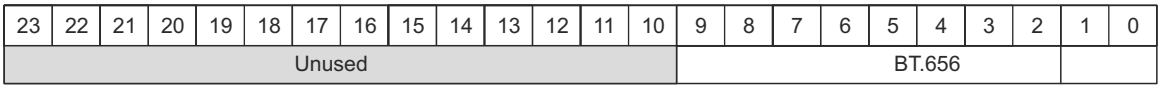


Figure 12-555. DISPC VP Data Mapping for BT.656 Mode

Figure 12-556 shows signal mapping on video port DATA[23:0] output data bus for BT.1120 mode. Bits [19-10] (CbCr) and [9-0] (Y) are used in 20-bit mode. Bits [19-12] (CbCr) and [9-2] (Y) are used in 16-bit mode (YCbCr422).

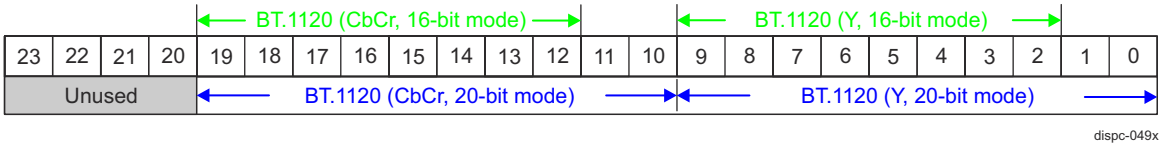


Figure 12-556. DISPC VP Data Mapping for BT.1120 Mode

VP outputs support both interlace and progressive content in BT.656/BT.1120 modes. For more information on timings configuration, see Section 12.9.1.4.1.10.7, *DISPC VP Timing Generator and Display Panel Settings*.

Note

In progressive BT.656/BT.1120 mode the maximum output resolution will be limited, as it requires two pixel clock cycles to send out one pixel.

12.9.1.4.1.10.4.1 DISPC BT Mode Blanking

During the transmission of the video signal, the portion of the stream in-between active video data segments is known as the horizontal blanking interval.

Strictly speaking this entire region is the blanking interval, but this interval also includes the EAV and SAV codes. The remaining bytes of information in a digital blanking interval are filled with values corresponding to the

Copyright © 2026 Texas Instruments Incorporated

blanking levels of the Cb, Y and Cr signals respectively, and in accordance with the standard multiplex sequence for the stream (CbYCrY..). The blanking levels are as follows:

- Cb = 80h
- Y = 10h
- Cr = 80h.

The sequence in the BLANKING region of the data stream is therefore: *80h, 10h, 80h, 10h.....80h, 10h*

For more details on setting the blanking timing values for BT.1120 and/or BT.656 mode, see Section 12.9.1.4.1.10.7, *DISPC VP Timing Generator and Display Panel Settings*.

12.9.1.4.1.10.4.2 DISPC BT Mode EAV and SAV

The End of Active Video (EAV) and Start of Active Video (SAV) parts of the stream are timing codes. Their function can be summarized as follows:

- EAV – marks the end of the active video data within the current line and therefore also the start of the subsequent line.
- SAV – heralds the start of the active video data within the current line.

These codes are embedded within the BT.656 video data stream, thereby eliminating the need for additional timing signals (HSYNC, VSYNC) to be included as part of the interface.

Both EAV and SAV codes are comprised of a sequence of four bytes (FFh – 00h – 00h - XY). The first three bytes in the sequence constitute a fixed preamble. The fourth byte, contains information about the field being transmitted (Field 1 or Field 2 in an interlaced video signal), the state of field blanking (Vertical) and the state of line blanking (Horizontal). The bit assignment for this byte of the code is shown in Figure 12-557, with the function of each bit described inTable 12-482.

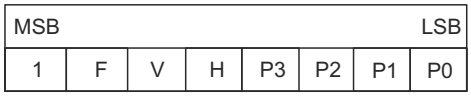


Figure 12-557. DISPC BT Mode Bit-Assignment for the Fourth Byte of EAV/SAV Codes

Table 12-482. DISPC BT Mode Bit Function

Bit	Symbol	Function
7	1	Always set to '1'.
6	F	Field bit. 0 – Field 1 1 – Field 2
5	V	Vertical Blanking Status bit. This bit goes High during a vertical field blanking interval, otherwise it remains Low.
4	H	Horizontal Blanking Status bit. 0 – byte is part of SAV code (i.e. stream is entering an active video data region for the current line) 1 – byte is part of EAV code (i.e. stream has entered a horizontal blanking interval - start of a new line)
3	P3	Protection bit 3
2	P2	Protection bit 2
1	P1	Protection bit 1
0	P0	Protection bit 0

The protection bits allow for detection and correction of 1-bit errors and the detection of 2-bit errors. The status of P3, P2, P1 and P0 depend on the states of bits F, V, and H. This dependency is shown in Table 12-483.

Table 12-483. DISPC BT Mode Status of Protection Bits in Function of F, V, and H

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0

Copyright © 2026 Texas Instruments Incorporated

Table 12-483. DISPC BT Mode Status of Protection Bits in Function of F, V, and H (continued)

F	V	H	P3	P2	P1	P0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

12.9.1.4.1.10.5 DISPC VP Spatial/Temporal Dithering

The active spatial/temporal dithering logic can be enabled to minimize the color banding when displaying the data on an LCD panel with color depth lower than 24-bit. The dithering logic is integrated after the color/gamma conversion blocks and before the TDM (Time Division Multiplexed) block. The spatial/temporal dithering algorithm is based on the (x,y) pixel position and frame rate control (the value of removed bits and the frame number). The dithering logic can process the pixels over one frame, two frames, or four frames. The number of frames is selected by setting the DSS_VP1_CONTROL[31-30] SPATIALTEMPORALDITHERINGFRAMES register field. In the case of a single frame, only spatial processing is applied. In case of multiple frames, both spatial and temporal processing are applied to the pixels. The number of frame is initialized before enabling the spatial/temporal dithering logic. It must be never changed by the software while the spatial/temporal logic is enabled. The spatial/temporal dithering logic is enabled by setting the DSS_VP1_CONTROL[7] STDITHERENABLE bit to 0x1.

Note

- If the interface data bus is smaller than the pixel format size and spatial/temporal dithering is not enabled, the MSBs of the pixel color components are output on the interface data bus.
- If the interface data bus is larger than the pixel format size, then by programming the pixel components replication active/inactive the MSB is replicated to the LSB of the interface data bus.

12.9.1.4.1.10.6 DISPC VP Multiple Cycle Output Format (TDM)

The pixels (only RGB components) after the active matrix dithering unit are formatted on one or multiple cycles (from 1 to 3 cycles). On three cycles, two pixels can concatenate and send to the panel. The cycle format is selected through the DSS_VP1_CONTROL[24-23] TDMCYCLEFORMAT bit field. The number of bits for each cycle is set in the DSS_VP1_DATA_CYCLE_0 register for the first cycle, the DSS_VP1_DATA_CYCLE_1 register for the second cycle, and the DSS_VP1_DATA_CYCLE_2 register for the third cycle. The output interface data bus width, when TDM mode is enabled (DSS_VP1_CONTROL[20] TDMENABLE register bit = 1), can be 8, 9, 12, or 16 bits, configurable through the DSS_VP1_CONTROL[22-21] TDMPARALLELMODE register field.

When the TDM is disabled (DSS_VP1_CONTROL[20] TDMENABLE = 0), the video port output interface data bus width is configured through the DSS_VP1_CONTROL[10-8] DATALINES register field.

When using TDM mode, only up to 24 bits per pixel can be output on the interface. For higher color depth, only the upper bits are kept before converting each pixel into TDM output.

Figure 12-558 through Figure 12-561 show various examples of TDM settings in the function of pixel data formats and the interface data bus width.

Copyright © 2026 Texas Instruments Incorporated

	24-bpp		
	1st cycle	2nd cycle	3rd cycle
Data[7]	R0[7]	G0[7]	B0[7]
Data[6]	R0[6]	G0[6]	B0[6]
Data[5]	R0[5]	G0[5]	B0[5]
Data[4]	R0[4]	G0[4]	B0[4]
Data[3]	R0[3]	G0[3]	B0[3]
Data[2]	R0[2]	G0[2]	B0[2]
Data[1]	R0[1]	G0[1]	B0[1]
Data[0]	R0[0]	G0[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2
DATA_CYCLE0 = 0x00000008
DATA_CYCLE1 = 0x00000008
DATA_CYCLE2 = 0x00000008

	18-bpp		
	1st cycle	2nd cycle	3rd cycle
Data[7]	R0[5]	G0[3]	x
Data[6]	R0[4]	G0[2]	x
Data[5]	R0[3]	G0[1]	x
Data[4]	R0[2]	G0[0]	x
Data[3]	R0[1]	B0[5]	x
Data[2]	R0[0]	B0[4]	x
Data[1]	G0[5]	B0[3]	B0[1]
Data[0]	G0[4]	B0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2
DATA_CYCLE0 = 0x00000008
DATA_CYCLE1 = 0x00000008
DATA_CYCLE2 = 0x00000002

	16-bpp	
	1st cycle	2nd cycle
Data[7]	R0[4]	G0[2]
Data[6]	R0[3]	G0[1]
Data[5]	R0[2]	G0[0]
Data[4]	R0[1]	B0[4]
Data[3]	R0[0]	B0[3]
Data[2]	G0[5]	B0[2]
Data[1]	G0[4]	B0[1]
Data[0]	G0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000008
DATA_CYCLE1 = 0x00000008

	12-bpp	
	1st cycle	2nd cycle
Data[7]	R0[3]	x
Data[6]	R0[2]	x
Data[5]	R0[1]	x
Data[4]	R0[0]	x
Data[3]	G0[3]	B0[3]
Data[2]	G0[2]	B0[2]
Data[1]	G0[1]	B0[1]
Data[0]	G0[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000008
DATA_CYCLE1 = 0x00000004

dispc-057

Figure 12-558. DISPC VP TDM 8-Bit Interface Settings

Copyright © 2026 Texas Instruments Incorporated

	24-bpp		
	1st cycle	2nd cycle	3rd cycle
Data[8]	R0[7]	G0[6]	x
Data[7]	R0[6]	G0[5]	x
Data[6]	R0[5]	G0[4]	x
Data[5]	R0[4]	G0[3]	B0[5]
Data[4]	R0[3]	G0[2]	B0[4]
Data[3]	R0[2]	G0[1]	B0[3]
Data[2]	R0[1]	G0[0]	B0[2]
Data[1]	R0[0]	B0[7]	B0[1]
Data[0]	G0[7]	B0[6]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x2
DATA_CYCLE0 = 0x00000009
DATA_CYCLE1 = 0x00000009
DATA_CYCLE2 = 0x00000006

	18-bpp	
	1st cycle	2nd cycle
Data[8]	R0[5]	G0[2]
Data[7]	R0[4]	G0[1]
Data[6]	R0[3]	G0[0]
Data[5]	R0[2]	B0[5]
Data[4]	R0[1]	B0[4]
Data[3]	R0[0]	B0[3]
Data[2]	G0[5]	B0[2]
Data[1]	G0[4]	B0[1]
Data[0]	G0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000009
DATA_CYCLE1 = 0x00000009

	16-bpp	
	1st cycle	2nd cycle
Data[8]	R0[4]	x
Data[7]	R0[3]	x
Data[6]	R0[2]	G0[1]
Data[5]	R0[1]	G0[0]
Data[4]	R0[0]	B0[4]
Data[3]	G0[5]	B0[3]
Data[2]	G0[4]	B0[2]
Data[1]	G0[3]	B0[1]
Data[0]	G0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000009
DATA_CYCLE1 = 0x00000007

	12-bpp	
	1st cycle	2nd cycle
Data[8]	R0[3]	x
Data[7]	R0[2]	x
Data[6]	R0[1]	x
Data[5]	R0[0]	x
Data[4]	G0[3]	x
Data[3]	G0[2]	x
Data[2]	G0[1]	B0[2]
Data[1]	G0[0]	B0[1]
Data[0]	B0[3]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000009
DATA_CYCLE1 = 0x00000003

dispc-058

Figure 12-559. DISPC VP TDM 9-Bit Interface Settings

Copyright © 2026 Texas Instruments Incorporated

	24-bpp	
	1st cycle	2nd cycle
Data[11]	R0[7]	G0[3]
Data[10]	R0[6]	G0[2]
Data[9]	R0[5]	G0[1]
Data[8]	R0[4]	G0[0]
Data[7]	R0[3]	B0[7]
Data[6]	R0[2]	B0[6]
Data[5]	R0[1]	B0[5]
Data[4]	R0[0]	B0[4]
Data[3]	G0[7]	B0[3]
Data[2]	G0[6]	B0[2]
Data[1]	G0[5]	B0[1]
Data[0]	G0[4]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x0000000C
DATA_CYCLE1 = 0x0000000C

	18-bpp		
	1st cycle	2nd cycle	3rd cycle
Data[11]	R0[5]	B0[5]	G1[5]
Data[10]	R0[4]	B0[4]	G1[4]
Data[9]	R0[3]	B0[3]	G1[3]
Data[8]	R0[2]	B0[2]	G1[2]
Data[7]	R0[1]	B0[1]	G1[1]
Data[6]	R0[0]	B0[0]	G1[0]
Data[5]	G0[5]	R1[5]	B1[5]
Data[4]	G0[4]	R1[4]	B1[4]
Data[3]	G0[3]	R1[3]	B1[3]
Data[2]	G0[2]	R1[2]	B1[2]
Data[1]	G0[1]	R1[1]	B1[1]
Data[0]	G0[0]	R1[0]	B1[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x3
DATA_CYCLE0 = 0x0000000C
DATA_CYCLE1 = 0x00060606
DATA_CYCLE2 = 0x000C0000

	16-bpp	
	1st cycle	2nd cycle
Data[11]	R0[4]	x
Data[10]	R0[3]	x
Data[9]	R0[2]	x
Data[8]	R0[1]	x
Data[7]	R0[0]	x
Data[6]	G0[5]	x
Data[5]	G0[4]	x
Data[4]	G0[3]	x
Data[3]	G0[2]	B0[3]
Data[2]	G0[1]	B0[2]
Data[1]	G0[0]	B0[1]
Data[0]	B0[4]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x0000000C
DATA_CYCLE1 = 0x00000004

	12-bpp
	1st cycle
Data[11]	R0[3]
Data[10]	R0[2]
Data[9]	R0[1]
Data[8]	R0[0]
Data[7]	G0[3]
Data[6]	G0[2]
Data[5]	G0[1]
Data[4]	G0[0]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0
DATA_CYCLE0 = 0x0000000C

dispc-059

Figure 12-560. DISPC VP TDM 12-Bit Interface Settings

Copyright © 2026 Texas Instruments Incorporated

	24-bpp		
	1st cycle	2nd cycle	3rd cycle
Data[15]	R0[7]	B0[7]	G1[7]
Data[14]	R0[6]	B0[6]	G1[6]
Data[13]	R0[5]	B0[5]	G1[5]
Data[12]	R0[4]	B0[4]	G1[4]
Data[11]	R0[3]	B0[3]	G1[3]
Data[10]	R0[2]	B0[2]	G1[2]
Data[9]	R0[1]	B0[1]	G1[1]
Data[8]	R0[0]	B0[0]	G1[0]
Data[7]	G0[7]	R1[7]	B1[7]
Data[6]	G0[6]	R1[6]	B1[6]
Data[5]	G0[5]	R1[5]	B1[5]
Data[4]	G0[4]	R1[4]	B1[4]
Data[3]	G0[3]	R1[3]	B1[3]
Data[2]	G0[2]	R1[2]	B1[2]
Data[1]	G0[1]	R1[1]	B1[1]
Data[0]	G0[0]	R1[0]	B1[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x3
DATA_CYCLE0 = 0x00000010
DATA_CYCLE1 = 0x00080808
DATA_CYCLE2 = 0x00100000

	18-bpp	
	1st cycle	2nd cycle
Data[15]	R0[5]	x
Data[14]	R0[4]	x
Data[13]	R0[3]	x
Data[12]	R0[2]	x
Data[11]	R0[1]	x
Data[10]	R0[0]	x
Data[9]	G0[5]	x
Data[8]	G0[4]	x
Data[7]	G0[3]	X
Data[6]	G0[2]	x
Data[5]	G0[1]	x
Data[4]	G0[0]	x
Data[3]	B0[5]	x
Data[2]	B0[4]	x
Data[1]	B0[3]	B0[1]
Data[0]	B0[2]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x1
DATA_CYCLE0 = 0x00000010
DATA_CYCLE1 = 0x00000002

	16-bpp
	1st cycle
Data[15]	R0[4]
Data[14]	R0[3]
Data[13]	R0[2]
Data[12]	R0[1]
Data[11]	R0[0]
Data[10]	G0[5]
Data[9]	G0[4]
Data[8]	G0[3]
Data[7]	G0[2]
Data[6]	G0[1]
Data[5]	G0[0]
Data[4]	B0[4]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0
DATA_CYCLE0 = 0x00000010

	12-bpp
	1st cycle
Data[15]	x
Data[14]	x
Data[13]	x
Data[12]	x
Data[11]	R0[3]
Data[10]	R0[2]
Data[9]	R0[1]
Data[8]	R0[0]
Data[7]	G0[3]
Data[6]	G0[2]
Data[5]	G0[1]
Data[4]	G0[0]
Data[3]	B0[3]
Data[2]	B0[2]
Data[1]	B0[1]
Data[0]	B0[0]

CONTROL[24-23] TDMCYCLEFORMAT = 0x0
DATA_CYCLE0 = 0x0000000C

dispc-060

Figure 12-561. DISPC VP TDM 16-Bit Interface Settings

12.9.1.4.1.10.7 DISPC VP Timing Generator and Display Panel Settings

Each video port output has a dedicated timing generator supporting progressive and interlaced mode. It is clocked using the pixel clock and is configured to generate the video data and sync signals to match the desired video display standard timings.

Two-level configuration for enabling the video ports exists:

Copyright © 2026 Texas Instruments Incorporated

- Via the individual DSS_VP1_CONTROL[0] ENABLE register bit for each VP. It allows each VP to be independently controlled by two different hosts.
- Via the common DSS_COMMON_DISPC_GLOBAL_OUTPUT_ENABLE[2-0] VP_ENABLE register field. It allows two or more VP timing generators to be in sync by enabling them simultaneously with a single register write.

Both ENABLE and VP_ENABLE register fields must be set in order for a VP (timing generator) to start.

Figure 12-562 shows the timing generator display parameters.

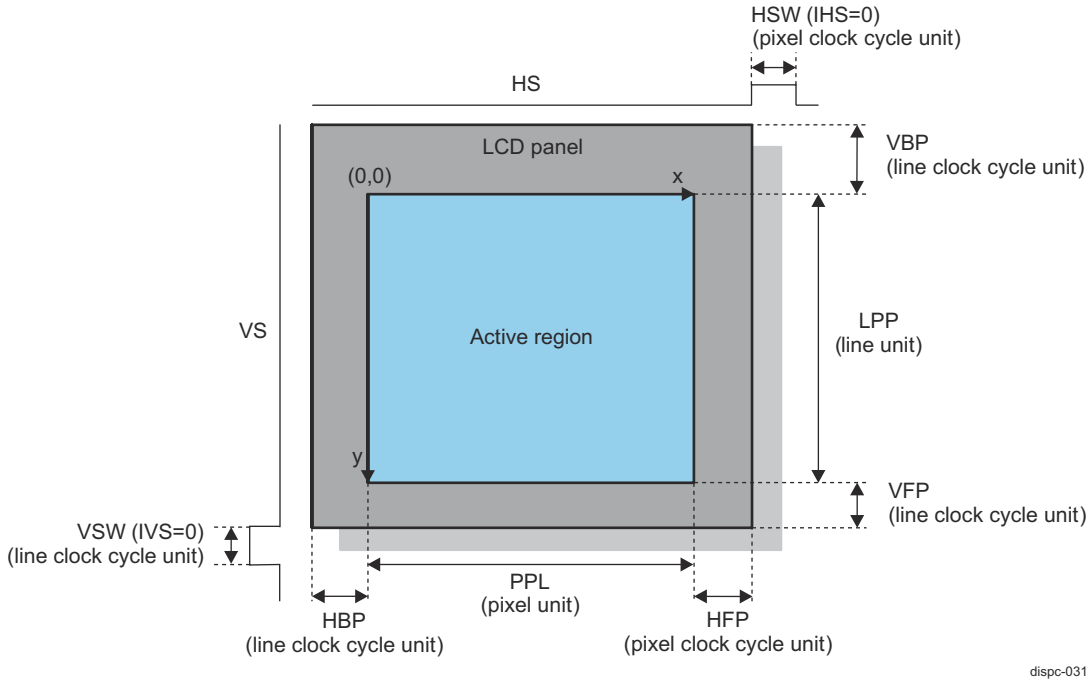


Figure 12-562. DISPC VP Display Timing Parameters

The width of VP output data bus is configured in the DSS_VP1_CONTROL[10-8] DATALINES register field, when TDM feature is disabled. When TDM is enabled, the width of the output data bus is defined according to Section 12.9.1.4.1.10.6, *DISPC VP Multiple Cycle Output Format (TDM)*.

The size of the display panel is defined by:

- Number of lines per frame, DSS_VP1_SIZE_SCREEN[27-16] LPP bit field, with a value from 1 to 4096
- Number of pixels per line, DSS_VP1_SIZE_SCREEN[11-0] PPL bit field, with a value from 1 to 4096

Standard HSYNC/VSYNC timing generation is programmable as follows:

- Horizontal front porch is set in the DSS_VP1_TIMING_H[19-8] HFP bit field.
- Horizontal back porch is set in the DSS_VP1_TIMING_H[31-20] HBP bit field.
- Horizontal synchronization pulse width is set in the DSS_VP1_TIMING_H[7-0] HSW bit field.
- Vertical front porch is set in the DSS_VP1_TIMING_V[19-8] VFP bit field.
- Vertical back porch is set in the DSS_VP1_TIMING_V[31-20] VBP bit field.
- Vertical synchronization pulse width is set in the DSS_VP1_TIMING_V[7-0] VSW bit field.

When the output is in BT.1120 or BT.656 mode, the following timing constants are mapped onto the DSS_VP1_TIMING_H and DSS_VP1_TIMING_V registers:

- Progressive mode:
  - Horizontal blanking (12 bits) is set in the {DSS_VP1_TIMING_V[3-0] VSW, DSS_VP1_TIMING_H[7-0] HSW} register fields (up to 2048 bytes of horizontal blanking supported).

Copyright © 2026 Texas Instruments Incorporated

  - Vertical frame blanking No 1 is set in the DSS_VP1_TIMING_V[19-8] VFP bit field.
  - Vertical frame blanking No 2 is set in the DSS_VP1_TIMING_V[31-20] VBP bit field.
  - Number of lines per frame is set in the DSS_VP1_SIZE_SCREEN[27-16] LPP bit field.
  - Number of pixels per line is set in the DSS_VP1_SIZE_SCREEN[11-0] PPL bit field.
- Interlaced mode:
  - Horizontal blanking (12 bits) is set in the {DSS_VP1_TIMING_V[3-0] VSW, DSS_VP1_TIMING_H[7-0] HSW} register fields (up to 2048 bytes of horizontal blanking supported).
  - Vertical field blanking No.1 for Even Field is set in the DSS_VP1_TIMING_H[19-8] HFP bit field.
  - Vertical field blanking No.2 for Even Field is set in the DSS_VP1_TIMING_H[31-20] HBP bit field.
  - Vertical field blanking No.1 for Odd Field is set in the DSS_VP1_TIMING_V[19-8] VFP bit field.
  - Vertical field blanking No.2 for Odd Field is set in the DSS_VP1_TIMING_V[31-20] VBP bit field.
  - Number of lines per field (even) is set in the DSS_VP1_SIZE_SCREEN[27-16] LPP bit field.
  - Delta number of odd field compared to even field (in a single line) is set in the DSS_VP1_SIZE_SCREEN[15-14] DELTA_LPP bit field. The DELTA_LPP field only controls the output channel and not the size of the field fetched from the frame buffer in memory. This field must be set to zero for YUV420 format.
  - Number of pixels per line is set in the DSS_VP1_SIZE_SCREEN[11-0] PPL bit field.

Horizontal/vertical synchronization and output enable signals polarity are programmable by setting the DSS_VP1_POL_FREQ[12] IVS, [13] IHS, and [15] IEO register bits. These signals can be gated by setting the DSS_VP1_CONFIG[7] VSYNCGATED and [6] HSYNCGATED register bits. In addition, the alignment between VSYNC and HSYNC signals can be programmed via the DSS_VP1_POL_FREQ[18] ALIGN register bit.

The latch of data can be driven on the rising or falling edge of the pixel clock by setting the DSS_VP1_POL_FREQ[14] IPC register bit. The drive of the SYNC and VSYNC signals in the function of the pixel clock is done by setting the DSS_VP1_POL_FREQ[16] RF bit.

Note

To set the pixel clock, CTRL_MMR_DPI0_CLK_CTRL[8] DPI0_CLK_CTRL_DATA_CLK_INVDIS setting should always be the same as the DSS_VP1_POL_FREQ[14] IPC setting, and CTRL_MMR_DPI0_CLK_CTRL[9] DPI0_CLK_CTRL_SYNC_CLK_INVDIS setting should always be the same as the DSS_VP1_POL_FREQ[16] RF setting.

For example, if CTRL_MMR_DPI0_CLK_CTRL[8] DPI0_CLK_CTRL_DATA_CLK_INVDIS is set to 1, DSS_VP1_POL_FREQ[14] IPC should be set to 1.

Each VP output can be configured in progressive output mode or interlaced output mode. The selection is done by writing into the bit-field DSS_VP1_CONFIG [22] OUTPUTMODEENABLE register bit. The default setting is for progressive mode. The selection can be changed only when the corresponding VP output is disabled. The configuration is independent for each VP output. It is possible to change the configuration of one of the VP outputs while the other VP is enabled.

The pixel clock for each VP output can be gated by setting the DSS_VP1_CONFIG[5] PIXELCLOCKGATED register bit.

The hold time of the pixels on the data bus is determined in clock cycles by the DSS_VP1_CONTROL[16-14] HT register field.

12.9.1.4.1.11 DISPC Safety Features

For safety critical system applications, DSS supports the following safety features in hardware:

- Data correctness check: To verify intended data is shown correctly on the display.
- Freeze frame detection: To notify a possible frame freeze, when there is no change in the display frame over multiple frame periods.

These features are implemented by collecting signatures from user-defined regions within each pipeline output frame and the final display output frame, and comparing them to reference signatures provided by the user

Copyright © 2026 Texas Instruments Incorporated

(software) and/or to previously saved signatures. The signature from each region is generated by using a MISR (Multiple Input Signature Register) module.

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.11.1 Safety Check Regions

DSS supports the following safety check regions to implement the safety features:

- Video pipelines: One safety check region at the output of each video pipeline.
- Video port outputs: Up to four sub-regions within the active video output area of the final display output of each video port.

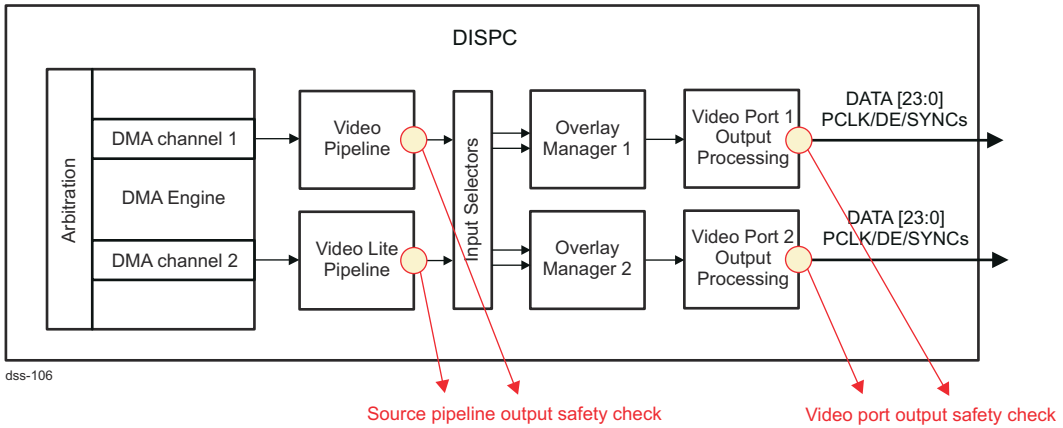


Figure 12-563. DISPC Safety Check Regions

Table 12-484 lists the parameters supported by each of the safety check regions, together with the associated register control fields.

Table 12-484. DISPC Safety Check Regions Parameters

Safety Region Parameter	Video Pipeline Control Register	Video Port Sub-region 0 Control Register	Video Port Sub-region 1 Control Register	Video Port Sub-region 2 Control Register	Video Port Sub-region 3 Control Register
X position	DSS_VID_SAFETY_POSITION[11-0] POSX	DSS_VP1_SAFETY_POSITION_0[11-0] POSX	DSS_VP1_SAFETY_POSITION_1[11-0] POSX	DSS_VP1_SAFETY_POSITION_2[11-0] POSX	DSS_VP1_SAFETY_POSITION_3[11-0] POSX
Y position	DSS_VID_SAFETY_POSITION[27-16] POSY	DSS_VP1_SAFETY_POSITION_0[27-16] POSY	DSS_VP1_SAFETY_POSITION_1[27-16] POSY	DSS_VP1_SAFETY_POSITION_2[27-16] POSY	DSS_VP1_SAFETY_POSITION_3[27-16] POSY
Width	DSS_VID_SIZE[11-0] SIZEX	DSS_VP1_SAFETY_SIZE_0[11-0] SIZEX	DSS_VP1_SAFETY_SIZE_1[11-0] SIZEX	DSS_VP1_SAFETY_SIZE_2[11-0] SIZEX	DSS_VP1_SAFETY_SIZE_3[11-0] SIZEX
Height	DSS_VID_SIZE[27-16] SIZEY	DSS_VP1_SAFETY_SIZE_0[27-16] SIZEY	DSS_VP1_SAFETY_SIZE_1[27-16] SIZEY	DSS_VP1_SAFETY_SIZE_2[27-16] SIZEY	DSS_VP1_SAFETY_SIZE_3[27-16] SIZEY
Data Correctness Check Mode Enable	DSS_VID_SAFETY_ATTRIBUTES[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_0[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_1[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_2[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_3[1] CAPTUREMODE

Copyright © 2026 Texas Instruments Incorporated

Table 12-484. DISPC Safety Check Regions Parameters (continued)

Safety Region Parameter	Video Pipeline Control Register	Video Port Sub-region 0 Control Register	Video Port Sub-region 1 Control Register	Video Port Sub-region 2 Control Register	Video Port Sub-region 3 Control Register
Freeze Frame Detection Mode Enable	DSS_VID_SAFETY_ATTRIBUTES[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_0[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_1[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_2[1] CAPTUREMODE	DSS_VP1_SAFETY_ATTRIBUTES_3[1] CAPTUREMODE
Region Safety Check Enable	DSS_VID_SAFETY_ATTRIBUTES[0] ENABLE	DSS_VP1_SAFETY_ATTRIBUTES_0[0] ENABLE	DSS_VP1_SAFETY_ATTRIBUTES_1[0] ENABLE	DSS_VP1_SAFETY_ATTRIBUTES_2[0] ENABLE	DSS_VP1_SAFETY_ATTRIBUTES_3[0] ENABLE
Freeze Frame Detection Threshold	DSS_VID_SAFETY_ATTRIBUTES[10-3] THRESHOLD	DSS_VP1_SAFETY_ATTRIBUTES_0[10-3] THRESHOLD	DSS_VP1_SAFETY_ATTRIBUTES_1[10-3] THRESHOLD	DSS_VP1_SAFETY_ATTRIBUTES_2[10-3] THRESHOLD	DSS_VP1_SAFETY_ATTRIBUTES_3[10-3] THRESHOLD
Frames to Skip	DSS_VID_SAFETY_ATTRIBUTES[12-11] FRAMESKIP	DSS_VP1_SAFETY_ATTRIBUTES_0[12-11] FRAMESKIP	DSS_VP1_SAFETY_ATTRIBUTES_1[12-11] FRAMESKIP	DSS_VP1_SAFETY_ATTRIBUTES_2[12-11] FRAMESKIP	DSS_VP1_SAFETY_ATTRIBUTES_3[12-11] FRAMESKIP
Reference Signature	DSS_VID_SAFETY_REF_SIGNATURE[31-0] SIGNATURE	DSS_VP1_SAFETY_REF_SIGNATURE_0[31-0] SIGNATURE	DSS_VP1_SAFETY_REF_SIGNATURE_1[31-0] SIGNATURE	DSS_VP1_SAFETY_REF_SIGNATURE_2[31-0] SIGNATURE	DSS_VP1_SAFETY_REF_SIGNATURE_3[31-0] SIGNATURE
MISR Seed	DSS_VID_SAFETY_LFSR_SEED[31-0] SEED	DSS_VID_SAFETY_LFSR_SEED	DSS_VID_SAFETY_LFSR_SEED	DSS_VID_SAFETY_LFSR_SEED	DSS_VID_SAFETY_LFSR_SEED
MISR Seed Selection	DSS_VID_SAFETY_ATTRIBUTES[2] SEEDSELECT	DSS_VP1_SAFETY_ATTRIBUTES_0[2] SEEDSELECT	DSS_VP1_SAFETY_ATTRIBUTES_1[2] SEEDSELECT	DSS_VP1_SAFETY_ATTRIBUTES_2[2] SEEDSELECT	DSS_VP1_SAFETY_ATTRIBUTES_3[2] SEEDSELECT
MISR Captured Signature	DSS_VID_SAFETY_CAPT_SIGNATURE[31-0] SIGNATURE	DSS_VP1_SAFETY_CAPT_SIGNATURE_0[31-0] SIGNATURE	DSS_VP1_SAFETY_CAPT_SIGNATURE_1[31-0] SIGNATURE	DSS_VP1_SAFETY_CAPT_SIGNATURE_2[31-0] SIGNATURE	DSS_VP1_SAFETY_CAPT_SIGNATURE_3[31-0] SIGNATURE

Copyright © 2026 Texas Instruments Incorporated

Figure 12-564 shows examples of one safety region in the video pipeline output stage and up to 4 possible regions in the final video port output stage.

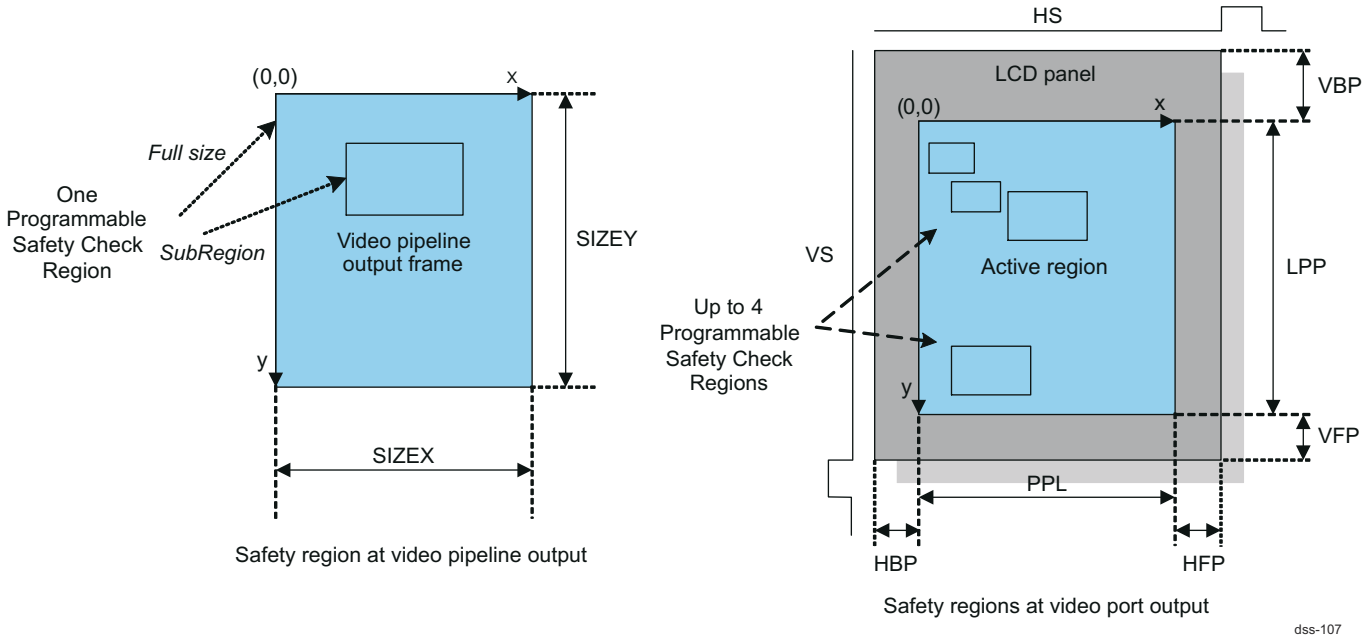


Figure 12-564. DISPC Safety Check Region Examples

The safety region in the video pipeline captures data only if the embedded alpha data is not equal to 0 (that is, non-transparent pixels). The safety regions in the display video port output captures all active video pixels within the region boundary. The (up to four) regions in the display output should be typically non-overlapping areas of the screen, but the hardware does not restrict them to be non-overlapping.

Figure 12-565 shows the locations within DISPC data path where the data is analyzed.

Copyright © 2026 Texas Instruments Incorporated

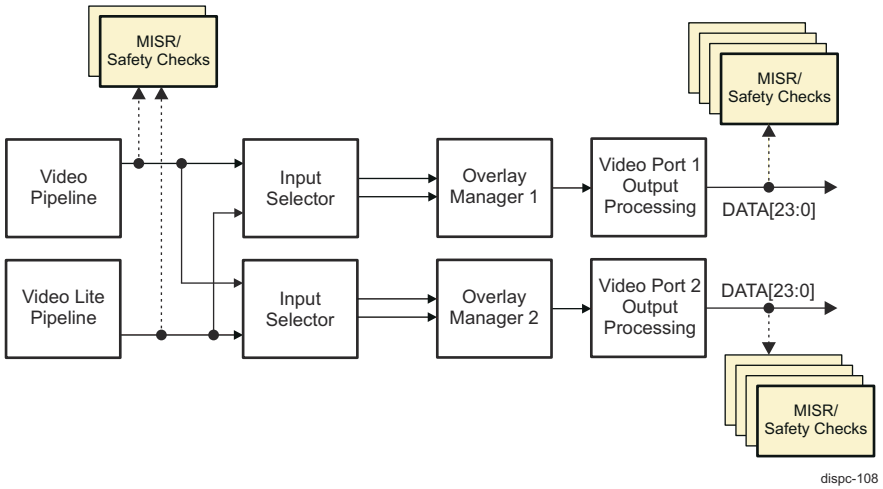


Figure 12-565. DISPC Safety Check Locations

Copyright © 2026 Texas Instruments Incorporated

12.9.1.4.1.11.2 Safety Signature Generator Using MISR

A 32-bit multiple input signature register (MISR) with 32-bit Galois LFSR polynomial, P(x)=X32+X22+X2+x+1, is used to capture a signature of active video pixel data (the 10 MSB bits of each color component) for each safety region, as shown in Figure 12-566 and the algorithm below it.

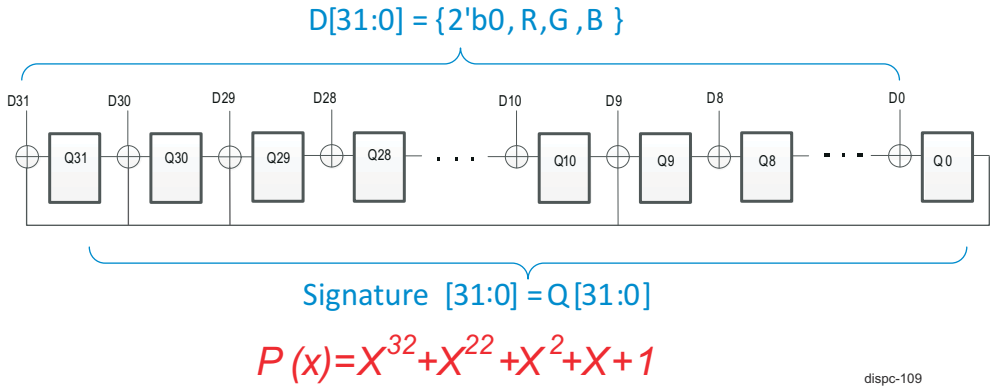


Figure 12-566. DISPC Safety 32-bit MISR Implementation

The MISR (32-bit Galois LFSR based) signature generation algorithm is as follows:

Tap _polynomial = 32'hE000_0200

lfsr_q = seed // initial LFSR value - any none zero value should work

For each (pixel data in the safety region) {

data_in[31:0] = { 2'b0, r,g,b } // 10 MSB bits of R,G,B components

lfsr_d = (data_in$^{(lfsr_q >> 1)}$)

If (lfsr_q[0]) lfsr_d = lfsr_d^{Tap_polynomial}

lsfr_q = lfsr_d

}

signature = lfsr_q

All MISRs are initialized at the beginning of each frame with a non-zero "seed" value either with the default constant value (0xFFFF_FFFF) or with the programmable seed configuration value in SAFETY_LFSR_SEED register for each group of regions (refer to Table 12-484, *DISPC Safety Check Regions Parameters*). Note that a same seed must be used to compare signatures between two frames. If different than the default seed value is desired, then a SEEDSELECT parameter must be set before the corresponding SAFETY_LFSR_SEED register is configured with the customer seed value.

All captured signatures from safety regions are memory mapped to read-only registers SAFETY_CAPT_SIGNATURE for each video port (refer to Table 12-484, *DISPC Safety Check Regions Parameters*). A SAFETY_CAPT_SIGNATURE register is updated with the signature from each sub-region at the end of every frame. This register returns the signature of the last frame data when the MISR is enabled. When MISR is disabled, this register is cleared.

The MISR aliasing (faulty signature matches fault-free signature) probability is:

$(2^{(L-n)}-1)/(2^{L}-1) = \sim 2^{(L-n)}/2^{L} = 2^{-n}$ for large L, where

n = length of signature register

L = length of input sequence

Using the above approximation, the result is:

Copyright © 2026 Texas Instruments Incorporated

n=4, aliasing probability = 6.25%

n=16, aliasing probability = 0.0015%

...

n=32 (as in DISPC MISR), aliasing probability is ~2^{-32} (negligible)

12.9.1.4.1.11.3 Safety Checks

If the data correctness check is enabled (see Table 12-484, *DISPC Safety Check Regions Parameters*):

- When a new signature is generated, it is compared against the reference signature provided by the software. A SAFETY_REF_SIGNATURE register must be configured with a reference signature data, see Table 12-484, *DISPC Safety Check Regions Parameters*.
- If signatures do not match, an interrupt event is generated to indicate data mismatch.

If the freeze frame detection is enabled (see Table 12-484, *DISPC Safety Check Regions Parameters*):

- When a new signature is generated, it is first compared against the saved signature (from previous frame).
- If signatures match, an internal counter used to keep track of the number of frames with no data change (for the region) is incremented.
- If signatures do not match, the counter is cleared.
- If the counter value is greater than the user programmed freeze frame detection threshold value (see Table 12-484, *DISPC Safety Check Regions Parameters*), an interrupt event (VIDSAFETYREGION_IRQ or VPSAFETYREGION_IRQ, see Section 12.9.1.4.1.5, *DISPC Interrupt Requests*) is generated to indicate a possible freeze frame detection. The threshold value must be configured with the maximum number of identical successive frames allowed before an interrupt is generated.
- After the comparison, the signature is saved as the previous signature.
- The counter is cleared when the interrupt event is generated or when freeze frame detection check is disabled.

This frame freeze is different from the display frozen due to pipeline lock up. In that case, the DISPC will generate an SYNCLOST_IRQ and/or DMA VIDBUFFERUNDERFLOW_IRQ interrupt. The freeze frame detection is for source data frozen while the DSS is working normally.

The two safety checks are continuously performed over multiple frames as long as their mode enable register bits are set.

12.9.1.4.1.11.4 Safety Check Limitations

The safety check will only be available when the DISPC is outputting RGB/YUV component data with separate sync signals. The safety functions are not available for following modes:

- YUV422 embedded sync modes (BT.656 and BT.1120)
- RGB TDM (Time Division Multiplex) mode

12.9.1.4.1.12 DISPC Security Management

12.9.1.4.1.12.1 Security Implementation

DSS supports the following security features:

- Secure mode configuration
- Illegal connection prevention

Secure Mode

DISPC supports a secure mode configuration register (DSS_COMMON1_DISPC_SECURE) which defines the "secure mode: attribute of each pipeline, overlay manager, and video port instance in DISPC. This register can only be modified by a host with an appropriate secure privilege (MReqSecure=1). When the secure bit corresponding to an instance is set by a secure host, the instance is deemed to be in "secure mode" and the DISPC hardware prevents the output of the instance getting connected to a non-secure downstream module. Also, any DMA transfer initiated by a secure pipeline will have its OCP in-band signal MReqSecure set to HIGH to indicate that is a secure mode transaction request.

Copyright © 2026 Texas Instruments Incorporated

By default, all pipelines, overlay managers, and video port instances are in a "non-secure mode". The DSS_COMMON1_DISPC_SECURE register bits are active, only when the DSS_COMMON_DISPC_SECURE_DISABLE[0] SECURE_DISABLE register bit is configured properly to 0x0. When the SECURE_DISABLE bit is set to 0x1, the DSS_COMMON1_DISPC_SECURE register bits are non-active and DISPC will behave as non-secure module.

Illegal Connection Prevention

DISPC hardware enforces the following rules to prevent an illegal connection:

- Secure input pipeline can only connect to a secure overlay manager: If any host (secure or non-secure) configures an overlay manager input selection to connect a secure input pipeline to a non-secure overlay manager, then the DISPC hardware will block the connection and issue a "security violation" interrupt (SECURITYVIOLATION_IRQ) to alert the host.

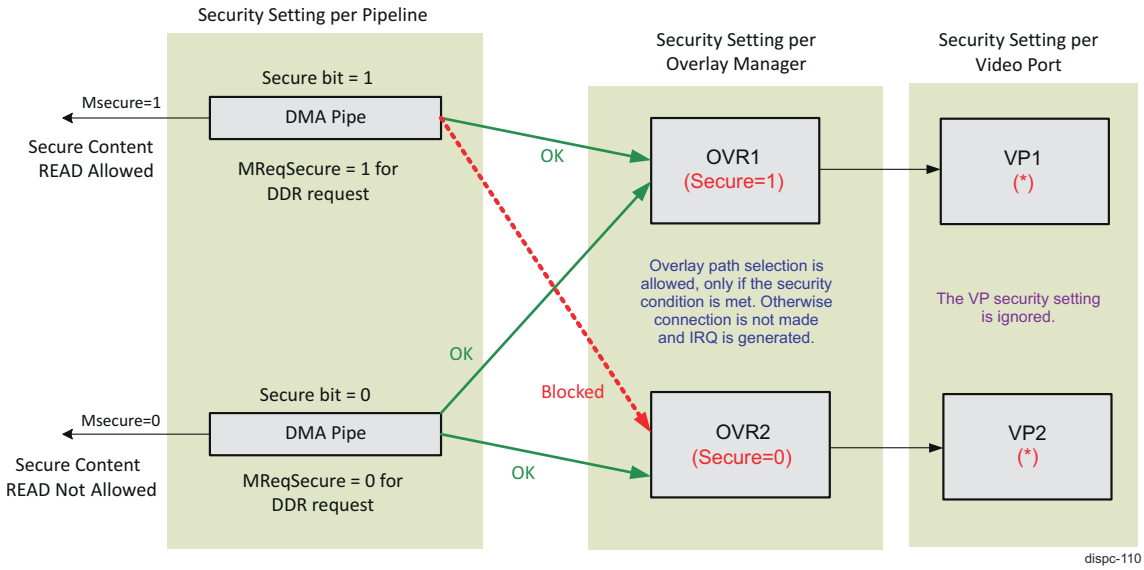


Figure 12-567. DISPC Secure Bit Setting and Illegal Connection Block

12.9.1.4.1.12.2 Secure Mode Configuration

The secure bit of DSS_COMMON1_DISPC_SECURE register is set/reset by a secure transaction. When the secure bit has been set, the software in "secure mode" is responsible for checking the DISPC configuration. The secure bit is propagated by DISPC to the system Interconnect in order to qualify all DISPC requests as secure or non-secure requests, based on the secure bits defined in the control register.

When DISPC accesses the frame buffer, in the case the secure bit has been reset and the frame buffer has been set secure, the display controller will receive an "error" in response of non-secure requests.

12.9.1.4.1.13 DISPC Shadow Registers

Some DISPC registers are termed *shadow registers*. The shadow registers allow the software to modify them at any time, without direct effect on the DISPC hardware configuration. When all the values for a given configuration are written into the shadow registers, software must set only one register bit to validate the configuration. When the hardware reaches the end of the current frame and sees that the bit has been set by software, the new configuration is now the configuration used by the hardware.

The DSS_VP1_CONTROL[5] GOBIT bit enables the hardware to use the new configuration, for all shadow registers associated with each VP output.

The registers are statically associated to a particular output (for example, timing registers) or dynamically associated to one output at a time (for example, video registers).

Copyright © 2026 Texas Instruments Incorporated

12.9.2 MIPI Display Serial Interface (DSI) Controller

12.9.2.1 DSI Block Diagram

The MIPI DSI v1.3.1 Transmitter Controller (DSITX) provides an interface that receives data and control from the host processor display system using either the DPI or SDI input bus interfaces. The DSITX will translate the incoming pixel information and control signals into an internal packed byte format, in the case of DPI, or pass in the prepacked SDI byte format, before the internal byte format data is packetized and sent to the MIPI DSI Compatible display via the MIPI D-PHY physical interface. It supports video and command mode displays and can work in multi-display mode using virtual channel identification on the packets.

The DSITX controller provides two interfaces for connection to a display panel. Normal operation for a panel supporting DCS commands can be done using either SDI interface, or using the APB (Advanced Peripheral Bus) access to DIRECT_CMD registers. Video streaming applications can only use the SDI or DPI interface. The DSI controller supports flow control using the SDI video interface.

The DSITX implements the stream arbitration and low-level protocol layer functionalities required by MIPI DSI specification v1.3. It supports up to 4 x 2.5 Gbps D-PHY data lanes in a single-link configuration and handles the byte lane mapping per use case (1, 2, 3, or 4-lanes).

The DSITX controller block diagram is illustrated in Figure 12-568.

Copyright © 2026 Texas Instruments Incorporated

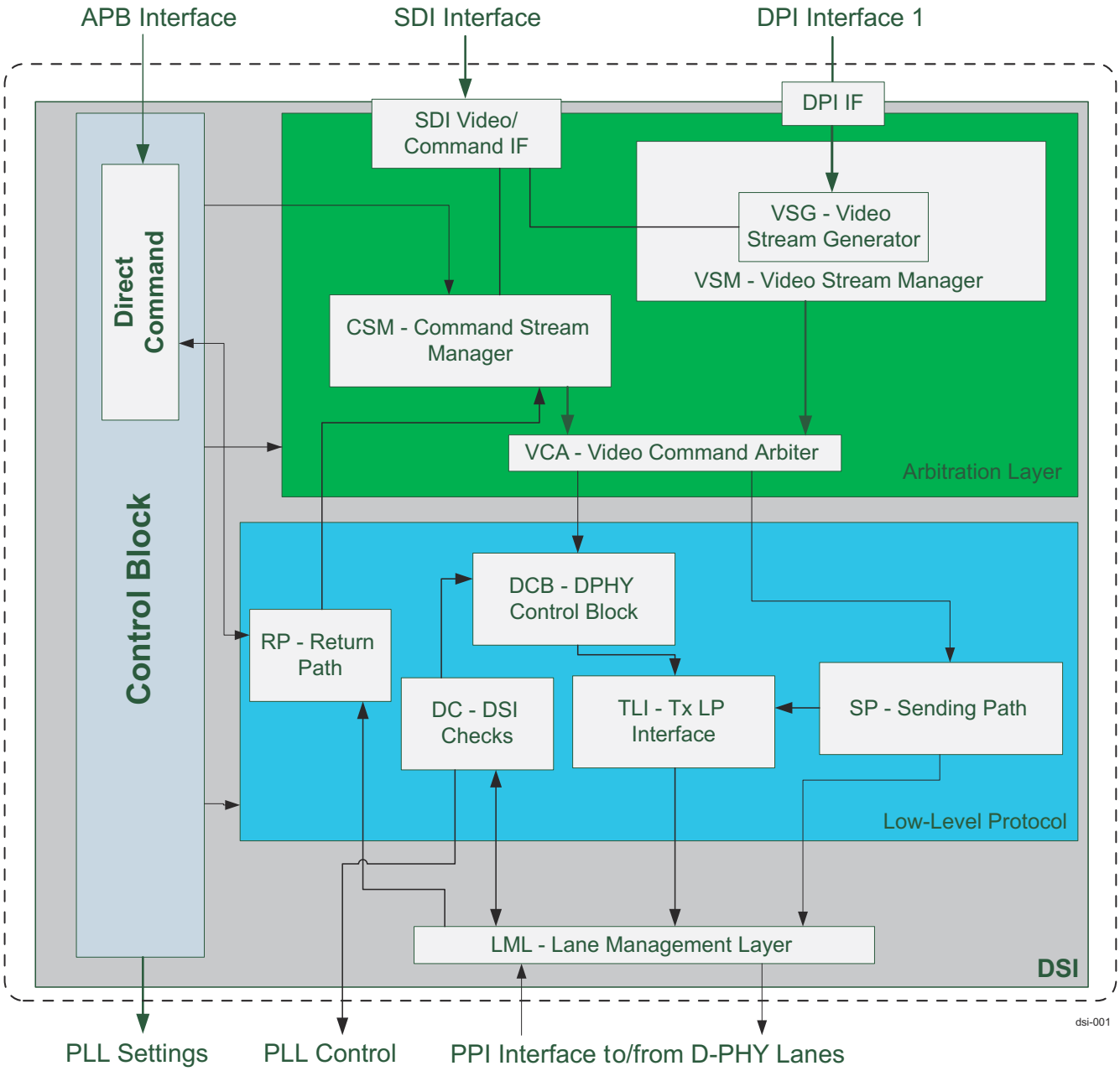


Figure 12-568. DSITX Controller Block Architecture

12.9.2.2 DSI Clocking

The DSI receives all the clocks from DISPC or DPHY_TX modules. There are no other dedicated PLLs. For more details on DSI clocks mapping at DSS and SoC level, see *DSI Integration*.

The DSI requires multiple clocks running asynchronously, that are shown in Table 12-485.

Table 12-485. DSI Clocking

Clock	Direction	Min	Max	Constraints	Description
dpi_0_clk (Async to all other clocks)	Input	7 MHz	425 MHz	Fpixel_clk = Ftx_byte_hs_clk × active_lanes × 8/(bits_per_pixel)	Used for DPI interface only. Frequency range depends on expected data rate with respect of number of data lanes, data lane frequency and frame rate

Copyright © 2026 Texas Instruments Incorporated

Table 12-485. DSI Clocking (continued)

Clock	Direction	Min	Max	Constraints	Description
sys_clk (Async to all other clocks)	Input	7 MHz	250 MHz	1. Cannot be slower than tx_byte_hs_clk x datapath_size/ if_datasize (risk of underrun) in SDI mode. 2. Must be greater than: rx_esc_clk x 8. 3. Speed should be sufficient to provide enough data in SDI operation. 4. Must be faster than the tx_byte_hs_clk for Direct command operation.	Main functional clock Also used for the VBUS/APB interface
dphy_0_tx_byte_hs_clk (Async to all other clocks)	Input	10 MHz	312.5 MHz	The max value in SDI interface operation depends on sys_clk: can't exceed sys_clk freq x if_datasize datapath_size Otherwise 312.5 MHz to match the 2.5Gbps limit on DPHY v1.3 specification	DPHY PPI Byte Clock Frequency set by the DPHY and its High Speed input clock (tx_byte_hs_clk = dphy bit rate /8) Maximum limit due to risk of underrun
dphy_0_tx_esc_clk (Async to all other clocks)	Input	1 MHz	20 MHz	No minimum limit is given by the DSI itself - while regular function mode max limit is 20 MHz	TX Escape Clock
dphy_0_rx_esc_clk (Async to all other clocks)	Input	1 MHz	10 MHz	No minimum limit is given by the DSI itself - while regular function mode max limit is 10 MHz	RX Escape Clock

Figure 12-569 describes the clock scheme and domains of the DSI_TX.

Copyright © 2026 Texas Instruments Incorporated

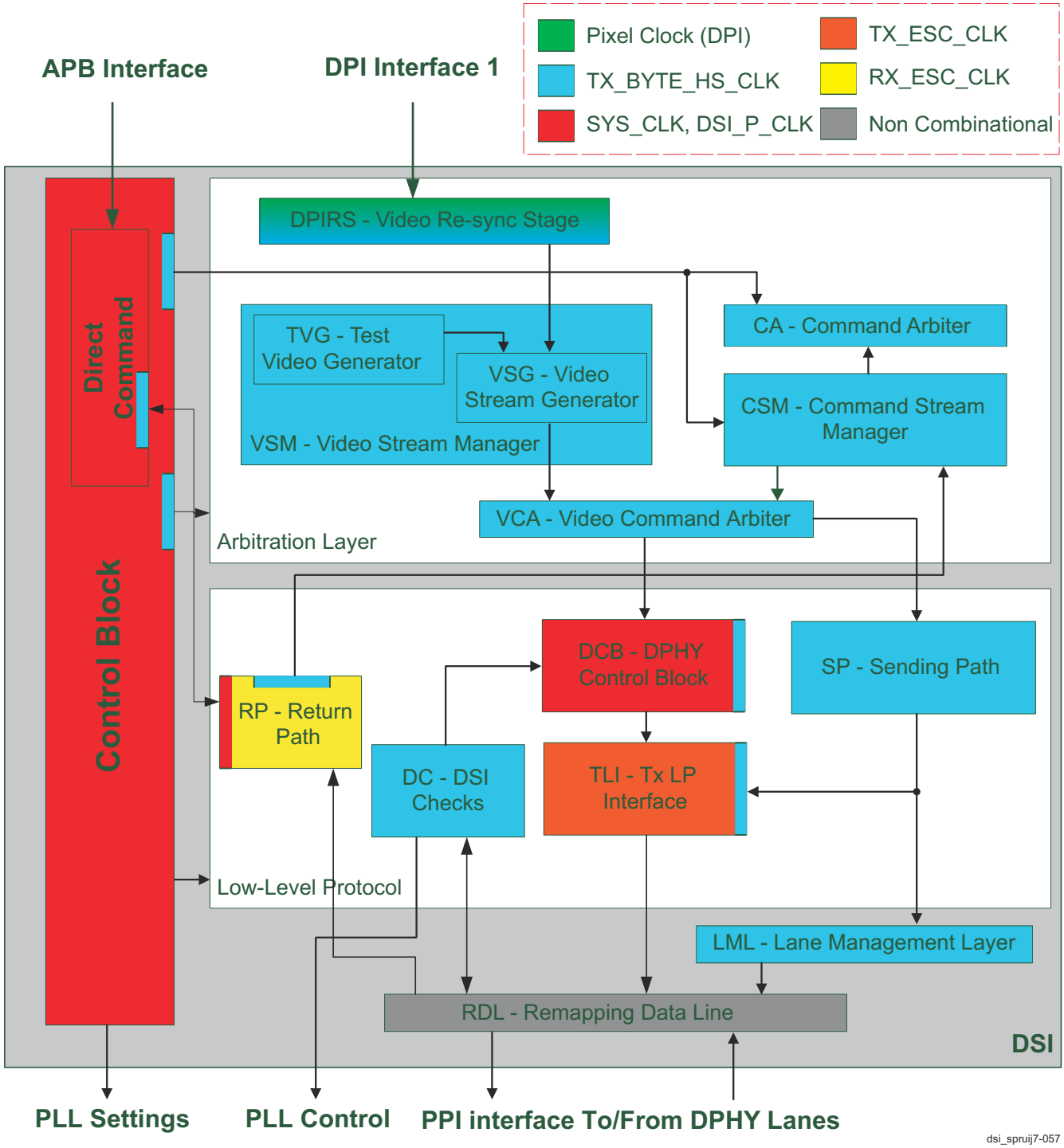


Figure 12-569. DSITX Clock Scheme and Domains

12.9.2.3 DSI Reset

The DSI has one active low reset input. The entire internal logic is reset by this reset. Internal resets (synchronous and asynchronous) for different clock domains are generated internally (synchronized to the respective clock domain).

Copyright © 2026 Texas Instruments Incorporated

12.9.2.4 DSI Power Management

The DSI implements a power management protocol to interface to a PSC (Power and Sleep Controller) module SoC level.

Figure 12-570 shows the expected sequence from SW while performing a *clkstop_req* to the DSI.

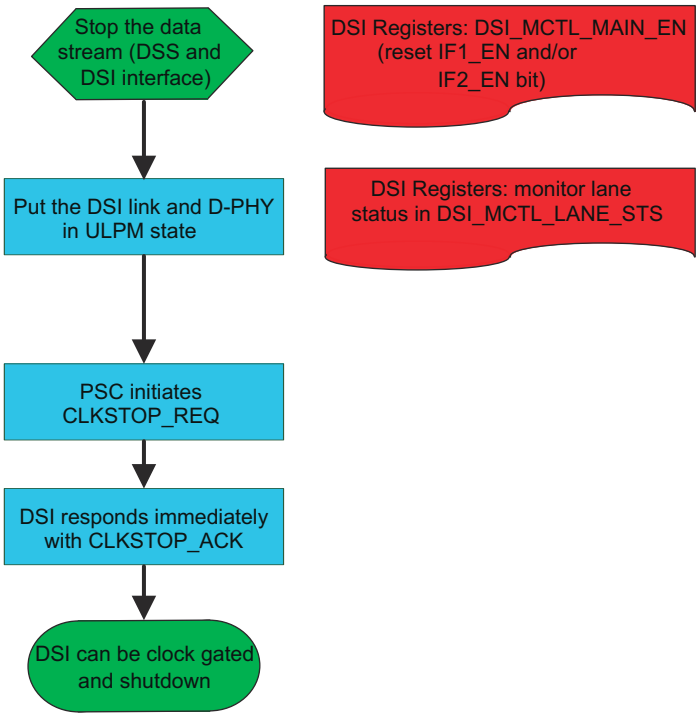


Figure 12-570. DSI Clock Gate / Power Off Procedure

12.9.2.5 DSI Interrupts

Table 12-486 shows the interrupts that are generated by the DSI module.

Table 12-486. DSI Interrupts

Interrupt (n = 0)	Description	Source Module	Source Signal Name
dsi_#n_func_intr	DSI-#n Func Interrupt	DSI Controller	dsi_irq_n
dsi_#n_safety_error_fatal_intr	DSI #n Internal Diagnostic Fatal interrupt	DSI Controller	asf_int_fatal
dsi_#n_safety_error_nonfatal_intr	DSI #n Internal Diagnostic Non-fatal interrupt	DSI Controller	asf_int_nonfatal
ecc_intr_uncorr_level_sys_intr	ECC aggr uncorrectable error	-	-

12.9.2.6 DSI Internal Interfaces

12.9.2.6.1 Video Input Interfaces

DSITX supports following video streaming interfaces:

- DPI (Uncompressed video stream)
- SDI (Serial Display Interface)

The DSI does not require DSI sub-link support. Therefore, only one video mode source is supported.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.6.1.1 Pixel Mapping

The pixel bit color mapping for the DSITX input interfaces are as shown in Figure 12-571. This matches with the pixel bit mapping on the output of DISPC for the supported formats.

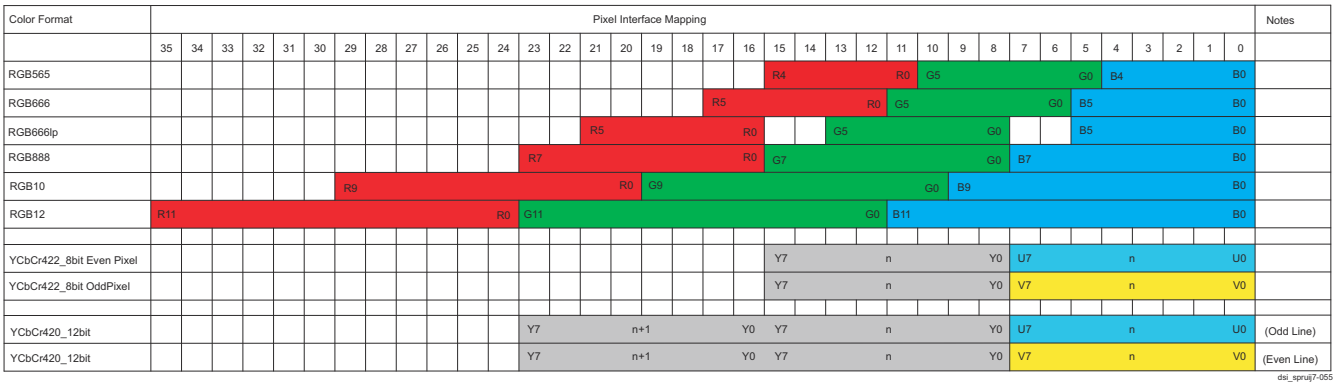


Figure 12-571. DSI Pixel Mapping On The DSI-DPI Interface

Table 12-487. DSI Pixel Format Interoperability between DISPC and DSITX

DSITX DPI Pixel Format	DISPC VPOUT Pixel Format[1] DSS0_VP_CONTROL[10-8] DATALINES Register Field Setting	DPI Interface Support	SDI Interface Support
RGB565	16-bit (DATALINES = 0x1)	Supported	Supported
RGB666	18-bit (DATALINES = 0x2)	Supported	Supported
RGB666lp	N/A	Not Supported	Not Supported
RGB888	24-bit (DATALINES = 0x3)	Supported	Supported
RGB10	30-bit (DATALINES = 0x4)	Supported	Not Supported
RGB12	36-bit (DATALINES = 0x5)	Supported	Not Supported
YcbCr422 8-bit	N/A	Not Supported	Not Supported
YCbCr420 12-bit	N/A	Not Supported	Not Supported

(1) Output aligned on the LSB of the pixel data interface.

12.9.2.6.2 DPI (Pixel Stream Interface)

- One uncompressed video stream input
- CEA-861 signaling/timing compatible interface
- Pixel clock rate range from 7 MHz - 425 MHz
- Frame Resolution – 4MPix@60fps equivalent frame size – supporting various frame resolutions and aspect ratios (with pixel clock freq < MAX_dsi_pixel rate = 425 MHz)
- Progressive video timing only
- Maximum Width - 8K (should only be constrained by the throughput)
- RGB format only –color space conversion or chroma up-sampling NOT supported

12.9.2.6.2.1 Signals

DPI interface supports the following set of DPI standard signals:

- VSYNC – Video Vertical Blanking indication signal as defined in the above standards
- HSYNC - Video Horizontal Blanking indication signal as defined in the above standards
- DATA ENABLE – Video data enable indication signal as defined in the above standards
- DATA - 36 bits of data for supporting up to 12bpp
- Clock – As defined for each format from 7 MHz - 425 MHz

Copyright © 2026 Texas Instruments Incorporated

12.9.2.6.3 SDI (Serial Data Interface)

DSITX uses a SDI interface (32-bit) to receive active video data from an internal frame memory using a DMA or other data pull mechanism. The interface uses req/ready handshake signaling to control the data flow.

The DSITX SDI Interface supports one of the following modes:

- Command-only

SDI-DSI Interface requires following signals:

- dsi_if_valid
- dsi_if_stall
- dsi_if_start (Line Start)
- dsi_if_frame_sync (frame_sync)
- dsi_if_data

The SDI interface of DSI_TX receives data directly from DSS VP output and is functional when the user configures the DSS to operate in COMMAND_STALLMODE (DSS0_VP_CONTROL[11] STALLMODE = '1' and DSS0_VP_CONTROL[12] STALLMODETYPE = '0').

12.9.2.6.3.1 Secure Display Support

Only secure peripherals can be connected to a secure DPI (VP) output of DSS. In order to support this requirement, the DSS exports a secure qualifier to the DSI_TX. Inside the DSI a SECURE register is implemented (DSI_WRAP_DPI_SECURE), which contains SECURE bits, [0] DPI_0_SECURE and [1] DPI_0_SECURE_VIOLATION, corresponding to the DPI port. These SECURE bits can only be set or reset by a secure host (vbusp transactions with secure qualifier set).

The behavior of DSI module for different settings of SECURE register bit and SECURE qualifier from DSS is as shown below:

Table 12-488. Secure Display Support

SECURE Register Bit (DSI)	SECURE mqualifier (DSS)	Security Violation Status	Comments
0	0	0	Data is passed (non-secure DSS data through non-secure DSI)
1	0	0	Data is passed (non-secure DSS data through secure DSI)
1	1	0	Data is passed (secure DSS data through secure DSI)
0	1	1	**Security Violation**. Data is blocked (secure DSS data through non-secure DSI)

Once a security violation is detected, it is captured in the DSI_WRAP_DPI_SECURE[1] DPI_0_SECURE_VIOLATION register status bit.

12.9.2.7 DSI Programming Guide

This section describes the programming guidelines for the DSI Controller and D-PHY.

The goal of this section is to present more in detail how the DSI link should be used at application level plus some scenarios of use of the DSI link (start-up, display switch on/off, dual-display).

12.9.2.7.1 Application Guidelines

The purpose of this section is to provide guidelines on how to drive the DSI host controller, provide general sequence of operations and some typical application notes.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.1.1 Overview of a Display Subsystem

Figure 12-572 shows an overview of a display subsystem. It covers the point to point interfaces between the DPI/SDI and DSI blocks. Other blocks and interfaces are included to give a better understanding of the environment for this interface.

A DSI interface is made up from one DSI controller and one D-PHY block. D-PHY is the physical layer of the DSI interface. The DSI block handles protocol. The DSI interfaces support displays that comply with the MIPI DSI standard.

The displays can be of either command mode type or video mode type. Command mode displays have a frame memory that can hold the entire image. The display is refreshed from this memory and only changes of the display content are sent over the display interface. Video mode displays do not have this memory and thus all data for every refresh must be transferred in real time.

The SDI block can support one display in command mode. By using command mode displays and rerouting of connections, between display updates, more than two displays can be connected and active.

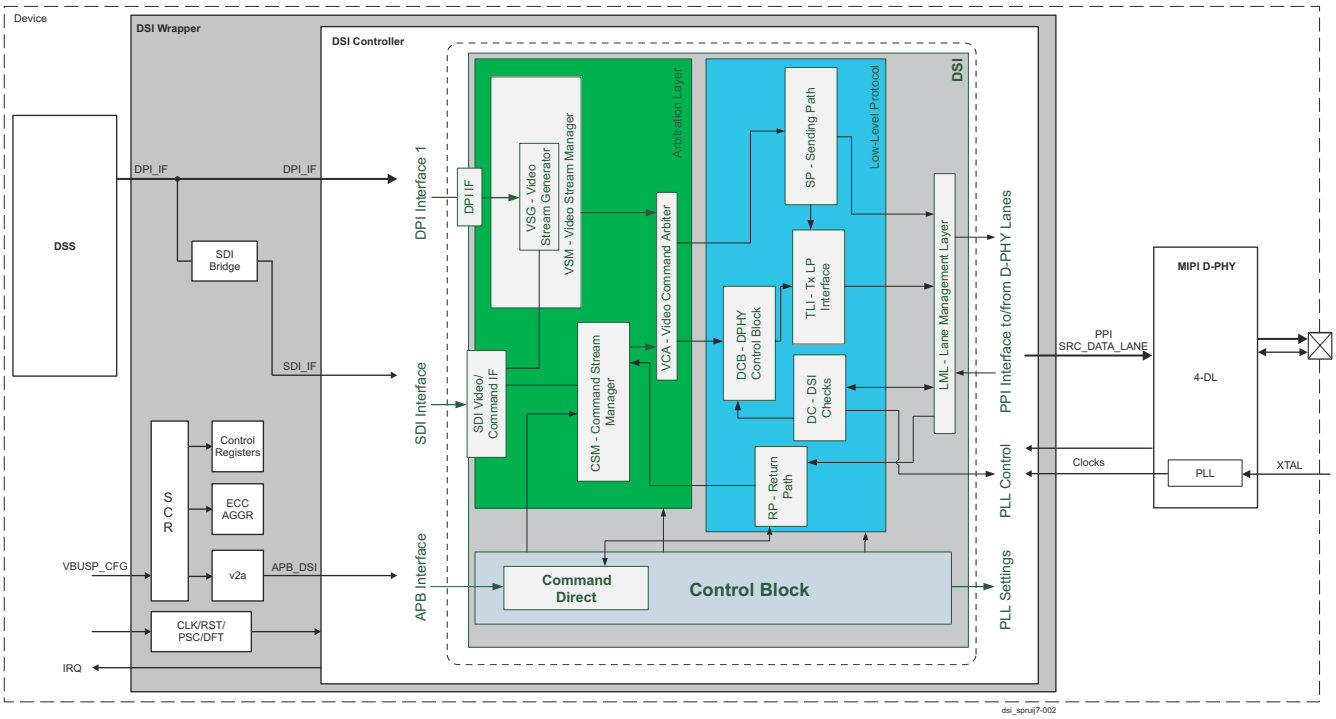


Figure 12-572. Example Display Subsystem Showing Controller and PHY Blocks

12.9.2.7.1.2 D-PHY And DSI Configuration

The DPHY and DSI controller must be configured before any panel video display can be used. All accesses to the DPHY to configure it for use with the DSI Controller should be performed through the DPHY own APB interface .

12.9.2.7.1.3 DSI Controller Initialization

The DSI controller will always require the DPHY and data path configuration to be set after power up and any hard reset. The minimum sequence should define the data path lane configuration; mappings, available lanes, DPHY power, resets and lane swap. All these register values should be set before the DSI controller is enabled for functional operation.

The DSI registers are accessed through the APB interface. The programming sequence for command, video and controller configuration must follow a sequence of accesses to ensure the clock gate control updates the internal nodes before the link is enabled.

Copyright © 2026 Texas Instruments Incorporated

All DSI_VID_xxx registers must be programmed, if required, before the write to the DSI_VID_MAIN_CTL register.

All DSI_MCTL_DPHY_xxx registers must be programmed, if required, before the write to the DSI_MCTL_MAIN_DATA_CTL register to set the [0] LINK_EN bit.

12.9.2.7.1.4 Panel Configuration Using Command Mode

The DSI controller can provide DCS command access using either the SDI interface or through the APB interface control of the DIRECT COMMAND registers (see Table 12-489). The panel registers will be programmed using the LP mode before the high-speed video is enabled. Commands can either be sent using the command interface or though the Command FIFO using the APB registers.

The command FIFO access is done though the APB Command mode registers by forming packets of DCS commands. The DCS commands can be found in the MIPI DCS specification or the display panel data sheet.

DCS commands are essentially register write or read transfers to panel registers. The initialization of a panel will require APB register writes to fill the FIFO with the DCS commands. The commands can be grouped into a long packet by sending the packet split into 4 bytes to fill the FIFO using a write to the DSI_DIRECT_CMD_WRDAT register.

Table 12-489. DSI Direct Command Mode Registers

DSI Register	Description
DSI_DIRECT_CMD_SEND	Direct Command - trigger the direct command sending - write only
DSI_DIRECT_CMD_MAIN_SETTINGS	Direct Command - main settings
DSI_DIRECT_CMD_STS	Direct Command - status - read only
DSI_DIRECT_CMD_RD_INIT	Direct Command - stop read operation
DSI_DIRECT_CMD_WRDAT	Direct Command - data to write byte 0 to 3
DSI_DIRECT_CMD_FIFO_RST	Direct Command- reset the write FIFO pointer

Each DCS command packet to be sent can be built by first selecting the packet configuration using the DSI_DIRECT_CMD_MAIN_SETTINGS. The DCS Command data types are defined in the DSI specification based on the number of parameters that are expected to be sent. Configuration of a panel display will normally use DCS write data types.

Table 12-490 shows the DSI_DIRECT_CMD_MAIN_SETTINGS register bit description.

Table 12-490. DSI Main Settings Register Description

DSI_DIRECT_CMD_MAIN_SETTINGS Register Bit	Description			
[24] CMD_LP_EN	Enables LP sending for the command request.			
[23:16] CMD_SIZE	Size of the command in case of write command - when written value is bigger than 0x10, the value is rounded to 0x10 for write - when size is bigger to 0x2 for read it is rounded to 0x2.			
[15:14] CMD_ID	In case of read/write command, Virtual Channel of the command.			
[13:8] CMD_HEAD	In case of read/write command, data type of the command:			
	0x05	DCS Write	0 parameters	Short Packet
	0x15	DCS Write	1 parameter	Short packet
	0x39	DCS Long Write	N parameters	Long Packet
[3] CMD_LONGNOTSHORT	This bit must be tied to one.			
[2:0] CMD_NAT	Nature of the direct command: 000: write command.			

Once the packet is loaded, issue a DIRECT_CMD_SEND (via the DSI_DIRECT_CMD_SEND register) and the data will be sent in LP mode.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.1.5 VIDEO Interface Configuration

The DSI video operation requires the configuration of the VSG and TVG registers to match the panel configuration. The two video interface options will also require careful selection of the clock and registers to achieve error free video streaming.

The DSITX controller supports the mechanism for initial skew calibration for D-PHY data rates greater than 1.5 Gbps.

DPI Video Interface Operation

The time taken to output a frame on the PPI needs to match what is coming in on the DPI. This is best achieved by using the recommended clock ratio between the pixel and byte clocks. If this is the case, then the blanking and active data periods must be matched up. The DSI controller will match its frame timing to the incoming DPI **VSYNC**, if it is programmed to generate a frame of slightly less than the same size when the VFP blanking is considered.

The controller works by transitioning to LP during the last programmed line of VFP. It will then remain in LP until the start of the next frame. So, programming the controller to match the DPI, but with at least one fewer line of VFP should result in a reliable configuration.

Program the DSI vertical size registers as follows:

- VSA = DPI_VSI (lines, minimum of 2)
- VBP = DPI_VBP (lines, minimum of 0)
- VACT = DPI_VACT (lines)
- VFP < DPI_VFP (minimum of 1)

The timing should also be matched per line, therefore the blanking and active periods should match. DPI horizontal timing is measured in pixels, whereas the DSI controller uses bytes. If any of the DPI related interrupts are triggered, then this highlights that the FIFO depth and/or the vsync_delay settings require to be tuned to the current configuration. Simulating the core operation with the expected clocks is the best way to ensure the FIFO depth and vsync_delay is suitable.

The relationship therefore depends upon the pixel format, which could be 24, 18 or 16 bpp. Additionally, the PPI short packets and packet headers that are inserted by the controller must be accounted for. HSA should be reduced by 14 bytes to account for the HSS short packet (4 bytes), the long blanking packet header and footer (4 + 2 bytes) and the HSE short packet (4 bytes). HBP should be reduced by 12 to account for the header and footer on the blanking packet (6 bytes) plus the header on the active data packet (6 bytes). HFP should be reduced by 6 bytes to account for the long packet header and footer. Finally, for lines with no active data, the BLKLINE_PULSE_PCK is the total size minus 20 bytes (14 for HSA, 6 for the remaining blacking which is all combined into a single packet).

Program the DSI horizontal size registers as follows:

- HSA = (DPI_HSA × bpp/8) - 14
- HBP = (DPI_HBP × bpp/8) - 12
- HACT = (DPI_HACT × bpp/8) NOTE: (DPI_HACT × bpp/32) must be an integer.
- HFP = (DPI_HFP × bpp/8)-6
- BLKLINE_PULSE_PCK = ((DPI_HSA + DPI_HBP + DPI_HACT + DPI_HFP) × bpp/8) - 20;

TVG Generator

The TVG oversees generating dummy data (to display something even if there is no video stream running). It is also able to ease verification or validation of the DSI without having a complete environment (application or verification test bench). The controller has a Test Video Generator that can be programmed to generate a set of test colour patterns based on the display panel that will be used. The panel parameters for horizontal and vertical resolution along with the frame rate and pixel colour depth need to be set based on the datasheet information for the panel.

Figure 12-573 presents TVG MODE patterns.

Copyright © 2026 Texas Instruments Incorporated

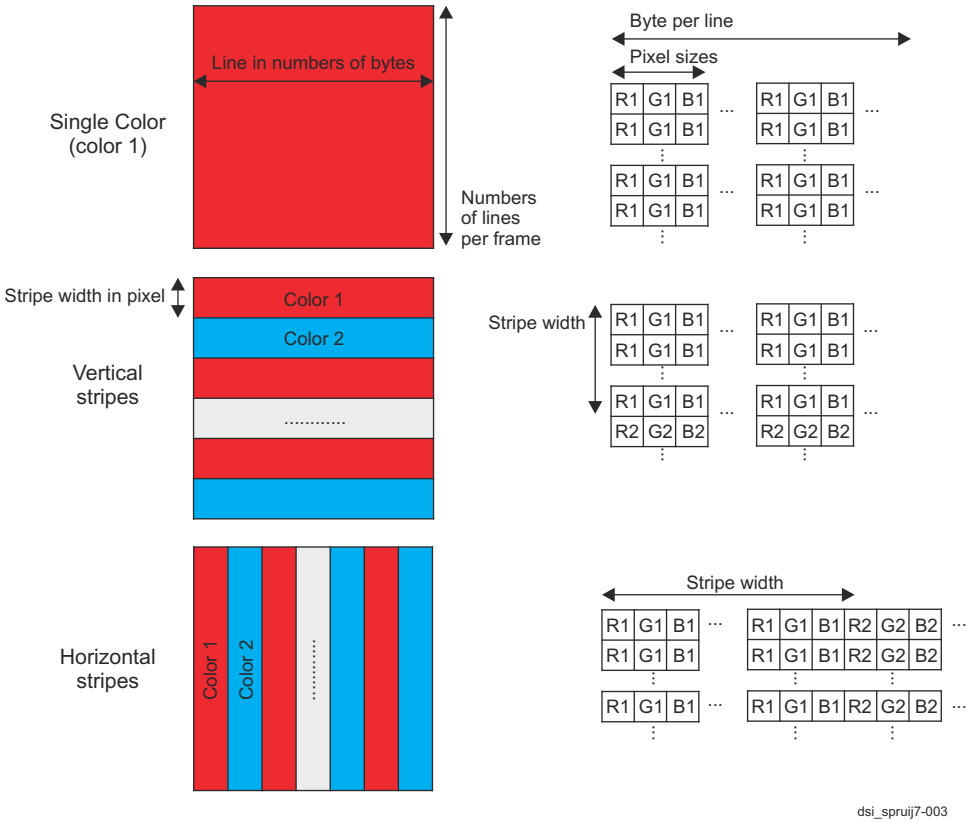


Figure 12-573. TVG MODE Patterns

The TVG generates a data stream in the same format than the one specified for the normal functional video stream (as set in VSG register). The content of that flow is specified by registers. Those registers specify the following information: Image size: Number of bytes per line (see TVG_LINE_SIZE[14:0] register) and number of lines per frame (TVG_NBLINE[12:0]).

Image kind: Single color, vertical stripes or horizontal stripes (see TVG_MODE[1:0] register). Stripe width: Significant when a stripe mode is enabled. Possible values are 1, 2, 4, 8, 16, 32, 64 and 128 (see TVG_STRIPE_SIZE[2:0] register).

Pixel kind: 16 bits RGB, 18 bits RGB, 18 bits loosely RGB, 24 bits RGB, 30 bits RGB or 36 bits RGB (see VID_PIXEL_MODE[3:0] register).

Color one: Color used in single color mode or first color when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant ones are the ones considered (see COL1_GREEN[11:0], COL1_RED[11:0], COL1_BLUE[11:0] registers).

Color two: Color two when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant are the one considered. (see COL2_GREEN[11:0], COL2_RED[11:0], COL2_BLUE[11:0] registers). Start pulse and stop handshake (+ stop mode) (see TVG_CTL register).

An example of the sequence is as follows:

- Select TVG for video stream generation instead of SDI 1 interface:
  - IF1_EN bit to 0 (stall signal at 1) in MCTL_MAIN_EN.
- Select video mode for interface 1 in MCTL_MAIN_DATA_CTL:
  - TVG_SEL bit to 1 in MCTL_MAIN_DATA_CTL register to select stream from TVG.
- Select generated frame format in TVG_CTL register:
  - Image format (single color, H stripes, V stripes) in TVG_MODE field.

Copyright © 2026 Texas Instruments Incorporated

  - Image color selection in TVG_COLOR1, TVG_COLOR1_BIS, TVG_COLOR2, VG_COLOR2_BIS.
  - Stripes size in TVG_STRIPE_SIZE field.
  - Image size in TVG_IMG_SIZE register.

  Note: The number of lines per frame and number of bytes per line. It is required that TVG settings on active area match VSG settings on active area. Any mismatch will create an error that is detected in the VSG and forces recovery mode in TVG, stopping test frame generation.
- Select TVG stop mode with TVG_STOPMODE field of TVG_CTL:
  - TVG video stream generation start/stop controlled by the TVG_RUN bit in TVG_CTL register. Polling TVG run status from the TVG_RUNNING bit in TVG_STS register is useful when setting TVG_RUN to 0. TVG video stream generation does not stop instantaneously (depends on TVG stop mode), so this should be checked to confirm the TVG has stopped.

12.9.2.7.2 Application Considerations

The following sections outline the normal operation of the DSITX controller and the programming and sequences require to operate the DSITX controller for normal video and command operation.

12.9.2.7.2.1 D-PHY Timings Control

Several DPHY timing constraints must be respected by the system, which are described in this section. When a switch in ULP is requested, the application should be aware that the D-PHY needs 1 ms to leave this state (this timing is guaranteed by the DSI itself). It makes no sense to start an ULP request if the expected time in ULP is short.

When PLL power down is asserted, the system should continue to assert for a specific minimum period of time (PLL internal requirement – please refer to DPHY documentation). Please note that shutting down the PLL implies that no clock is present on tx_byte_hs_clk input and thus the DSI link is stopped; even if the system-side interface clocks remain active, no more HS transfers can be done.

Force_stop, ppi_c_force_tx_stop (clock lane) and ppi_d_force_tx_stop (1 per data lane), can be useful to resolve certain deadlock situations, for example: DSI target does not give back control to controller, or critical DPHY error. When a 'force stop mode' is issued by the application, the system should maintain this state on the bus to ensure the request is correctly considered by the D-PHY cells and the **stop_state** is asserted and direction signal is deasserted.

reg_wait_burst_time (for the LML) must be programmed to ensure that two HS bursts are separated by at least 100 ns. The value will be based on the period of the tx_byte_clk cycle.

VCA: When trying to interleave commands to video stream, it is assumed that only write/read commands and BTA request are sent (no TE). The reason is that such transfers are slow and difficult to predict in term of duration. They could take longer than the slots available for video. The read commands are unpredictable and may cause some break in the video stream, however, they can still be used, if an appropriate response time can be guaranteed. It is the responsibility of the system integrator and the application to plan and implement recovery procedures when the video stream is stalled due to a read that takes too long.

12.9.2.7.2.2 Control Block

Most of the register contents must be resynchronized against a DSI internal clock (tx_hs_byte_clk). This means there is a certain time to pass data from one clock domain to the other. If two writes in the same register are too close, the register itself is updated but the synchronization may fail.

The formula to calculate the number of system clock periods between two writes is:.

nb_cycle(dsi_p_clk) = 6*($f_{dsi_p_clk}/f_{tx_byte_hs_clk}$ + 1)

The application must respect a period of TX ns between two write accesses to the same register (there is no issue writing different registers back-to-back).

Table 12-491 shows an example of required time between two write accesses.

Copyright © 2026 Texas Instruments Incorporated

Table 12-491. Example of Required Time between Two Write Accesses

$f_{dsi_p_clk}$	$f_{tx_byte_hs_clk}$	TX minimum time between 2 writes accesses
200 MHz	106 MHz	~ 90 ns
160 MHz	10 MHz	~ 650 ns
200 MHz	10 MHz	~ 650 ns

Another place where asynchronous behaviour may interfere with programming in the CB is the direct command status register DIRECT_CMD_STS. Some of the bits of this register are set when a pulse (that lasts one clock period) coming from tx_hs_byte_clk domain is observed (after resynchronization on dsi_p_clk) and is reset when the clear bit is written. In use case where tx_byte_hs_clk is slow (10 MHz range), the bit can be read and a clear attempted before the end of the source pulse. This can result in the re-assertion of the bit immediately after it is cleared. For that reason, it is recommended to wait for a while between reading the bits of the DIRECT_CMD_STS register and clearing them.

Application Issues

Some register fields cannot take all the possible values but are restricted to a certain number of combinations (mode control). The DSI controller does not check that all the fields match a valid setup so it is the responsibility of the system integrator and the application to check that the written values are amongst the permitted combinations. Amongst the register fields that fall in to this category are most of the mode settings (stop mode, recovery mode, direct command type, image sizes, etc.)

Programming Coherency

The application level must ensure that the register configurations are valid for the system to operate correctly. There are several possible combinations are possible in the CB registers however should be avoided. A few examples:

- No HS transfer should be enabled when PLL is shut-off.
- TVG should not be run when the video interface is running.
- No access (from interface or from register) should be attempted while TBG is active.
- Prior to shut-down of the DPHY PLL, application should verify that the DSI link is in proper state. If the bit that enables BTA is not set and that any operation implying a BTA is sent, the system is stalled. Then the bit that enables BTA must be aligned with application needs. Moreover, read enable or TE enable cannot be set if bta_en is not set (these operations cannot be enabled when BTA is not enabled).
- TE feature must not be enabled in SDI interface.

12.9.2.7.2.3 Video Coherency

In video mode, there are a lot of sizes defined for the video stream generation. This data corresponds to the payload of the various generated packets and not to their duration. However, they are closely linked. It is the responsibility of the system integrator and the application to use correct sizes to ensure the correct video timing and behaviour, because the system may use 1 to 4 active data lanes with a fixed number of bytes for the header/checksum overhead. In the same way, when programming D-PHY time (to switch from LP to HS), the application must consider D-PHY time plus overhead due to LLP and LML crossing time (see Section 12.9.2.7.7.2, *Video stream settings (VSG)*).

VSG Control

Start and stop sequences can take a long time to be performed. If the requests are not maintained up to the time the status bit are indicating effective start or stop, they can be ignored and the VSG may not have started or stopped as expected by application. It is then up to the application to carefully manage the request and to verify that requests are being processed before changing the state of the VSG.

Test Generator:

The DSITX provides test generator to provide a video data stream. When using TVG, all sources can be used (provided VCA permits it) however the SDI interface must not be programmed to send video data and must be restricted to command mode.

Copyright © 2026 Texas Instruments Incorporated

The pixel modes supported are: RGB 16-, 18-, 24-, 30- and 36-bit; YCbCr422 16-bit; and YCbCr420 12-bit (note that additional YCbCr422 20-bit and YCbCr420 24-bit may be supported using SDI interface, but are restricted to command mode only).

The supported display sizes are listed below:

- QQVGA (160 x 120) - 15/20/30/60 fps - RGB 16-18-24 (-30 and 36) bits per pixels
- QCIF (176x144), QCIF + (176x208 and 176x220) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- QVGA (320x240) - 15/20/30/60 fps - RGB R16-18 and 24 bits per pixels
- CIF (352x288), CIF + (352x416 or 352x440) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- 1/2 VGA (320x480) and 2/3 VGA (640x320) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- VGA (640x480) - 15/20/30/60 fps - RGB 16-18-24(-30 and 36) bits per pixels
- WVGA (800x480 - 848x480 -854x480 - 852x480) - RGB 15/20/30/60 fps - 16-18 and 24 bits per pixels
- SVGA (800x600) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- QHD (960x540) - 15/20/30/60 fps - RGB 16-18 and 24 bits per pixels
- XVGA (1024x768) - 15/20/30/60 fps - 16-18 and 24 bits per pixels
- Full HD (1920x1080) - 15/20/30/60 fps - 16-18 and 24 bits per pixels
- WUXGA(1920x1200) - 15/20/30/60/(120 fps – 16 bpp only) 16, 18 and 24 bits per pixels
- 4Kx4K(4096x4096) - 15/20/30 fps - 16-18 and 24 bits per pixels

12.9.2.7.3 Start-up Procedure

The start-up sequence described in Figure 12-574 assumes that the rest of the chip is ready (all clocks are present except tx_byte_hs_clk, application it is ready...) and that the DSI is ready (reset de-asserted).

The following is the most common start up sequence:

Firstly, application layer programs and enables the PLL, then waits for a period for the PLL to lock (minimum time will be defined by the PHY documentation). At the end of the minimum wait period, the application layer should poll the PLL lock status register periodically to confirm lock status, repeating until confirmed. At the end of this step, it is assumed that a clock is present on the tx_byte_hs_clk input (this step may be replaced by an interrupt-based alternative if available from the selected DPHY or provided in the system integration logic).

During the time taken by the PLL to lock, the application can program the configuration registers and prepare the DSI link. This can also be done after the PLL is locked. It should at least configure enough to be able to use LP mode to send direct commands.

Enable clock and data lane(s) per the needs. This action is only concerned with programming registers that control the D-PHY lane enable signal. The active lane configuration with the DSI_MCTL_MAIN_PHY_CTL and DSI_MCTL_MAIN_EN registers must be programmed to match, i.e. for two data lanes (0 and1).

- DSI_MCTL_MAIN_PHY_CTL (0x08) register – LANE3/4_EN - bits[2:1] set to zero to disable lanes 2/3.
- DSI_MCTL_MAIN_EN (0x24) register – DAT3/4_EN fields - bits[6:5] set to zero to disable lanes 2/3.

Start the lane in the DCB. The DCB state machine puts the lanes in the correct state to be ready.

At this step, the DSI link should be able to access to display - at least in LP mode. If the system (DSI link and display) is fully programmed, code to display image can be started. If display characteristics must be read from the display and the system is programmed in LP mode, this is time to read the displays parameters to allow correct programming of the DSI. A new round of programming can then take place to set-up the system, based on the parameters which have been read fro the display.

The DSI internal setting is now completed and it is ready to accept data. The application can now enable the interface so that the stall signal is removed and that DSI link can accept data.

When the procedure described above is terminated, it does not imply that display itself is ready. At this stage, the D-PHY and DSI link are ready to send/receive data to/from display and thus application may still need to initialize the display itself.

Copyright © 2026 Texas Instruments Incorporated

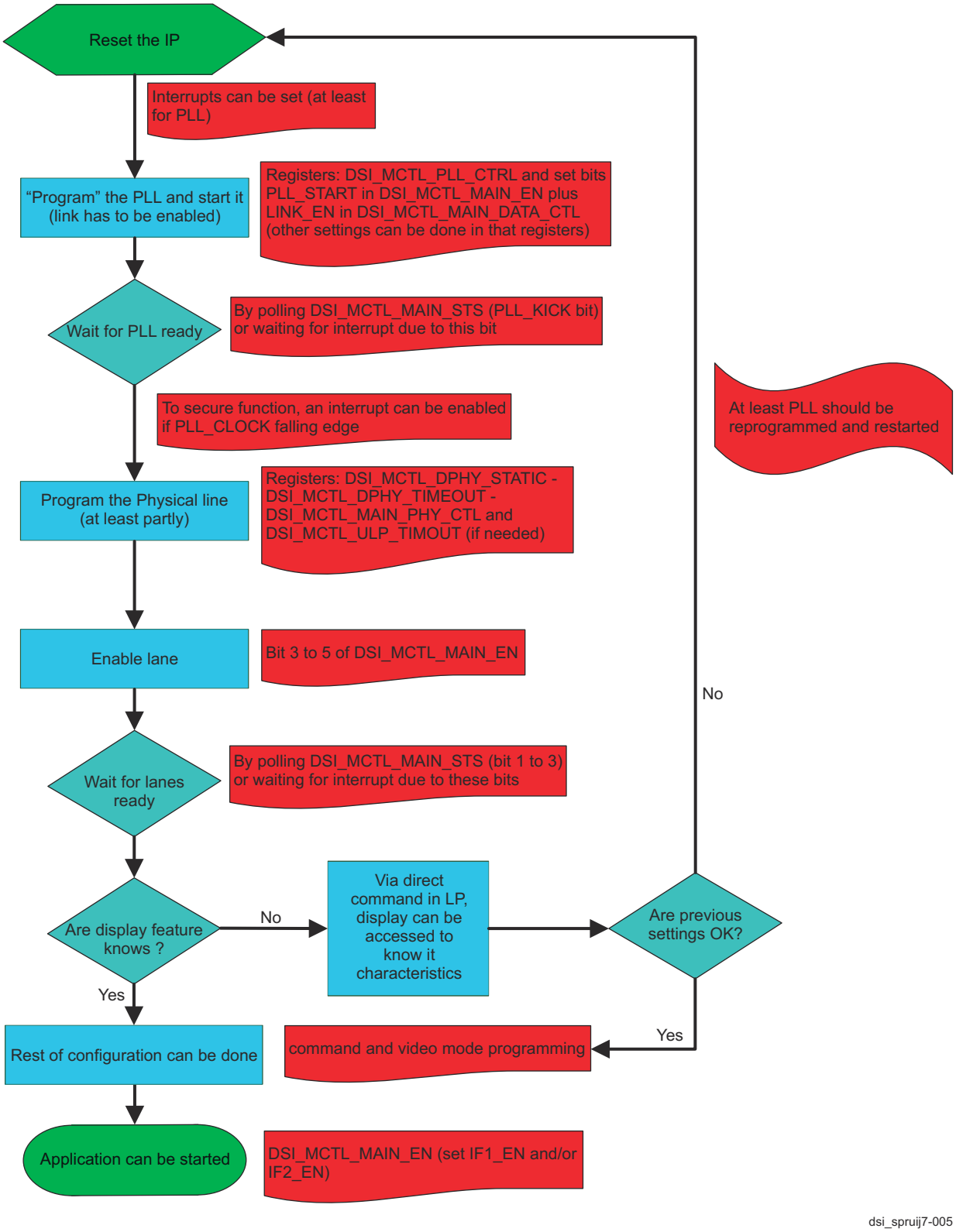


Figure 12-574. Start-up Procedure Summary

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.4 Interrupt Management

There are numerous error and status conditions that can be monitored via the interrupt mechanism. The edge on which these conditions are generated is programmable.

Each interrupt source has four associated registers:

1. The status register itself "<sts_reg>"
2. a control register "<sts_reg>_ctl" to control corresponding interrupt behaviour
3. a flag register "<sts_reg>_flag" to observe when interrupt/event has been triggered
4. A clear register "<sts_reg>_clr" to clear the event flag register.

The decision of which status bits (<sts_bit>) can generate interrupt is taken when the corresponding enable bit "<sts_bit>_en" is set in the status control register. In the same register the status bit edge "<sts_bit>_edge" states on which edge of the status bit the interrupt is triggered. When the selected edge is observed, the flag bit "<sts_bit>_flag" is set. The interrupt signal is an OR of all flag bits that are enabled. When the register "<sts_reg>_clr "is written with the bit "<sts_bit>_clr ", the " <sts_bit>_flag" is cleared.

12.9.2.7.4.1 Error and Status Registers

The error register bank handles storing all error flags that are identified. It also stores some status bits that must be observable and detectable by the application. These status and error registers can generate an interrupt on the rising or falling edge if this generation is enabled.

This mechanism is different depending on the nature of the status or error bits. For those that are generated in the DSI block (basically all status and error bits except direct command), the status bits cannot be easily controlled and toggle according to the internal status meaning only the current value is observed in the status/ error register itself.

If the interrupt generation is enabled with the associated enable bit, a rising edge (associated edge bit set to 0) or a falling edge (associated edge bit set to 1) of that status bit toggles the corresponding flag bit. At the end, all the flag bits are put together with an OR to generate the interrupt signal. The flag register can be reset by writing in the clear register.

The described behavior can be summarized by the following pseudo-HDL. The signal named signal_sts_bit is the bit that is observable by reading status bit.

- reg_sts_q is the status register (accessible via APB)
- reg_flag_sts_q is the flag register
- clear_the_flag is set when the register clear is written
- edge_sts_ctl is the edge bit
- enable_sts_ctl is the enable bit

reg_sts_d <= **signal_sts_bit**;

reg_flag_sts_d <= '0' WHEN clear_the_flag = '1' ELSE
- 1' WHEN (reg_sts_q = '0' AND reg_sts_d = '1' AND edge_sts_ctl = '0') ELSE – detect rising
- '1' WHEN (reg_sts_q = '1' AND reg_sts_d = '0' AND edge_sts_ctl = '1') ELSE – detect falling
- reg_flag_sts_q;

irq_n <= NOT (OR_OF_ALL(reg_flag_sts_q AND enable_sts_ctl));

The code is slightly different in cases where the observed bit is a generated condition (in the control block), as is the case for the direct command status and error flags. The status/error information is automatically generated but is cleared only when writing in the corresponding clear bit. The interrupt generation behaves similarly to the error condition cases, however, as the status falling edge is observed only when the bit is cleared, it is not possible to use the falling edge detection on flag (however the code is kept as-is to provide a standard implementation method). This leads to the following code:

- reg_sts_d <= '0' WHEN clear_the_flag = '1' ELSE
  - '1' WHEN the_sts_bit (Note 1) = '1' AND reg_sts_q = '0' ELSE -- (see Note)

Copyright © 2026 Texas Instruments Incorporated

  - reg_sts_q;
- reg_flag_sts_d <= '0' WHEN clear_the_flag = '1' ELSE
  - '1' WHEN (reg_sts_q = '0' AND reg_sts_d = '1' AND edge_sts_ctl = '0') ELSE – detect rising
  - '1' WHEN (reg_sts_q = '1' AND reg_sts_d = '0' AND edge_sts_ctl = '1') ELSE – detect falling
  - reg_flag_sts_q;
- int <= NOT(OR_OF_ALL(reg_flag_sts_q AND enable_sts_ctl));

Note: In some case (pulse generated), the rising edge detection is not done and the code becomes simply '1' WHEN signal_sts_bit = '1' ELSE...

12.9.2.7.4.2 Interrupt Management for Direct Command Registers

The following direct command status bits/registers are only set when error is generated and only cleared when the clear bit is written:

- all the status bits of the register DSI_DIRECT_CMD_RD_STS
- all the status bits of the register DSI_DIRECT_CMD_STS
- all the status bits except bit 0 (cmd_transmission) of the register DSI_DIRECT_CMD_STS

As these registers are reset only when the clear register is written, it is not possible to detect the falling edge on them to generate interrupts. Only the rising edges can generate the interrupt.

Note: There can be issues with detection when using the bits for all the signals that are a pulse generated in the tx_byte_hs_clk domain. When the speed of that clock is slower in relation to dsi_p_clk, it is possible to have the interrupt set, the bit read and the clear attempted before the end of the tx_byte_hs_clk pulse, after the clear, the bit is set again.

12.9.2.7.5 Direct Command Usage

The direct command mechanism is a way to send commands (with a maximum size of direct_cmd_fifodepth bytes) by writing and reading registers. It allows commands to be sent that are not allowed through the SDI-DSI interface (DSI commands, triggers), to perform read operations or to send DCS / generic write / generic read operations when the DSI link is not fully programmed and cannot use the SDI interface for commands.

Figure 12-575 shows direct command management sequence.

Basically, the procedure to do so is as follows:

- Clear the status bits (when reading the status bits, only the DSI reads the current access status).
- Program the various registers that define the command to send.
- Write the direct_cmd_send register. Writing in that register (any value) triggers the command sending mechanism.

  The command should be started only if the command request is amongst the defined ones for DCS or Generic Write or Read with 0, 1, or more parameters:
  - DCS Write codes 0x05, 0x15, 0x39
  - DCS Read codes 0x06
  - Generic Write codes 0x03, 0x13, 0x23, 0x29
  - Generic Read codes 0x04, 0x14, 0x24

  Note: The direct_cmd_main_settings register can use other data type values for cmd_head to support future types and no check is performed for invalid types.
- Wait for the assertion of the command_completed bit in the direct_cmd_sts register (or any of the relevant bits that indicate a specific command termination). Instead of polling the registers, it is possible to enable the corresponding interrupts and wait for its detection.
- Read the received data and (if any) the different status bits.

Copyright © 2026 Texas Instruments Incorporated



Figure 12-575. Direct Command Management

There are several checks performed when sending a direct command to verify that requested command sizes are supported and to check that the requested command exists (otherwise the request to send the command is ignored).

- For Short packets commands the DCS and Generic Read and Write the data size is rounded to maximum of 2.
- For Long packet DCS and Generic Read and Write the data size is rounded to the maximum size of the FIFO.
- No checks are made for the write FIFO having the expected number of bytes programmed into the CMD_SIZE in the dir_cmd_main register. In this case the data written out will loop back through the value in the FIFO.

At completion of a read command, registers can be read (they contain read data and/or errors detected during the communication) and trigger events are sent to the application to announce the request completion (using the status and error register bank).

Copyright © 2026 Texas Instruments Incorporated

When a read request is performed, the read data is put in the RP FIFO. If more bytes are sent, they are ignored and errors are set accordingly. The received read can then be accessed by reading several times the register that gives an access to the FIFO.

For every new command, several status bits are present: one signals that a command is being transmitted (cmd_transmission). When the command is complete, several bits can be toggled to signal it (write_completed, trigger_completed, te_received, read_completed and read_completed_with_err).

Some complex commands provide intermediate information. For BTA requests, there is a bit that signals when the BTA request has been sent (BTA_completed) and another one that signals when the DSI is controller of the D-PHY interface again (BTA_finished). For commands that imply a BTA, some other bits specify whether specific data has been received (such as trigger, acknowledge with or without error).

12.9.2.7.5.1 Trigger Mapping Information

Triggers are one of the groups of direct commands that can be sent and received using the DPHY in Low Power escape mode. The trigger_val[3:0] bits (in the registers DIRECT_CMD_MAIN_SETTINGS and DIRECT_CMD_STS) are used to define which of the four possible trigger entry codes has to be sent or has been received. The register fields described above are copied in signal ppi_d1_tx_trigger_esc[3:0] (for trigger_val[3:0] of DIRECT_CMD_MAIN_SETTINGS) or are a copy of ppi_d1_rx_trigger_esc[3:0] (for trigger_val[3:0] of DIRECT_CMD_STS). Only one bit out of the four should be set else the D-PHY behavior is not very precisely defined. Table 12-492 gives the trigger mapping in the system.

Table 12-492. Trigger Mapping

Trigger name	Trigger entry code	trigger_val	meaning in TX direction	meaning in RX direction
trigger 0 - reset	01100010	1b0001	Reset	Not affected by DSI spec
trigger 1 - unknown 3	01011101	1b0010	Not affected by DSI spec	TE response(1)
trigger 2- unknown 4	00100001	1b0100	Not affected by DSI spec	Acknowledge with no error(1)
trigger 3 - unknown 5	10100000	1b1000	Not affected by DSI spec	Not affected by DSI spec

(1) These triggers are not observed in the trigger_val field of the direct_cmd_sts register because they are used by the DSI link for a specific purpose.

Trigger Reset – This will request from the display and expects an immediate reset of the DSI command/video mode with all pending requests in TX sending path FIFO for transfer to be discarded.

Acknowledge – This is a response to DSITX controller. The host processor may request a command acknowledge and error information related to any transmission by asserting Bus Turnaround with the transmission. The peripheral shall respond with ACK Trigger Message if there are no errors and with Acknowledge and Error Report packet if any errors were detected in previous transmissions. Appropriate flags shall be set to indicate what errors were detected on the preceding transmissions.

If the transmission was a Read request, the peripheral shall return READ data without issuing additional ACK Trigger Message or an Acknowledge and Error Report packet if no errors were detected. If there was an error in the Read request, the peripheral shall return the appropriate Acknowledge and Error Report unless the error was a single-bit correctable error. In that case, the peripheral shall return the requested READ data packet followed by Acknowledge and Error Report packet with appropriate error bits set.

Tearing Effect - The Tearing effect on the display is avoided by having synchronization information from the display. It is used only in command mode. Users are responsible for selecting the command mode for the VC using the TE feature and selecting the polling or automatic mechanism.

The user is responsible for ensuring that there is a delay after a trigger is issued by the host to the panel, before any other action, message or trigger is performed. When the host issues a reset trigger, the system will expect the controller to stop and re initialise the clock and data links. All other triggers issued by the host will be application specific and are outside the scope of this document.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.5.2 Command Mode Settings

The command mode is enabled on the SDI interface. This is controlled using the register mctl_main_en. Other settings can be set using the registers: mctl_main_data_ctrl (to decide about TE usage, read enable...); and cmd_mode_ctl (Virtual Channel of the command packets, arbitration between requests of the two interfaces, possibility to use LPDT, TE timer programming and padding value in case of an error).

When programming Direct READ commands, a BTA request is sent automatically following the read transmission, for the peripheral to respond – it is therefore not necessary to explicitly send a BTA request at that time. The system must allow the BTA request to be completed and the bus returned to the host before issuing any new read or write command. The system must either: poll the read_completed (and read_completed_with_err) status bits; or, wait for an interrupt caused by these bits before issuing any new command.

When programming Direct WRITE commands, it is advisable to ensure that each transaction can complete successfully before moving on to a subsequent command. There are two recommended methods to achieve this, either: explicitly request a BTA between write transactions while checking for the associated interrupt or polling the appropriate status flag (this method also provides the added security of an ACK response from the peripheral); or allowing a gap between commands sufficient to allow completion of the first before commencing with the second. The recommended approximate gap time required to ensure that all commands complete successfully is equivalent to 100 TX ESC byte periods. If required (for instance, to minimize the start-up time), it is also possible to calculate the minimum delay time required for each transaction, based on the TX escape clock frequency used in the application and the length of each command (no active video transmission should be enabled at the time – a typical use case would be peripheral parameter configuration). As with read operations, any write operation followed by an explicit BTA request must allow the BTA to be completed and the bus returned to the host before issuing any new read or write command.

For single parameter commands, the upper bytes of the 32-bit value written to the Direct Command write data register (direct_cmd_wrdat) should be masked to zero to comply with the DSI specification requirement that unused parameter locations are set to zero. For zero parameter writes, assuming there have been previous write commands sent, it is necessary to clear the sending path FIFO by writing to the direct_cmd_fifo_rst register to clear the data path, then write a zero value to the direct_cmd_wrdat register to send the zero parameter write command.

Note: Failure to follow these procedures may result in non-zero data in the unused parameter locations; this may or may not affect the peripheral, however it is recommended to use the process described to ensure compliance with the DSI specification in this regard.

If STOP mode is forced during a command mode read or write transaction, all outstanding transactions should be considered lost and all status and error flags should be disregarded and cleared. Similarly, if an LPDT read data operation is stopped by writing to the STOP_READ_OPERATION field of the DIRECT_CMD_RD_INIT register, the status for the aborted read operation should be disregarded and cleared before resuming normal operation.

12.9.2.7.5.3 Bus Turnaround Sequence

The command operation can request a read or an acknowledge response from the panel after a long sequence of packets is sent. The Bus turnaround is also used to control the Tearing Effect control.

When a Bus Turnaround is requested from the command stream manager, the DPHY will be given a turn request. The request will make the DPHY interface perform a sequence of LP states (LP11-10-00-10-00) to indicate that it is releasing control of the DP/DN signals. The DPHY Direction signal will change once it detects the Panel side driving the LP00-10-11 states. The panel will then be able to send any response packets (see section 0) and release the bus by issuing a bus turnaround sequence to the controller DPHY. The DSITX controller DPHY will detect the sequence and change the direction signal once the pattern is sent and LP00 state is valid for 2-3 TX_ESC_CLK cycles.

Figure 12-576 shows Bus turnaround timing sequence from controller to panel and panel to controller.

Copyright © 2026 Texas Instruments Incorporated

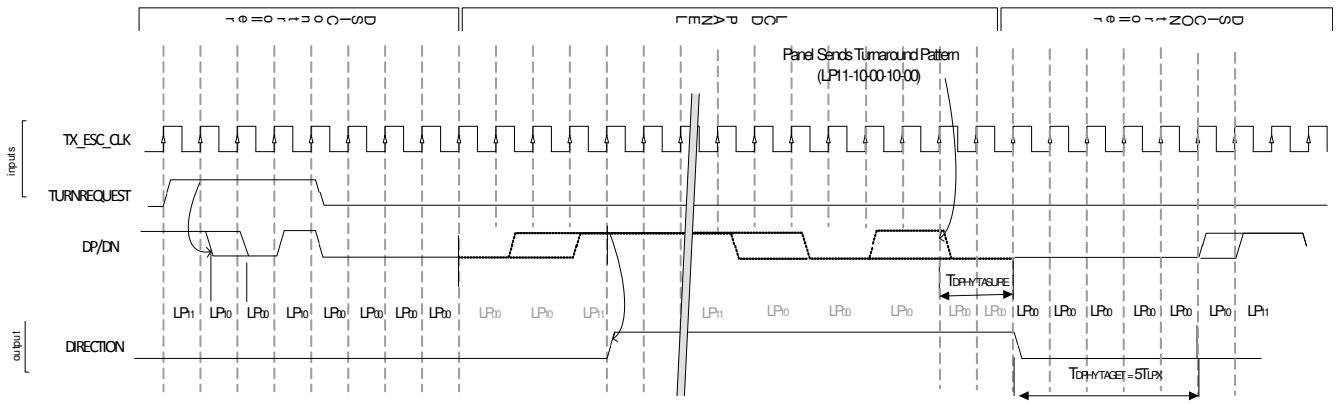


Figure 12-576. Bus Turnaround Timing Sequence from Controller to Panel and Panel to Controller

12.9.2.7.5.4 Tearing Effect Control

The DSI command control for a tearing effect request can be performed using the polling method or automatically. The enable bits for BTA and TE must be enabled for the mechanism to be performed with either the SDI or DIRCMD interface. For automatic operation, the DSI will send a first BTA, and then waits for BTA reception and if no TE has been received during the first BTA (1) , it sends another BTA and then waits for TE. There are three possible error conditions that can happen (but only the two first are detected by the DSITX controller and passed to registers):

- If the display answers to the second BTA with another BTA and thus does not sent the TE - this means that the display does not support TE generation OR that TE generation mechanism has not been enabled (reg_err_no_te).
- If the TE has not arrived within a given programmable period – this means that the TE request arrived too late and that the Display Application Processor is not synchronized with the display. The error reported is reg_err_te_miss but the system continues to wait until the TE is received. (2)
- If a time-out has been detected by the DSI showing that a problem happened during the BTA and that the system is forced back to idle without BTA – in this case, a BTA without TE is seen and the CSM generates a reg_err_no_te (as for first case).

For the polling method to request a tearing effect (TE polling mode is enabled with te_hw_polling_en = 1) the DSI sends a first BTA, waits for peripheral TE + BTA, if only BTA has been received after the first BTA, it keeps sending BTAs until peripheral responds to the BTA by a TE + BTA instead of BTA only; in this specific mode reg_err_no_te will never be asserted. To stop sending BTA and waiting for TE a force stop (3)) is needed.

Note 1: TE received is checked on the first BTA in case a BTA was the last command issued by the DSI link (BTA or read command sent via direct command interface).

Note 2: Again to support the cases where a BTA was emitted just before the TE request, the TE windows is counted on the first BTA (in case the display understands this BTA as the TE request and thus may emits too late the TE), and is restarted after the second BTA.

Note 3: The time to wait before forcing a stop should be around the duration of a frame. The reason is that if you miss the TE time at the end of the current frame, you've to wait till the end of the next frame before new TE can be sent by display.

The following table describes the TE timeout counter operation, Programming of the IP. The counter value should be calculated based on the tx_byte_clk period.

Table 12-493. TE Timeout Programming

te_timeout(11)	te_timeout(10)	timeout value
0	0	256 × te_timeout<9:0>
0	1	512 × te_timeout<9:0>
1	0	1024 × te_timeout<9:0>

Copyright © 2026 Texas Instruments Incorporated

Table 12-493. TE Timeout Programming (continued)

te_timeout(11)	te_timeout(10)	timeout value
1	1	2048 × te_timeout<9:0>

12.9.2.7.5.5 Tearing Effect Control on Panels with Frame Buffer

Display panels will use command mode sequencing when an on-board frame buffer is used to store the video image rather than using the video streaming and synchronisation packets to control the update of the image information. This means that the host display driver must know when the panel is processing the image and avoid changing pixel information before it is updated. The mechanism is known as the tearing effect. The DSITX controller can support two schemes as outlined in the MIPI DSI specification.

TE control using the Polling Method

For polling to the display module, the host processor shall detect the current scan line information with a DCS command such as **get_scan_line** to avoid Tearing Effects. DSITX controller will issue a READ using this command until it gets to the expected value for the last line.

DCS Command - get_scanline (45h)

Command

Direction H->D

Hex Code 0 1 0 0 0 1 0 1 45h

Parameter 1

Direction D->H

Hex Code N15 N14 N13 N12 N11 N10 N9 N8 XXh

Parameter 2

Direction D->H

Hex Code N7 N6 N5 N4 N3 N2 N1 N0 XXh

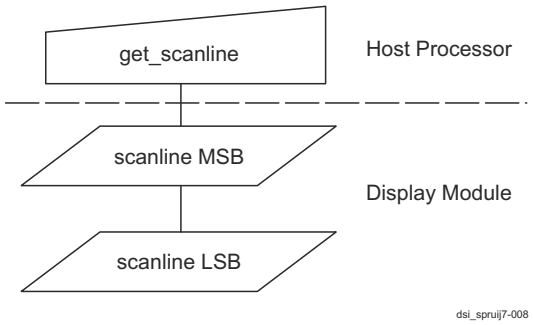


Figure 12-577. get_scan_line Command

Description

The display module returns the current scanline, N, used to update the display device. The total number of scanlines on a display device is defined as VSYNC + VBP + VACT + VFP. The first scanline is defined as the first line of V Sync and is denoted as Line 0.

In Sleep Mode, the value returned by get_scanline is undefined. See [MIPI-DPI] for definitions of VSYNC, VBP, VACT, and VFP.

- In 2D mode, the scanline value of the display memory and the display panel is the same.
- In 3D Mode, the scanline value of the display memory and the display panel can be different; get_scanline shall return the current scanline of the display panel.

Copyright © 2026 Texas Instruments Incorporated

Restrictions

None

TE control using the Automatic Method

For TE-reporting from the display module, the TE-reporting function is enabled and disabled by three DCS commands to the display module controller: set_tear_on, set_tear_scanline, and set_tear_off. See [MIPI-DCS] for details.

set_tear_on and set_tear_scanline are sent to the display module as DSI Data Type 0x15 (DCS Short Write, one parameter) and DSI Data Type 0x39 (DCS Long Write/write_LUT), respectively. The host processor ends the transmission with Bus Turn-Around asserted, giving bus possession to the display module.

Since the display module DSI Protocol layer does not interpret DCS commands, but only passes them through to the display controller, it responds with a normal Acknowledge and returns bus possession to the host processor. In this state, the display module cannot report TE events to the host processor since it does not have bus possession.

To enable TE-reporting, the host processor shall give bus possession to the display module without an accompanying DSI command transmission after TE reporting has been enabled. This is accomplished by the host processor protocol logic asserting (internal) Bus Turn-Around signal to its D-PHY functional block. The PHY layer will then initiate a Bus Turn-Around sequence in LP mode, which gives bus possession to the display module.

Since the timing of a TE event is unknown to the host processor, the host processor shall give bus possession to the display module and then wait for up to one video frame period for the TE response. During this time, the host processor cannot send new commands, or requests to the display module, because it does not have bus possession.

DCS Command - set_tear_on (35h)

Command

Direction H->D

Hex Code 35h

Parameter

Direction H->D

Hex Code X X X X X X X M XXh

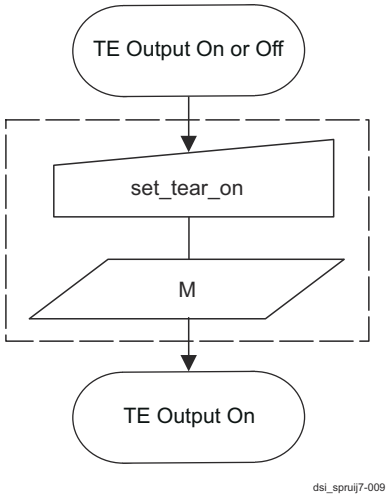


Figure 12-578. set_tear_on Command - 1

Copyright © 2026 Texas Instruments Incorporated

Description

This command turns on the display module Tearing Effect output signal on the TE signal line. The TE signal is not affected by changing set_address_mode bit B4. set_tear_on has one parameter that describes the Tearing Effect Output Line mode.

If M = 0 (Mode 0):

The Tearing Effect Output line consists of V-Blanking information only.

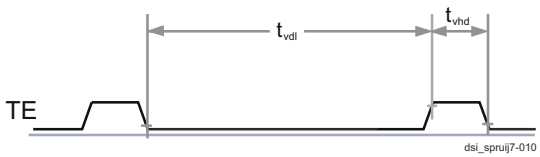


Figure 12-579. set_tear_on Command - 2

If M = 1 (Mode 1):

The Tearing Effect Output Line consists of both V-Blanking and H-Blanking information.

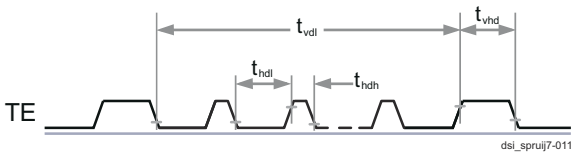


Figure 12-580. set_tear_on Command - 3

The Tearing Effect Output line shall be active low when the display module is in Sleep mode. See [MIPI- DPI] for definitions of tvdl, tvdh, thdl and thdh.

Restrictions

This command takes effect on the frame following the current frame. Therefore, if the Tearing Effect (TE) output is already ON, the TE output shall continue to operate as programmed by the previous set_tear_on, or set_tear_scanline, command until the end of the frame.

DCS Command - set_tear_scanline (44h)

Command

Direction H->D

Hex Code 0 1 0 0 0 1 0 0 44h

Parameter 1

Direction H->D

Hex Code N15 N14 N13 N12 N11 N10 N9 N8 XXh

Parameter 2

Direction H->D

Hex Code N7 N6 N5 N4 N3 N2 N1 N0 XXh

Copyright © 2026 Texas Instruments Incorporated

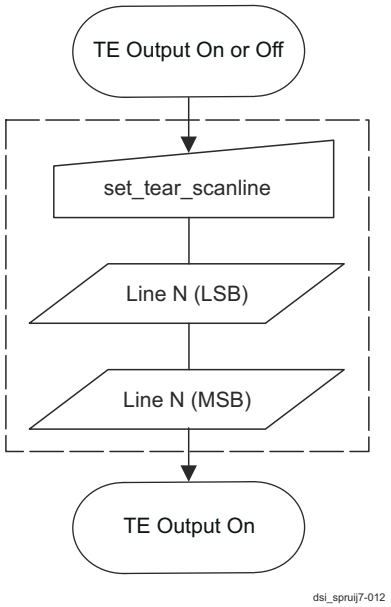


Figure 12-581. set_tear_scanline Command - 1

Description

This command turns on the display module Tearing Effect output signal on the TE signal line when the display module reaches line N. The TE signal is not affected by changing set_address_mode bit B4.

The Tearing Effect Line On has one parameter that describes the Tearing Effect Output Line mode.

After issuing a set_tear_scanline command to the display module, the Tearing Effect output signal, e.g. as in DBI- 2 systems, shall be a delayed version of V-Blanking information as illustrated in Figure 12-582.

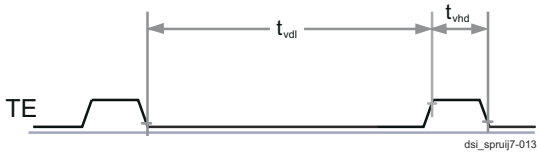


Figure 12-582. set_tear_scanline Command - 2

Note that set_tear_scanline with N = 0 is equivalent to set_tear_on with M = 0. The Tearing Effect Output line shall be active low when the display module is in Sleep mode. See [MIPI-DBI] for definitions of tvdl and tvdh and [MIPI-DSI] for definition of display module line numbers. In 2D mode, the scanline value of the display memory and the display panel is the same.

Restrictions

This command takes effect on the frame following the current frame. Therefore, if the Tear Effect (TE) output is already ON, the TE output shall continue to operate as programmed by the previous set_tear_on, or set_tear_scanline, command until the end of the frame.

DCS Command - set_tear_off (34h)

Command

Direction H->D

Hex Code 0 0 1 1 0 1 0 0 34h

Parameters None

Copyright © 2026 Texas Instruments Incorporated

Description

This command turns off the display module Tearing Effect output signal on the TE signal line.

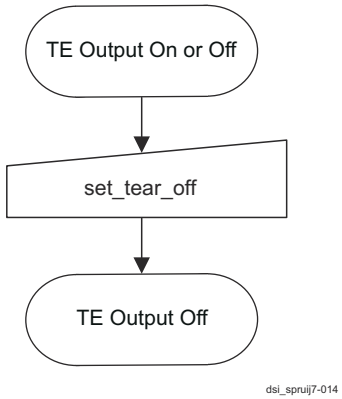


Figure 12-583. set_tear_off Command

Restrictions

This command has no effect when the Tearing Effect output is already off.

12.9.2.7.5.6 Return Path Operation

The DSI Return Path behaves like a target, by collecting data from the DPHY receive interface when the DPHY is configured to accept a transmission from the peripheral. The RP receives 6 signals from the DPHY (see Table 12-494).

The data (and trigger) reception is inactive if the DPHY PPI direction signal is at low level. When it is active, data is received using the RX ESC_CLK clock (the clock is transmitted with the data itself - see D-PHY protocol description from MIPI) and may stop if there is a pause in the data transmission but data is generated on clock falling edge. New data is available on rx_data_esc on a rising edge of the clock, if both rx_lpdt_esc and rx_valid_esc are asserted.

Table 12-494. Interface Between Return Path and DPHY

Signal Name	I/O	From/To	Description
rx_lpdt_esc	I	DPHY	Escape Low Power data receive mode This active high signal is asserted to indicate that the lane module is in low-power data receive mode. While in this mode, received data bytes are driven onto the rx_data_esc output when rx_valid_esc is active. The lane module remains in this mode with rx_lpdt_esc asserted until a stop state is detected on the lane interconnect.
rx_trigger_esc(3:0)	I	DPHY	Escape trigger received These signals indicate that an escape trigger command has been received. Only one of these signals will be asserted at any time and the signal will remain asserted until the lane module returns to stop state.
rx_data_esc(7:0)	I	DPHY	Escape receive data For Low Power data receptions, this eight-bit value is driven by the D-PHY and is valid on rising edges of rx_clk_esc with rx_valid_esc asserted. The bit connected to rx_data_esc[0] was received first.
rx_valid_esc	I	DPHY	Escape receive data valid This signal indicates that the lane module is driving data on rx_data_esc and expects the protocol layers to take the data at the current rising edge of rx_clk_esc. There is no "ready" signal to throttle the receive data.

Copyright © 2026 Texas Instruments Incorporated

Table 12-494. Interface Between Return Path and DPHY (continued)

Signal Name	I/O	From/To	Description
stop_state_dl1	I	DPHY	Lane is in stop state This signal indicates that the lane module is in STOP state. Note that this signal is asynchronous to any clock in the protocol interface.
direction	I	DPHY	Lane direction. When this signal is high, the lane module is in receive mode. When direction is low, the lane module is in transmit mode. (Mnemonic: 1 = Input, 0 = Output.)

The received data messages are identified as one of two possible types:

- Trigger
- Read packet

Trigger messages are passed almost directly to the register (reg_req = reg_trigger = 1 and reg_rd_data<3:0> equals to trigger value during one clock cycle). However, triggers are decoded to see if they are a TE trigger or an acknowledge with no error.

- The TE data is passed separately to the CSM using csm_te_received pin. It needs to be resynchronized with tx_byte_clk byte clock because of the CSM working clock domain.
- The acknowledge in only passed to register using reg_req and reg_ack pins. All other trigger values are passed undecoded to the registers without any processing. It is the responsibility of the application to decode and decide what to do with them.

Read Packet: When data is received, the system waits for the 4 bytes of the **HEADER**, and performs an ECC correction (if it is enabled) and then decodes the packets.

The following cases are considered when the system performs the ECC check:

- If the received command is not in the list of legal display opcodes, (i.e. errors in the packet header), it discards that packet and all further incoming bytes, even if they are legal packets (until the next BTA - i.e. direction change). It sends errors to the register bank (err_undecodable and uncorrectable_err). This may also result in an EOT error being reported as no further bytes have been decoded. But even if ECC correction shows an uncorrected error, decoding is attempted (if the errors are in the ECC byte or in the payload bytes, the opcode can be understood and data correctly recovered). In this case only err_uncorrectable error bit is sent to register and it falls in the situation of regular commands (acknowledge with error or read).
- If "an acknowledge with error", the two status bytes are sent to the registers (reg_req = 1 and reg_ack_err = 1 and reg_rd_data<15:0> contains the correct value).
- If the message type is a short read (either DCS or generic), data is passed to the registers (2 bytes without any specific decoding). Interface signals must be set accordingly and the two bytes are sent to registers (reg_req = reg_start = reg_end = reg_read = 1 with data and dscnotgen set accordingly). Size information is decoded in the first byte of the received packet depending on the data type information and then sent to register using reg_size port.
- If a long read, the size is sent to the register using reg_size. The system goes to **LONG state**. All data received are sent to register using reg_data and are used for checksum detection. If during receive, return packet fifo (in control block) becomes full (rd_data_fifo_full), the error flag err_oversize must be set. All remaining data is used for checksum calculation but are not transferred to registers. When the number of received bytes is equal to the packet size, the system will check the **CHECKSUM**. The two checksum bytes are never transferred to register. They are used to make checksum detection and indicate err_checkum status.
- If the transfer is shorter than the size specified in the header, the error err_wrong_length is issued along with any subsequent errors such as err_missing_eot (if needed). The case where it is longer drives to a tentative checksum decoding that generates an error and by a tentative to decode a new header (that is in fact a part of the previous packet) that drives to an error err_undecodable and the throwing of the rest of the received bytes (with corresponding errors such as err_missing_eot).

Copyright © 2026 Texas Instruments Incorporated

- If it is an EoT packet, it is not passed to the control block. The following data is ignored (if any). If the system is waiting for **EoT**, this is the only action. If system is not waiting for EoT packet, the err_eot_with_err is sent.
- If after an "acknowledge with error", the system waits for EoT packet and counts 4 bytes but is not able to decode them, it emits an err_eot_with_err and err_undecodable.
- If direction is removed at a "wrong" time (i.e before the fourth byte of a small packet, the system stops to works but emit errors to the register: err_receive with eventually err_missing_eot.

The DSI is designed to support multi-peripheral integration, the virtual channel of the received packet is also stored in register using reg_vc port.

The application FW layer is responsible of all read actions, it is assumed that once a read is requested, application reads return packets before requesting other read actions. Based on this assumption, the receive FIFO implemented in the register block is cleared each time a read action is started. All data not read before the new read command are lost.

A re-initialization of the return path is also performed using register access on reg_init_rp.

All errors related to short or long received data packets should be captured in the register when the direction changes (to capture all errors at the same time and not generate several errors and possibly several interrupts on the same reverse transaction). As soon as an error is detected on a packet, the error must be recorded and maintained up to the end of the packet transfer.

The maximum packet size supported by the DSI return path is 16 bytes. If a longer packet is received, only the first 16 bytes are available (by register way) for application.

12.9.2.7.5.7 EoT Packet Management

The DSI protocol states that, in the sending path, only the HS transmission requires use of EoT packet. For transfer from display to host, it is not recommended to use EoT but it is possible. The host needs to be able to detect the EoT packet and then behaves accordingly.

In case the display makes use of the EoT packet, the bit disp_eot_gen must be set to one. In that case, if after the ECC correction, the packet header is 0x08, we've detected an EoT packet and all bytes that may arrive between this packet and the direction change must be ignored.

If additional data is received between the EOT packet and the next BTA (end of read process), or if an unwanted EOT packet is detected, the unwanted bytes must be rejected and err_receive is asserted.

If the EoT packet is not detected although it should be present, the error err_missing_eot is set. If after an "acknowledge with error", 4 bytes are received before BTA but are not identified as EoT (because of errors), it is assumed these 4 bytes are the EoT. In that case, the error err_eot_with_err has to be set with err_undecodable.

12.9.2.7.5.8 ECC Correction

The DSITX controller uses the packet header ECC information to detect and correct one error, and can detect two or more errors without correction.

12.9.2.7.5.9 LP Transmission and BTA

The DSI host may send a command sequence and then request an acknowledge message from the panel. The host will send the command and release the bus by performing a BTA sequence. The panel will then respond with an acknowledge message (or error response) followed by a BTA to return the bus to the host. This sequence is illustrated in Overview of a Display Subsystem.

Overview of a Display Subsystem.

Figure 12-584 shows LP transmission timing diagram.

Copyright © 2026 Texas Instruments Incorporated

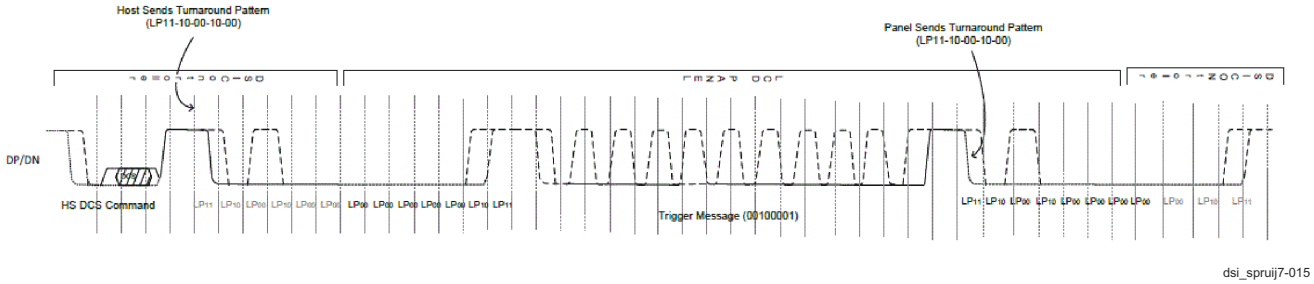


Figure 12-584. LP Transmission Timing Diagram for Host Read and Trigger ACK

The Host will not be able to transmit any other packets once the BTA is sent to the panel and will expect either an ACK trigger or response LPDT message with a payload. The system must ensure there is sufficient time to send any command and issue the BTA, receive the response (Trigger, Short or long packet) and accept the BTA from the panel, if this request is made in the line blanking intervals.

Figure 12-585 presents panel read response sequences.

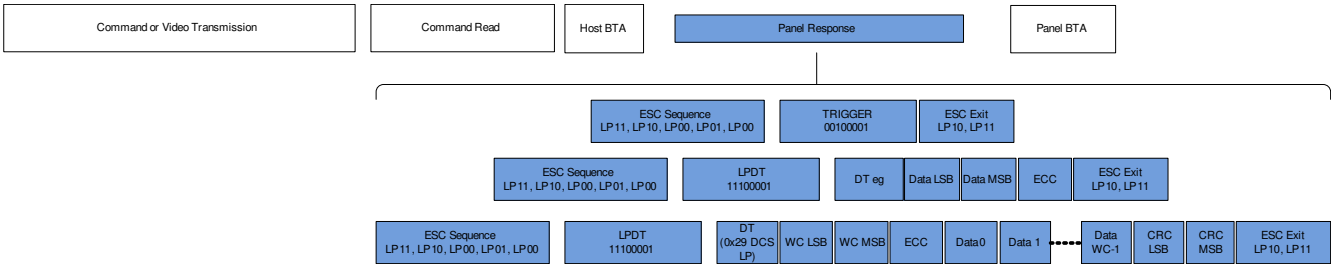


Figure 12-585. Panel Read Response Sequences

The panel response time will be based on the tx_esc_clk used inside the panel and the expect number of bytes for the payload. The expected reposes are illustrated in the blocks in the MIPI DSI standard.

12.9.2.7.6 Low-power Management

There are several low-power scenarios which consist of switching off various parts of the design:

1. Switch D-PHY cells in ULP mode when there are is no data to be sent (bits clklane_ulpm_en, dat1_ulpm_en, dat2_ulpm_en, dat3_ulpm_en and dat4_ulpm_en in register mctl_main_en set to one with ULP mode enabled in register mctl_main_phy_ctl). Please, note that the time to leave ULP state is 1 ms and needs to be set in clock cycles in the register mctl_ulpout_time.
2. Switch the DPHY byte/bit clock PLL off (refer to DPHY documentation for control sequence). In this case, the D-PHY cell can only transmit data in LPDT thus maximum bandwidth is 10 Mbits/s).
3. More drastic methods are to shut-down some parts of the design. The D-PHY and the DSI themselves can be shut down.

When D-PHY and DSI are both shut-down and the display is powered down, the restart is a normal start procedure as described in Section 12.9.2.7.3, *Start-up procedure*.

When only the D-PHY is stopped, it is placed in either the stop or ULP state. Restarting the D-PHY is similar to the regular start-up procedure, however, after the lanes are enabled by the DSI Controller (clklane_en, dat1_en, dat2_en, dat3_en and dat4_en in register mctl_main_en are set), the D-PHY generates a rising edge on the stopstate signal to indicate to the DSI Controller that the enable request has been effective within the D-PHY. After that the start-up procedure may continue up to the point that the DSI link is ready to begin transmission.

Note that when setting the D-PHY into the ULP state, the application needs to wait enough time between the time it sets the ULP request in register and the time the D-PHY is disabled – this ensures that the ULP request is detected on the D-PHY interface.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.7 Video Mode Settings

All values contained in the Control Block (CB) registers must be set with meaningful values and are expressed with a range of different units: clock cycles, bytes, number of lines.

12.9.2.7.7.1 Video Stream Presentation

In Figure 12-586, it shows the list of packets that can be part of a video stream and the associated register fields. In the four following figures (Figure 12-587, Figure 12-588, Figure 12-589, Figure 12-590), different video streams that can be generated are presented.

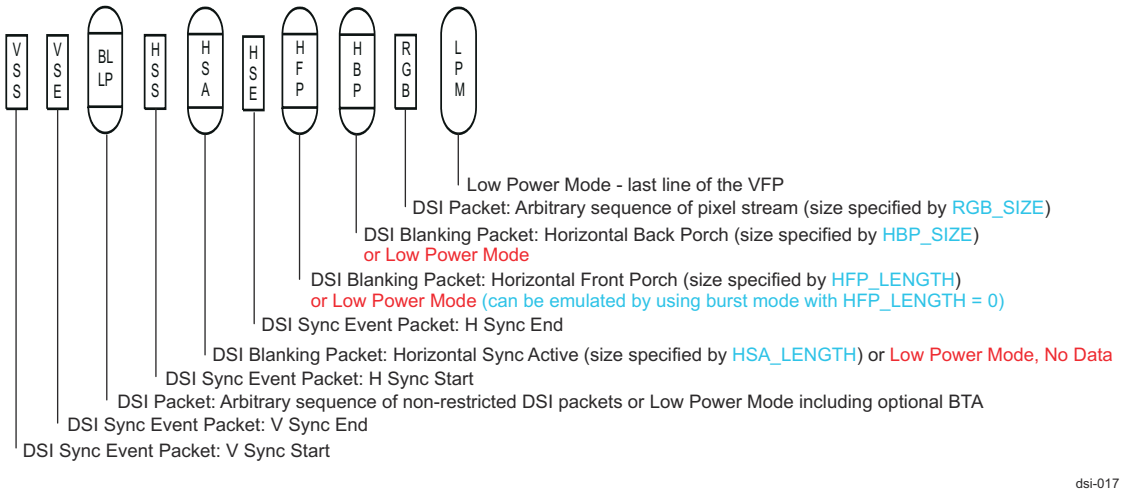


Figure 12-586. DSI Packet Terminology

Red text - behaviour not possible.

Blue text - register fields used to specify the packet size.

The BLLP (Blanking – Low Power) periods allow the DPHY link to perform the following sequences:

- Host Transmit blanking packets in HS.
- Host Transmit command packets in HS or LP (Escape mode).
- Host Transmit packets using interleaving to a different virtual Channel in HS or LP (Escape mode).
- Host performs BTA and waits for response packets from the panel using Escape Mode followed by a panel BTA.

Copyright © 2026 Texas Instruments Incorporated

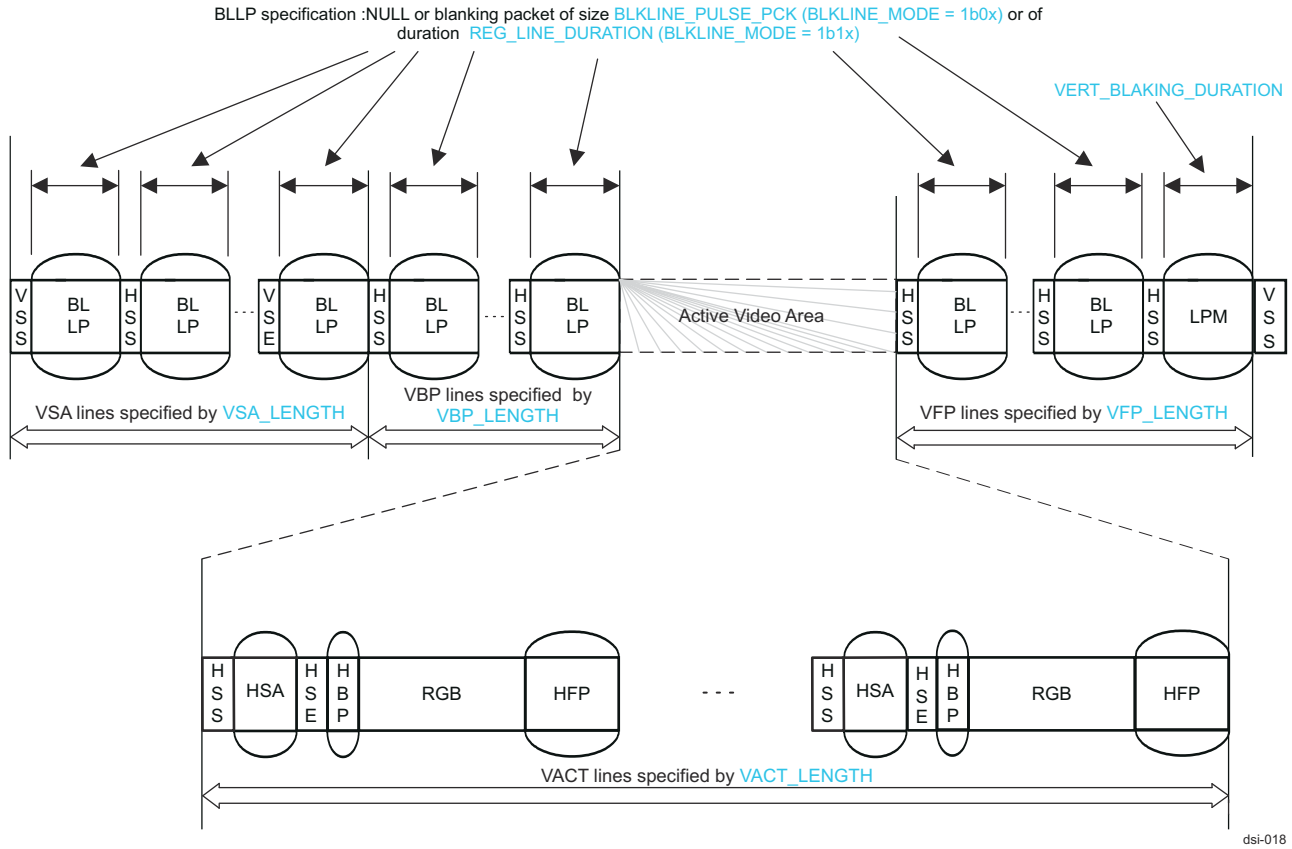


Figure 12-587. Video Pulse Mode (Non-Burst) Timing Diagram

Note

This sequence is no longer specified in the MIPI spec Version 1.02.00 28-Jun-2010 but supported by the DSI-controller.

Copyright © 2026 Texas Instruments Incorporated

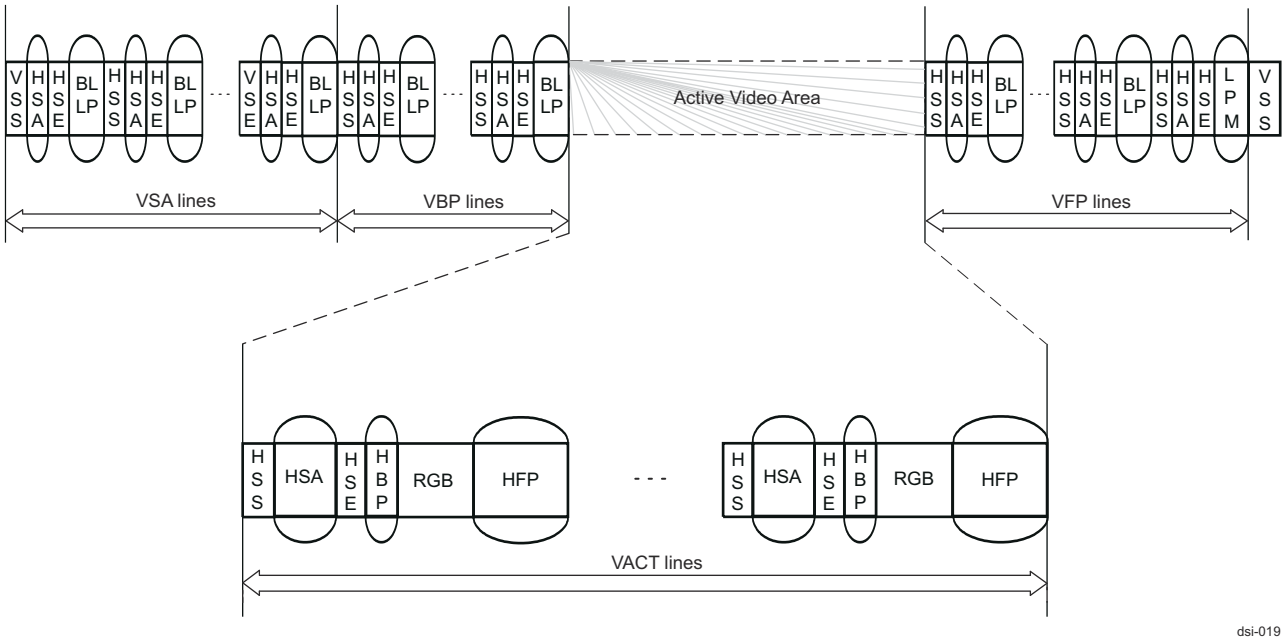


Figure 12-588. Video Pulse Mode (Non-Burst)

Note

In the MIPI spec 1.02.00 June 2010, the line (VSE/HSA/HSE/BLLP) has been moved from VSA lines part to the VBP lines part. The sequence is the same but in the design the VSA lines has one more line and the VBP lines has one line less. This must be considered during the VSA and VBP register programming if the DSI MIPI spec 1.02.00 is used as reference.

Copyright © 2026 Texas Instruments Incorporated

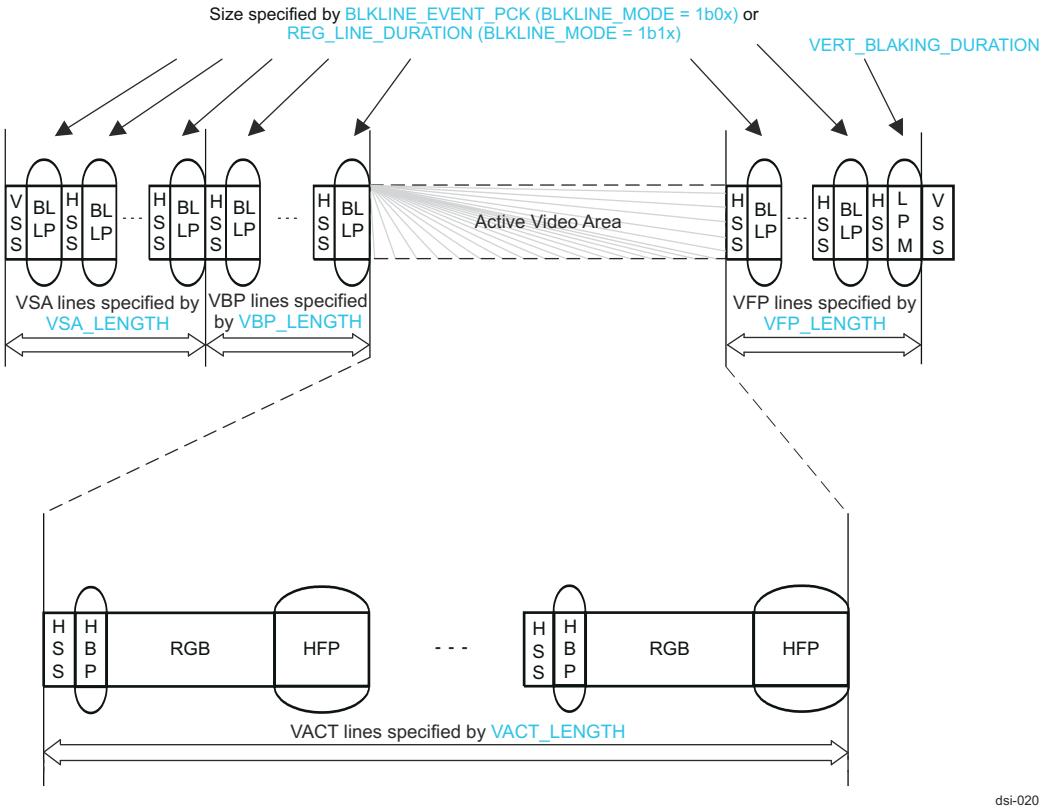


Figure 12-589. Video Event Mode (Non-Burst)

burst_mode = sync_pulse_active = sync_pulse_horizontal = 0 - all spec versions.

Copyright © 2026 Texas Instruments Incorporated

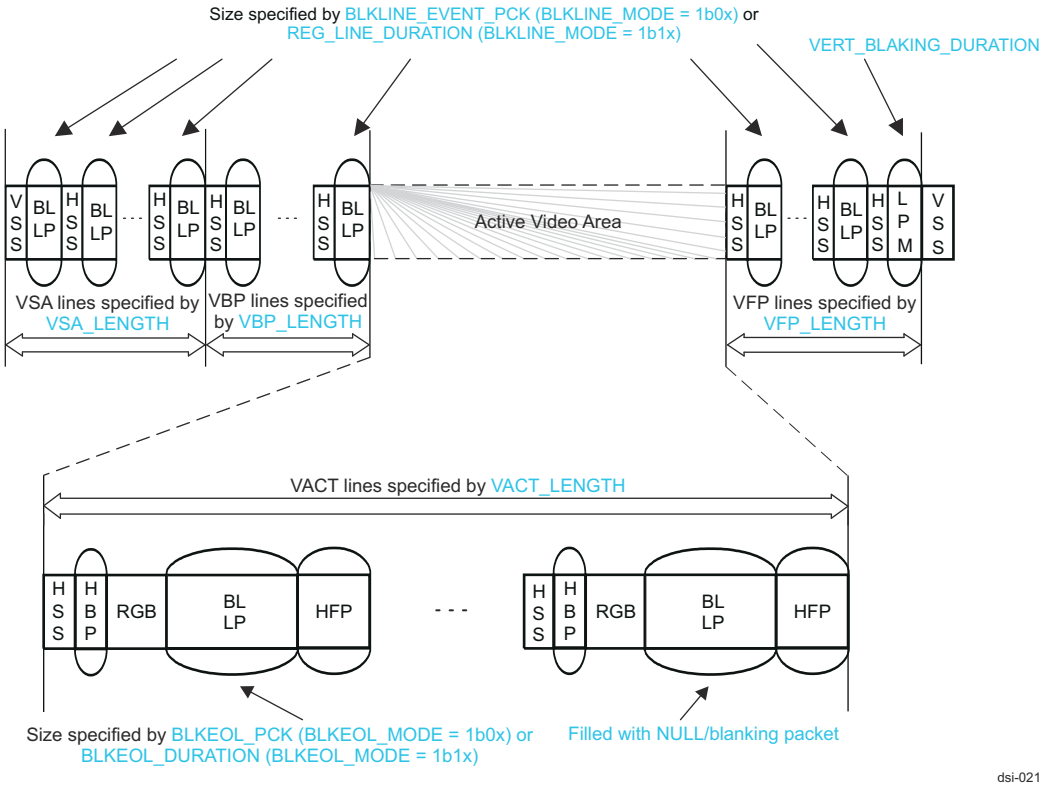


Figure 12-590. Video Sync Event in Burst Mode

burst_mode = 1 and sync_pulse_active = sync_pulse_horizontal = 0 - all spec versions.

Burst Mode operation requires the tx_byte_clk to be set to double the bit rate compared to the non-burst case. The calculations for the video parameters must then be increased to match the higher rate. The DPI FIFO must also be large enough to store at least half of the active line bytes before the output transmission begins otherwise it will underrun.

12.9.2.7.7.2 Video Stream Settings (VSG)

The following is all the registers that need to be set before using the video in SDI or DPI interface operation.

- vid_vsize1, 2
- vid_hsize1, 2
- vid_blksize1, 2
- vid_pck_time
- vid_dphy_time
- vid_error_color1, 2
- vid_vca_setting1, 2
- vid_main_ctl – programmed last in sequence of vid registers to apply to DSI registers

This section addresses the dependencies between all video registers, and details how the TVG and VCA registers must be set per VSG registers.

vid_main_ctl

- header information must correspond with the vid_pixel_mode used to control the generation of the recovery streams. The header information is used to be enable management of future cases where the DSI link could be used to pass streams with formats that differ feom those currently supported directly.
- sync_pulse_horizontal can only be asserted if sync_pulse_active = 1.

Copyright © 2026 Texas Instruments Incorporated

When sync_pulse_active is set to one, the HSYNC pulses are emulated only during the active part of the frame. When both sync_pulse_horizontal and sync_pulse_active are set, the HSYNC pulses are present on all lines of the frame.

vid_vsize - the variables are expressed in line numbers in this register.

- vsa_length should be at least one (in order VSG can insert at least the vertical synchronization start VSS line) and greater or equal to 2 when pulse mode (sync_pulse_active = 1).
- vbp_length 0 to 63.
- vfp_length must be greater than 0.
- vact_length must be greater than 0.
- vid_hsize1: - the numbers are expressed in number of bytes (the fields are specifying the payload).

vid_hsize1 - the numbers are expressed in number of bytes (the fields are specifying the payload of the corresponding packet).

- hsa_length need to be at least set to one if pulse mode is enabled. The expected normal operating range is 1 to 255, large values for the HAS length will impact on the size of the DPI FIFO required.
- hbp_length can have any value. However, when set to 0, a HBP packet is generated with 0 payload.

vid_hsize2 - again the fields are specifying the payload of the corresponding packet (in byte).

- hfp_length can have any value. However, when set to 0, no HFP blanking packet is generated.
- rgb_size must match the number of pixel per line of the display multiplied by the number of byte per pixel.

vid_blksize1 - still specifying size of the payload in byte.

- blkline_event_pck defines the payload length of the blanking or NULL packet to be inserted in blanking line (with sync event). This value is used when reg_blkline_mode = 0b01 or 0b00 and when sync_pulse_active = 0. It is also needed by the VSG when reg_blkline_mode = 0b1x. Its value depends on the hbp, rgb, blkeol, and hfp packet length.
- blkline_event_pck defines the payload length of the blanking or NULL packet to be inserted in blanking line (with sync event). This value is used when reg_blkline_mode = 0b01 or 0b00 and when sync_pulse_active = 0. It is also needed by the VSG when reg_blkline_mode = 0b1x. Its value depends on the hbp, rgb, blkeol, and hfp packet length.

  -- 32767 = 0x7FFF = max value on 15 bits and -100 to take

  margin blkeol_pck < 32767 - hfp_length - hbp_length - hsa_length - rgb_size - 100;

The blkeol_pck (and blkeol_duration) must be adapted to the reg_wakeup_time in case VCA is authorized to go back in LP.

vid_blksize2 - still specifying size of the payload in byte.

- blkline_pulse_pck has almost the same definition than blk_event_pck but it concerns blanking lines with sync pulse.

vid_pck_time - specifies duration in clock cycles.

- blkeol_duration: blkeol_duration = div_round_up (1) ((blkeol_pck + 6), lane_nb);
- vert_blanking_duration is no longer used and can always be set to 0. It remains present in the design for legacy reason.

Note 1: The function div_round_up is a function that performs a division then round the result to the first entire number superior or equal to the division result.

vid_dphy_time: specifies duration in clock cycles. These are values that have external dependencies and thus need to be calculated first (and rest of VSG programming is calculated from these values).

reg_line_duration: depends on display type and size and of its programming (in some display, the blanking sizes can be adapted to provide a given number of frame per second with one of the D-PHY possible clock frequency (as PLL can only generate a given number of discrete frequencies).

if (pulse mode) {

Copyright © 2026 Texas Instruments Incorporated

line_length = (blkline_pulse_pck + 6);

}

else (event mode) {

line_length = (blkline_event_pck + 6);

if (burst_mode and burst_lp and lane_nb == 2 and

hsa_length[0:0] == 1 and hfp_length[0:0] == 1)

line_length = line_length - 1;

}

result = div_round_up(line_length, lane_nb);

Note 1: The function div_round_up is a function that performs a division then round the result to the first entire number superior or equal to the division result.

Additional Notes about the line duration:

- If line length (in bytes) is a multiple of the number of lanes enabled, DSI will never face any timing jitter except due to the resynchronization in SP/DCB.
- If line length (in bytes) is NOT a multiple of the number of lanes enabled For Non LP operation, there should not have any problem since the line is once longer, once shorter than the "normal" value. For example: if line length is 491 bytes, it will be 245 then 246 in clock cycles with 2 lanes for an average value of 245.5.

  For LP enabled operation, the controller cannot allow underflow of the line and frame timing. With the previous example, the LP line length in clock cycles will be 246 since we cannot start the HS transmission in the middle of a clock cycle. And there is no mechanism to have one line in 245 and the next one in 246 cycles. For LP see section 0.

reg_wakeup_time must be shorter than line duration and depends on the D-PHY cell plus some pipelines delays inserted by the DSI link. This value strongly depends on the DPHY PLL programming and configuration, and is a mix of both internal and external analog and digital timing factors, therefore it is very difficult to provide an exact formula. The recommended approach to selecting a suitable value for this parameter is to characterize behaviour in at the system level environment using simulation, emulation (e.g. Palladium) or validation (e.g. FPGA).

The DPHY needs 100 ns + time for THXS-EXIT to go to Low Power mode.

The timing for the clock lane TXRequestHS going active to the TXReadyHS will be determined by the DPHY DDR bit rate clock frequency. The timing will be the total number of DDR bit clk cycles for the TXRequestHS to be detected and the clock lane to transition to High Speed, so TLPX + TCLK-PREPARE + TCLK_ZERO. Additional time is required between the clock lane driving HS clocks and the Data Lanes being activate, TCLK-PRE. All of these parameters can be calculated from the DPHY datasheet.

Figure 12-591 shows timing diagram for clock lane activation/deactivation.

Copyright © 2026 Texas Instruments Incorporated

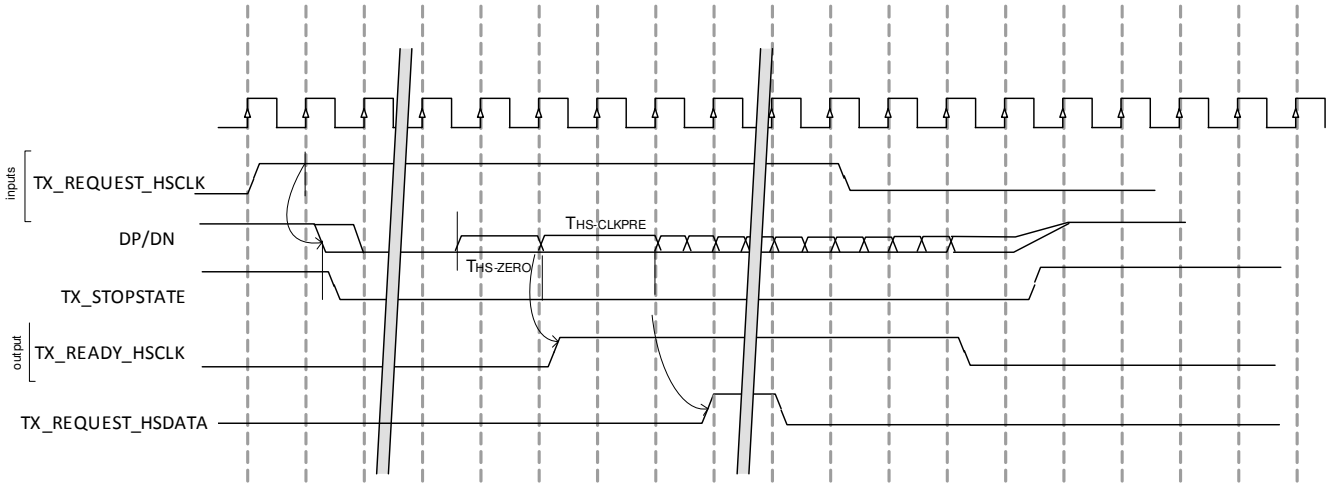


Figure 12-591. Timing Diagram for Clock Lane Activation/Deactivation

The data lane request follows a similar timing sequence from the TX_Request_HS to TX_Ready_HS, TLPX + THS-PREPARE + THS_ZERO. The data lane requests will be activated once the clock lane is ready, i.e TX_Ready_HSCLK is high.

Figure 12-592 shows timing diagram for data lane activation/deactivation.

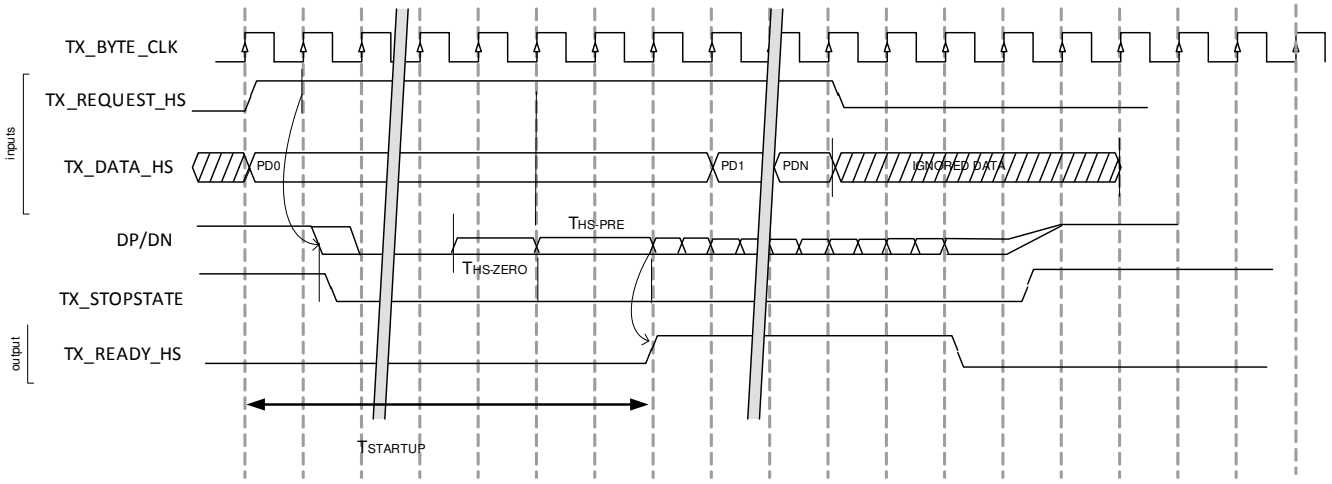


Figure 12-592. Timing Diagram for Data Lane Activation/Deactivation

Table 12-495. Timing Parameters

Parameter	Description	Min	Max	Example 650 Mbps (UI = 1.538 ns)
TLPX	Transmitted length of any Low- Power state period.	50 ns		50 ns
TCLK-PREPARE	Time that the transmitter drives the Clock Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	38 ns	95 ns	38-95 ns
TCLK-PREPARE + TCLK-ZERO	TCLK-PREPARE + time that the transmitter drives the HS-0 state prior to starting the Clock.	300 ns		300 ns
Wakeup Time CL				388-445 ns
TCLK-PRE	Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode.	8 UI		12.3 ns

Copyright © 2026 Texas Instruments Incorporated

Table 12-495. Timing Parameters (continued)

Parameter	Description	Min	Max	Example 650 Mbps (UI = 1.538 ns)
THS-PREPARE	Time that the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	40 ns + 4 × UI	85 ns + 6 × UI	~ 46-95 ns
THS-PREPARE + THS-ZERO	THS-PREPARE + time that the transmitter drives the HS-0 state prior to transmitting the Sync sequence.	145 ns + 10 × UI		160 ns
Wakeup Time DL				~ 270 ns

So the wakeup time values expressed as tx_byte_clk (12.3ns) cycles, CL = 36 and CLKPRE + DL = 22.

Figure 12-593 shows wakeup time timing diagram.

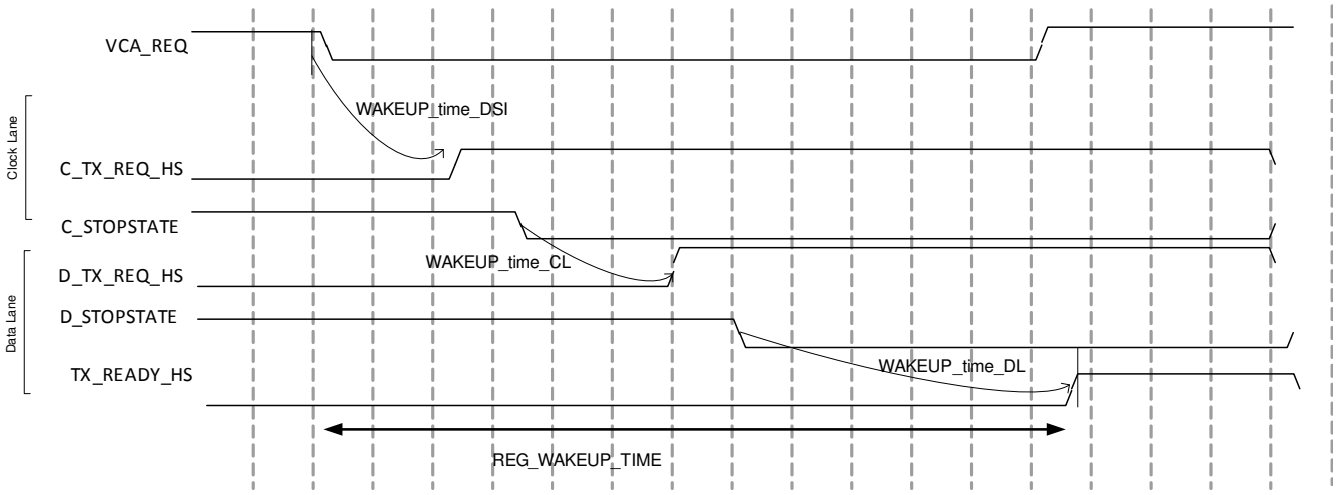


Figure 12-593. Wakeup Time Timing Diagram

Here is the example of the calculation done for the system running at **650 MHz**.

if (clk_continuous == TRUE) {

wakeup_time_cl = 0x1;

} else {

wakeup_time_cl = 0x24;

};

wakeup_time_dsi = 0xA; -- from request on VSG to request HS on DL1 (+ 1 for VSG internal cycle)

wakeup_time_dl = 0x14; -- from stop state falling edge to tx_ready rising edge

reg_wakeup_time = wakeup_time_dsi + wakeup_time_cl + wakeup_time_dl + (hs_host_eot × 4 / lane_nb)

Note: It is not necessary to program all the register values every time because some of them are used only in some modes. For instance, blkline_event_pck and blkline_pulse_pck are used depending on the way SYNC is generated, and then in pulse mode, there is no need to program blkline_event_pck and vice-versa.

12.9.2.7.7.3 VCA Configuration

The VCA setting done in two registers: vid_vca_setting_1 and vid_vca_setting_2. Their programming must respect the following rules:

- vid_vca_setting_1:

Copyright © 2026 Texas Instruments Incorporated

  - max_burst_limit specifies the size of the bigger packet that generates the following sequence RGB + packet + NULL packet + BLK packet; it must be linked with blkeol_pck (same size minus the smaller NULL packet). It is equal to blkeol_pck - 6.
    - burst_lp can be set to 1'b1 if blkeol_pck > 2 × reg_wakeup_time × lane_number (reg_wakeup_time is only specifying the time to go from LP to HS thus need to consider the time to go from HS to LP too).
- vid_vca_setting_2:
  - exact_burst_limit: as it specifies (in byte) the payload of the packet that generates a sequence RGB + packet + HFP, it must have the same value as of blkeol_pck when the DSI link is in burst mode.
  - max_line_limit specifies the maximum size of a packet that generates HSS + (HSA + HSE) + packet + NULL packet. It then depends on line_duration, hsa_length regardless if the system is using pulse or event synchronizing. Its size is linked to vert_blanking_duration.

if (sync_pulse_active == 0) {

-- size of the payload of the inserted packet followed by a NULL pkt

max_line_limit = (blkline_event_pck-6);

};

if (sync_pulse_active == 1) {

-- size of the payload of the inserted packet followed by a NULL pkt

max_line_limit = gen_blkline_pulse_pck-6;

Another constraint to respect when using VCA is to ensure that only supported operations are sent from command side when video is active. For instance, no TE should be attempted, because it could break the video stream (TE completion time can be long). Command messages can be inserted into the LP states provided there is sufficient space for the message bytes and any associate bytes for the header and CRC, if a long packet type.

12.9.2.7.7.4 TVG Configuration

The TVG has a single register: tvg_img_size that should complies with the following rules against VSG settings:

tvg_line_size: must be equal to rgb_size set for VSG

tvg_nbline: must be equal to vact_length set for VSG

The TVG generates a data stream in the same format than the one specified at the output of the VRS. The content of that flow is specified by the following registers:

- Image size - TVG_LINE_SIZE[14:0] and TVG_NBLINE[12:0] registers: Number of bytes per line and number of lines per frame.
- Image kind - TVG_MODE[1:0] register: Single color, vertical stripes or horizontal stripes.
- Stripe width - TVG_STRIPE_SIZE[2:0] register: Significant when a stripe mode is enabled. Possible values are 1, 2, 4, 8, 16, 32, 64 and 128.
- Pixel kind VID_PIXEL_MODE[3:0] register: 16 bits RGB, 18 bits RGB, 18 bits loosely RGB, 24 bits RGB, 30 bits RGB or 36 bits RGB.
- Color one - COL1_GREEN[11:0], COL1_RED[11:0], COL1_BLUE[11:0] registers: Color used in single color mode or first color when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant ones are the ones considered.
- Color two - COL2_GREEN[11:0], COL2_RED[11:0], COL2_BLUE[11:0] registers: Color two when in stripe mode. 12 bits per component R/G/B. For formats using fewer bits than 12, the least significant are the one considered.
- Start pulse and stop handshake (+ stop mode) (see TVG_CTL register).

The Test Video Generator can be programmed to generate a set of test colour patterns based on the display panel that will be used. The panel parameters for horizontal and vertical resolution along with the frame rate and pixel colour depth need to be set based on the datasheet information for the panel.

Figure 12-594 presents TVG MODE patterns.

Copyright © 2026 Texas Instruments Incorporated

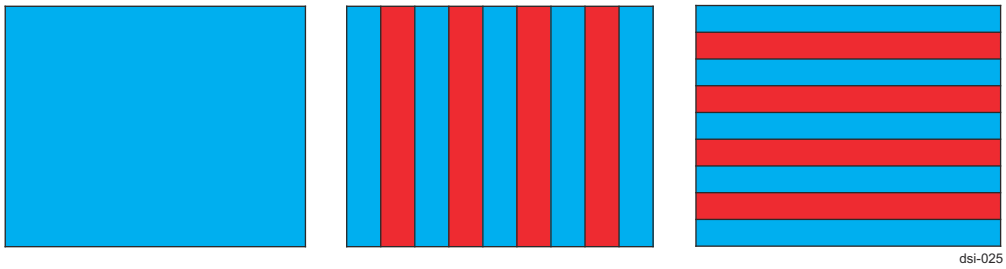


Figure 12-594. TVG MODE Patterns

An example of the sequence is as follows:

- Select TVG for video stream generation instead of DPI/SDI 1 interface.
  - IF1_EN bit to 0 (SDI stall signal at 1) in MCTL_MAIN_EN.
- Select video mode for interface 1 in MCTL_MAIN_DATA_CTL.
  - TVG_SEL bit to 1 in MCTL_MAIN_DATA_CTL register to select stream from TVG.
- Select generated frame format in TVG_CTL register.
  - Image format (single color, H stripes, V stripes) in TVG_MODE field.
  - Image color selection in TVG_COLOR1, TVG_COLOR1_BIS, TVG_COLOR2, VG_COLOR2_BIS.
  - Stripes size in TVG_STRIPE_SIZE field.
  - Image size in TVG_IMG_SIZE register.

  Note: The TVG settings on active area for the number of lines per frame and number of bytes per line must match VSG settings on active area. Any mismatch will create an error that is detected in the VSG and forces recovery mode in TVG, stopping test frame generation.
- Select TVG stop mode with TVG_STOPMODE field of TVG_CTL.
  - TVG video stream generation start/stop controlled by the TVG_RUN bit in TVG_CTL register.
  - Polling TVG run status from the TVG_RUNNING bit in TVG_STS register is useful when setting TVG_RUN to 0. TVG video stream generation does not stop the instantaneously (depends on TVG stop mode), so this should be checked to confirm the TVG has stopped.

12.9.2.7.8 DPI To DSI Programming

The DPI interface will normally be driven from a graphics driver that will be configured to match a frame geometry and refresh rate for a panel display. The DPI driver will use the expected refresh rate and geometry for the active and blanking parts of a frame to determine the pixel rate clock. The DSITX controller will then translate this to match the incoming byte information to send across the DSI DPHY link.

A system will always expect the bandwidth of the incoming pixels × BPP to match the transmission of bytes from the DSI DPHY based on the number of active lanes. The relationship of DPI clock to TX byte clock must ideally hold with the following formula:

$F_{pixel_clk} = F_{tx_byte_clk} \times active_lanes \times 8 \div bits_per_pixel;$

The DSI will supply packets to the DPHY interface using the internal sequencing and counters clocked from the tx_byte clock. The normal system operation will mean that the DSI must always have the correct number of tx_byte clock cycles to match the number of pixel clock cycles. This means that all the events used for the packet generation will be directly impacted by the delay that exists in the DPHY High Speed request to ready.

The DSITX controller relies upon the DPHY PLL being programmed and locked before starting the DPI video. The DSITX controller can then be configured and enabled (clock lane and data lanes) before starting the video transmission.

The DSITX controller will begin data transmission with a High Speed request after it receives the falling edge of the VSYNC on the DPI interface. The controller will then begin to build the bytes for the VSYNC packets (VSS, HSA, Blanking packets) during the delay period before the ready is returned from the DPHY. This delay will add an interval effect on all the packets that follow.

Copyright © 2026 Texas Instruments Incorporated

The DPI FIFO size must be sufficient to buffer all the incoming pixel data during the active line stage while the DSITX controller continues to send the previous lines packet information due to this interval effect.

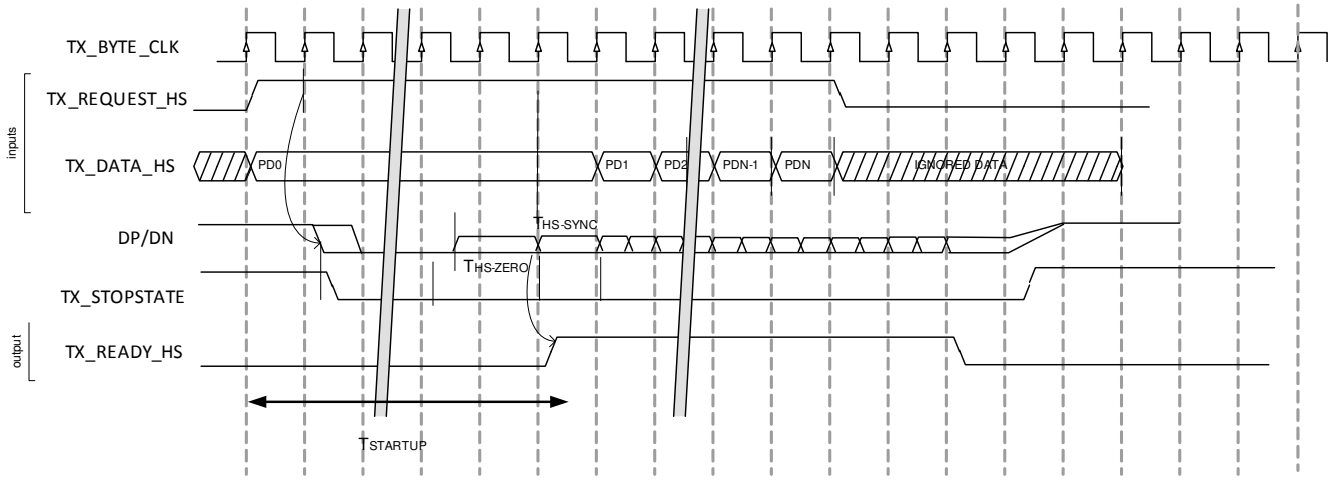


Figure 12-595. Timing Diagram

The DPI and DSI system may not be able to guarantee the exact frequency on the pixel clock input or tx_byte clock from the DPHY. The DSI configuration can be adjusted to support a small deviation in frequencies using the size of the HFP packet so that it changes the alignment of packets on the active lanes to absorb the different pixel clock cycles during the line. More details are given in section0.

12.9.2.7.8.1 DSI and DPHY Operation

The DPI FIFO is used to buffer all the active pixel data from the DPI interface during the active line. The depth of the FIFO must be selected to allow the pixel to be stored after the DPHY request to ready delay is introduced at the start of every frame. The FIFO will fill to a depth based on the tx_byte clock cycles used from the point where VSYNC is identified to the response from the DPHY link after it has exited Low Power and entered the zeroes state on the active data lanes.

Figure 12-596 presents DPI interface to DSI DPHY timing diagram.

Copyright © 2026 Texas Instruments Incorporated

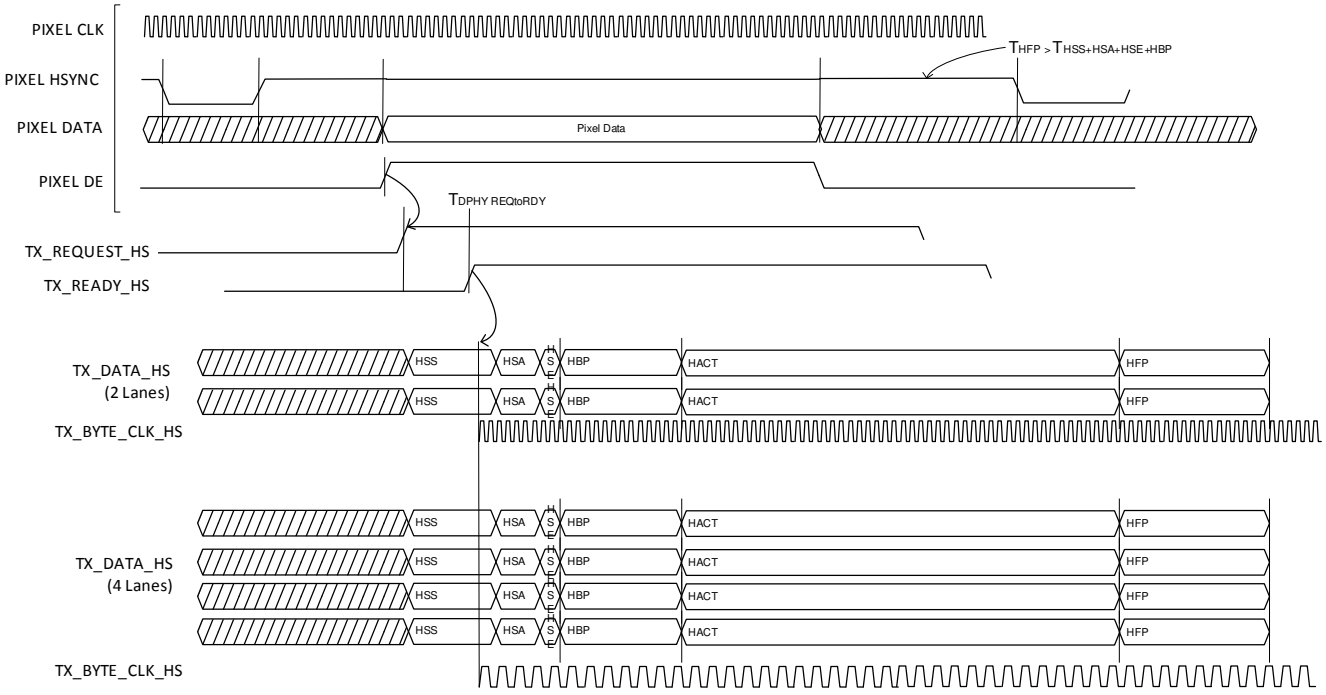


Figure 12-596. DPI Interface to DSI DPHY Timing Diagram

The timing diagram above illustrates how the packets are generated for a DPHY with two and with four lanes activated. The diagram shows that the tx_byte clock changes based on the number of lanes, and that the configuration register values for the number of bytes used for HSA, HBP and HFP will remain the same.

The register values calculated for each of the line synchronization stages needs to take the number of bytes used to form the packet into account so that the timing for each stage aligns to the timing of the DPI input and most importantly that the total number of bytes exactly matches the number of byte clock cycles. The values used for the Horizontal Front porch must be larger than the combined horizontal sync and back porch to allow the DPI FIFO to empty.

The basic rule is that the values used for the DSI will be based on the DPI × BPP/8.

The DPI graphics driver must be configured with horizontal values that can be directly translated to DSI packet payloads.

- The minimum values for the DPI HSA must convert to be greater than:
  - Non-Burst Sync Pulses – 15 bytes
  - Non-Burst Sync Events – 11 bytes
- The minimum values for the DPI HBP must convert to be greater than:
  - Non-Burst Sync Pulses – 7 bytes
  - Non-Burst Sync Events – 7 bytes

These values will then form the number of tx_byte_clk cycles used to send the bytes over the number of active lanes. So, for an RGB888, DSI configuration the minimum values will be:

- HSA = roundup(15/3) = 5
- HBP = roundup(11/3) = 4

The diagram below, D-PHY Timings Control (see Figure 12-597), illustrates the byte mapping for a four lane DPHY link. In this example, the calculation for the horizontal sync stage aligns exactly across the four byte lanes.

- HSS – Four Bytes
- HSA – Blank Packet with Header (4) + HSA_Count + CRC(2)
- HSE – Four Bytes

Copyright © 2026 Texas Instruments Incorporated

The next stage is the horizontal back porch:

- HBP – Blank Packet with Header (4) + HBP_Count + CRC(2)

The value used in the HBP count can be used to account for the Header bytes used in the active data packet in next stage, although it is best to add this to the HFP stage calculation and allow time to empty the FIFO.

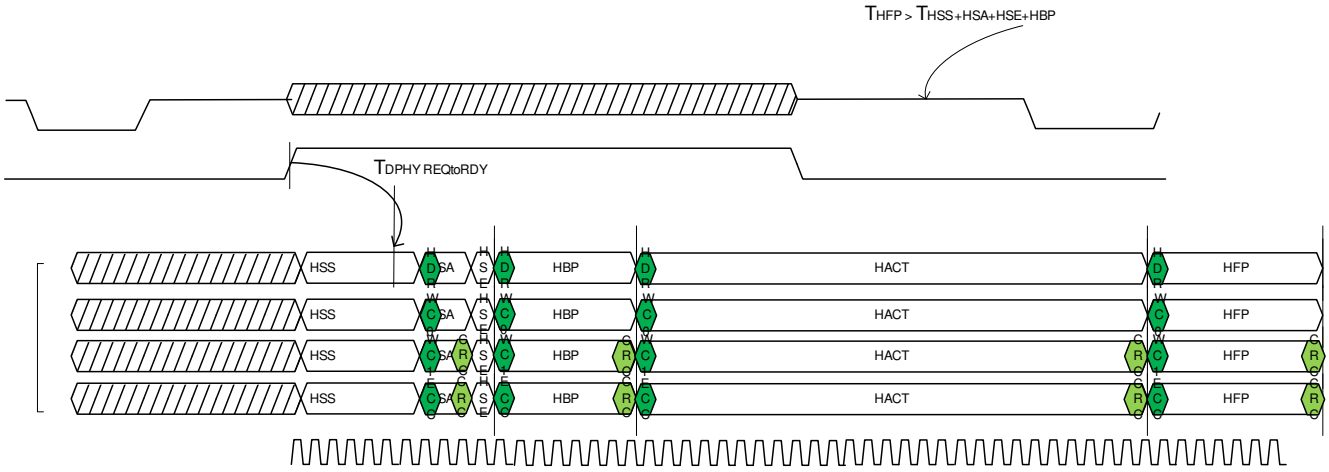


Figure 12-597. Horizontal Line Packet Timing Diagram

The Active part of the line will be a fixed size based on the line size and BPP, and include the six bytes for the packet header and CRC. The final stage is the horizontal front porch.

- HFP – Blank Packet with Header (4) + HFP_Count + CRC(2)

The HFP value should be adjusted to ensure that the number of tx_byte clock cycles for the horizontal line exactly matches the pixel clock cycles for the line times the BPP.

The timing diagram below, Control Block, illustrates the byte alignment changing when the HBP calculated is two bytes shorter than the previous case. This leads to the active and front porch packet headers beginning two bytes earlier, however the HFP count is adjusted to keep the end of the line on exactly the same tx_byte_clk cycle count.

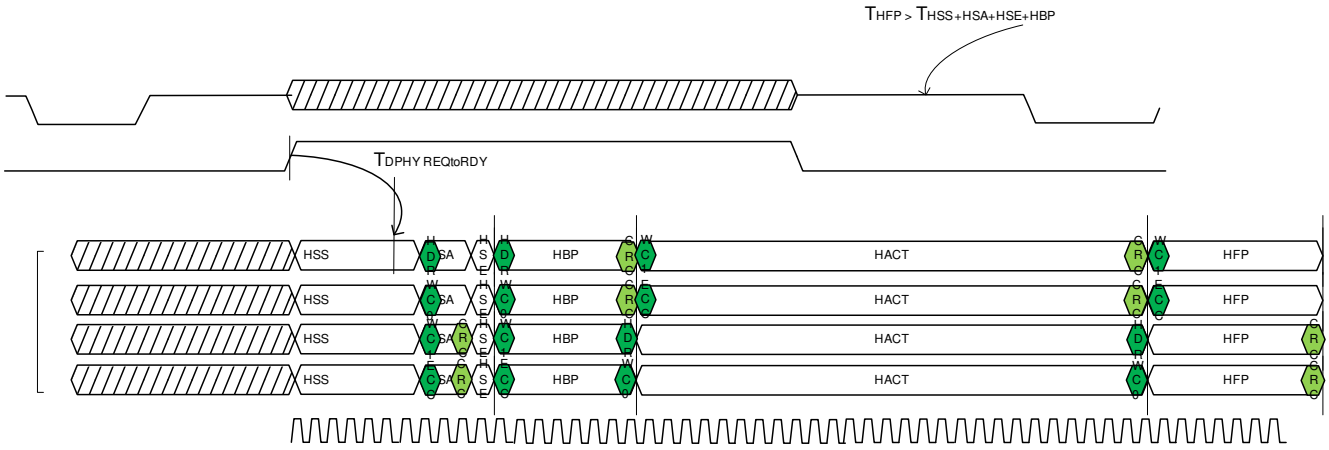


Figure 12-598. Horizontal Line Packet - Shorter HBP - Longer HFP Timing Diagram

The recommendation is to ensure that the total bytes for each horizontal line results in an exact integer tx_byte clock cycles (Total_bytes MOD num_of_lanes = zero), except for the case where the clocks are not exactly matched.

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.8.2 Pixel Clock to TX_BYTE_CLK Variation

The selection of the Pixel clock rate and DPHY Bit clock rate should ideally match based on the ratio of bpp on the pixel side compared to the tx_byte_clk. The timing for the horizontal sync, back and front porch will also be configured to allow the pixel data to be transferred without any over or underflow of the DPI FIFO.

The DPI FIFO fill level register (DPI_CFG) can be used to track that the parameters are sufficient and that any clock variation can still be handled by the FIFO depth. This register will show the pixel data held in the buffer, so during the HSA and HBP packet generation stages DPI pixel data will be stored and this value will increment. Once the data packet is being generated, the DPI_CFG level will remain constant, as the buffer is emptied at the same bandwidth as it is filled. Finally, the value will reduce to zero during the HFP stage.

The packet length of the HFP can be adjusted to help absorb small differences in the pixel_clk to tx_byte_clk relationship. The DSITX controller will concatenate all of the packets during High Speed transmission, so for example adding to the byte value of the HFP will make the bytes at the end of the packet move to align on another lane.

The packets that follow will then be concatenated and aligned to the next available lane, so if the tx_byte_clk is faster than the expected pixel clock times bpp the extra byte will delay the start of the new line tx_byte_clk × extra_byte/lanes.

Figure 12-599, Video coherency, illustrates the impact of increasing the HFP by one byte so that the next line begins ¼ tx_byte clock later. This allows averaging out across the line length of small differences in the pixel to tx_byte clocks when the tx_byte clock is not ideally matched to the expected pixel clock multiplied by bpp.

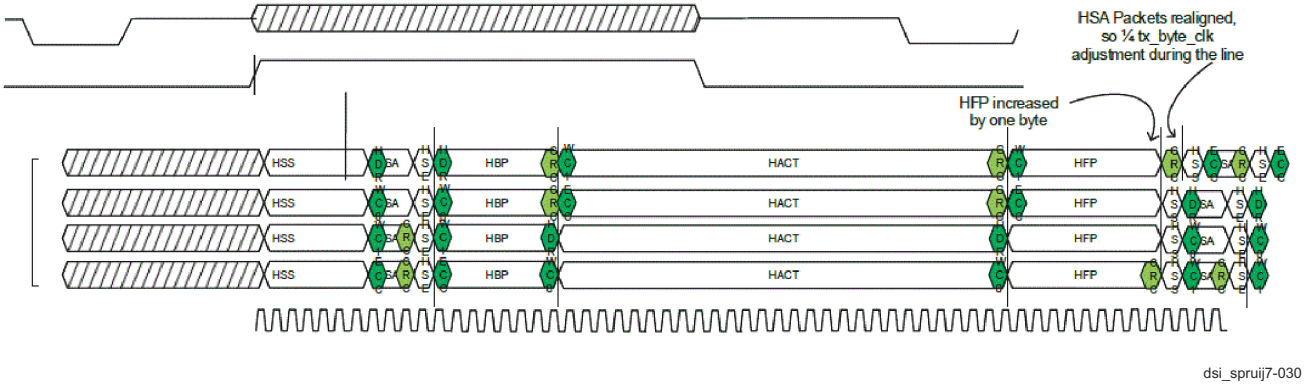


Figure 12-599. Packet Alignment Control Using HFP Byte Increase - tx_byte_clk Faster than Ideal Rate

The DPI_CFG register will show any clock misalignment as an increasing/decreasing value for the FIFO fill level on each line of pixels (must be read mid-line for an accurate reading of the level). When the byte count adjustment is made, the DPI_CFG register will return to the original fill level value every four lines.

12.9.2.7.8.3 LP Operation

The DSITX controller can be configured to perform a transition between LP state and HS state during the horizontal lines which have long periods of blanking. The registers controlling the timing for the DPHY wakeup time and the reg_line_duration must be programmed to exactly match the DPI cycles of the horizontal line to the tx_byte_clk cycles required for the number of active lanes selected.

Copyright © 2026 Texas Instruments Incorporated

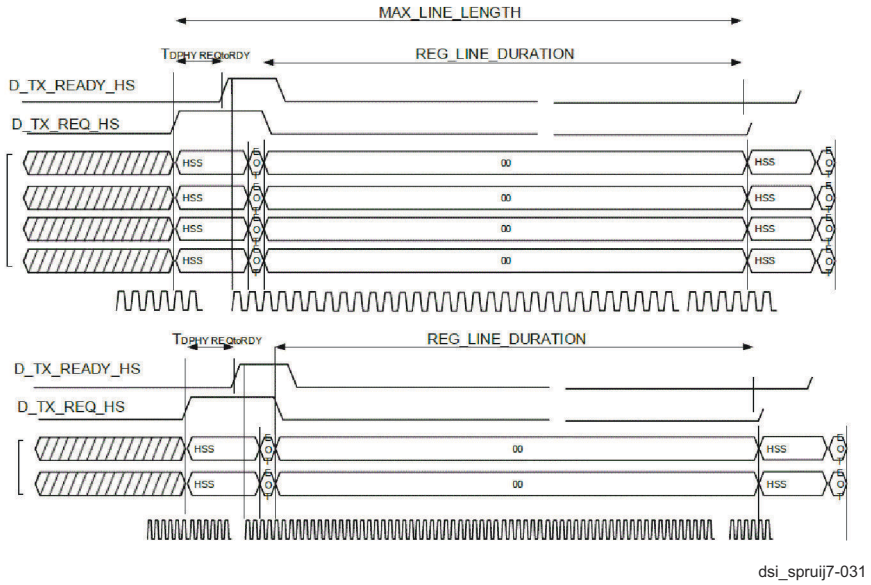


Figure 12-600. Low Power Operation with Four and Two Active Lanes

The LP operation for each vertical blanking line will require the reg_line_duration to be configured to match the following:

- Max_line_length (in tx_byte_clk cycles see 2.2.4) minus Eot (if enabled), also minus further 10 if CLOCK lane is non-continuous.

Note REG_WAKEUP_TIME is used internally to adjust the cycles for each vertical LP line.

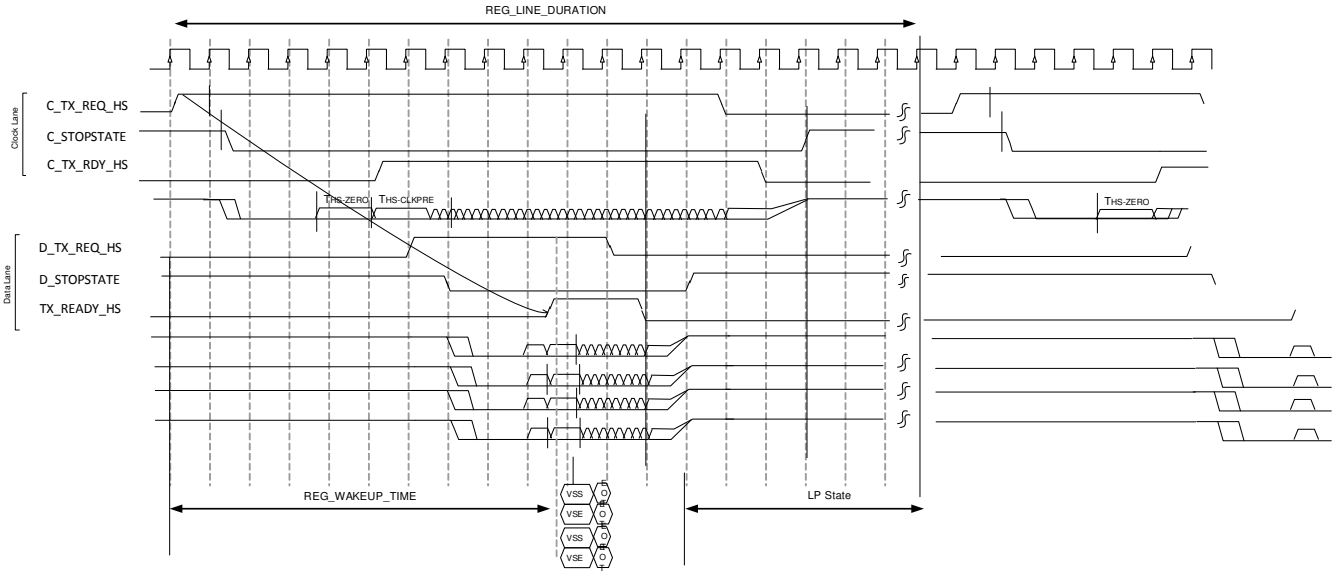


Figure 12-601. REG_LINE_DURATION Timing Example for LP Operation

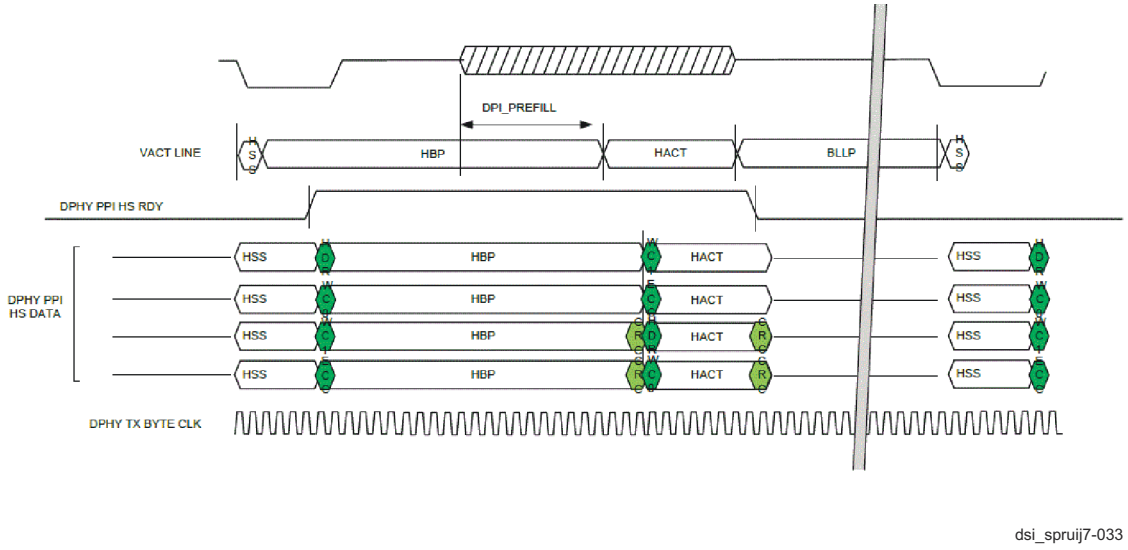
12.9.2.7.8.4 DPI Interface Burst Operation

The DSITX controller can support burst operation on the DPI interface provided the DPI FIFO size is large enough to store the line pixel data with enough data to not underflow. The burst operation will use a tx_byte_clk that is faster than normal event based operation, so the DPI FIFO size will be impacted by the ratio of the pixel_clk and BPP input to the tx_byte_clk and number of lanes on the output.

Copyright © 2026 Texas Instruments Incorporated

The recommended operation is to set the DSI HSA and DSI HFP registers to zero, and adjust the DSI HBP size to control the number of bytes prefilled in the DPI FIFO before the burst is sent.

Figure 12-602 shows DPI operation with event burst mode.



dsi_spruij7-033

Figure 12-602. DPI Operation with Event Burst Mode

12.9.2.7.9 Programming the DSITX Controller to Match the Incoming DPI Stream

The time taken to output a frame on the PPI needs to match what is coming in on the DPI. This is best achieved by using the recommended clock ratio between the pixel and byte clocks. Assuming that is the case, then the blanking and active data periods must be matched up.

For each frame of video the time will be:

$(V_{total}*H_{total})_{@pixel_clk} = (V_{total}*H_{total})_{@tx_byte_clk} \times bits_per_pixel/8;$

So for example, the time for a full HD frame 1920x1080 with reduced blanking in RGB565 will be:

2200x1200 pixels = 2200x1200 tx_bytes × 16/8;

The DSITX controller will match its frame timing to the incoming DPI vsync, as long as it is programmed to generate a frame of slightly less than the same size when the VFP blanking is considered. The controller works by transitioning to LP during the last programmed line of VFP. It will then remain in LP until the start of the next frame. So programming the controller to match the DPI, but with at least one fewer line of VFP should result in a reliable configuration.

12.9.2.7.9.1 Vertical Timing

The DSITX controller will generate vertical packets for each part of the frame beginning with the VSS and VSE combined with blanking packets to fill the VSA. The blanking packets for the pulse mode program the DSI vertical size registers as follows:

- VSA = DPI_VSI (lines, minimum of 2 –> VSS and VSE)
- VBP = DPI_VBP (lines, minimum of 0)
- VACT = DPI_VACT (lines)
- VFP < DPI_VFP (minimum of 1)

DPI horizontal timing is measured in pixel clocks, whereas the DSITX controller uses byte clocks. The horizontal timing should also be matched per line; therefore the blanking and active periods of each line should match. The relationship therefore depends upon the pixel format, which could be 24, 18 or 16 bpp. The timing for a horizontal line for the DPI driver side will be the number of pixel cycles for the **HLINE** = (HSA + HBP + HACT +

Copyright © 2026 Texas Instruments Incorporated

HFP) and this will form **HLINE** × bpp/8 bytes for the controller. These bytes will then be sent using 1, 2, 3 or 4 data lanes, so the DSITX controller will required 1, ½, 1/3, ¼ this number of tx_byte_clk cycles respectively.

e.g a DPI with HSA = 12, HBP = 12, HACT = 1920, HFP = 24 and 16bpp will be 1968 pixel clocks for each horizontal line.

This gives the DSI horizontal lines 3936 bytes to transmit and would use 984 tx_byte_clks on a four lane system, or 1312 on a three lane system etc..

The horizontal configuration uses different registers depending on the Mode (Pulse or Event) and the control of the BLKLINE_MODE register bit.

- Non-Burst Mode with Sync Pulses – enables the peripheral to accurately reconstruct original video timing, including sync pulse widths.

  Note that for accurate reconstruction of timing, packet overhead including Data ID, ECC, and Checksum bytes should be taken into consideration.
- Non-Burst Mode with Sync Events – similar to above, but accurate reconstruction of sync pulse widths is not required, so a single Sync Event is substituted.
- Burst mode – RGB pixel packets (active video) portion are time-compressed, leaving more time during a scan line for LP mode (saving power) or for multiplexing other transmissions onto the DSI link.

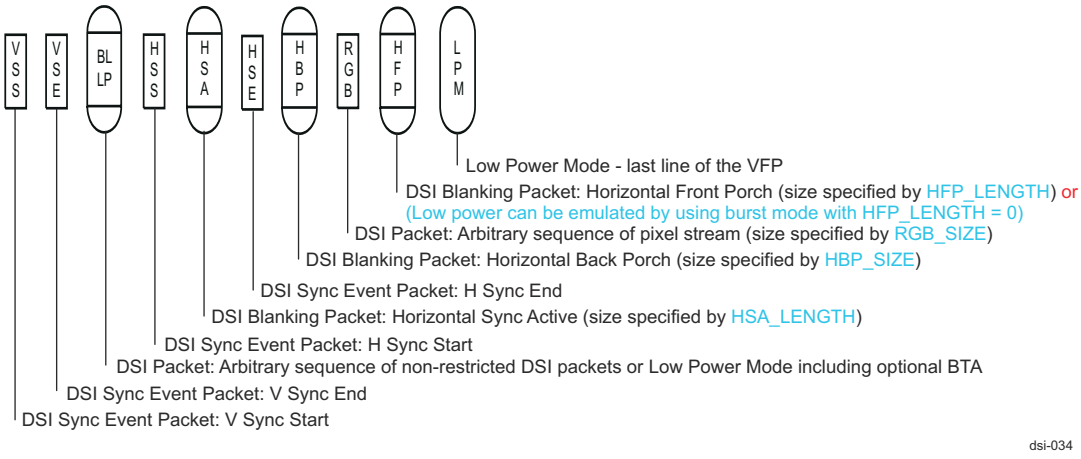


Figure 12-603. Vertical Timing

12.9.2.7.9.2 Horizontal Timing for Non-Burst Mode with Sync Pulses

Sync Pulse Mode uses the Short packet and Long packet structures to accurately track the DPI interface timing. Figure 12-604 illustrates the packet sequence for a single frame.

Copyright © 2026 Texas Instruments Incorporated

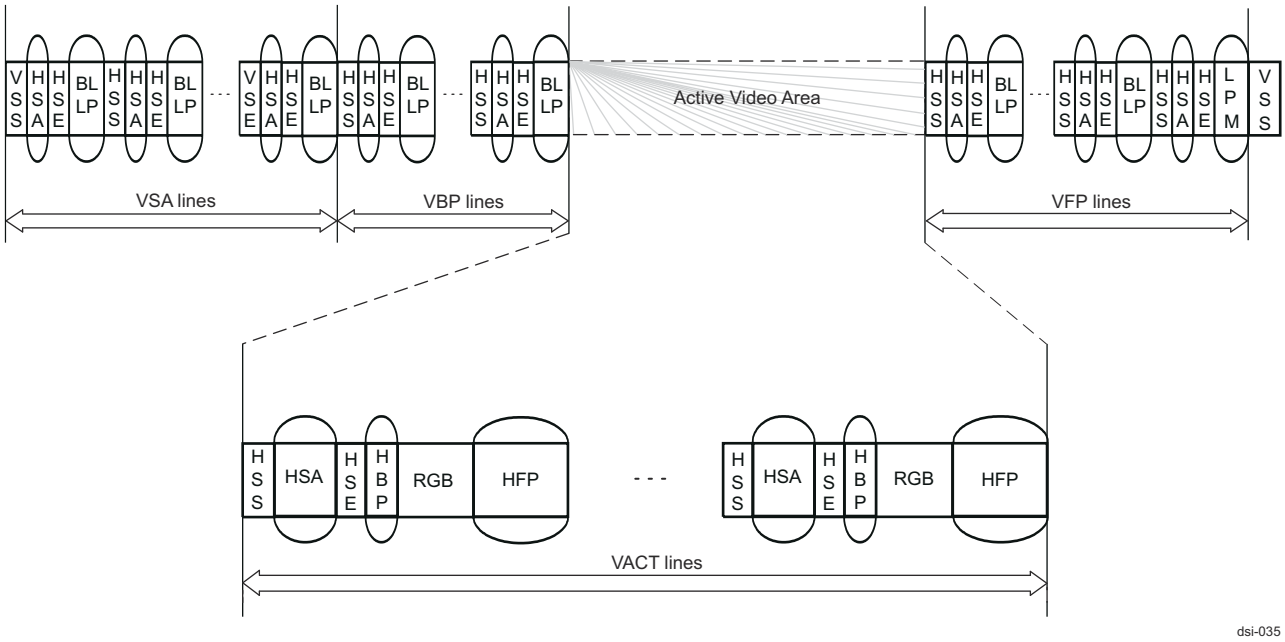


Figure 12-604. Vertical Timing

The DSITX controller registers for this mode are used to generate the exact number of bytes required for each horizontal line based on the DPI line configuration and the BPP for the colour pixel data.

The packet structure for each type of line during the frame is shown below, Trigger Mapping Information. The calculation of the total number of bytes in any horizontal line must always be HLINE × bpp/8 bytes.

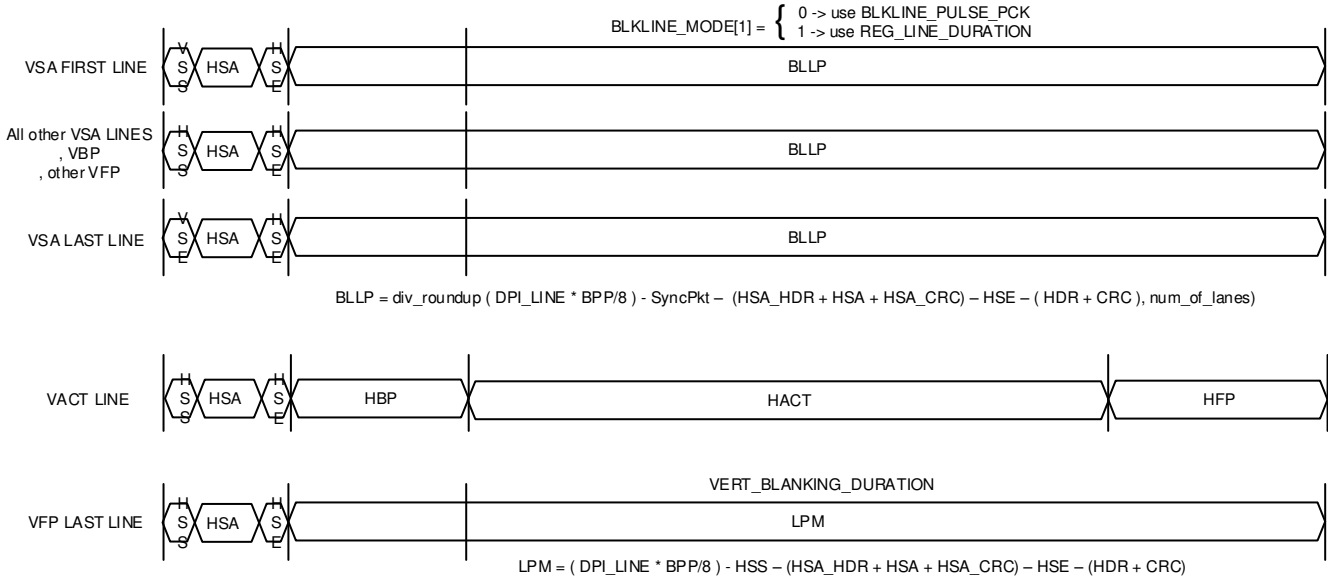


Figure 12-605. Non-Burst Mode With Sync Pulse Line Structures

The DSI short packets and packet headers that are inserted by the controller must be accounted for:

- HSA should be reduced by 14 bytes to account for the HSS short packet (4 bytes), the long blanking packet header and CRC footer (4 + 2 bytes) and the HSE short packet (4 bytes).
- HBP should be reduced by 12 to account for the header and footer on the blanking packet (6 bytes) plus the header/footer on the active data packet (6 bytes).
- HFP should be reduced by 6 bytes to account for the long packet header and CRC footer.

Copyright © 2026 Texas Instruments Incorporated

Finally for lines with no active data, the controller will use either:

- BLKLINE_PULSE_PCK: Total line size (HLINE × bpp/8) in bytes minus the HSA 20 bytes (14 for HSA header and CRC, 6 for the remaining blanking which is all combined into a single packet).
- REG_LINE_DURATION: Total line size (HLINE × bpp/8) in tx_byte_clk cycles minus the calculated HSA in tx_byte_clk cycles (HSA × bpp/8 minus 14 bytes (14 for HSA header and CRC) minus the EOT packet (4 bytes) if required, divided by number of lanes). This should be aligned to the number of active lane so rounding the value so that REG_LINE_DURATION MOD Lanes equals zero.
- VERT_BLANKING_DURATION: Total line size (HLINE × bpp/8) in tx_byte_clk cycles minus the calculated HSA in tx_byte_clk cycles (HSA × bpp/8 minus 14 bytes (14 for HSA header and CRC) divided by number of lanes).

Program the DSI horizontal size registers as follows:

burst_mode = 0 and sync_pulse_active = sync_pulse_horizontal = 1;

HSA = (DPI_HSA × bpp/8)-14 – DPI HSA min value 5 for RGB888

HBP = (DPI_HBP × bpp/8)-12– DPI HBP min value 5 for RGB888

HACT = (DPI_HACT × bpp/8)

HFP = (DPI_HFP × bpp/8)-6 – DPI HFP min value 10 for RGB888

Note: (DPI_HACT × bpp/32) must be an integer. Total Line Length = div_roundup((HLINE × bpp/8), num of lanes);

(BLKLINE_PULSE_PCK) bytes} = (HLINE × bpp/8)-20-HSA;

(REG_LINE_DURATION)tx_{byte}clks} = Total Line Length - div_roundup((HSA × bpp/8) - 14, number of lanes);

(VERT_BLANKING_DURATION)tx_{byte}clks} = Total Line Length - div_roundup((HSA × bpp/8) - 14, number of lanes);

For example; a DPI with HSA = 12, HBP = 12, HACT = 1920, HFP = 24 and 16bpp will be 1968 pixel clocks for each horizontal line. This give each DSI HLINE of 3936 bytes. So for 4 lanes, 984 tx_byte_clk cycles;

HSA = (12 × 16/8) -14 = 10

HBP = (12 × 16/8) -12 = 12

HACT = (1920 × 16/8) = 3840

HFP = (24 × 16/8) -6 = 42

Total Line = div_roundup((12 + 12 + 1920 + 24) × 16/8, 4) = 984 tx_byte_clk cycles = 3936 bytes;

BLKLINE_PULSE_PCK = (3936) -20 -10 = 3906

REG_LINE_DURATION = 984 – div_roundup(12 × 2, 4) = 978

VERT_BLANKING_DURATION = 984 – div_roundup(12 × 2, 4) = 978

Confirming the calculation for each line with BLKLINE_MODE = 0;

VSS = VSSPkt + HSAPkt + HSEPkt + BLK_LINE_PULSE_PCKPkt = 4 + (4 + 10 + 2) +4 + (4 + 3906 + 2) = 3936;

VSALine = HSSPkt + HSAPkt + HSEPkt + BLK_LINE_PULSE_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936;

VSE = VSEPkt + HSAPkt + HSEPkt + BLK_LINE_PULSE_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936;

VACT = HSSPkt + HSAPkt + HSEPkt + HBPPkt + HACTPkt + HFPPkt = 4 + (4 +10 + 2) +4 + (4 + 12 + 2)+ (4 + 3840 +2) + (4 + 42 + 2) = 3936;

Copyright © 2026 Texas Instruments Incorporated

VFP = HSSPkt + HSAPkt + HSEPkt + BLK_LINE_PULSE_PCKPkt = 4 + (4 + 10 + 2) + 4 + (4 + 3906 + 2) = 3936;

VFPLast = HSSPkt + HSAPkt + HSEPkt + VERT_BLANKING_DURATION = 4 + (4 + 10 + 2) + 4 + (978 × 4) = 3936;

Confirming the calculation for each line with BLKLINE_MODE = 1 matches the tx_byte cycles;

VSS = VSSPkt + HSAPkt + HSEPkt + REG_LINE_DURATION = div_roundup((4 + (4 +10 + 2) + 4), 4) + 978 = 984;

VSALine = HSSPkt + HSAPkt + HSEPkt + REG_LINE_DURATION = div_roundup((4 + (4 +10 + 2) + 4), 4) + 978 = 984;

VSE = VSEPkt + HSAPkt + HSEPkt + REG_LINE_DURATION = div_roundup((4 + (4 + 10 + 2) + 4), 4) + 978 = 984;

VACT = HSSPkt + HSAPkt + HSEPkt + HBPPkt + HACTPkt + HFPPkt = div_roundup((4 + (4 + 10 + 2) +4 + (4 + 12 + 2)+ (4 + 3840 +2) + (4 + 42 + 2) = div_roundup((3936, 4) = 984;

VFP = HSSPkt + HSAPkt + HSEPkt + REG_LINE_DURATION = div_roundup((4 + (4 + 10 + 2) + 4), 4) + 978 = 984;

VFPLast = HSSPkt + HSAPkt + HSEPkt + VERT_BLANKING_DURATION = div_roundup((4 + (4 + 10 + 2) + 4 + (3912), 4) = 984;

12.9.2.7.9.3 Event Mode Horizontal Timing

Non Burst Event Mode uses the Short packet and Long packet structures to track the DPI interface timing without accurate reconstruction of sync pulse widths, so a single Sync Event is substituted. Figure 12-606 illustrates the packet sequence for a single frame.

The DSITX controller registers for this mode are used to generate the exact number of bytes required for each horizontal.

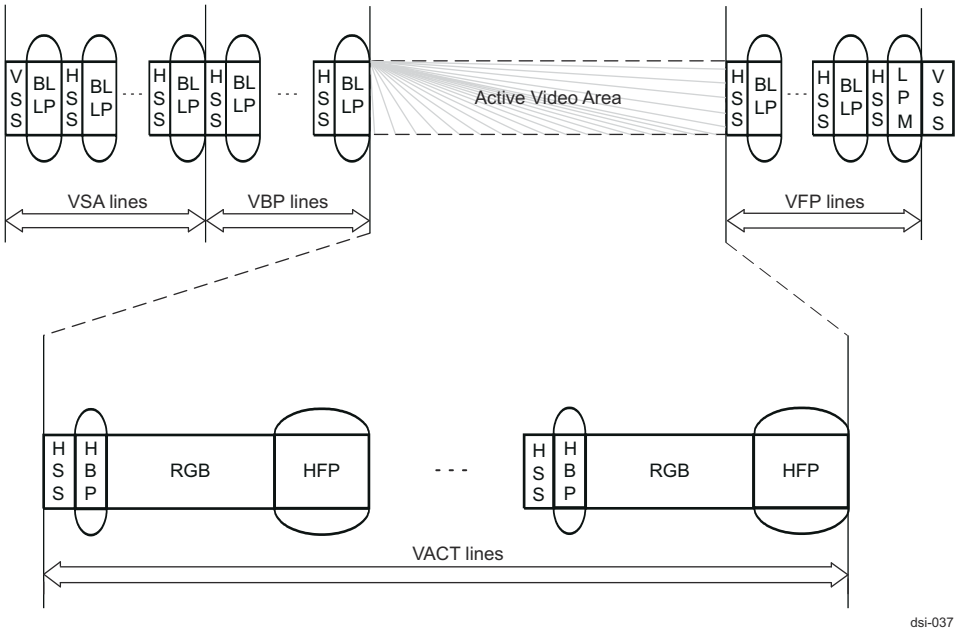


Figure 12-606. Horizontal Timing - 1

line based on the DPI line configuration and the BPP for the colour pixel data. The packet structure for each type of line during the frame is shown in Figure 12-607:

Copyright © 2026 Texas Instruments Incorporated

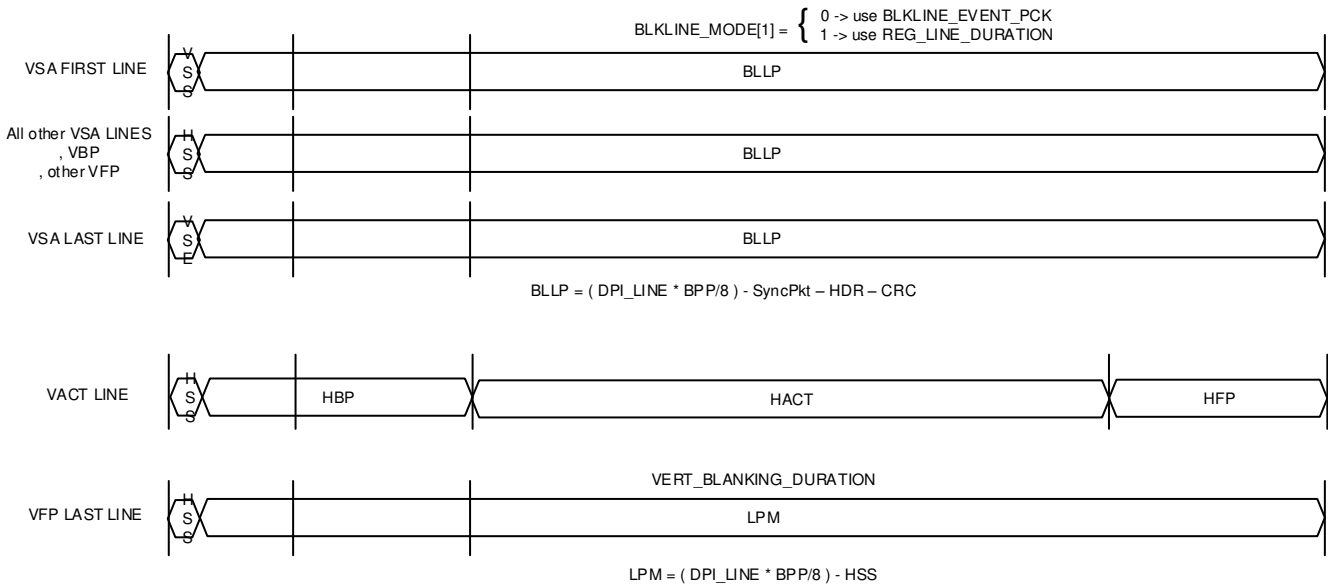


Figure 12-607. Horizontal Timing - 1

The DSI short packets and packet headers that are inserted by the controller must be accounted for:

- VSS/VSE/HSS short packet (4 bytes).
- HBP should account for the header and footer on the blanking packet (6 bytes) plus the header on the active data packet (2 bytes). This will match the DPI_HSA + DPI_HBP minus the Sync Short packet.
- HFP should be reduced by 6 bytes to account for the long packet header and CRC footer and footer of the active data.

Finally, for lines with no active data, the controller will use either:

- BLKLINE_EVENT_PCK: Total line size (HLINE × bpp/8) in bytes minus 10 bytes (4 for VSS/VSE/HSS, 6 for the remaining blanking header/footer which is all combined into a single packet).
- REG_LINE_DURATION: Total line size div_roundup(HLINE × bpp/8, num_of_lanes) in tx_byte_clk cycles minus tx_byte cycles for the 8 bytes (4 for VSS/VSE/HSS, and 4 for EOT). This should be aligned to the number of active lane so round the value so that REG_LINE_DURATION MOD Lanes equals zero.
- VERT_BLANKING_DURATION: Total line size div_roundup (HLINE × bpp/8, num_of_lanes) in tx_byte_clk cycles minus tx_byte cycles for the 4 bytes (4 for HSS).

Program the DSI horizontal size registers as follows:

burst_mode = 0 and sync_pulse_active = sync_pulse_horizontal = 0;

- HSA = 0
- HBP = ((DPI_HSA + DPI_HBP) × bpp/8)-12 – DPI HSA + HBP min value 5 for RGB888
- HACT = (DPI_HACT × bpp/8)
- HFP = (DPI_HFP × bpp/8)-6 – DPI HFP min value 10 for RGB888

Note: (DPI_HACT × bpp/32) must be an integer. Total Line Length = div_roundup((HLINE × bpp/8), num of lanes).

(BLKLINE_EVENT_PCK) bytes} = (HLINE × bpp/8) - 10

(REG_LINE_DURATION) tx_{byte}clks} = Total Line Length - div_roundup(8, number of lanes)

(VERT_BLANKING_DURATION) tx_{byte}clks} = Total Line Length - div_roundup(4, number of lanes)

For example: a DPI with HSA = 12, HBP = 12, HACT = 1920, HFP = 24 and 16bpp will be 1968 pixel clocks for each horizontal line. This give each DSI HLINE of 3936 bytes.

HSA = 0

Copyright © 2026 Texas Instruments Incorporated

HBP = ((12 + 12) × 16/8) -12 = 24

HACT = (1920 ×16/8) = 3840

HFP = (24 ×16/8) -6 = 42

Total Line = div_roundup((12 + 12+1920 + 24) × 16/8, 4) = 984

BLKLINE_EVENT_PCK = (HLINE × bpp/8) – 10 = 981

REG_LINE_DURATION = 984 – div_roundup(8, 4) = 982

VERT_BLANKING_DURATION = 984 – div_roundup(4, 4) = 983

Confirming the calculation for each line:

VSS = VSSPkt + BLK_LINE_EVENT_PCKPkt = 4 + (4 + 3926 + 2) = 3936

VSALine = HSSPkt + BLK_LINE_EVENT_PCKPkt = 4 + (4 + 3926 + 2) = 3936

VSE = VSEPkt + BLK_LINE_EVENT_PCKPkt = 4 + (4 + 3926 + 2) = 3936

VACT = HSSPkt + HBPPkt + HACTPkt + HFPPkt = 4 + (4 + 24 + 2) + (4 + 3840 +2) + (4 + 42 + 2) = 3936

VFP = HSSPkt + BLK_LINE_EVENT_PCKPkt = 4 + (4 + 3926 +2) = 3936

VFPLast = HSSPkt + VERT_BLANKING_DURATION = 4 + (983 × 4) = 3936

12.9.2.7.9.4 Burst Event Mode Horizontal Timing

Burst Event Mode uses the Short packet and Long packet structures to transmit the video information in short bursts and allow the system to use the Low Power states to save active power. This can only be supported with DSITX controllers able to store a horizontal line in the DPI FIFO. The burst mode will expect the tx_byte_clk to be twice the rate of the non burst event mode, and the register calculation will expect the total bytes for each line to double. Figure 12-608 illustrates the packet sequence for a single frame.

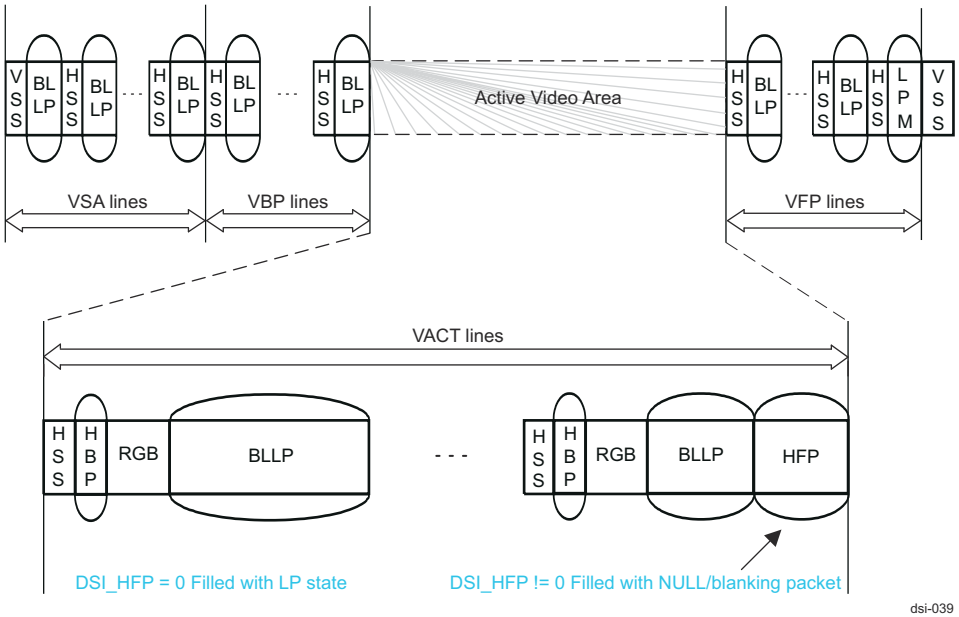
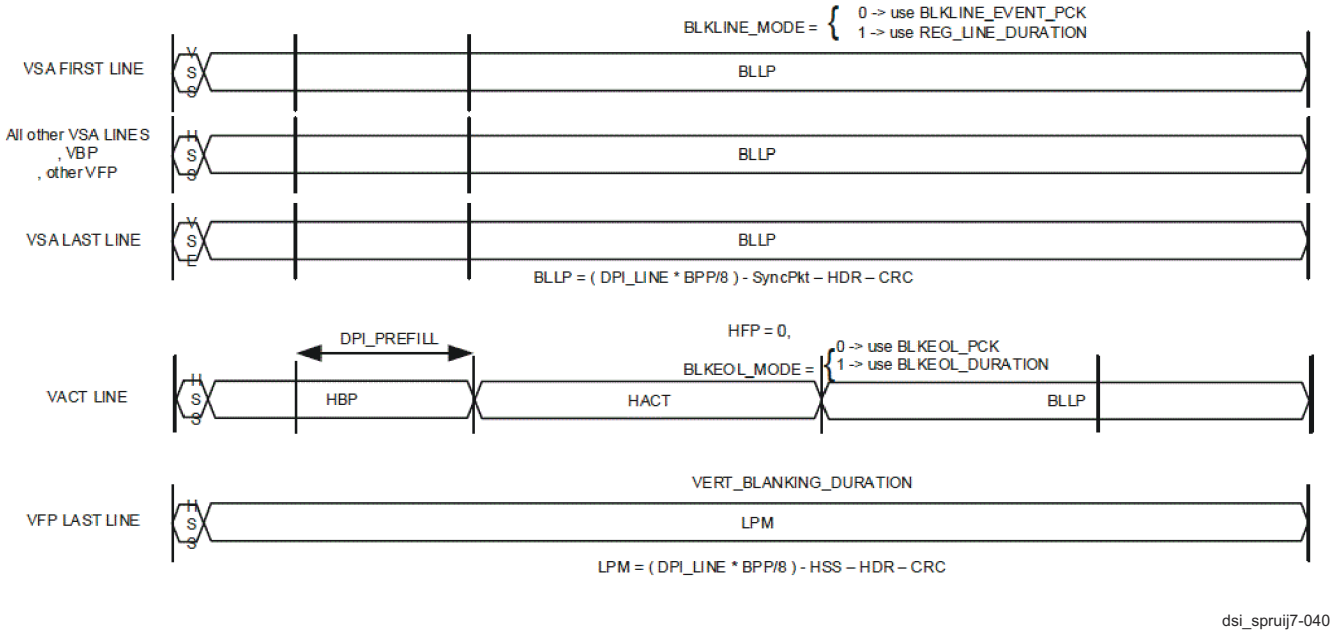


Figure 12-608. Burst Event Mode Horizontal Timing - 1

The DSITX controller registers for this mode are used to generate the exact number of bytes required for each horizontal line based on the DPI line configuration and the BPP for the colour pixel data.

Copyright © 2026 Texas Instruments Incorporated

The packet structure for each type of line during the frame is shown below, with the DSI HFP = 0 to select BLLP operation.



dsi_spruij7-040

Figure 12-609. Burst Event Mode Horizontal Timing - 2

The DSI short packets and packet headers that are inserted by the controller for active lines must be accounted for:

- VSS/VSE/HSS short packet (4 bytes).
- HBP should account for the header and footer on the blanking packet (6 bytes) plus the header/footer on the active data packet (6 bytes). This will match the DPI_HSA + DPI_HBP minus the Sync Short packet. In this case the DPI must also reduce the HSA and HFP timing.
- BLLP uses BLKEOL_PCK or BLKEOL_DURATION and will change depending on the HFP use case:
  - HFP can be zero to extend the BLKEOL_x values(recommended).
  - or a non-zero value to transmit a Blanking Packet instead of LP state. The Blanking Packet HFP should be reduced by 6 bytes to account for the long packet header and CRC footer.
  - The host_eot state will cause the BLLP to either add and EOT cycle or not. The BLKEOL_PCK or BLKEOL_DURATION value will need to be adjusted to include this extra cycle if host_eot is disabled.

Finally for lines with no active data, the controller will use either:

- BLKLINE_EVENT_PCK in bytes, excluding the header and crc.
- REG_LINE_DURATION or VERT_BLANKING_DURATION.
- Both are the total expected tx_byte_clk cycle for the line minus the number of cycles required to send the header packet bytes (4 for VSS/VSE/HSS) over the active lanes.

Program the DSI horizontal size registers as follows:

burst_mode = 1 and sync_pulse_active = sync_pulse_horizontal = 0;

- HSA = 0
- HBP = (2 × (DPI_HSA + DPI_HBP) × bpp/8)-12 + DPI FIFO Prefill
- HACT = (DPI_HACT × bpp/8) NOTE: (DPI_HACT × bpp/32) must be an integer.
- HFP = Zero

Total Line Length = div_roundup((HLINE × bpp/8), num of lanes);

(BLKLINE_EVENT_PCK)} = (HLINE × bpp/8) × 2 - 4

Copyright © 2026 Texas Instruments Incorporated

(REG_LINE_DURATION)} = Total Line Length × 2 - div_roundup(4, number of lanes)

(VERT_BLANKING_DURATION)} = Total Line Length × 2 - div_roundup(4, number of lanes)

(BLKEOL_DURATION)} = Total Line Length × 2 - TX_BYTE_CYCLES

(BLKEOL_PCK) bytes} = Total Line Length × 2 - (HACT + HBP + HSS)

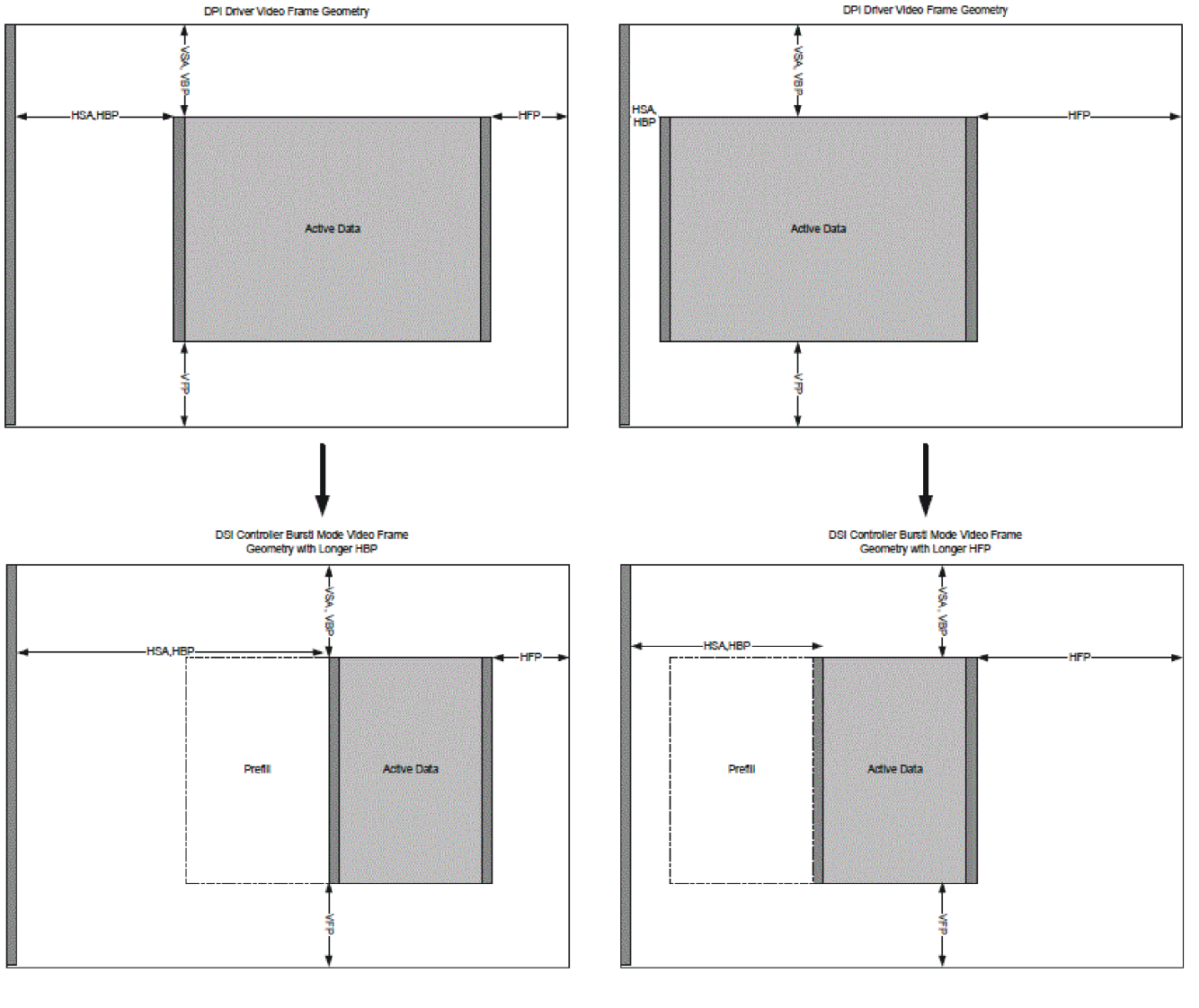
Note: TX_BYTE_CYCLES = TX Byte clock cycles to send Active Line bytes on the number of active lanes = div_roundup (HBP + HACT, number of lanes).

Burst Operation Frame Configuration

Burst mode operation can be used to save power provided the system configuration can support the higher tx_byte clock rate. The user should consider if the power saving during LP states is more than reducing the number of active lanes and running in non-burst mode.

The burst mode can only be used if the controller configuration can support it; either using SDI video interface, of DPI interface with a buffer size large enough to prefill the data. The video driver timing from the DPI side will also need to be changed.

Copyright © 2026 Texas Instruments Incorporated



dsi_spruij7-041

Figure 12-610. DSITX Controller Video Frames for Burst Mode Compared with DPI Driver Side

For example:

The VESA configuration for a 1920x1200 frame at 60-frames per second with normal blanking would have a pixel clock at 193.25 MHz, HSA 200 pixels, HBP 336 pixels, HFP 136 pixels and total line 2592 pixels. This would need to be reconfigured to move the active area earlier in the horizontal line for the DSITX controller, so could be changed to; HSA 20 pixels, HBP 30 pixels, HFP 622 pixels. This would allow the HFP LP state to be almost ¼ of the line (622/2592).

12.9.2.7.9.5 Burst Mode Operation

The DSI IP offers different options for behavior in the BLLP areas:

- Insertion of blanking or null packets.
- Switch to LP (power saving).
- Insertion of commands (if any) + blanking or null packet.

Figure 12-611 shows burst mode operation.

Copyright © 2026 Texas Instruments Incorporated



Figure 12-611. Burst Mode Operation

The user should balance the increase in the clock frequencies for transmission of the active display, against the power saving by using BLLP(Null) or LP state.

For Burst with the LP state the register fields must be programmed for burst_lp = 1 and reg_blkeol_mode = 2'b2.

Command Insertion Operation

The DSITX can use the burst operation to allow the blanking/LP stages to be used for DCS/GEN command insertion by programming the size of the space available after the sync of each line.

Only one command packet can be passed within each video line. The decision to insert a command in a video line is done at the beginning of the slot just after the previous video packet. If a command arrives in the middle of the slot, it is not processed in that line, the decision to accept the command will be taken in the next line.

If the packet does not fit in a short slot (size bigger than reg_max_burst_limit and not equal to reg_exact_burst_limit), it is delayed up to next long slot. The signal reg_err_burstwrite is flagged (it is only an information that can help application software control). If its length is longer than reg_max_line_limit, it is discarded and the error reg_err_linewrite is generated.

Read commands and BTA requests are only passed during long slots, as there is no way to calculate their duration exactly. In the case where their duration takes longer time than the vertical blanking period, the error reg_err_longread is flagged.

Copyright © 2026 Texas Instruments Incorporated

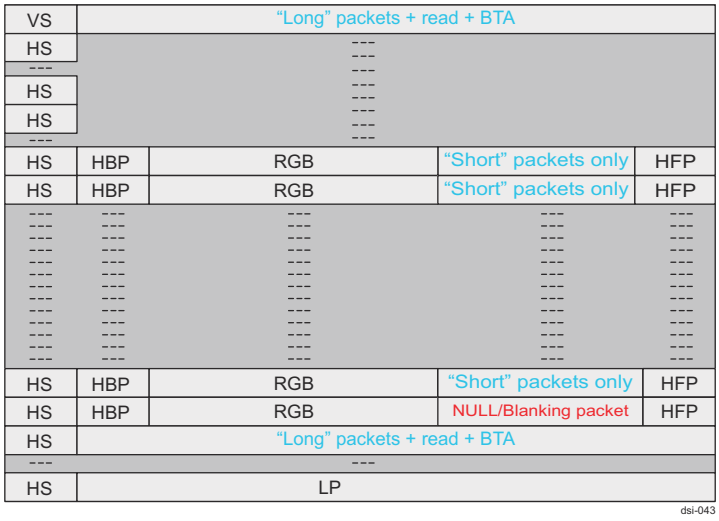
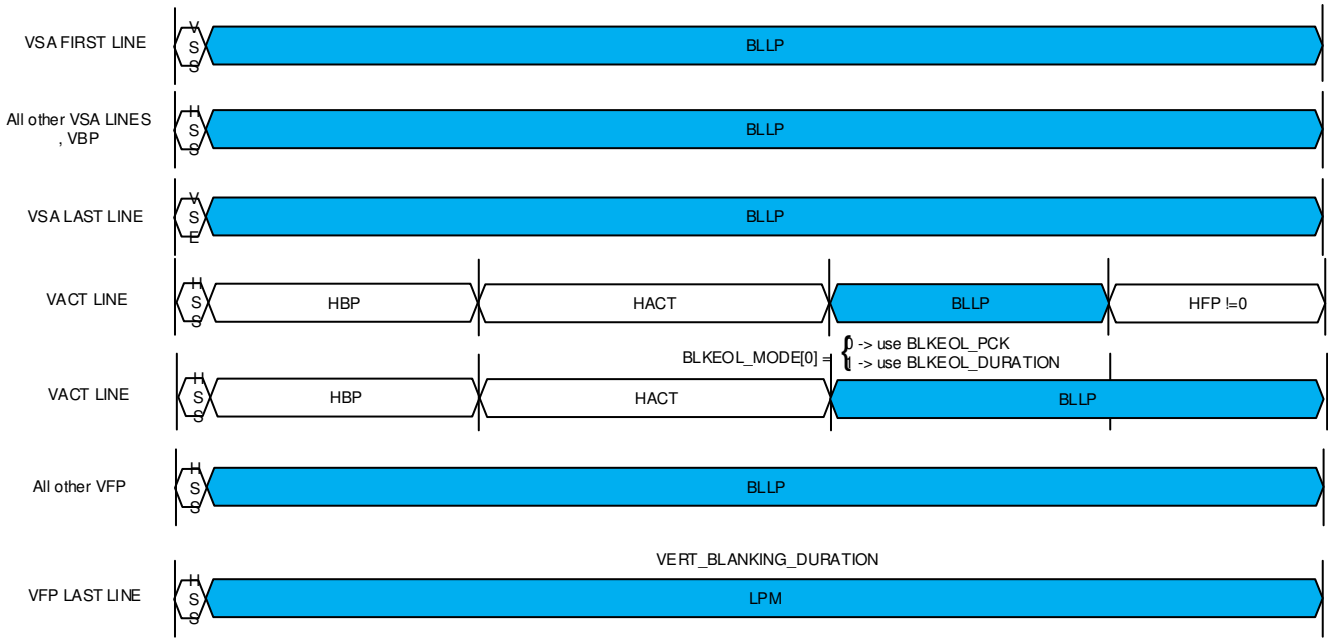


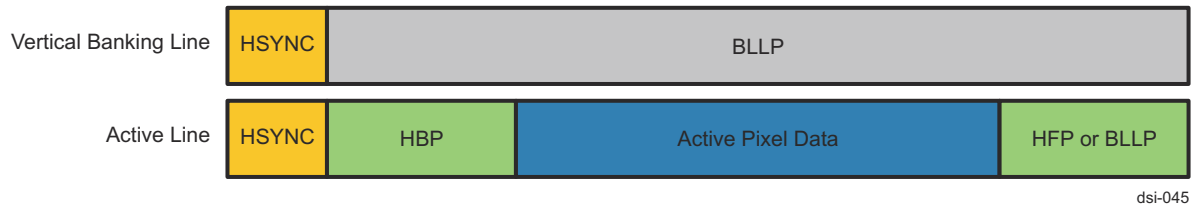
Figure 12-612. Burst Mode Command Locations During Video Frame

Command Insertion Registers in Burst Operation

The burst operation requires the configuraton of the registers to control the size of the packets that will fit in the space available on each type of horizontal line.



In VSA, VBP or VFP: Insertion of short and long command packets, read or BTA are allowed.



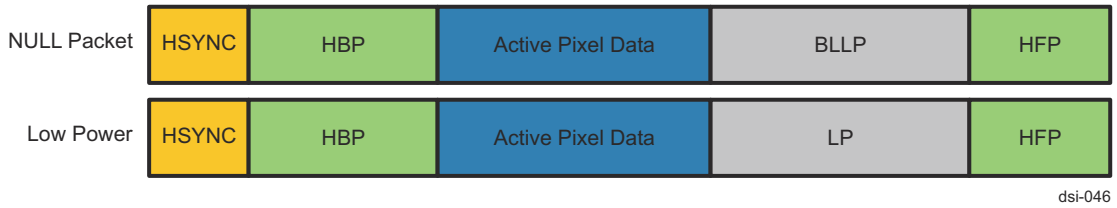
- In VACT: Only short packets can be inserted.
- Command, if inserted, is inserted right after the HSA or HSE or the video packet.
- Only one command can be inserted by line.

Copyright © 2026 Texas Instruments Incorporated

- When a command is inserted on a video line, no switch to LP but Null packet insertion.
- Trigger and TE insertion not supported.

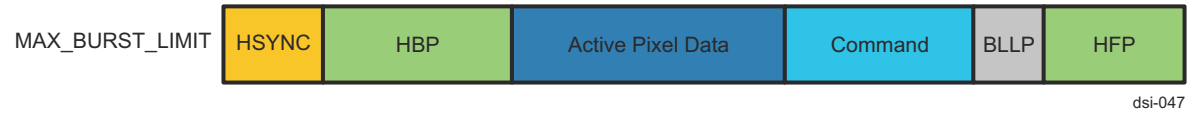
VID_VCA_SETTING1 register:

[16] BURST_LP: After an active line, in burst mode, the system can switch in LP (1) or should complete the line with NULL packet (0).



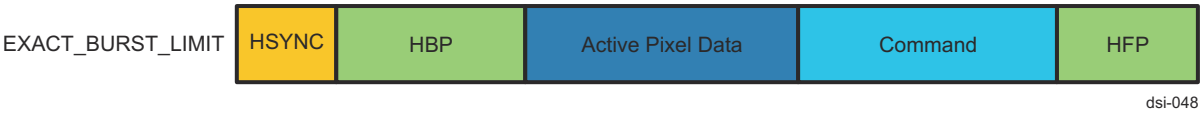
VID_VCA_SETTING1 register:

[15:0] MAX_BURST_LIMIT: Maximum length of a packet (number of bytes) that can be passed during the blanking period at the end of an active line (in burst mode) followed by a NULL packet with no data (6 bytes).



VID_VCA_SETTING2 register:

[15:0] EXACT_BURST_LIMIT:Exact maximum size of the burst packet (packet that fits after pixel data in burst mode), that is with no Null packet insertion afterwards.



VID_VCA_SETTING2 register:

[31:16] MAX_LINE_LIMIT: The "maximum" packet (number of bytes) that can be passed during the vertical blanking period.



If the packet does not fit in a short slot (size bigger than MAX_BURST_LIMIT and not equal to EXACT_BURST_LIMIT), it is delayed up to next long slot.

- If the packet length is longer than MAX_LINE_LIMIT, it is thrown away (an error is asserted).

Table 12-496 outlines the expected behaviour of each packet type during the video frame.

Table 12-496. Behavior of the DSITX Transmission when Both Video and Command are Running

Packet type	Packet Size	Behavior	Error Flagged
VACT Active Line - Read and BTA do not pass			
No packet	-	LP mode null packet sent	
Short packet		Packet passed followed by a null packet	
Long packet	size <reg_max_burst_limit	Packet passed followed by a null packet	

Copyright © 2026 Texas Instruments Incorporated

Table 12-496. Behavior of the DSITX Transmission when Both Video and Command are Running (continued)

Packet type	Packet Size	Behavior	Error Flagged
Long packet	size = reg_exact_burst_limit	Packet passed	
Long packet	reg_max_burst_limit < size < reg_exact_burst_limit	Packet delayed, LP mode packet delayed, null packet sent	Burstwrite Burstwrite
Long packet	reg_exact_burst_limit < size <= reg_max_line_limit	packet is delayed to non-active part of frame: Packet delayed, LP mode packet delayed, null packet sent	Burstwrite Burstwrite
Packet type	Packet Size	Behavior	Error Flagged
Long packet	size > reg_max_line_limit	Packet thrown, LP mode packet thrown, null packet sent	Linewrite Linewrite
Read packet	-	Packet delayed, LP mode Packet delayed, null packet sent	-
BTA request	-	Packet delayed, LP mode Packet delayed, null packet sent	-
VSA, VBP, VFP - Long packet, read and BTA can be passed			
No packet	-	LP mode	-
Short packet	-	Packet passed followed by a null packet	-
Long packet	size < reg_max_burst_limit	Packet passed followed by a null packet	
Long packet	size >reg_exact_burst_limit	packet thrown, LP mode	Linewrite
Read packet	read completed (direction) before end of slot	Packet passed	
Read packet	read not completed (direction) before end of slot	Packet passed	Longread If operation lasts too long
BTA request	-	Packet passed	Longread If operation lasts too long

Note that for burst operation with Low Power, the command will be sent as a high-speed packet even if the request is made to send as Low Power. Also, the next active line will return to using LP during the HFP phase if no other commands are requested.

12.9.2.7.9.6 Example Configurations

1. Here is an example for a VGA frame with 24bpp @60fps using non burst pulse sync mode: With blanking, total size 500 lines, each line is 800 pixels.

400000 DPI cycles/frame @ 24 MHz pixel clock <-> 300000 DSI cycles/frame @ 18 MHz byte clock;

800 DPI cycles/line @ 24 MHz <-> 600 DSI cycles/line @ 18 MHz;

Table 12-497. DPI and DSI Parameters - 1

Parameter	DPI	DSI
VSA	5	5
VBP	5	5
VACT	480	480

Copyright © 2026 Texas Instruments Incorporated

Table 12-497. DPI and DSI Parameters - 1 (continued)

Parameter	DPI	DSI
VFP	10	5
HSA	40	106 (120-14)
HBP	40	108 (120-12)
HACT	640	1920
HFP	80	234 (240-6)
HTOTAL	800 pixel cycles	2400 bytes

burst_mode = 0

sync_pulse_active = 1

sync_pulse_horizontal = 1

BLKLINE_PULSE_PCK = 2380 (2400-20)

REG_LINE_DURATION = 2380

VERT_BLANKING_DURATION = 2384

Note: The controller will send 4 VFP lines then transition to LP for the duration equivalent to 6 lines.

2. Here are some example values for a 24fps UHD frame size with 24bpp using non-burst event mode: With blanking, total size 2250 lines, each line is 4000 pixels.

9000000 DPI cycles/frame @ 216 MHz pixel clock <-> 6750000 DSI cycles/frame @ 162 MHz byte clock

4000 DPI cycles/line @ 216 MHz <-> 3000 DSI cycles/line @ 162 MHz

Table 12-498. DPI and DSI Parameters - 2

Parameter	DPI	DSI
VSA	2	2
VBP	0	0
VACT	2160	2160
VFP	40	1
HSA	40	0
HBP	40	228 (120 + 120 - 12)
HACT	3840	11520
HFP	80	234 (240 - 6)
HTOTAL	4000	12000

burst_mode = 0

sync_pulse_active = 0

sync_pulse_horizontal = 0

BLKLINE_EVENT_PCK = 11990 (12000-10)

REG_LINE_DURATION = 11990

VERT_BLANKING_DURATION = 11996

Note: After the active data the controller will transition immediately to LP for the duration equivalent to 88 lines, giving the maximum power saving between lines.

Copyright © 2026 Texas Instruments Incorporated

If any of the DPI related interrupts are triggered, then this highlights that the FIFO depth and/or the vsync_delay settings require to be tuned to the current configuration. Simulating the core operation with the expected clocks is the best way to ensure.

12.9.2.7.9.7 Stereoscopic Video Support

The DSI controller can support stereoscopic display format (SDF) using both command and video modes.

- Command Mode
  - APB Direct Command 3D Commands – Use get_3D_control with Read
  - SDI Command Mode – DCS Writes
- Video Mode – Single Link Operation
  - VSYNC Parameter1 Definition (L/R, Mode, Format)
  - SDI Video Mode
  - DPI Video Mode – Video stream will expect to follow expected format; frame, line, pixel
    - 3DVSYNC = 0 not currently supported.

The DSI must be configured to enable the 3D operation and identify the format that is being used using the register control to enable the 3D features and the configuration that matches the video stream from the system.

The SDF format requires the system to provide frame and synchronisation information using the two parameter fields within the vertical sync short packet (VSS).

Configured the DSI mctl_3dvideo_ctl register to match format – L/R Pixel, L/R Line, L/R Frame Drive video stream based on orientation and format.

Note: DSI Controller will not perform any on-the-fly pixel manipulation for L-R ordering.

Example Video Operation

For a 3D video stream where the DSI video is given the combined images as one continuous large frame, the VSYNC packet would identify that the left image is sent first, it is a line based format and in landscape mode (see Table 12-499).

Table 12-499. Example Video Operation

Position	D7	D6	D5	D4	D3	D2	D1	D0
Value	0	0	0	0	0	0	1	0
Description	Rsvd	Rsvd	Left First	No Sync	Line-based		Landscape	

mctrl_3dvideo_ctl (0x20)

Table 12-500. VSYNC Parameter 1

[7]	vid_vsync_3d_en	1
[5]	vid_vsync_3d_lr	0
[4]	vid_vsync_3d_second_en	1
[3:2]	vid_vsync_3dformat	00
[1:0]	vid_vsync_3dmode	10

The video stream would then be configured in the DSI to be twice the line length of the normal image.

Copyright © 2026 Texas Instruments Incorporated

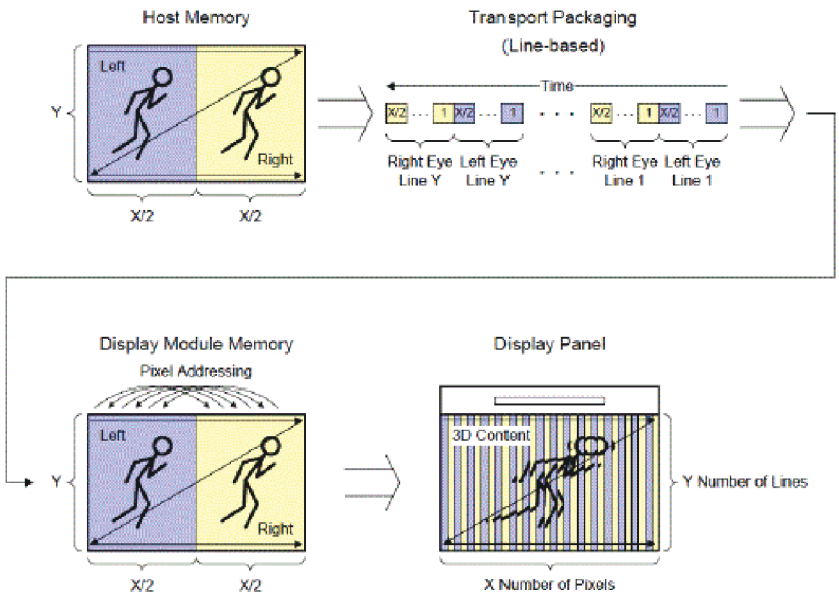


Figure 12-613. Example DPI Image Format for SDF Combining Left And Right Pixels

Note: The vertical operation DSI controller will expect the system to provide the left and right as two individual frames with SYNC to allow the blanking to be generated. The DSI does not insert the vertical blanking area when VSYNC = 0.

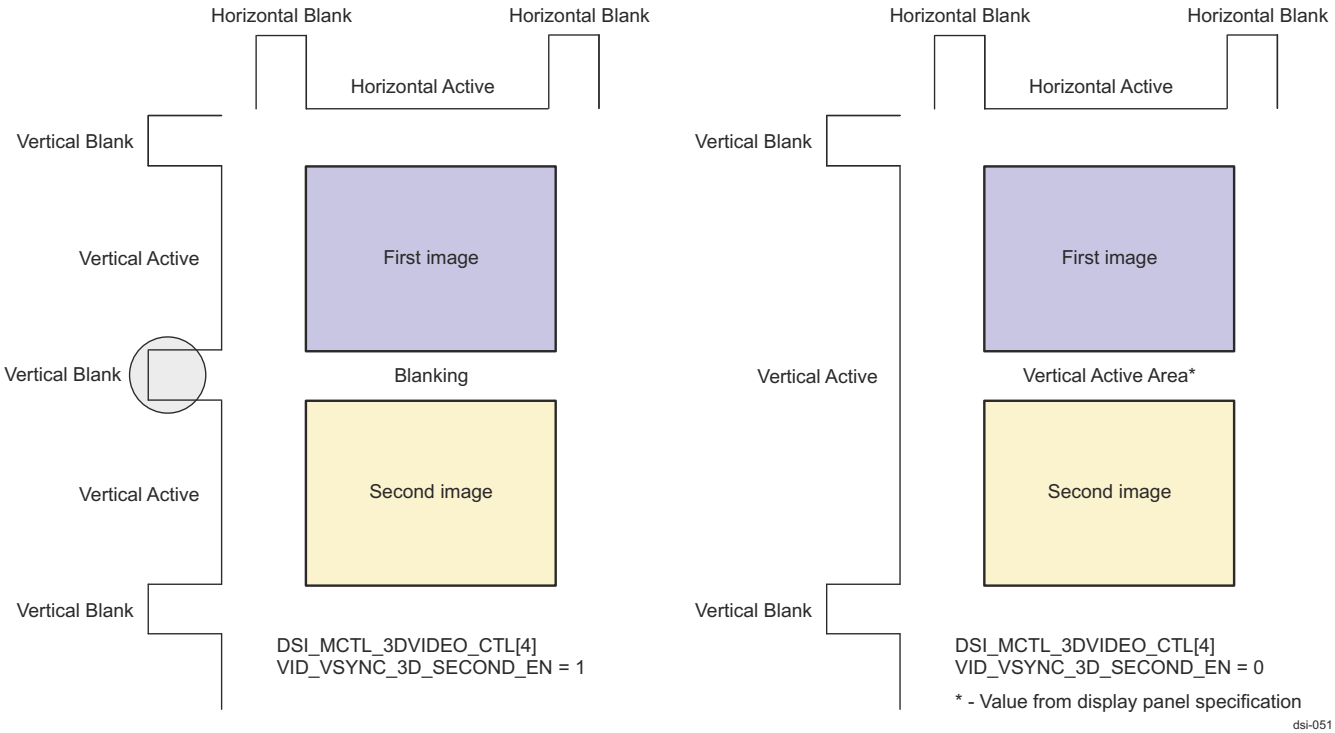


Figure 12-614. 3D Image format for 3DVSYNC splitting first and second images

Copyright © 2026 Texas Instruments Incorporated

12.9.2.7.10 DSITX Video Stream Variable Refresh

The DSITX Controller can be programmed to allow the video stream sequence from a video source to halt between frames and enter Low Power state. The DPI video stream can halt before the new VSYNC is sent and remain in this state until the host video system wants to begin transmitting the new frame.

The DSITX will not perform the normal frame recovery mechanism when this is enabled, however it will continue to recover from line errors until the end of the frame.

Table 12-501. DSITX Video Stream Variable Refresh

Variable refresh rate operation enable control	Configures how the video generation timing is controlled.0, 1 DPI mode operation supports = 1 (enabled). For SDI operation, this parameter can optionally be set to FALSE (= 0) when the system clock and tx_byte_clk are generated from the same reference clock source, or TRUE to allow variable refresh rate control.

Copyright © 2026 Texas Instruments Incorporated

Chapter 13

On-Chip Debug

This chapter describes the on-chip debug support.

13.1 On-Chip Debug Overview 2463
13.2 On-Chip Debug Features 2463
13.3 On-Chip Debug Functional Description 2464

Copyright © 2026 Texas Instruments Incorporated

13.1 On-Chip Debug Overview

This chapter describes the properties and capabilities of the various features available through the On-Chip Debug framework deployed on this device.

The On-Chip Debug framework enables various Debug and Trace use-cases, including:

- **JTAG Tooling** – Access to on-chip debug resources is supported by an IEEE 1149.1 (JTAG) compliant interface that is supported by an Arm® CoreSight™ DAP JTAG-DP.
- **Self-Hosted Tooling** – code running on programmable cores within the device is able to use on-chip debug resources to enable embedded tooling solutions.
- **PCB-level interconnect testing** – IEEE 1149.1 and IEEE 1149.6 compliant Boundary Scan supports product level integration testing
- **Low Power Operation** – When on-chip JTAG and debug management logic is in a low power state, activity sensed on the JTAG interface will cause this logic to be placed in a functional state.
- **Stop Mode debugging** – Debug of embedded processors is supported using various mechanisms that can halt the pipeline of a CPU. Breakpoints (Software and Hardware), Watchpoints, Cross-Triggering, and On-demand (e.g. user requested) halt request mechanisms may be supported based on the capabilities of a given processor.
- **Debug-aware Peripherals** – Peripheral awareness of processor execution state allows safe suspension of peripheral operation. Supported by select peripherals.
- **Synchronized Debug** – Wide deployment of Cross-Triggering allows multiple processors and/or debug elements to be grouped together to process various actions based on a common event occurrence.
- **Processor Trace** – Support for the generation of a trace stream with the encoding of processor state that may include some combination of program flow, timing details (execution and stall), and memory references (address and/or data) with the goal of facilitating processor state reconstruction for debug purposes.
- **Software Messaging Trace** – Support for software messaging trace where embedded code running within the device can be instrumented to use memory writes to send important debug information to a trace stream.
- **System Traffic Trace** – Strategically selected points in the SoC topology are instrumented with trace-capable bus probes that support transaction trace, read-latency monitoring, and throughput computation.
- **Trace Correlation (through timestamping)** – Support for the correlation of different trace streams is enabled through the use of a common global timestamp that is distributed to supported trace sources.
- **Trace data movement** – Trace data movement on chip is supported using standard Arm ATB trace infrastructure components. Concurrent use of the trace bus by multiple trace sources is supported, with each trace source identifiable through a unique ID. An Arm® CoreSight™ Trace Router supports sending a trace stream off-chip (TPIU), to dedicated memory on-chip (TBR), or broadcasted to both (TPIU + TBR).
- **On-Chip trace collection via dedicated buffer** – An on-chip trace buffer is supported by logic that implements capturing trace data until either the memory fills (stop-on-full, system-bridge) or continuously until a request to stop is received (circular buffer). Interleaving of multiple trace streams is made possible through the use of a standardized encoding that embeds trace data along with the corresponding trace source ID.
- **Trace export over TPIU** – Trace data is exported over device LVCMOS pins using a standard protocol that embeds trace data along with the corresponding trace source ID.

CAUTION

Debug features that are intrusive in nature may affect the ability of the system to meet critical real-time deadlines and should be used with caution.

13.2 On-Chip Debug Features

The On-Chip Debug framework provides a comprehensive hardware platform for a rich debug and development experience. The On-Chip Debug framework supports these features:

- An IEEE 1149.1 (JTAG and Boundary Scan) compliant device interface
- Cross-triggering and Debug boot mode device interface
- Trace port device interface
- Advanced power, reset, and clock management to facilitate debug across complex low power modes

Copyright © 2026 Texas Instruments Incorporated

- Breakpoint-based debug
- Cross-triggering
- Program flow trace
- System traffic monitoring trace
- Software instrumentation trace
- Trace export via trace port device interface
- Trace capture on-chip via dedicated buffer with options to support stop-on-full, circular-buffer, or data hand-off to various peripherals
- Arm® CoreSight™ compliant debug components deployed to streamline 3rd party tooling support

Note

Some features may not be available. See *Module Integration* for more information.

13.3 On-Chip Debug Functional Description

13.3.1 On-Chip Debug Block Diagram

A logical partitioning of the On-Chip Debug features deployed on this device is illustrated in Figure 13-1.

Copyright © 2026 Texas Instruments Incorporated

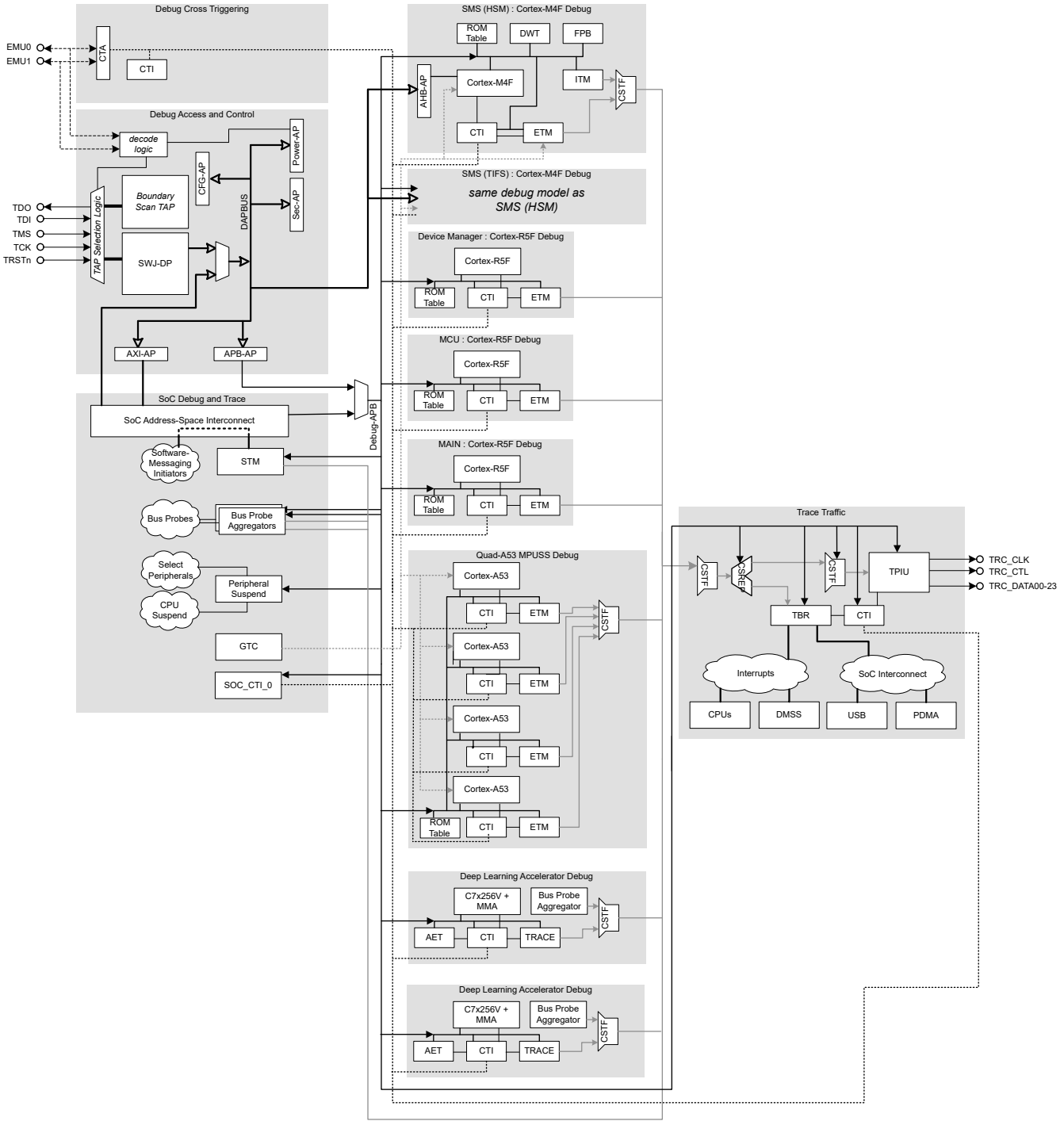


Figure 13-1. On-Chip Debug Block Diagram

13.3.2 Device Interfaces

On-Chip Debug features are supported through three device interfaces:

- **JTAG**: IEEE 1149.1 compliant interface that provides access to Boundary Scan and acts as the primary interface for off-chip access to On-Chip debug resources (See Section 13.3.2.1)
- **Trigger and Debug Boot Mode**: Multi-functional interface that supports product level cross-triggering and debug-related boot modes (See Section 13.3.6, Section 13.3.4)

Copyright © 2026 Texas Instruments Incorporated

- **Trace Port**: Arm TPIU compliant Trace Port interface is used to facilitate export of trace (See Section 13.3.12)

Texas Instruments supports a variety of eXtended Development System (XDS™) JTAG controllers with various debug capabilities beyond only JTAG support. The following document is a good reference for guidelines: Emulation and Trace Headers. More information can also be found here:XDS Target Connection Guide .

The previous link connects to TI community resources. Linked contents are provided “AS IS” by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

13.3.2.1 JTAG Interface

Table 13-1. JTAG Interface Signals

SignalName	I/OType(1)	Description
TRSTn	I	*Test Reset*. Initializes and disables the test interface.
TCK	I	*Test Clock*. Controls the timing of the test interface independently from any system clocks. TCK is pulsed by the equipment controlling the test and not by the tested device.
TMS	I	*Test Mode Select*. Controls the transitions of the test interface state machine.
TDI	I	*Test Data Input*. Supplies the data to the JTAG registers.
TDO	O/Z	*Test Data Output*. Used to serially output the data from the JTAG registers to the equipment controlling the test.

(1) I = Input; I/O = Input or output; O = Output; O/Z = Output or three-state

13.3.2.2 Trigger and Debug Boot Mode Interface

Table 13-2. Trigger and Debug Boot Mode Interface Signals

Signal Name	I/O Type(1)	Description
EMU0	I/O	Channel 0 trigger or boot mode select
EMU1	I/O	Channel 1 trigger or boot mode select

(1) I = Input; I/O = Input or output; O = Output; O/Z = Output or three-state

13.3.2.3 Trace Port Interface

Table 13-3. Trace Port Signals

Pin Name	TPIU Signal Name	I/O Type(1)	Description
TRC_CLK	TRACECLK	O	TraceClock
TRC_CTL	TRACECTL	O	TraceControl
TRC_DATA[*n*:0]	TRACEDATA[*n*:0]	O	TraceData [*n*:0]

(1) I = Input; I/O = Input or output; O = Output; O/Z = Output or three-state

Note

The Trace Port interface signals are associated with device pins that are multiplexed with other signal functions.

13.3.3 Debug and Boundary Scan Access and Control

On-Chip debug resources are made available through three mechanisms:

- JTAG access via DAP and related APs
- JTAG access via Boundary Scan TAP
- Embedded access to debug resources via SoC address space

DAP

Copyright © 2026 Texas Instruments Incorporated

Off-chip debug tools are able to access On-Chip debug resources via the JTAG interface when not in Boundary Scan mode (see below). A CoreSight™ Compliant DAP architecture provides access via a DP and a collection of APs:

- SWJ-DP: Arm® CoreSight™ compliant SWJ Debug Port provides support for a JTAG interface with a 4-bit IR

Note

Even though an SWJ-DP is implemented on-chip, only JTAG is supported. This device does not support SWD.

- APB-AP: Arm® CoreSight™ APB Access Port provides access to the Debug-APB address space which is the primary configuration space for On-Chip Debug resources.
- AXI-AP: Arm® CoreSight™ AXI Access Port provides access to the SoC address space, allowing visibility and control over system resources.
- Config-AP: TI Configuration AP supports access to SoC debug management registers
- Power-AP: TI Configuration AP supports advanced power, reset, and clock management. (See *Power, Reset, Clock Management*)
- SEC-AP: TI AP supports debug authentication

Table 13-4 describes the APSEL assignment for this device.

Table 13-4. DAP APSEL Assignment

APSEL	AP	Description
0	Config-AP	TI AP - Reserved
1	APB-AP	Provides access to Debug Config Plane
2	AXI-AP	Provides access to SoC Address Space
3	Power-AP	TI AP - Extended Power/Reset/Clock Control
4-5	Reserved	Reserved for future use
6	SEC-AP	TI AP - Authentication Interface
7	SMS TIFS M4F	
8	Reserved	Reserved for future use
9	SMS HSM M4F	
10-31	Reserved	Reserved for future use

Boundary Scan

This device supports boundary scan using an IEEE 1149.1 compliant JTAG TAP that is made visible through the use of a compliance-enable mode (see *Debug Boot Modes and Boundary Scan Compliance*). IEEE 1149.1 and 1149.6 Boundary Scan support are defined in device-specific BSDL files that can be found in the respective device's product folder on ti.com.

SoC Address Space

The device architecture provides access to On-Chip debug resources through the SoC Address Space. This includes access to all DAP Access Ports (APs) as well as the Debug-APB address space.

13.3.4 Debug Boot Modes and Boundary Scan Compliance

The EMU1 and EMU0 device pins are used to convey debug boot mode behavior and can also be used as part of a boundary scan compliance sequence to enable boundary scan features.

Debug Boot Mode

EMU1 and EMU0 are sampled when MCU_PORz is de-asserted and the decoded value determines the debug boot mode behavior as detailed in Table 13-5.

Copyright © 2026 Texas Instruments Incorporated

Table 13-5. Debug Boot Modes

Sample Time	EMU1	EMU0	Boot Mode	Behavior
MCU_PORz deassertion	0	0	Reserved	Reserved for future use
	0	1	Reserved	Reserved for future use
	1	0	Wait-In-Reset (WIR)	Device will remain quiescent until a debug connection is established and the debugger releases WIR. This mode is useful for debugging boot sequences.
	1	1	Normal	Device boots normally. Debug can connect at any time but the context will be post-boot.

Boundary Scan Compliance

EMU1 and EMU0 are sampled when TRSTz is de-asserted and the decoded value determines the debug boot mode behavior as detailed in Table 13-6.

Table 13-6. Boundary Scan Compliance

Sample Time	EMU1	EMU0	Boundary Scan Compliance	Behavior
TRSTn deassertion	0	1	Boundary Scan Enabled	The device's Boundary Scan TAP is connected to the device's JTAG interface.
	0	0		
	1	0	Boundary Scan Disabled	The device's SWJ-DP is connected to the device's JTAG interface.
	1	1		

13.3.5 Power, Reset, Clock Management

This device includes advanced power, reset, and clock management debug capabilities, including:

- **Wakeup support for debug logic**: logic within the device is able to sense JTAG activity and ensure that debug logic is on and able to service JTAG requests.
- **Reset isolation**: critical configuration and trace datapaths and logic are not sensitive to warm reset
- **Configuration independence**: debug configuration occurs over a debug-only interconnect, separate from SoC traffic to ensure debug logic remains available even during deadlock scenarios.
- **Power-AP**: a CoreSight™ compliant Access Port (AP) developed by TI that provides a standard interface for debug tooling to access status and control over power, reset, and clocking for the system and various LPSC-defined sub-domains.

13.3.6 Debug Cross Triggering

This device supports an Arm® CoreSight™ compliant four-channel programmable on-chip Cross Triggering network. In addition to the four-channel on-chip network, this device implements two channels of product level triggering via the EMU0 and EMU1 device pins.

Conceptually, each channel of Cross Triggering can be viewed as mapping of a user-defined set of events to a user-defined set of actions, where the occurrence of any event in the set-of-events results in the generation of the set-of-actions. Table 13-7 provides a domain-level summary of the supported events and actions.

Table 13-7. Domain-Level Summary of Triggering Capability

Domain	Events	Actions
Product-Level	Zero detected on EMU0 input pin	EMU0 output pin driven to Zero
	Zero detected on EMU1 input pin	EMU1 output pin driven to Zero
SoC Debug	TBR Acquisition Complete	TPIU insert trigger packet
	TBR Embedded buffer is full	TPIU start flush process
	System reset asserted	TBR insert trigger packet
	Bus Probe-n match	TBR start flush process
	STM write to a TRIG location	Bus Probe-n Start
	STM write to a trigger-enabled stimulus port	Bus Probe-n Stop

Copyright © 2026 Texas Instruments Incorporated

Table 13-7. Domain-Level Summary of Triggering Capability (continued)

Domain	Events	Actions
R5FSS0	R5FSS0 has halted	R5FSS0 – halt request
	PMU generated interrupt	R5FSS0 – resume request
	ETM Trigger outputs	ETM Trigger inputs
MCU_R5FSS0	MCU_R5FSS0 has halted	MCU_R5FSS0 - halt request
	PMU generated interrupt	MCU_R5FSS0 - resume request
	ETM External Out (EXTOUT) trigger	ETM External In (EXTIN) trigger
WKUP_R5FSS0	WKUP_R5FSS0 has halted	WKUP_R5FSS0 – halt request
	PMU generated interrupt	WKUP_R5FSS0 – resume request
	ETM External Out (EXTOUT) trigger	ETM External In (EXTIN) trigger
A53SS0	A53SS0 core-n has halted	A53SS0 core-n – halt request
	PMU generated interrupt	A53SS0 core-n – resume request
	ETM generated trigger	CTI interrupt
	--	ETM External In (EXTIN) trigger
C7XSS0/1	C7XSS0 core has halted	C7XSS0 core - halt request
	--	C7XSS0 core - resume request
	C7XSS0 AET External Out trigger	C7XSS0 AET External In trigger

13.3.7 Peripheral Suspend

Some peripherals on the device include support for a debug suspend capability that allows the peripheral's behavior to alter when an assigned CPU enters the debug suspend state. To use this feature, it must be enabled within the peripheral and a connection made between the desired suspend source (CPU) and the suspend destination (peripheral) of interest.

Details on the debug suspend feature for a specific peripheral, including how to enable it, is covered in the associated peripheral section.

On this device, there is a dedicated interrupt router (INTR), DBGSUSPENDROUTER0, that supports the routing of a suspend source to a suspend destination. Details on the assignment of DBGSUSPENDROUTER0 inputs and outputs are provided in the following tables.

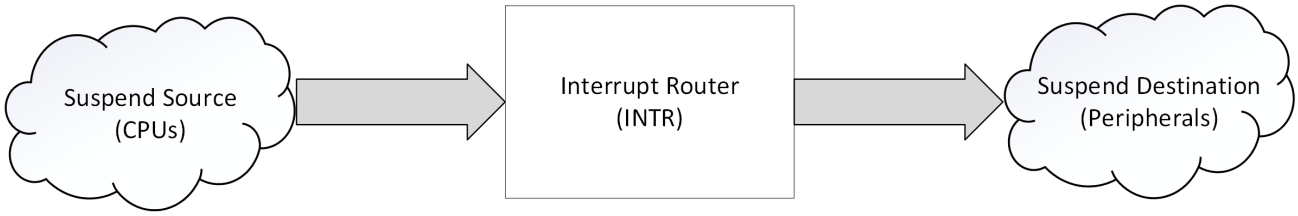


Figure 13-2. Peripheral Suspend Support

Table 13-8. Suspend Source Assignment

Index	Suspend Source	Description
1	A53SS0_0	A53SS0 Core 0
2	A53SS0_1	A53SS0 Core 1
3	A53SS0_2	A53SS0 Core 2
4	A53SS0_3	A53SS0 Core 3
9	C7X256V0	C7X256V0 Core 0
10	C7X256V1	C7X256V1 Core 0

Copyright © 2026 Texas Instruments Incorporated

Table 13-8. Suspend Source Assignment (continued)

Index	Suspend Source	Description
22	R5FSS0	R5FSS0 CPU
23	WKUP_R5FSS	Device Management Cortex R5F
24	MCU_M4FSS	MCU M4F
25	SMS0_HSM	SMS HSM

Table 13-9. Suspend Destination Assignment

Index	Suspend Destination	Description
0	WKUP_GTC0	Global Time Counter
1	CPSW0	Ethernet Switch (SoC)
2	DMSS	DMA SS (SoC)
3	DMSS_HSM	SA3SS DMSS_HSM
4	DMSS_CSI	DMA SS (CSI-RX)
7-5	Reserved	Reserved
8	TIMER0	Dual Mode Timer (Main)
9	TIMER1	
10	TIMER2	
11	TIMER3	
12	TIMER4	
13	TIMER5	
14	TIMER6	
15	TIMER7	
16	MCU_TIMER0	Dual Mode Timer (MCU)
17	MCU_TIMER1	
18	MCU_TIMER2	
19	MCU_TIMER3	
20	Reserved	Reserved
21	WKUP_TIMER0	Dual Mode Timer (Wakeup)
22	WKUP_TIMER1	
23	Reserved	Reserved
24	EPWM0	Enhanced PWM (Main)
25	EPWM1	
26	EPWM2	
27	Reserved	Reserved
28	MCAN0	MCAN (Main)
29	MCU_MCAN0	MCAN (MCU)
30	MCU_MCAN1	
31	MCAN1	MCAN (Main)
32	MCRC64_0	CRC (Main)
33	MCU_MCRC64_0	CRC (MCU)
34	Reserved	Reserved
35	I2C0	I2C (Main)
36	I2C1	
37	I2C2	
38	I2C3	
39	MCU_I2C0	I2C (MCU)

Copyright © 2026 Texas Instruments Incorporated

Table 13-9. Suspend Destination Assignment (continued)

Index	Suspend Destination	Description
40	WKUP_I2C0	I2C (Wakeup)
41	Reserved	Reserved
42	ECAP0	ECAP (Main)
43	ECAP1	
44	ECAP2	
45	EQEP0	EQEP (Main)
46	EQEP1	
47	EQEP2	
48	I2C4	I2C (Main)
49	Reserved	Reserved
50	PDMA0	Peripheral DMA
51	PDMA1	
52	PDMA2	
53	PDMA3	
54	RTI2	Real-time Interrupt (A53)
55	RTI3	
56	RTI0	
57	RTI1	
58	RTI4	Real-time Interrupt (C7X256V0)
59	MCU_RTI0	Real-time Interrupt (MCU)
60	WKUP_RTI0	Real-time Interrupt (DM)
61	RTI15	Real-time Interrupt (GPU)
62	RTI5	Real-time Interrupt (C7X256V1)
63	RTI8	Real-time Interrupt (R5FSS0)
127-61	Reserved	Reserved

13.3.8 R5FSS0, MCU_R5FSS0 and WKUP_R5FSS0 Debug

The R5FSS0, MCU_R5FSS0 and WKUP_R5FSS0 includes an Arm Cortex R5F with embedded debug capability, including:

- Independent debug configuration
- ROM Table
- Basic debug functionality
- Cross Triggering
- PMU: Performance Monitor Unit
- ETM: Embedded Trace Macrocell

A summary of the R5FSS0, MCU_R5FSS0 and WKUP_R5FSS0 debug capabilities is detailed in Table 13-10.

Copyright © 2026 Texas Instruments Incorporated

Table 13-10. R5FSS0, MCU_R5FSS0 and WKUP_R5FSS0 Debug Capabilities

Capability	Feature	Notes
Basic Debug	Independent debug configuration	Debug resource configuration is performed over a configuration interface that is isolated from functional traffic
	ROM table	Facilitates discovery of debug resources within debug configuration address space
	Processor halt	Support user-requested entry into the suspended state
	Single step	Execution of a single instruction before entering the suspended state
	Software breakpoints	Software breakpoints are supported via opcode replacement
	Hardware breakpoints	Eight Debug Breakpoint resources support hardware breakpoints
	Hardware watchpoints	Eight Debug Watchpoint resources support data address breakpoints
	Core register access	Access to processor core registers
	System memory access	Access to memory from perspective of CPU
	Vector catch	Halting in response to an exception
	Arm® TrustZone® debug authentication	Provisioning for DBGEN and NIDEN
Cross Triggering	Debug state	Support for controlling execution state (run, halt) via triggers and creating triggers upon entry into debug state
	PMU	PMU interrupt trigger
	ETM	Five ETM external triggers (one ETM Trigger, two external out event triggers, two external in event triggers)
PMU	Profile Counters	Three counters can be used to count different events available for gathering statistics on the operation of the processor and memory system
	Cycle Counter	One dedicated counter available for counting CPU clock cycles
ETM	Triggering Infrastructure	Comprehensive triggering infrastructure supports use of comparators (address, data value, context ID), counters, sequencer state, an external inputs and outputs to control the enabling of trace.
	Instruction trace	Supports tracing of instruction flow
	Cycle-accurate tracing	Supports inclusion of a precise cycle count of executed instructions.
	Branch broadcast tracing	Support for tracing branch address details even in circumstances where that information might be discoverable from object code
	Data tracing	Support for data address and data value tracing

13.3.9 A53SS0 Debug

The A53SS0 is a multi-core Arm Cortex-A53 subsystem with embedded debug capability and features, including:

- Independent debug configuration
- ROM Table
- Basic debug functionality
- Cross Triggering
- PMU: Performance Monitor Unit
- ETM: Embedded Trace Macrocell

A summary of the A53SS0 debug capabilities and features is detailed in Table 13-11.

Copyright © 2026 Texas Instruments Incorporated

Table 13-11. A53SS0 Debug Capabilities

Capability	Feature	Notes
Basic Debug	Independent debug configuration	Debug resource configuration is performed over a configuration interface that is isolated from functional traffic
	ROM table	Facilitates discovery of debug resources within debug configuration address space
	Processor halt	Support user-requested entry into the suspended state
	Single step	Execution of a single instruction before entering the suspended state
	Software breakpoints	Software breakpoints are supported via opcode replacement
	Hardware breakpoints	Six Debug Breakpoint resources support hardware breakpoints
	Hardware watchpoints	Four Debug Watchpoint resources support data address breakpoints
	Core register access	Access to processor core registers
	System memory access	Access to memory from perspective of CPU
	Vector catch	Halting in response to an exception
	Arm® TrustZone® debug authentication	Provisioning for DBGEN, NIDEN, SPIDEN, SPNIDEN, SUIDEN, and SUNIDEN
Cross Triggering	Debug state	Support for controlling execution state (run, halt) via triggers and creating triggers upon entry into debug state
	PMU	PMU interrupt trigger
	ETM	Four ETM external triggers
	CPU	CTI interrupt into CPU
PMU	Profile Counters	Six counters can be used to count different events available for gathering statistics on the operation of the processor and memory system
	Cycle Counter	One dedicated counter available for counting CPU clock cycles
ETM	Instruction trace with cycle counting	Support for tracing instruction execution with timing information
	Branch broadcast tracing	Support for tracing branch address details even in circumstances where that information might be discoverable from object code
	Return stack tracing	Support for tracing of the return stack
	Stall Control	Support for stalling the A53 processor to avoid ETM overflows
	Global timestamping	Support for attaching a global timestamp to trace traffic

13.3.10 C7XSS0 and C7XSS1 Debug

The C7XSS0 and C7XSS1 are a single-core C7xV-256 subsystem with embedded debug capability and features, including:

- Independent debug configuration
- Basic debug functionality
- Cross Triggering
- AET: Advanced Event Triggering
- Processor Trace
- Traffic Monitoring Trace

A summary of the C7XSS0 and C7XSS1 debug capabilities and features is detailed in the following table.

Copyright © 2026 Texas Instruments Incorporated

Table 13-12. C7XSS0 and C7XSS1 Debug Capabilities

Capability	Feature	Notes
Basic Debug	Independent debug configuration	Debug resource configuration is performed over a configuration interface that is isolated from functional traffic
	ROM table	Facilitates discovery of debug resources within debug configuration address space
	Processor halt	Support user-requested entry into the suspended state
	Single step	Execution of a single instruction before entering the suspended state
	Software breakpoints	Software breakpoints are supported via opcode replacement
	Hardware breakpoints	Four Debug Breakpoint resources support hardware breakpoints
	Core register access	Access to processor core registers
	System memory access	Access to memory from perspective of CPU
	Arm TrustZone debug authentication	Provisioning for DBGEN, NIDEN, SPIDEN, SPNIDEN, SUIDEN, and SUNIDEN
Cross Triggering	Debug state	Support for controlling execution state (run, halt) via triggers and creating triggers upon entry into debug state
	AET	Two channels of AET external triggers (input and output)
AET	Profile Counters	Two counters can be used to count different events available for gathering statistics on the operation of the processor and memory system. Includes watermark support.
	Application Context Sensitive Triggering	Supports generation of events based on the context of application execution (e.g. Thread-ID, Virtual Machine ID, Program bus activity, Memory bus activity)
	Visibility and detection of events	provided by CPU, Subsystem, and SOC
	Range detection	of Program and Memory bus activity
	Programmable State Machine	supports complex, state-based event detection
	Trace Stream Filtering / Windowing	supports the management of trace windows
Trace	Historical Record	Creates a customized (based on trace stream filtering/ windowing) historical account of the CPU's behavior through the use of different types of trace streams
	Program Flow Trace	records program flow execution, including discontinuities resulting from branches, interrupts, or exceptions
	Cycle Accurate Trace	records cycle-accurate timing information
	Memory Reference Trace	records data log activity (address, data, address and data)
	Event Trace	records information about the occurrence, duration, or relationship of one or more events
	Stream Buffer Parameter Trace	records the parameters used to start a stream buffer operation allowing off-line reconstruction
	Application Context Trace	records changes in the context of the application (virtual machine ID, thread-ID)
	Stall Control	Support for stalling the C7X processor to avoid Trace overflows
	Global timestamping	Support for attaching a global timestamp to trace traffic
	Traffic Monitoring	Traffic monitoring bus probes are deployed within C7XSS0 memory subsystem

13.3.11 SoC Debug and Trace

This device includes debug capabilities deployed at the system level, including:

Copyright © 2026 Texas Instruments Incorporated

- Software messaging trace
- Debug-aware peripherals
- Traffic monitoring with bus probes
- Global timestamping for trace

More details for each of these capability areas can be found in the corresponding sections below.

13.3.11.1 Software Messaging Trace

Software messaging trace is supported on this device by an MIPI STP-V2 compliant Arm® CoreSight™ STM with supporting logic that maps initiator IDs to specific STP Major Source IDs. The following device initiators support software messaging:

- A53SS0 cores
- C7XSS0
- C7XSS1
- R5FSS0
- MCU_R5FSS0
- WKUP_R5FSS0
- SMS0_HSM
- SMS0_TIFS
- DMSS BCDMA
- DMSS PktDMA
- DAP AXI-AP

13.3.11.2 Debug-Aware Peripherals

Select peripherals support a debug feature that allows them to react to the debug state of a controlling processor. For instance, a timer peripheral that is allocated to a particular processor could be configured to stop counting when the associated processor is in the halted state.

This device includes programmable support for shared peripherals that allows the developer to select the processor whose debug state a given peripheral should receive. Peripherals that are debug aware include programmable support to enable or disable sensitivity to debug.

A list of the processors and peripherals that support this debug model can be found in Table 13-13.

Table 13-13. Debug-Aware Peripheral Support

Supported Processors	Supported Peripherals
A53SS0 cores	GTC
C7XSS0	CPSW3G
C7XSS1	DMSS
R5FSS0	SA3SS
MCU_R5FSS0	TIMER instances
WKUP_R5FSS0	EPWM instances
SMS0_HSM	MCAN instances
SMS0_TIFS	I2C instances
--	MCRC64 instances
--	ECAP instances
--	EQEP instances
--	PDMA_SPI
--	PDMA_UART
--	PDMA_MCASP
--	RTI/WWDT instances(1)

(1) RTI/WWDT suspend is associated with the processor that it is allocated to

Copyright © 2026 Texas Instruments Incorporated

13.3.11.3 Traffic Monitoring With Bus Probes

Traffic monitoring bus probes are deployed at strategic points in the system interconnect and support the following capabilities:

- MIPI STP-V2 compliant trace messaging support with global timestamping support
- Latency Statistics Collection: Statistics on the timeliness of responses to read requests are collected during a sampling window and then read by a profiling entity (application thread or encoded into a trace message)
- Throughput Statistics Collection: Statistics on the amount of data moving across an interface are collected during a sampling window and then read by a profiling entity (application thread or encoded into a trace message)
- Transaction Trace: Information about specific transactions occurring at a probed location are captured and encoded into a trace message
- Complex filtering support allows discrimination of traffic being monitored
- Cross-Trigger support for enabling and disabling probe monitoring

Table 13-14 details the locations in the system interconnect that are probed:

Table 13-14. Bus Probe Points

Probe Point	Bus Monitoring Details
A53SS0 Write Initiator	Provides visibility to all write requests initiated by A53SS0
A53SS0 Read Initiator	Provides visibility to all read requests initiated by A53SS0
A53SS0 ACP Target	Monitors traffic to the A53SS0 ACP port
C7XSS0 Deep Learning Accelerator MDMA Initiator	Monitors traffic initiated by DLA MDMA
C7XSS0 Deep Learning Accelerator SDMA Target	Monitors traffic to DLA SDMA port
C7XSS1 Deep Learning Accelerator MDMA Initiator	Monitors traffic initiated by DLA MDMA
C7XSS1 Deep Learning Accelerator SDMA Target	Monitors traffic to DLA SDMA port
DDR Target	Monitors RT and Non-RT traffic to DDR
GMPC Target	Monitors traffic to GPMC
FSS Target	Monitors traffic to FSS0
DSS and CSI DMA Initiator	Provides visibility to all requests initiated by DSS and CSI DMA instances
JPEG Encoder Read Initiator	Provides visibility to all read requests initiated by JPEG Encoder
JPEG Encoder Write Initiator	Provides visibility to all write requests initiated by JPEG Encoder
OCSRAM 2 Bank 1 – future use	Monitors traffic to OCSRAM Bank 1 (future use)
OCSRAM 2 Bank 2 – future use	Monitors traffic to OCSRAM Bank 2 (future use)
PCIe Target	Monitors traffic to PCIe Target interface
Video Encoder/Decoder Initiator	Provides visibility to all requests initiated by Video Encoder/Decoder
VPAC LDC Initiator	Monitors requests initiated by VPAC LDC initiator
VPAC Real-Time Initiator	Monitors requests initiated by VPAC RT initiator
VPAC Non-Real-Time Initiator	Monitors requests initiated by VPAC NRT initiator
VPAC Target	Monitors traffic to VPAC
Peripheral CBASS Targets	Monitors traffic to these targets: ECAP# EQEP# I2C# MCAN# PDMA_SPI PDMA_UART SPI# UART#

Copyright © 2026 Texas Instruments Incorporated

Table 13-14. Bus Probe Points (continued)

Probe Point	Bus Monitoring Details
McASP CBASS Targets	Monitors traffic to these targets: EPWM# MCASP#
WKUP CBASS Targets	Monitors traffic from MAIN to WKUP, including these targets: WKUP_GTC WKUP_R5F WKUP_TIMER# WKUP_RTI WKUP_I2C0 WKUP_CTRL_MMR0 WKUP_VTM0
MCU CBASS Targets	Monitors traffic from MAIN to MCU, including these targets: MCU_R5F MCU_SPI# MCU_MCRC64 MCU_UART0 MCU_MCAN# MCU_TIMER# MCU_DCC0 MCU_RTI0 MCU_I2C0
Infrastructure CBASS Targets	Monitors traffic to these targets: CMP_EVENT_INTROUTER CTRL_MMR DCC# DDPA DFTSS ESM PSRAM# GPIO# EFUSE GPIOMUX_INTROUTER PADCFG_CTRL_MMR0 PLL_MMR0 PLLCTRL PSC TIMESYNC_EVENT_ROUTER

13.3.11.4 Global Timestamping for Trace

Support for a global timestamp for use by debug is enabled by a 48-bit continuous timestamp managed by the device's GTC. This 48-bit global timestamp is readable by embedded software, via the GTC, and distributed across the device to various trace sources where it is used as a time reference when timestamping trace streams. Trace sources that support global timestamping, include:

- A53SS0 cores: ETM
- C7XSS0: Trace
- C7XSS1: Trace
- SMS0_HSM: ITM and ETM
- SMS0_TIFS: ITM and ETM
- STM

Copyright © 2026 Texas Instruments Incorporated

- Bus Probes

13.3.12 Trace Traffic

Trace traffic originates from a trace source, is distributed across the device using Arm® CoreSight™ compliant trace infrastructure components, and reaches one of two possible trace sinks.

13.3.12.1 Trace Sources

A summary of the trace sources present on this device is included in Table 13-15.

Table 13-15. Trace Sources

Domain	Trace Source
SoC Debug	Bus Probes
	STM
R5FSS0	ETM
MCU_R5FSS0	ETM
WKUP_R5FSS0	ETM
SMS0_HSM	ETM, ITM / DWT
SMS0_TIFS	ETM, ITM / DWT
A53SS0	ETM (each core)
C7XSS0	Processor Trace
C7XSS1	Processor Trace

13.3.12.2 Trace Infrastructure

Trace distribution is accomplished using standard Arm® CoreSight™ compliant trace infrastructure components:

- CoreSight™ Trace Funnels (CSTF): Non-programmable CSTFs are used at points of interleaving where multiple trace sources converge and form a single stream of trace traffic. This device includes one instance of a programmable CSTF that is deployed immediately before the TPIU.
- CoreSight™ Trace Replicator (CSREP): A programmable CSREP is used as a routing device, that can be used to forward trace traffic, based on its ID, to one, both, or none of the device trace sinks.

13.3.12.3 Trace Sinks

Two trace sinks are supported on this device:

- Arm® CoreSight™ TPIU: TPIU supports export of trace off-chip via LVCMOS device pins (See 1.3.3.3) for capture by an external receiver.
- TI Trace Buffer Router (TBR) with 64KB of storage: the TBR can function as a trace buffer or as a system bridge. As a trace buffer, the TBR can be setup to capture trace data until the internal buffer fills or it can operate as a circular buffer that will capture continuously. When configured as a system bridge the TBR's storage becomes an elastic buffer that supports the concurrent queueing and dequeuing of trace traffic. The system bridge mode supports interrupt and event notification capabilities that support integration with device level CPUs and/or DMAs to support a variety of use cases, including, for e.g., relocation of trace data to DDR and off-chip export over USB.

13.3.13 Application Support

TI includes a set of system level debug facilities in the Debug Subsystem of devices known as Chip Tools (CTools). For easy integration of A53 processor trace and system trace into applications, a set of libraries commonly referred to as CToolsLib are provided. CToolsLib is a collection of embedded target APIs/libraries focused on enabling easy access to the CTools. CToolsLib encompasses the following libraries to assist in trace integration:

- STM Library
- CP Tracer Library
- TBR Library

Copyright © 2026 Texas Instruments Incorporated

- ETM Library

For more information about these libraries, downloadable files, and other useful links, please visit the CToolsLib Wiki site.

The previous link connects to TI community resources. Linked contents are provided “AS IS” by the respectivecontributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

Copyright © 2026 Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from November 22, 2025 to August 19, 2026 (from Revision C (November 2025) to Revision D (August 2026)) **Page**

- Added trademark to C7™ DSP reference ... 14
- Added section *QoS Summary Table* ... 78
- Added section *Route ID* ... 82
- Added section *ISC and Firewall* ... 83
- Updated the GPIO1 interrupts not being supported to [143:72] in section *GPIO Unsupported Features*, as GPIO1[71:52] interrupts are supported ... 211
- Added section *DCC Input Source Clock Mapping* ... 320
- Updated BOOTMODE Pin Mapping table ... 362
- Updated Primary Boot Mode Selection and Configuration section ... 362
- Updated sections *PSC0 LPSC Features* and *WKUP_PSC0 LPSC Features* ... 418
- Updated CSI_RX and data interface connections in figure *VPAC Overview* ... 633
- Removed references to Noise Filter (NF) from section *Vision Pre-processing Accelerator (VPAC)* ... 633
- Added section *VPAC Subsystem Block Diagram* ... 637
- Updated tables *VPAC Subsystem VISS Interrupt Events*, *VPAC Subsystem NF Interrupt Events*, *VPAC Subsystem VISS Interrupt Events*, *VPAC Subsystem UTC Error Interrupt Events*, and *VPAC Subsystem UTC TR Complete Interrupt Events* ... 640
- Updated table *VPAC Subsystem CTSET Event List* ... 646
- Updated figure *VISS On-the-fly Operation* and changed UDMA to UTC in section *VISS On-the-fly Processing* ... 654
- Updated figure *VISS Memory to Memory Operation* and changed UDMA to UTC in section *VISS Memory to Memory Image Processing* ... 656
- Added section *Chromatic Aberration Correction* ... 658
- Added section *VISS Blanking Requirements* ... 660
- Updated FCFA_CFG_ERR_INTR and RAWFE_CFG_ERR_INTR information; Added FCC_OUTIF_OVF_INTR interrupt in section *Interrupts Merging* ... 661
- Added section *RAWFE Not Supported Features* ... 666
- Added section *RAWFE Integration in VISS* ... 667
- Added section *RAWFE Memory Map* ... 668
- Updated table *RAM ECC Capabilities* ... 669
- Updated LSC_CFG_ERR and H3A_BUF_OVERFLOW_PULSE descriptions, and added DPC_STATS_READ_ERR interrupt to table *RAWFE Interrupts* ... 669
- Added section *RAWFE Defective Pixel Correction (DPC) for 4x4 RGBIR CFA* ... 677
- Removed duplicated information from section *RAWFE LSC Features Supported*. ... 679
- Added section *RAWFE LSC Gain Map Upsampling in 2x2 Mode (LSC CHN_MODE = 0)* ... 680
- Added section *RAWFE LSC Gain Map Upsampling in 4x4 Color Mapping Mode (LSC CNH_MODE = 1 or 2)* 681
- Added VISS prefix to section *VISS Pattern Conversion and IR Demosaicing (PCID) Module* title and moved section location ... 699
- Added section *VISS Chromatic Aberration Correction (CAC) Module* ... 701

Copyright © 2026 Texas Instruments Incorporated

- Added section *BCDMA Event Triggers* 823
- Updated the interrupt mapping tables in section *Interrupt Sources* 833
- Fix statement regarding Case 1 and Case 4..... 1762

Copyright © 2026 Texas Instruments Incorporated

This page intentionally left blank.

Copyright © 2026 Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025