

OMAP5910 Dual-Core Processor Universal Serial Bus (USB) and Frame Adjustment Counter (FAC) Reference Guide

Literature Number: SPRU677
October 2003



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DSP	dsp.ti.com
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital Control	www.ti.com/digitalcontrol
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2003, Texas Instruments Incorporated

Read This First

About This Manual

This document describes the universal serial bus (USB) and frame adjustment counter (FAC) host of the OMAP5910 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5910 Dual-Core Processor MPU Subsystem Reference Guide (literature number SPRU671)

OMAP5910 Dual-Core Processor DSP Subsystem Reference Guide (literature number SPRU672)

OMAP5910 Dual-Core Processor Memory Interface Traffic Controller Reference Guide (literature number SPRU673)

OMAP5910 Dual-Core Processor System DMA Controller Reference Guide (literature number SPRU674)

OMAP5910 Dual-Core Processor LCD Controller Reference Guide (literature number SPRU675)

OMAP5910 Dual-Core Processor Universal Asynchronous Receiver/Transmitter (UART) Devices Reference Guide (literature number SPRU676)

OMAP5910 Dual-Core Processor Universal Serial Bus (USB) and Frame Adjustment Counter (FAC) Reference Guide (literature number SPRU677)

OMAP5910 Dual-Core Processor Clock Generation and System Reset Management Reference Guide (literature number SPRU678)

OMAP5910 Dual-Core Processor General-Purpose Input/Output (GPIO) Reference Guide (literature number SPRU679)

OMAP5910 Dual-Core Processor MMC/SD Reference Guide (literature number SPRU680)

OMAP5910 Dual-Core Processor Inter-Integrated Circuit (I2C) Controller Reference Guide (literature number SPRU681)

OMAP5910 Dual-Core Processor Timer Reference Guide (literature number SPRU682)

OMAP5910 Dual-Core Processor Inter-Processor Communication Reference Guide (literature number SPRU683)

OMAP5910 Dual-Core Processor Camera Interface Reference Guide (literature number SPRU684)

OMAP5905 Dual-Core Processor Multichannel Serial Interface (MCSI) Reference Guide (literature number SPRU685)

OMAP5910 Dual-Core Processor Micro-Wire Interface Reference Guide (literature number SPRU686)

OMAP5910 Dual-Core Processor Real-Time Clock (RTC) Reference Guide (literature number SPRU687)

OMAP5910 Dual-Core Processor HDQ/1-Wire Interface Reference Guide (literature number SPRU688)

OMAP5910 Dual-Core Processor PWL, PWT, and LED Peripheral Reference Guide (literature number SPRU689)

OMAP5910 Dual-Core Processor Multichannel Buffered Serial Port (McBSP) Reference Guide (literature number SPRU708)

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

1	USB Host Controller	11
2	USB Open Host Controller Interface Functionality	14
2.1	OHCI Controller Overview	14
2.2	OMAP5910 USB Host Controller/OHCI Specification Differences	14
2.2.1	Power Switching Output Pins Not Supported	14
2.2.2	Overcurrent Protection Input Pins Not Supported	15
2.2.3	HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers	15
2.2.4	No Ownership Change Interrupt	15
2.2.5	Valid Address Ranges for Pointers to Data Structures	15
2.3	OMAP5910 Implementation of OHCI Specification for USB	15
2.3.1	Isochronous TD OFFSETX/PSWX Values	15
2.3.2	OMAP5910 USB Host Controller Endpoint Descriptor (ED) List Head Pointers	16
3	USB Host Controller Registers	17
3.1	USB Host Controller Reserved Registers and Reserved Bit Fields	53
3.2	Endianism and USB Host Controller Registers	53
3.3	USB Host Controller Registers, USB Reset, and USB Clocking	54
4	USB Host Controller Interrupt Sources	55
4.1	OHCI Interrupts	55
4.1.1	OHCI Scheduling Overrun Interrupt	55
4.1.2	OHCI HcDoneHead Writeback Interrupt	55
4.1.3	OHCI Start Of Frame Interrupt	55
4.1.4	OHCI Resume Detect Interrupt	55
4.1.5	OHCI Unrecoverable Error Interrupt	55
4.1.6	OHCI Frame Number Overflow	56
4.1.7	OHCI Root Hub Status Change	56
4.1.8	OHCI Ownership Change Interrupt	56
4.2	Local Bus MMU Interrupts	56
5	USB Pin Multiplexing	57
5.1	Host Controller Connectivity With USB Transceivers	57
5.2	USB Function Controller Connectivity With USB Transceivers	58
5.3	On-Board Transceiverless Connection Using OMAP5910 Transceiverless Link Logic	60

5.4	USB Signal Multiplexing Mode Diagrams	63
5.5	Ports Shown as Unconnected	91
5.6	Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing	92
6	USB Host Controller Access to System Memory	93
6.1	Local Bus Virtual Addressing	94
6.2	Cache Coherency in OHCI Data Structures and Data Buffers	96
6.3	Local Bus Addressing and OHCI Data Structure Pointers	97
6.3.1	MPUVAtoLBVA()—MPU Virtual Address to Local Bus Virtual Address Conversion Function	97
6.3.2	LBVAtoMPUVA()—Local Bus Virtual Address to MPU Virtual Address Conversion Function	98
6.3.3	MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function	98
6.3.4	LBVAtoPA()—Local Bus Virtual Address to Physical Address Conversion Function	99
6.3.5	PAtoLBVA()—Physical Address to Local Bus Virtual Address Conversion Function	99
6.3.6	PAtoMPUVA()—Physical Address to MPU Virtual Address Conversion Function	99
6.3.7	Physical, MPU Virtual, and Local Bus Virtual Addresses—an Example	100
6.4	NULL Pointers	103
6.5	Endianism and USB Host Controller Access to System Memory	103
6.5.1	Endianism and OHCI Endpoint and Transfer Descriptors	103
6.5.2	Endianism and OHCI Data Buffers	104
7	OMAP5910 Local Bus	105
7.1	LB Register Descriptions	105
7.2	LB MPU Time-out Register (LB_MPU_TIMEOUT)	106
7.3	LB Hold Timer Register (LB_HOLD_TIMER)	106
7.4	LB Priority Register (LB_PRIORITY_REG)	107
7.5	LB Clock Divider Register (LB_CLOCK_DIV)	107
7.6	LB Abort Address Register (LB_ABORT_ADD)	110
7.7	LB Abort Data Register (LB_ABORT_DATA)	110
7.8	LB Abort Status Register (LB_ABORT_STATUS)	111
7.9	LB IRQ Output Register (LB_IRQ_OUTPUT)	112
7.10	LB IRQ Input Register (LB_IRQ_INPUT)	112
7.11	Local Bus Initialization	113
7.12	Local Bus Virtual Addressing	113
8	OMAP5910 Local Bus MMU	114
8.1	OMAP5910 Local Bus MMU Registers	115
8.2	Local Bus MMU Programming for USB Host Controller Operation	127
8.2.1	Local Bus MMU Page Size and the USB Host Controller	127
8.2.2	Local Bus MMU and Page Protection	127
8.2.3	Local Bus MMU Page Miss	127

9	USB Host Controller Reset and Clock Control	129
9.1	USB Host Controller Clock Control	129
9.2	Initializing ULPD to Generate the 48-MHz Clock	129
9.3	USB Host Controller Hardware Reset	130
9.4	USB Host Controller OHCI Reset	130
9.5	USB Host Controller Power Management	130
9.6	Local Bus Clock	131
10	OMAP5910 USB Hardware Considerations	132
10.1	VBUS Power Switching For USB Type A Host Receptacles	132
10.2	Transient Suppression for USB Connectors	132
10.3	VBUS Monitoring for USB Function Controller	132
10.4	USB D+ Pullup Enable for USB Function Controller	132
10.5	Port Passthrough Mode	133
10.6	UART1 Connectivity when CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24	133
10.7	MPU_BOOT Signal Sharing	134
10.8	USB D+, D– Pulldown for USB Function Controller	134
11	Overview of the OMAP5910 USB Functional Module	135
11.1	OMAP5910 Inputs/Outputs	135
11.2	USB Function Interrupts	135
11.3	USB Function Clocks and Reset	137
11.4	USB Function DMA Requests	137
11.5	USB Detection	138
11.5.1	Software Detection	138
11.5.2	Hardware Detection	138
	GPIO0 Detection	139
	I/O Power Supply Detection	139
11.6	Software Disconnect	140
12	Register Map	141
12.1	Revision Register (REV)	143
12.1.1	REV_NB	143
12.2	Endpoint Selection Register (EP_NUM)	143
12.2.1	Setup FIFO Select (Setup_Sel)	144
12.2.2	TX/RX FIFO Select (EP_Sel)	144
12.2.3	Endpoint Direction (EP_Dir)	145
12.2.4	Endpoint Number (EP_Num)	145
12.3	Data Register (DATA)	146
12.3.1	Transmit/Receive FIFO Data (DATA)	146
12.4	Control Register (CTRL)	147
12.4.1	Clear Halt Endpoint (Clr_Halt)	147
12.4.2	Set Halt Endpoint (Set_Halt)	148
12.4.3	Set FIFO Enable (Set_FIFO_En)	148

12.4.4	Clear Endpoint (Clr_EP)	149
12.4.5	Endpoint Reset (Reset_EP)	149
12.5	Status Register (STAT_FLG)	150
12.5.1	Isochronous Missed IN Token (Miss_In)	151
12.5.2	Isochronous Receive Data Flush (Data_Flush)	151
12.5.3	Isochronous Receive Data Error (ISO_Err)	152
12.5.4	Isochronous FIFO Empty (ISO_FIFO_Empty)	152
12.5.5	Isochronous FIFO Full (ISO_FIFO_Full)	152
12.5.6	Endpoint Halted Flag (EP_Halted)	153
12.5.7	Transaction Stall (STALL)	153
12.5.8	Transmit Non-Acknowledge (NAK)	153
12.5.9	Transaction Acknowledge (ACK)	154
12.5.10	FIFO Enable (FIFO_En)	154
12.5.11	Non-Isochronous FIFO Empty (Non_ISO_FIFO_Empty)	154
12.5.12	Non-Isochronous FIFO Full (Non_ISO_FIFO_Full)	155
12.6	Receive FIFO Status Register (RXFSTAT)	155
12.6.1	Receive FIFO Byte Count (RXF_Count)	155
12.7	System Configuration Register 1 (SYSCON1)	156
12.7.1	Device Configuration Locked (Cfg_lock)	156
12.7.2	NAK Enable (Nak_En)	157
12.7.3	Self-Powered (Self_Pwr)	157
12.7.4	Shutoff Disable (SOFF_Dis)	157
12.7.5	External Pullup Enable (Pullup_En)	158
12.8	System Configuration Register 2 (SYSCON2)	159
12.8.1	Remote Wakeup (Rmt_Wkp)	159
12.8.2	Stall Command (Stall_Cmd)	160
12.8.3	Device Cond (Dev_Cfg)	160
12.8.4	Clear Cond (Clr_Cfg)	160
12.9	Device Status Register (DEVSTAT)	161
12.9.1	Remote Wakeup Enabled (R_WK_OK)	162
12.9.2	USB Reset Signaling (USB_Reset)	162
12.9.3	Suspended State (SUS)	162
12.9.4	Cond State (CFG)	163
12.9.5	Addressed State (ADD)	163
12.9.6	Default State (DEF)	163
12.9.7	Attached State (ATT)	163
12.10	Start of Frame Register (SOF)	164
12.10.1	Frame Timer Locked (FT_Lock)	164
12.10.2	Time Stamp OK (TS_OK)	165
12.10.3	Time Stamp Number (TS)	165
12.11	Interrupt Enable Register (IRQ_EN)	166
12.12	Interrupt Source Register (IRQ_SRC)	167
12.12.1	Transmit DMA CH.n Done Interrupt Flag (TXn_Done)	168
12.12.2	RX DMA CH.n Transactions Count Interrupt Flag (RXn_Cnt)	168

12.12.3 Receive DMA CH.n EOT Interrupt Flag (RXn_EOT)	169
12.12.4 Start Of Frame Interrupt Flag (SOF)	169
12.12.5 OUT Transaction Endpoint n Interrupt Flag (EPn_RX)	170
12.12.6 IN Transaction Endpoint n Interrupt Flag (EPn_TX)	170
12.12.7 Device State Changed Interrupt Flag (DS_Chg)	170
12.12.8 Setup Transaction Interrupt Flag (Setup)	170
12.12.9 OUT Transaction Endpoint 0 Interrupt Flag (EP0_RX)	171
12.12.10 IRQ_SRC[0].EP0_TX: IN Transaction Endpoint 0 Interrupt Flag	171
12.13 Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)	171
12.13.1 Receive Endpoint Interrupt Source (EPn_RX_IT_src)	172
12.13.2 Transmit Endpoint Interrupt Source (EPn_TX_IT_src)	172
12.14 Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)	173
12.14.1 DMA Receive Single Byte (DMAn_RX_SB)	173
12.14.2 DMA Receive Interrupt Source (DMAn_RX_IT_src)	174
12.14.3 DMA Transmit Interrupt Source (DMAn_TX_IT_src)	174
12.15 Receive DMA Channels Configuration Register (RXDMA_CFG)	175
12.15.1 Receive Endpoint Number for DMA Channel 2 (RXDMA2_EP)	175
12.15.2 Receive Endpoint Number for DMA Channel 1 (RXDMA1_EP)	176
12.15.3 Receive Endpoint Number for DMA Channel 0 (RXDMA0_EP)	176
12.16 Transmit DMA Channels Configuration Register (TXDMA_CFG)	177
12.16.1 Transmit Endpoint Number for DMA Channel 2 (TXDMA2_EP)	177
12.16.2 Transmit Endpoint Number for DMA Channel 1 (TXDMA1_EP)	178
12.16.3 Transmit Endpoint Number for DMA Channel 0 (TXDMA0_EP)	178
12.17 DMA FIFO Data Register (DATA_DMA)	179
12.17.1 DMA FIFO Data(DATA_DMA)	179
12.18 Transmit DMA Control Registers (TXDMA0...TXDMA2)	180
12.18.1 Transmit DMA Ch.n End of Transfer (TXn_EOT)	180
12.18.2 Transmit DMA Ch.n Start (TXn_Start)	181
12.18.3 Transmit DMA Ch.n Transfer Size Counter (TXn_TSC)	181
12.19 Receive DMA Control Registers (RXDMA...RXDMA2)	182
12.19.1 Receive DMA Ch.n Transfer Stop (RXn_Stop)	182
12.19.2 Receive DMA Ch.n Transactions Count (RXn_TC)	182
12.20 Endpoint 0 Configuration Register (EP0)	183
12.20.1 Endpoint 0 FIFO Size (EP0_Size)	183
12.20.2 Endpoint 0 Pointer (EP0_ptr)	183
12.21 Receive Endpoint Configuration Registers (EP1_RX...EP15_RX)	184
12.21.1 Receive Endpoint n Valid (EPn_RX_Valid)	184
12.21.2 Receive Endpoint n Double-Buffer (EPn_RX_Db)	185
12.21.3 Receive Endpoint n Size (EPn_RX_Size)	185
12.21.4 Receive Isochronous Endpoint n (EPn_RX_Iso)	186
12.21.5 Receive Endpoint n Pointer (EPn_RX_ptr)	186
12.22 Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)	187
12.22.1 EPn_TX[15].EPn_TX_Valid: Transmit Endpoint n Valid	187
12.22.2 Transmit Endpoint n Double-Buffer(EPn_TX_Db)	187

12.22.3 Transmit Endpoint n Size (EPn_TX_Size)	188
12.22.4 Transmit Isochronous Endpoint n (EPn_TX_Iso)	188
12.22.5 Transmit Endpoint n Pointer (EPn_TX_ptr)	188
13 USB Transactions	189
13.1 Non-Isochronous, Non-Setup OUT (USB HOST -> LH) Transactions	189
13.1.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions	190
Acknowledged Transactions (ACK)	191
Non-Acknowledged Transactions (NAK)	192
13.1.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions	192
STALLED Transactions	192
Packet Errors	193
Sequence Bit Errors	193
13.1.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions	193
13.2 Non-Isochronous IN (LH->USB HOST) Transactions	193
13.2.1 Non-Isochronous IN Endpoint Handshaking	195
Acknowledged Transactions (ACK)	196
Non-must Transactions (NAK)	196
13.2.2 Non-Isochronous IN Transaction Error Conditions	197
STALLED Transactions	197
Packet Errors	198
13.2.3 Non-Isochronous IN Endpoint FIFO Error Conditions	198
13.3 Isochronous OUT (USB HOST-> LH) Transactions	199
13.3.1 Isochronous OUT Endpoint Handshaking	200
13.3.2 Isochronous OUT Transaction Error Conditions	200
13.3.3 Isochronous OUT Endpoint FIFO Error Conditions	201
13.4 Isochronous IN (LH->USB HOST) Transactions	202
13.4.1 Isochronous IN Endpoint Handshaking	203
13.4.2 Isochronous IN Transaction Error Conditions	203
13.4.3 Isochronous IN Endpoint FIFO Error Conditions	203
13.5 Control Transfers on Endpoint 0	204
13.5.1 Autodecoded Control Write Transfers	208
Autodecoded Control Write Transfer Handshaking	208
Autodecoded Control Write Transfer Error Conditions	208
13.5.2 Autodecoded Control Read Transfers	209
Autodecoded Control Read Transfer Handshaking	209
Autodecoded Control Read Transfer Error Conditions	209
13.5.3 Non-Autodecoded Control Write Transfers	209
Specific Local Host Required Actions	211
Non-Autodecoded Control Write Transfer Handshaking	211
Non-Autodecoded Control Write Transfer Error Conditions	212
13.5.4 Non-Autodecoded Control Read Transfers	212
Non-Autodecoded Control Read Transfer Handshaking	213
Non-Autodecoded Control Read Transfer Error Conditions	214

13.5.5 Autodecoded Versus Non-Autodecoded Control Requests	214
13.5.6 Note on Control Transfers Data Stage Length	217
14 Device Initialization	219
15 Preparing for Transfers	223
16 Interrupt Service Routine (ISR) Flowcharts	226
16.1 Important Note on USB Interrupts	226
16.2 Parsing the General USB Interrupt	227
16.3 Setup Interrupt Handler	227
16.4 Endpoint 0 RX Interrupt Handler	232
16.5 Endpoint 0 TX Interrupt Handler	232
16.6 Device States Changed Handler	237
16.7 Device States Attached/Unattached Handler	240
16.8 USB Reset Interrupt Handler	241
16.9 Suspend/Resume Interrupt Handler	241
16.10 Parsing the Non-Isochronous Endpoint-Specific Interrupt	241
16.11 Non-Isochronous, Non-Control OUT Endpoint Receive Interrupt Handler	246
16.12 Non-Isochronous, Non-Control IN Endpoint Transmit Interrupt Handler	246
16.13 SOF Interrupt Handler	246
16.14 Summary of USB-Related Interrupts	254
17 DMA Operation	255
17.1 Receive DMA Channels Overview	255
17.2 Non-Isochronous OUT (USB HOST → LH) DMA Transactions	255
17.3 Isochronous OUT (USB HOST → LH) DMA Transactions	261
17.4 Transmit DMA Channels Overview	262
17.5 Non-Isochronous IN (LH → USB HOST) DMA Transactions	262
17.6 Isochronous IN (USB HOST → LH) DMA Transactions	266
17.7 Important Note on DMA Requests	266
17.8 Note on DMA Channel Deconfiguration	267
18 Power Management	269
19 Frame Adjustment Counter	271
20 Features	271
21 Synchronization and Counter Control	273
22 FAC Interrupt	276
23 FAC Clocks and Reset	276
24 Software Interface	277

Figures

1	OMAP5910 Block Diagram	12
2	OMAP5910 USB Host Controller	13
3	Typical USB Host Connections	57
4	Typical USB Function Connections	58
5	OMAP5910 USB Host Controller Connection—With and Without the OMAP5910 Transceiverless Link Logic	61
6	OMAP5910 USB Function Connection—With and Without the OMAP5910 Transceiverless Link Logic	62
7	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 0	66
8	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 1	67
9	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 2	68
10	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 3	69
11	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 4	70
12	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 5	71
13	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 6	72
14	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 7	73
15	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 9	74
16	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 10	75
17	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 11	76
18	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 12	77
19	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 13	78
20	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 14	79
21	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 15	80
22	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 16	81
23	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 17	82
24	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 18	83
25	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 19	84
26	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 20	85
27	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 21	86
28	OMAP5910 Cond for HMC_MODEs 22, 26-31	87
29	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 23 (Transceiverless Connection Uses TXD+, TXD- Signaling)	88
30	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 24 (Transceiverless Connection Uses TXD+, TXD- Signaling)	89
31	OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 25 (Transceiverless Connection Uses TXD+, TXD- Signaling)	90
32	OMAP5910 USB Host Controller Data Path to System Memory	93

33	Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address With Local Bus MMU Disabled	94
34	Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address With Local Bus MMU Enabled	95
35	USB Function Environment	136
36	Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions	190
37	Non-Isochronous IN Transaction Phases and Interrupts	195
38	Isochronous OUT Transaction Phases and Interrupts	200
39	Isochronous IN Transaction Phases and Interrupts	203
40	Stages and Transaction Phases of Autodecoded Control Transfers	205
41	Stages and Transaction Phases of Non-Autodecoded Control Transfers	206
42	Example of RAM Organization	220
43	Device Configuration Routine	221
44	Endpoint Configuration Routine	222
45	Prepare for USB RX Transfer Routine	224
46	Prepare for TX Transfer on Endpoint n Routine	225
47	General USB Interrupt ISR Source Parsing Flowchart	229
48	Setup Interrupt Handler	230
49	Parse Command Routine (Setup Stage Control Transfer Request)	231
50	Endpoint 0 RX Interrupt Handler	233
51	Prepare for Control Write Status Stage Routine	234
52	Endpoint 0 TX Interrupt Handler	235
53	Prepare for Control Read Status Stage Routine	236
54	USB Function Device State Transitions	238
55	Typical Operation for USB Device State Changed Interrupt Handler	239
56	Attached/Unattached Handler	240
57	USB Reset Handler Flowchart I	242
58	USB Reset Handler Flowchart II	243
59	Typical Operation for USB Suspend/Resume General USB Interrupt Handler	244
60	Non-Isochronous Endpoint-Specific (Except ER 0) ISR Flowchart	245
61	Non-Isochronous Non-Control Endpoint Receive Interrupt Handler	247
62	Read Non-Isochronous RX FIFO Data Flowchart	248
63	Non-Isochronous Non-control Endpoint Transmit Interrupt Handler	249
64	Write Non-Isochronous TX FIFO Data Flowchart	250
65	SOF Interrupt Handler Flowchart	251
66	Read Isochronous RX FIFO Data Flowchart	252
67	Write Isochronous TX FIFO Data Flowchart	253
68	Non-Isochronous RX DMA Transaction Example (RX_TC = 2)	257
69	Non-Isochronous RX DMA Start Routine	258
70	Non-Isochronous RX DMA EOT Interrupt Handler	259
71	Non-Isochronous RX DMA Transaction Count Interrupt Handler	260
72	Isochronous RX DMA Transaction	261
73	Isochronous RX DMA Start Routine	261
74	File Transfer Size	263

75	Non-Isochronous TX DMA DMA Start Routine	264
76	Non-Isochronous TX DMA Done Interrupt Handler	265
77	Isochronous TX DMA Start Routine	267
78	Power Management Signal Values	270
79	FAC Top-Level Diagram	272
80	FAC-Module Counters and Clock Synchronization	274
81	Synchronization Circuit for Frame-Synchronization and Frame-Start Signals	275
82	Synchronization Circuit Waveforms	275

Tables

1	USB Host Controller Registers	17
2	OHCI Revision Number Register (HcRevision)	19
3	HC Operating Mode Register (HcControl)	19
4	HC Command and Status Register (HcCommandStatus)	22
5	HC Interrupt Status Register (HcInterruptStatus)	23
6	HC Interrupt Enable Register (HcInterruptEnable)	24
7	HC Interrupt Disable Register (HcInterruptDisable)	27
8	HC HCAA Address Register (HcHCCA)	28
9	HC Current Periodic Register (HcPeriodCurrentED)	29
10	HC Head Control Register (HcControlHeadED)	29
11	HC Current Control Register (HcControlCurrentED)	30
12	HC Head Bulk Register (HcBulkHeadED)	31
13	HC Current Bulk Register (HcBulkCurrentED)	31
14	HC Head Done Register (HcDoneHead)	32
15	HC Frame Interval Register (HcFmInterval)	32
16	HC Frame Remaining Register (HcFmRemaining)	33
17	HC Frame Number Register (HcFmNumber)	33
18	HC Periodic Start Register (HcPeriodicStart)	34
19	HC Low-Speed Threshold Register (HcLSThreshold)	34
20	HC Root Hub A Register (HcRhDescriptorA)	35
21	HC Root Hub B Register (HcRhDescriptorB)	37
22	HC Root Hub Status Register (HcRhStatus)	38
23	HC Port 1 Status and Control Register (HcRhPortStatus1)	39
24	HC Port 2 Status and Control Register (HcRhPortStatus2)	43
25	HC Port 3 Status and Control Register (HcRhPortStatus3)	47
26	Host UE Address Register (HostUEAddr)	51
27	Host UE Status Register (HostUEStatus)	52
28	Host Time-out Control Register (HostTimeoutCtrl)	52
29	Host Revision Register (HostRevision)	53
30	USB Signal Multiplexing Modes	63
31	MPU MMU Programming for Address Conversion Example	100
32	MPU Memory Allocations for Address Conversion Example	100
33	Physical Addresses for Address Conversion Example	100
34	Local Bus MMU Programming for Address Conversion Example	101
35	Local Bus Virtual Addresses for Address Conversion Example	101
36	Some Data Structure Initializations for Address Conversion Example	102

37	Little-Endian Data Alignment Within 32-Bit Word	104
38	Local Bus Control Registers	105
39	LB MPU Time-out Register (LB_MPU_TIMEOUT)	106
40	LB Hold Timer Register (LB_HOLD_TIMER)	106
41	LB Priority Register (LB_PRIORITY_REG)	107
42	LB Clock Divider Register (LB_CLOCK_DIV)	108
43	LB Abort Address Register (LB_ABORT_ADD)	110
44	LB Abort Data Register (LB_ABORT_DATA)	110
45	LB Abort Status Register (LB_ABORT_STATUS)	111
46	LB IRQ Output Register (LB_IRQ_OUTPUT)	112
47	LB IRQ Input Register (LB_IRQ_INPUT)	112
48	Local Bus MMU Registers	115
49	LB MMU Walking Status Register (LB_MMU_WALKING_ST_REG)	116
50	LB MMU Control Register (LB_MMU_CNTL_REG)	116
51	LB MMU Fault Address High Register (LB_MMU_FAULT_AD_H_REG)	117
52	LB MMU Fault Address Low Register (LB_MMU_FAULT_AD_L_REG)	117
53	LB MMU Fault Status Register (LB_MMU_FAULT_ST_REG)	118
54	LB MMU Interrupt Acknowledge Register (LB_MMU_IT_ACK_REG)	118
55	LB MMU TTB Address High Register (LB_MMU_TTB_H_REG)	119
56	LB MMU TTB Address Low Register (LB_MMU_TTB_L_REG)	119
57	LB MMU Lock Counter Register (LB_MMU_LOCK_REG)	119
58	Local Bus MMU TLB Read/Write Register	120
59	Local Bus MMU CAM High Register	121
60	Local Bus MMU CAM Low Register	121
61	Local Bus MMU RAM High Register	123
62	Local Bus MMU RAM Low Register	123
63	Local Bus MMU Global Flush Register	124
64	Local Bus MMU Entry Flush Register	124
65	Local Bus MMU CAM Read High Register	124
66	Local Bus MMU RAM Read High Register	126
67	Local Bus MMU RAM Read Low Register	126
68	CONF_MOD_USB_HOST_HMC_MODE_R=7 Internal Connectivity	133
69	CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24 UART Signal Assignments	134
70	USB Function Module Registers	141
71	Revision Register (REV)	143
72	Endpoint Selection Register (EP_NUM)	143
73	Data Register (DATA)	146
74	Control Register (CTRL)	147
75	Status Register (STAT_FLG)	150
76	Receive FIFO Status Register (RXSTAT)	155
77	System Configuration Register 1(SYSCON1)	156
78	SYSCON2 – System Configuration Register 2 (SYSCON2)	159
79	Device Status Register (DEVSTAT)	161

80	Start of Frame Register (SOF)	164
81	Interrupt Enable Register (IRQ_EN)	166
82	Interrupt Source Register (IRQ_SRC)	167
83	Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)	171
84	Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)	173
85	Receive DMA Channels Configuration Register (RXDMA_CFG)	175
86	Transmit DMA Channels Configuration Register (TXDMA_CFG)	177
87	DMA FIFO Data Register (DATA_DMA)	179
88	Transmit DMA Control Registers (TXDMA...TXDMA2)	180
89	Receive DMA Control Registers (RXDMA0...RXDMA2)	182
90	Endpoint 0 Configuration Register (EP0)	183
91	Receive Endpoint n Configuration Registers (EP1_RX...EP15_RX)	184
92	Endpoint n Size Values	185
93	Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)	187
94	Autodecoded Versus Non-Autodecoded Control Requests	214
95	USB Interrupt Type by Endpoint Type	254
96	FAC Registers	277
97	Frame-Adjustment-Reference-Count Register (FARC)	277
98	Frame-Start-Count Register (FSC)	278
99	FAC-Control-and-Configuration Register (CTRL)	279
100	FAC-Status Register (STATUS)	279

Universal Serial Bus

This chapter describes the universal serial bus (USB) host of the OMAP5910 multimedia processor.

1 USB Host Controller

The OMAP5910 USB host controller (HC) is a three-port controller that communicates with USB devices at the USB low-speed (1.5M bit/s maximum) and full-speed (12M bit/s maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 1.1* and the *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB*. It is assumed that users of the OMAP5910 USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The OMAP5910 USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the OMAP5910 USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The OMAP5910 USB host controller accesses these data structures without direct intervention by the processor using the OMAP5910 local bus. These data structures and data buffers can be located in internal or external system RAM. The local bus MMU allows the USB host controller to access the full address range of internal and external memories.

The OMAP5910 USB host controller is connected to the OMAP5910 MPU public peripheral bus to enable MPU access to registers. The USB host controller gains access to the data structures in the OMAP5910 system memory via the internal OMAP5910 local bus (LB) interface. The USB host controller provides an interrupt to the MPU level 2 interrupt handler to signal certain hardware events to the host controller driver software.

Flexible multiplexing of signals from the OMAP5910 USB host controller, the OMAP5910 USB function controller, and other OMAP5910 peripherals allows

a wide variety of system-level USB functions. The OMAP5910 top-level pin multiplexing controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5910 USB signal multiplexing selects how the signals associated with the three OMAP5910 USB host ports and the OMAP5910 USB function controller can be brought out to OMAP5910 pins.

Figure 1 shows the OMAP5910 device with the USB host controller highlighted. Figure 2 shows the OMAP5910 USB host controller. See Section 1.11 for the OMAP5910 USB Functional Module.

Figure 1. OMAP5910 Block Diagram

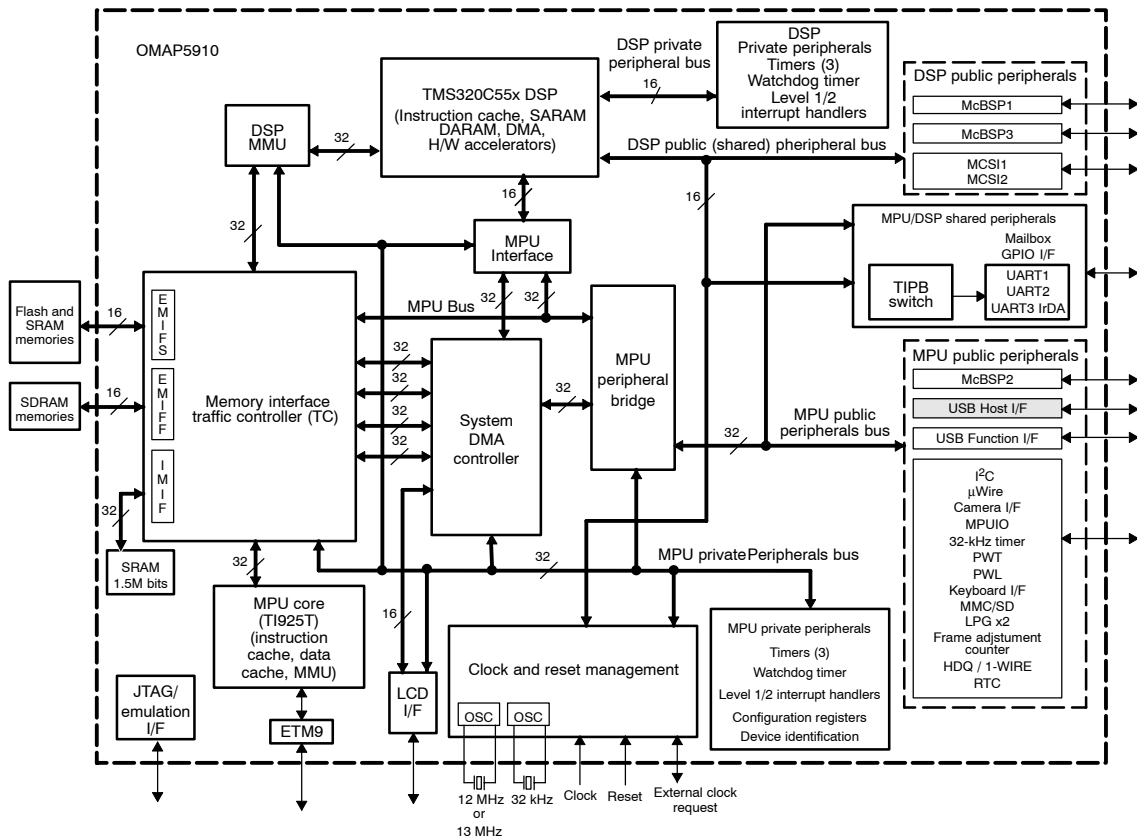
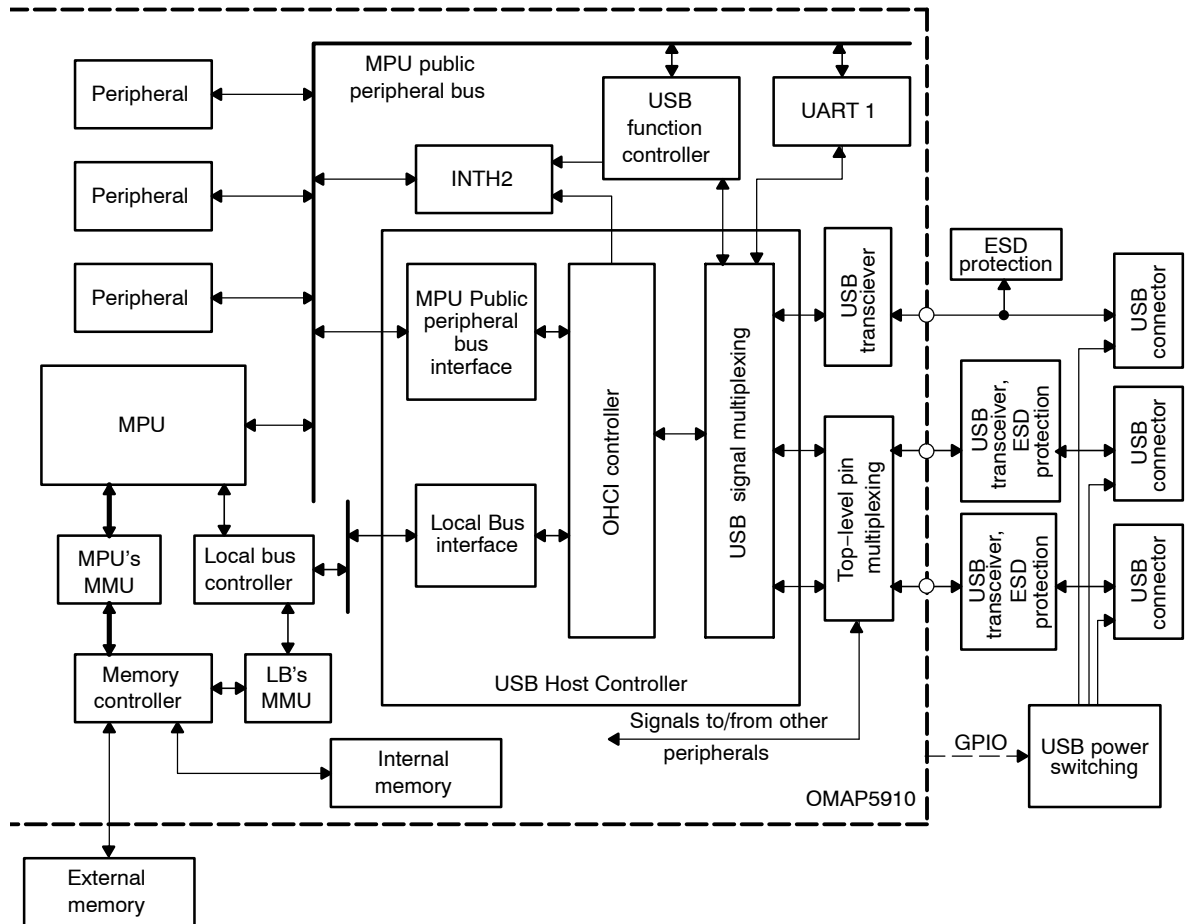


Figure 2. OMAP5910 USB Host Controller



2 USB Open Host Controller Interface Functionality

2.1 OHCI Controller Overview

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the *Universal Serial Bus Specification Version 1.1*, defines most of the USB functionality that the OMAP5910 USB host controller provides.

The *OHCI Specification for USB* focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. Issues such as interrupt generation, USB host controller state, USB frame management, and the methods that the hardware must use to process the lists of data structures in system memory are also described.

This document does not duplicate the information presented in the *OHCI Specification for USB* or the *USB Specification*. OMAP5910 USB host controller users should refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

2.2 OMAP5910 USB Host Controller/*OHCI Specification Differences*

The OMAP5910 USB host controller does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on power switching, overcurrent reporting, and the OHCI ownership change interrupt. Other restrictions are imposed by OMAP5910 system memory addressing mechanisms and the effects of OMAP5910 pin multiplexing options.

2.2.1 Power Switching Output Pins Not Supported

The OMAP5910 device does not provide pins that can be controlled directly by the USB host controller OHCI port power control features. The OHCI RhPortStatus(n) register port power control bits can be programmed by the USB host controller driver software, but this does not have any direct effect on any VBUS switching implemented on the board.

Users can use other GPIO pins or implementation-specific control mechanisms to control VBUS switching.

2.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP5910 device does not provide pins that allow the USB host controller OHCI RhPortStatus(n) overcurrent protection status bits to be directly controlled by external hardware.

Users can use GPIO pins or other implementation-specific control mechanisms to report port overcurrent information to the USB host controller driver.

2.2.3 HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers

The USB signal multiplexing modes provide selections to allow 0, 1, 2, or 3 USB host controller ports to be brought out to OMAP5910 pins. The OHCI RhDescriptorA register always reports three available USB host ports, regardless of the CONF_MOD_USB_HOST_HMC_MODE_R field of the MOD_CONF_CTRL_0 register or top-level pin multiplexing settings. When the CONF_MOD_USB_HOST_HMC_MODE_R field setting of the MOD_CONF_CTRL_0 register disables a USB host controller port, the USB host controller sees that port as unattached.

When OMAP5910 top-level pin multiplexing configures a pin for functionality other than the USB, the USB host controller is disconnected from that pin and that pin does not affect the USB host controller.

2.2.4 No Ownership Change Interrupt

The OMAP5910 USB host controller does not implement the OHCI ownership change interrupt.

2.2.5 Valid Address Ranges for Pointers to Data Structures

The mechanism that allows the OMAP5910 USB host controller to access the USB endpoint descriptor (ED), transfer descriptor (TD), and HCCA data structures in system memory places certain requirements on the registers that point to data structures in system memory and on the pointers within those data structures. Details can be found in Section 6, *USB Host Controller Access to System Memory*.

2.3 OMAP5910 Implementation of *OHCI Specification for USB*

2.3.1 Isochronous TD OFFSETX/PSWX Values

The OMAP5910 USB host controller implements the *OHCI Specification for USB* optional feature of checking isochronous OFFSETX/PSWX values. If

either OFFSETX or OFFSET(X+1) does not have a condition code of Not Accessed, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HostUEAddr, HostUEStatus, or HostTimeoutCtrl registers.

2.3.2 OMAP5910 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI *Specification for USB* provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in a malfunction. As a specific example, the HcControlHeadED and HcBulkHeadED pointer registers and the 32 HcCalInterruptTable pointers must all point to valid local bus addresses of valid endpoint descriptors.

The OMAP5910 USB host controller does not check HcControlHeadED registers, HcBulkHeadED registers, or the values in the 32 HcCalInterruptTable pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the OMAP5910 USB host controller attempts to access using the local bus virtual address of 0, which causes an unrecoverable error. Registers HostUEAddr, HostUEStatus, and HostTimeoutCtrl are updated in this case.

3 USB Host Controller Registers

Most of the OMAP5910 host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

Note:

The USB host controller registers must be accessed using 32-bit data operations. Use of smaller data access sizes may result in unexpected operation of the USB host controller. The USB host controller registers and the USB host controller data structures are organized for little-endian operation mode because the MPU processor on the OMAP5910 device must use little-endian mode.

The OMAP5910 USB host controller registers are listed in Table 1. Table 2 through Table 29 describe specific register bits.

Table 1. USB Host Controller Registers

Name	Description	R/W	Size [†]	Address
HcRevision	OHCI revision number	R	32	FFFB:A000h
HcControl	HC operating mode	R/W	32	FFFB:A004h
HcCommandStatus	HC command and status	R/W	32	FFFB:A008h
HcInterruptStatus	HC interrupt status	R/W	32	FFFB:A00Ch
HcInterruptEnable	HC interrupt enable	R/W	32	FFFB:A010h
HcInterruptDisable	HC interrupt disable	R	32	FFFB:A014h
HcHCCA	Local bus virtual address of the HCCA [‡]	R/W	32	FFFB:A018h
HcPeriodCurrentED	Local bus virtual address of the current periodic endpoint descriptor [‡]	R/W	32	FFFB:A01Ch
HcControlHeadED	Local bus virtual address of the head of the control endpoint descriptor list [‡]	R/W	32	FFFB:A020h
HcControlCurrentED	Local bus virtual address of the current control endpoint descriptor [‡]	R/W	32	FFFB:A024h

[†] Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

[‡] Restrictions apply to the local bus virtual addresses used in these registers. See Section 6.1, *Local Bus Addressing*.

[§] This register provides control and status for the OMAP5910 pins associated with the USB transceiver for some HMC_MODE values.

[¶] This register provides control and status for the OMAP5910 pins associated with USB port 1 for some HMC_MODE values.

[#] This register provides control and status for the OMAP5910 pins associated with USB port 2 for some HMC_MODE values.

Table 1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size [†]	Address
HcBulkHeadED	Local bus virtual address of the head of the bulk endpoint descriptor list [‡]	R/W	32	FFFB:A028h
HcBulkCurrentED	Local bus virtual of the current bulk endpoint descriptor [‡]	R/W	32	FFFB:A02Ch
HcDoneHead	Local bus virtual address of the head of the list of retired transfer descriptors [‡]	R	32	FFFB:A030h
HcFmInterval	HC frame interval	R/W	32	FFFB:A034h
HcFmRemaining	HC frame remaining	R	32	FFFB:A038h
HcFmNumber	HC frame number	R	32	FFFB:A03Ch
HcPeriodicStart	HC periodic start	R/W	32	FFFB:A040h
HcLSThreshold	HC low speed threshold	R/W	32	FFFB:A044h
HcRhDescriptorA	HC root hub A	R, R/W	32	FFFB:A048h
HcRhDescriptorB	HC root hub B	R/W	32	FFFB:A04Ch
HcRhStatus	HC root hub status	R, R/W	32	FFFB:A050h
HcRhPortStatus1	HC port 1 control and status [§]	R, R/W	32	FFFB:A054h
HcRhPortStatus2	HC port 2 control and status [¶]	R, R/W	32	FFFB:A058h
HcRhPortStatus3	HC port 3 control and status [#]	R, R/W	32	FFFB:A05Ch
Reserved	Reserved	None		FFFB:A060h to FFFB:A0DFh
HostUEAddr	Host UE address	R	32	FFFB:A0E0h
HostUEStatus	Host UE status	R	32	FFFB:A0E4h
HostTimeoutCtrl	Host timeout control	R/W	32	FFFB:A0E8h
HostRevision	Host revision	R	32	FFFB:A0ECh
Reserved	Reserved	None		FFFB:A0F0h to FFFB:AFFFh

[†] Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

[‡] Restrictions apply to the local bus virtual addresses used in these registers. See Section 6.1, *Local Bus Addressing*.

[§] This register provides control and status for the OMAP5910 pins associated with the USB transceiver for some HMC_MODE values.

[¶] This register provides control and status for the OMAP5910 pins associated with USB port 1 for some HMC_MODE values.

[#] This register provides control and status for the OMAP5910 pins associated with USB port 2 for some HMC_MODE values.

The other revision number register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Table 2. OHCI Revision Number Register (HcRevision)

Bits	Field	Description	Type	Reset Value
31–8	Reserved	Reserved		0x00 0000
7–0	REV	OHCI Specification revision—the OHCI revision number upon which the USB host controller is based. Write has no effect.	R	0x10

The HC operating mode register controls the operating mode of the USB host controller.

Table 3. HC Operating Mode Register (HcControl)

Bits	Field	Value	Description	Type	Reset Value
31–11	Reserved		Reserved		
10	RWE		Remote wake-up enable. This bit has no effect in OMAP5910. The OMAP5910 USB host controller does not provide a processor wake-up mechanism.	R/W	0
9	RWC		Remote wake up connected. This bit has no effect in OMAP5910. The OMAP5910 USB host controller does not provide a processor wake-up mechanism.	R/W	0
8	IR		Interrupt routing. The OMAP5910 USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the MPU level 2 interrupt controller.	R/W	0

Table 3. HC Operating Mode Register (HcControl) (Continued)

Bits	Field	Value	Description	Type	Reset Value
7–6	HCFS		Host controller functional state:	R/W	00
		00	USBReset		
		01	USBResume		
		10	USBOperational		
		11	USBSuspend		
			A transition to USBOperational causes SOF generation to begin in 1 ms. The USB host controller may automatically transition from USBSuspend to USBResume if a downstream resume is received. The USB host controller enters USBSuspend after a software reset. The USB host controller enters USBReset after a hardware reset. The USBReset state resets the root hub and causes downstream signaling of USBReset.		
5	BLE		Bulk list enable:	R/W	0
		0	Bulk ED list is not processed in the next 1-ms frame. The host controller driver can modify the list. If the driver removes the ED pointed to by the HcBulkCurrentED from the ED list, it must update HcBulk-CurrentED to point to an ED still on the list before it reenables the bulk list.		
		1	Enables processing of bulk ED List. HcBulkHeadED must be 0 or point to a valid ED before setting this bit. HcBulkCurrentED must point to a valid ED or be 0 before setting this bit.		
4	CLE		Control list enable:	R/W	0
		0	The host control ED list is not processed in the next 1-ms frame. Host controller driver may modify the control ED list. If the driver removes the ED pointed to by the HcControlCurrentED from the ED list, it must updateHcControlCurrentED to point to an ED still on the list before it reenables the control list.		
		1	Enables processing of the control ED list. HcControlHeadED must be 0 or point to a valid ED before setting this bit. HcControlCurrentED must be 0 or point to a valid ED before setting this bit		

Table 3. HC Operating Mode Register (HcControl) (Continued)

Bits	Field	Value	Description	Type	Reset Value
3	IE		Isochronous enable	R/W	0
		0	Enables processing of isochronous EDs.		
		1	Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list. When this bit is written to 1, processing of isochronous EDs might not occur in the current frame but is enabled in the next frame.		
2	PLE		Periodic list enable	R/W	0
		0	The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame.		
		1	Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.		
1–0	CBSR		Control/bulk service ratio	R/W	00
			Specifies the ratio between control and bulk EDs processed in a frame.		
		00	One control ED per bulk ED		
		01	Two control EDs per bulk ED		
		10	Three control EDs per bulk ED		
		11	Four control EDs per bulk ED		

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

Table 4. HC Command and Status Register (HcCommandStatus)

Bits	Field	Description	Type	Reset Value
31–18	Reserved	Reserved		
17–16	SOC	Scheduling overrun count Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.	R	00
15–4	Reserved	Reserved		
3	OCR	Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. OMAP5910 does not support SMI interrupts, so no ownership change interrupt occurs.	R/W	0
2	BLF	Bulk list filled The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HcBulkCurrentED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.	R/W	0
1	CLF	Control list filled The host controller driver must set this bit if it modifies the control list to include new TDs. If HcControlHeadED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.	R/W	0
0	HCR	Host controller reset Write of 0 has no effect. 1: This bit initiates a software reset of the USB host controller. This transitions the USB host controller to the USBSuspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this register returns a 0. A write of 1 to this bit does not reset the root hub, and does not signal USB reset to downstream USB functions.	R/W	0

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Table 5. HC Interrupt Status Register (HcInterruptStatus)

Bits	Field	Description	Type	Reset Value
31	Reserved	Reserved		
30	OC	Ownership change The OMAP5910 USB host controller does not implement ownership change interrupts.	R	0
29–7	Reserved	Reserved		
6	RHSC	Root hub status change When 1 indicates a root hub status change has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
5	FNO	Frame number overflow When 1 indicates a frame number overflow has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
4	UE	Unrecoverable error When 1 indicates that an unrecoverable error has occurred on the local bus or that an isochronous TD PSW field condition code was not set to <i>Not Accessed</i> when the USB host controller attempted to perform a transfer using that PSW/offset pair. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
3	RD	Resume detected When 1 indicates that a downstream device has issued a resume request. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

Table 5. HC Interrupt Status Register (HcInterruptStatus) (Continued)

Bits	Field	Description	Type	Reset Value
2	SF	Start of frame When 1 indicates that a SOF has been issued. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
1	WDH	Write done head When 1 indicates that the USB host controller has updated the HcDoneHead register. Write of 0 has no effect. Write of 1 clears this bit. The host controller driver must read the value from HcDoneHead before writing 1 to this bit.	R/W	0
0	SO	Scheduling overrun When 1 indicates that a scheduling overrun has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

The HC interrupt enable register enables various OHCI interrupt sources to generate interrupts to the OMAP5910 level 2 interrupt handler.

Table 6. HC Interrupt Enable Register (HcInterruptEnable)

Bits	Field	Description	Type	Reset Value
31	MIE	Master interrupt enable When 1, allows other enabled OHCI interrupt sources to propagate to the OMAP5910 level 2 interrupt controller. When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the OMAP5910 level 2 interrupt controller. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5910.	R	0
29–7	Reserved	Reserved		

Table 6. HC Interrupt Enable Register (HcInterruptEnable) (Continued)

Bits	Field	Description	Type	Reset Value
6	RHSC	<p>Root hub status change</p> <p>When 1 and MIE is 1, allows root hub status change interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, root hub status change interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0
5	FNO	<p>Frame number overflow</p> <p>When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, frame number overflow interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0
4	UE	<p>Unrecoverable error</p> <p>When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, unrecoverable error interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0
3	RD	<p>Resume detected</p> <p>When 1 and MIE is 1, allows resume detected interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, resume detected interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0

Table 6. HC Interrupt Enable Register (HcInterruptEnable) (Continued)

Bits	Field	Description	Type	Reset Value
2	SF	<p>Start of frame</p> <p>When 1 and MIE is 1, allows start of frame interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, start of frame interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0
1	WDH	<p>Write done head</p> <p>When 1 and MIE is 1, allows write done head interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, write done head interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0
0	SO	<p>Scheduling overrun</p> <p>When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the OMAP5910 level 2 interrupt controller.</p> <p>When 0, or when MIE is 0, scheduling overrun interrupts do not propagate.</p> <p>A write of 0 has no effect on this bit.</p> <p>A write of 1 sets this bit.</p>	R/W	0

The HC interrupt disable register is used to clear bits in the HcInterruptEnable register.

Table 7. HC Interrupt Disable Register (HcInterruptDisable)

Bits	Field	Description	Type	Reset Value
31	MIE	Master interrupt enable Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable MIE bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5910.	R	0
29–7	Reserved	Reserved		
6	RHSC	Root hub status change Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable RHSC bit.	R/W	0
5	FNO	Frame number overflow Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable FNO bit.	R/W	0
4	UE	Unrecoverable error Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable UE bit.	R/W	0
3	RD	Resume detected Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable RD bit.	R/W	0
2	SF	Start of frame Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable SF bit.	R/W	0

Table 7. HC Interrupt Disable Register (HcInterruptDisable) (Continued)

Bits	Field	Description	Type	Reset Value
1	WDH	Write done head Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable WDH bit.	R/W	0
0	SO	Scheduling overrun Read always returns 0. Write of 0 has no effect. Write of 1 clears the HcInterruptEnable SO bit.	R/W	0

The HCAA address register defines the local bus virtual address of the beginning of the HCCA.

Table 8. HC HCAA Address Register (HcHCCA)

Bits	Field	Description	Type	Reset Value
31–8	HCCA	See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses.	R/W	0
7–0	Reserved	Reserved	R	0

The HC current periodic register defines the local bus virtual address of the next endpoint descriptor (ED) on the periodic ED List.

Table 9. HC Current Periodic Register (HcPeriodCurrentED)

Bits	Field	Description	Type	Reset Value
31–4	PCED	Local bus virtual address of current ED on the periodic ED list. This field represents bits 31:4 of the local bus virtual address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses.	R/0	0x0000000
3–0	Reserved	Reserved	R	0x0

The HC head control register defines the local bus virtual address of the head ED of the control ED list.

Table 10. HC Head Control Register (HcControlHeadED)

Bits	Field	Description	Type	Reset Value
31–4	CHED	Local bus virtual address of head ED on the control ED list. This field represents bits 31:4 of the local bus virtual address of the head ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC current control register defines the local bus virtual address of the next ED on the control ED list.

Table 11. HC Current Control Register (HcControlCurrentED)

Bits	Field	Description	Type	Reset Value
31–4	CCED	<p>Local bus virtual address of current ED on the control ED list</p> <p>This field represents bits 31:4 of the local bus virtual address of the next ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 6.1, <i>Local Bus Addressing</i>, for the restrictions on local bus virtual addresses.</p> <p>A value of 0x0000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process.</p> <p>This register is automatically updated by the USB host controller.</p>	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC head bulk register defines the local bus virtual address of the head ED on the bulk ED list.

Table 12. HC Head Bulk Register (HcBulkHeadED)

Bits	Field	Description	Type	Reset Value
31–4	BHED	Local bus virtual address of head ED on the bulk ED list This field represents bits 31:4 of the local bus virtual address of the head ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC current bulk register defines the local bus virtual address of the next ED on the bulk ED list.

Table 13. HC Current Bulk Register (HcBulkCurrentED)

Bits	Field	Description	Type	Reset Value
31–4	BCED	Local bus virtual address of current ED on the bulk ED list This field represents bits 31:4 of the local bus virtual address of the next ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses. A value of 0x00000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. This register is automatically updated by the USB host controller.	R/W	0
3–0	Reserved	Reserved	R	0x0

The HC head done register defines the local bus virtual address of the current head of the done TD queue.

Table 14. HC Head Done Register (HcDoneHead)

Bits	Field	Description	Type	Reset Value
31–4	DH	Local bus virtual address of the last TD that was added to the done queue. This field represents bits 31:4 of the local bus virtual address of the top TD on the done TD queue. TDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 6.1, <i>Local Bus Addressing</i> , for the restrictions on local bus virtual addresses. A value of 0x00000000 indicates that there are no TDs on the done queue. This register is automatically updated by the USB host controller.	R	0x00000000
3–0	Reserved	Reserved	R	0x0

The HC frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Table 15. HC Frame Interval Register (HcFmInterval)

Bits	Field	Description	Type	Reset Value
31	FIT	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.	R/W	0
30–16	FSMPS	Largest data packet Largest data packet size allowed for full speed packets, in bit times.	R/W	0
15–14	Reserved	Reserved		
13–0	FI	Frame interval The number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999, to give a 1-ms frame. The host controller driver may make minor changes to this field to attempt to manually synchronize with another clock source.	R/W	0x2EDF

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.

Table 16. HC Frame Remaining Register (HcFmRemaining)

Bits	Field	Description	Type	Reset Value
31	FRT	Frame remaining toggle This bit is loaded with the frame interval toggle bit every time the USB host controller loads the frame interval field into the frame remaining field.	R	0
30–14	Reserved	Reserved		
13–0	FR	Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.	R	0

The HC frame number register reports the current USB frame number.

Table 17. HC Frame Number Register (HcFmNumber)

Bits	Field	Description	Type	Reset Value
31–16	Reserved	Reserved		
15–0	FN	Frame number This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from 0xFFFF to 0x0000. After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.	R	0

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

Table 18. HC Periodic Start Register (HcPeriodicStart)

Bits	Field	Description	Type	Reset Value
31–14	Reserved	Reserved		
13–0	PS	Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.	R/W	0

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

Table 19. HC Low-Speed Threshold Register (HcLSThreshold)

Bits	Field	Description	Type	Reset Value
31–14	Reserved	Reserved		
13–0	LST	Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller may not start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold. The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.	R/W	0x0628

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Table 20. HC Root Hub A Register (HcRhDescriptorA)

Bits	Field	Value	Description	Type	Reset Value
31–24	POTPG		<p>Power-on to power-good time</p> <p>This field defines the minimum amount of time ($2\text{ ms} \times \text{POTPG}$) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device.</p> <p>This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device.</p> <p>The required amount of time is implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller.</p> <p>The implementation-specific value must be computed and then written to this register before the USB host controller driver is initialized.</p> <p>Because OMAP5910 does not provide a direct control from the USB host controller to switch the VBUS on and off, this value must take into account any delays caused by other methods of controlling VBUS externally.</p>	R/W	0xA
23–13	Reserved		Reserved		
12	NOCP		<p>No overcurrent protection</p> <p>When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. OMAP5910 does not provide signals to allow connection of external overcurrent indication signals to the USB host controller, so this bit defaults to 1.</p>	R/W	1
11	OCPM		<p>Overcurrent protection mode</p> <p>OMAP5910 does not provide overcurrent protection input signals, so this bit has no effect.</p>	R/W	0

Table 20. HC Root Hub A Register (HcRhDescriptorA) (Continued)

Bits	Field	Value	Description	Type	Reset Value
10	DT		Device type This bit is always 0, which indicates that the USB host controller implemented is not a compound device.	R	0
9	NPS		No power switching	R/W	1
		0	Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.		
		1	Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered. Because OMAP5910 does not provide connections from the USB host controller to control external VBUS switching, this bit defaults to 1.		
8	PSM		Power switching mode	R/W	0
		0	Indicates that all ports are powered at the same time.		
		1	Indicates that individual port power switching is supported. Because OMAP5910 does not provide signals from the USB host controller to control external VBUS switching, this bit defaults to 0.		
7–0	NDP		Number of downstream ports This register defaults to 3 to indicate three downstream ports. The USB signal multiplexing mode and OMAP5910 top-level pin multiplexing features may place the OMAP5910 device in a mode where 0, 1, 2, or 3 of the USB host controller downstream ports are actually useable. This register reports three regardless of USB signal multiplexing mode and OMAP5910 top-level pin multiplexing mode. See Section 5, <i>USB Pin Multiplexing</i> , for information on USB signal multiplexing.	R	0x03

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Table 21. HC Root Hub B Register (HcRhDescriptorB)

Bits	Field	Description	Type	Reset Value
31–16	PPCM	<p>Port power control mask</p> <p>Each bit determines whether a corresponding downstream port has port power controlled by the global power control. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port.</p> <p>PPCM bit 0 is reserved.</p> <p>PPCM bit 1 is the port power control mask for downstream port 1.</p> <p>PPCM bit 2 is the port power control mask for downstream port 2.</p> <p>PPCM bit 3 is the port power control mask for downstream port 3.</p> <p>PPCM bits 4 through 15 are reserved.</p> <p>OMAP5910 does not provide connections from the USB host controller to pins to provide external port power switching. Systems that implement port power switching must use other mechanisms to control port power.</p>	R/W	0x0000
15–0	DR	<p>Device removable</p> <p>Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached.</p> <p>DR bit 0 is reserved.</p> <p>DR bit 1 is the device removable bit for downstream port 1.</p> <p>DR bit 2 is the device removable bit for downstream port 2.</p> <p>DR bit 3 is the device removable bit for downstream port 3.</p> <p>DR bits 4 through 15 are reserved.</p>	R/W	0x0000

The HC root hub status register reports the USB host controller root hub status.

Table 22. HC Root Hub Status Register (HcRhStatus)

Bits	Field	Description	Type	Reset Value
31	CRWE	Clear remote wake-up enable Write of 0 has no effect. Write of 1 clears the device remote wake-up enable bit.	R/W	0
30–18	Reserved	Reserved		
17	OCIC	Overcurrent indication change This bit is automatically set when the overcurrent indicator bit changes. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
16	LPSC	Local power status change This bit defaults to 0 since the root hub does not support the local power status feature. Write of 0 has no effect. Write of 1 sets the PortPowerStatus bits for all ports if PowerSwitchingMode is 0. A write of 1 sets PortPowerStatus bits for ports with their corresponding PortPowerControlMask bits cleared if PowerSwitchingMode is 1.	R/W	0
15	DRWE	Device remote wake-up enable When 1, this bit enables a ConnectStatusChange event to be treated as a resume event, which causes a transition from USBSuspend to USBResume state, and sets the ResumeDetected interrupt status bit. When 0, ConnectStatusChange events do not cause a transition from USBSuspend to USBResume state and the ResumeDetected interrupt is not changed. Write of 0 has no effect. Write of 1 sets the device remote wake-up enable bit.	R/W	0
14–2	Reserved	Reserved		

Table 22. HC Root Hub Status Register (HcRhStatus) (Continued)

Bits	Field	Description	Type	Reset Value
1	OCI	Overcurrent indicator This bit reports global overcurrent indication if global overcurrent reporting is selected. When 1, this bit indicates that an overcurrent condition has been sensed. When 0, no overcurrent condition has been sensed. Because OMAP5910 does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0.	R	0
0	LPS	Local power status The root hub does not support the local power status feature. This bit always reads as 0. Write of 0 has no effect. Write of 1 when in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), a write of 1 turns off power to those ports whose corresponding PortPowerControlMask bit is 0. Because OMAP5910 does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect.	R/W	0

The HC port 1 status and control register reports and controls the state of USB host port 1. HcRhPortStatus1 can provide status and control for the OMAP5910 USB port associated with the OMAP5910 integrated USB transceiver according to the HMC_MODE value. See Section 5, *USB Pin Multiplexing*.

Table 23. HC Port 1 Status and Control Register (HcRhPortStatus1)

Bits	Field	Description	Type	Reset Value
31–21	Reserved	Reserved		
20	PRSC	Port 1 reset status change This bit indicates, when 1, that the port 1 port reset status bit has changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

Table 23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bits	Field	Description	Type	Reset Value
19	OCIC	<p>Port 1 overcurrent indicator change</p> <p>This bit indicates, when 1, that the port 1 port overcurrent indicator has changed.</p> <p>Write of 0 has no effect.</p> <p>Write of 1 clears this bit.</p> <p>The OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p>	R/W	0
18	PSSC	<p>Port 1 suspend status change</p> <p>This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low speed EOP, and 3-ms synchronization delays have been completed.</p> <p>Write of 0 has no effect.</p> <p>Write of 1 clears this bit.</p>	R/W	0
17	PESC	<p>Port 1 enable status change</p> <p>This bit indicates, when 1, that the port 1 port enable status changed.</p> <p>Write of 0 has no effect.</p> <p>Write of 1 clears this bit.</p>	R/W	0
16	CSC	<p>Port 1 connect status change</p> <p>This bit indicates, when 1, that the port 1 current connect status has changed due to a connect or disconnect event. If CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, then this bit is set.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p> <p>If the HcRhDescriptorB.DR[1] bit is set to indicate a nonremovable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.</p>	R/W	0
15–10	Reserved	Reserved		

Table 23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bits	Field	Description	Type	Reset Value
9	LSDA/CPP	<p>Port 1 low-speed device attached/clear port power</p> <p>When read as 1, this bit indicates that a low-speed device is attached to port 1. A 0 in this bit indicates a full speed device.</p> <p>This bit is only valid when port 1 CurrentConnectStatus is 1.</p> <p>The host controller driver can write a 1 to this bit to clear the port 1 PortPowerStatus. A write of 0 to this bit has no effect.</p> <p>The OMAP5910 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.</p>	R/W	0
8	PPS/SPP	<p>Port 1 port power status/set port power</p> <p>When read as 1, this bit indicates that the port 1 power is enabled. When read as 0, port 1 power is not enabled.</p> <p>The OMAP5910 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control.</p> <p>A write of 1 to this bit sets the port 1 port power status bit. A write of 0 has no effect.</p>	R/W	1
7–5	Reserved	Reserved		
4	PRS/SPR	<p>Port 1 port reset status/set port reset</p> <p>When read as 1, indicates that port 1 is receiving the USB reset signaling. When read as 0, USB reset is not being sent to port 1.</p> <p>A write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1. A write of 0 to this bit has no effect.</p>	R/W	0

Table 23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bits	Field	Description	Type	Reset Value
3	POCI/CSS	<p>Port 1 port overcurrent indicator/clear suspend status</p> <p>When read as 1, indicates a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred.</p> <p>OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p> <p>A write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0
2	PSS/SPS	<p>Port 1 port suspend status/set port suspend</p> <p>When read as 1, indicates that port 1 is in the USB suspend state or is in the resume sequence. When 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.</p> <p>If port 1 CurrentConnectStatus is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.</p>	R/W	0
1	PES/SPE	<p>Port 1 port enable status/set port enable</p> <p>When read as 1, indicates that port 1 is enabled. When read as 0, this bit indicates that port 1 is not enabled. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.</p> <p>A write of 1 to this bit when port 1 CurrentConnectStatus is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0

Table 23. HC Port 1 Status and Control Register (HcRhPortStatus1) (Continued)

Bits	Field	Description	Type	Reset Value
0	CCS/CPE	<p>Port 1 current connection status/clear port enable</p> <p>When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1.</p> <p>This bit is set to 1 after root hub reset if the HcRhDescriptorB.DR[1] bit is set to indicate a non-removable device on port 1.</p> <p>A write of 1 to this bit clears the port 1 port enable bit. A write of 0 to this bit has no effect.</p>	R/W	0
<p>The HC port 2 status register reports and controls the state of USB host port 2. Depending on HMC_MODE value, HcRhPortStatus2 can provide status and control for the OMAP5910 USB Port 1 pins:</p> <ul style="list-style-type: none"> <input type="checkbox"/> CLK32K_OUT/USB1.SPEED <input type="checkbox"/> BOOT/USB1.SUSP <input type="checkbox"/> RST_HOST_OUT/USB1_SE0 <input type="checkbox"/> MCBSP.CLK/USB1_TXEN <input type="checkbox"/> MCS11.DOUT/USB1.TXD <input type="checkbox"/> MCS11.SYNC/USB1.VP <input type="checkbox"/> MCS11.CLK/USB1.VM <input type="checkbox"/> MCS11.DIN/USB1.RCV 				

Table 24. HC Port 2 Status and Control Register (HcRhPortStatus2)

Bits	Field	Description	Type	Reset Value
31–21	Reserved	Reserved		
20	PRSC	<p>Port 2 reset status change</p> <p>This bit indicates, when 1, that the port 2 port reset status bit has changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0

Table 24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bits	Field	Description	Type	Reset Value
19	OCIC	<p>Port 2 overcurrent indicator change</p> <p>This bit indicates, when 1, that the port 2 port overcurrent indicator has changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p> <p>The OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p>	R/W	0
18	PSSC	<p>Port 2 suspend status change</p> <p>This bit indicates, when 1, that the port 2 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0
17	PESC	<p>Port 2 enable status change</p> <p>This bit indicates, when 1, that the port 2 port enable status changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0
16	CSC	<p>Port 2 connect status change</p> <p>This bit indicates, when 1, that the port 2 current connect status has changed due to a connect or disconnect event. If CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, then this bit is set.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p> <p>If the HcRhDescriptorB.DR[2] bit is set to indicate a nonremovable USB device on port 2, this bit is only set after a root hub reset to inform the system that the device is attached.</p>	R/W	0
15–10	Reserved	Reserved		

Table 24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bits	Field	Description	Type	Reset Value
9	LSDA/CPP	<p>Port 2 low-speed device attached/clear port power</p> <p>This bit indicates, when read as 1, that a low-speed device is attached to port 2. A 0 in this bit indicates a full-speed device.</p> <p>This bit is only valid when port 2 CurrentConnectStatus is 1.</p> <p>The host controller driver can write a 1 to this bit to clear the port 2 PortPowerStatus. A write of 0 to this bit has no effect.</p> <p>The OMAP5910 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.</p>	R/W	0
8	PPS/SPP	<p>Port 2 port power status/set port power</p> <p>This bit indicates, when read as 1, that the port 2 power is enabled. When read as 0, port 2 power is not enabled.</p> <p>The OMAP5910 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits has no direct effect on external port power control.</p> <p>A write of 1 to this bit sets the port 2 port power status bit. A write of 0 has no effect.</p>	R/W	1
7–5	Reserved	Reserved		
4	PRS/SPR	<p>Port 2 port reset status/set port reset</p> <p>When read as 1, indicates that port 2 is receiving the USB reset signaling. When read as 0, USB reset is not being sent to port 2.</p> <p>A write of 1 to this bit sets the port 2 port reset status bit and cause the USB host controller to begin signaling USB reset to port 2. A write of 0 to this bit has no effect.</p>	R/W	0

Table 24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bits	Field	Description	Type	Reset Value
3	POCI/CSS	<p>Port 2 port overcurrent indicator/clear suspend status</p> <p>When read as 1, indicates that a port 2 port overcurrent condition has occurred. When 0, no port 2 port overcurrent condition has occurred.</p> <p>The OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p> <p>A write of 1 to this bit when port 2 port suspend status is 1 causes resume signaling on port 2. A write of 1 when port 2 port suspend status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0
2	PSS/SPS	<p>Port 2 port suspend status/set port suspend</p> <p>When read as 1, indicates that port 2 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 2 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.</p> <p>If port 2 CurrentConnectStatus is 1, a write of 1 to this bit sets the port 2 port suspend status bit and places port 2 in USB suspend state. If CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.</p>	R/W	0
1	PES/SPE	<p>Port 2 port enable status/set port enable</p> <p>When read as 1, this bit indicates that port 2 is enabled. When read as 0, it indicates that port 2 is not enabled. This bit is automatically set at completion of port 2 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.</p> <p>A write of 1 to this bit when port 2 CurrentConnectStatus is 1 sets the port 2 port enable status bit. A write of 1 when port 2 current connect status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0

Table 24. HC Port 2 Status and Control Register (HcRhPortStatus2) (Continued)

Bits	Field	Description	Type	Reset Value
0	CCS/CPE	<p>Port 2 current connection status/clear port enable</p> <p>When read as 1, this bit indicates that port 2 currently has a USB device attached. When 0, it indicates that no USB device is attached to port 2.</p> <p>This bit is set to 1 after root hub reset if the HcRhDescriptorB.DR[2] bit is set to indicate a non-removable device on port 2.</p> <p>A write of 1 to this bit clears the port 2 port enable bit. A write of 0 to this bit has no effect.</p>	R/W	0
<p>The HC port 3 status and control register reports and controls the state of USB host port 3. Depending on HMC_MODE value, HcRhPortStatus2 can provide status and control for the OMAP5910 USB port 2 pins:</p> <ul style="list-style-type: none"> <input type="checkbox"/> MCSI2.CLK/USB2_SUSP <input type="checkbox"/> UART2.RTS/USB2_SE0 <input type="checkbox"/> MCSI2.DOUT/USB2.TXEN <input type="checkbox"/> UART2.TX/USB2.TXD <input type="checkbox"/> MCSI2.DIN/USB2.VP <input type="checkbox"/> UART2.RX/USB2.VM <input type="checkbox"/> UART2.CTS/USB2.RCV. 				

Table 25. HC Port 3 Status and Control Register (HcRhPortStatus3)

Bits	Field	Description	Type	Reset Value
31–21	Reserved	Reserved		
20	PRSC	<p>Port 3 reset status change</p> <p>This bit indicates, when 1, that the port 3 port reset status bit has changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0

Table 25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bits	Field	Description	Type	Reset Value
19	OCIC	<p>Port 3 overcurrent indicator change</p> <p>This bit indicates, when 1, that the port 3 port overcurrent indicator has changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p> <p>The OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p>	R/W	0
18	PSSC	<p>Port 3 suspend status change</p> <p>When 1, this bit indicates that the port 3 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0
17	PESC	<p>Port 3 enable status change</p> <p>This bit indicates, when 1, that the port 3 port enable status changed.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p>	R/W	0
16	CSC	<p>Port 3 connect status change</p> <p>When i, this bit indicates that the port 3 current connect status has changed due to a connect or disconnect event. If CurrentConnectStatus is 0 when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, then this bit is set.</p> <p>A write of 1 clears this bit. A write of 0 has no effect.</p> <p>If the HcRhDescriptorB.DR[3] bit is set to indicate a non-removable USB device on Port 3, this bit is only set after a root hub reset to inform the system that the device is attached.</p>	R/W	0
15–10	Reserved	Reserved		

Table 25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bits	Field	Description	Type	Reset Value
9	LSDA/CPP	<p>Port 3 low-speed device attached/clear port power</p> <p>This bit indicates, when read as 1, that a low-speed device is attached to port 3. A 0 in this bit indicates a full-speed device.</p> <p>This bit is only valid when port 3 CurrentConnectStatus is 1.</p> <p>The host controller driver may write a 1 to this bit to clear the port 3 PortPowerStatus. A write of 0 to this bit has no effect.</p> <p>OMAP5910 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.</p>	R/W	0
8	PPS/SPP	<p>Port 3 port power status/set port power</p> <p>When read as 1, this bit indicates that the port 3 power is enabled. When read as 0, port 3 power is not enabled.</p> <p>The OMAP5910 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software may track the current power state using the port power status bit and other power control bits, but those bits has no direct effect on external port power control.</p> <p>A write of 1 to this bit sets the port 3 port power status bit. A write of 0 has no effect.</p>	R/W	1
7–5	Reserved	Reserved		
4	PRS/SPR	<p>Port 3 port reset status/set port reset</p> <p>When read as 1, this bit indicates that port 3 is receiving the USB reset signaling. When read as 0, USB reset is not being sent to port 3.</p> <p>A write of 1 to this bit sets the port 3 port reset status bit and cause the USB host controller to begin signaling USB reset to port 3. A write of 0 to this bit has no effect.</p>	R/W	0

Table 25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bits	Field	Description	Type	Reset Value
3	POCI/CSS	<p>Port 3 port overcurrent indicator/clear suspend status</p> <p>When read as 1, this bit indicates that a port 3 port overcurrent condition has occurred. When 0, no port 3 port overcurrent condition has occurred.</p> <p>The OMAP5910 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism.</p> <p>A write of 1 to this bit when port 3 port suspend status is 1 causes resume signaling on port 3. A write of 1 when port 3 port suspend status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0
2	PSS/SPS	<p>Port 3 port suspend status/set port suspend</p> <p>When read as 1, this bit indicates that port 3 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 3 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.</p> <p>If port 3 CurrentConnectStatus is 1, a write of 1 to this bit sets the port 3 port suspend status bit and places port 3 in USB suspend state. If CurrentConnectState is 0, a write of 1 instead sets ConnectStatusChange to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.</p>	R/W	0
1	PES/SPE	<p>Port 3 port enable status/set port enable</p> <p>When read as 1, this bit indicates that port 3 is enabled. When read as 0, this bit indicates that port 3 is not enabled. This bit is automatically set at completion of port 3 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.</p> <p>A write of 1 to this bit when port 3 CurrentConnectStatus is 1 sets the port 3 port enable status bit. A write of 1 when port 3 current connect status is 0 has no effect. A write of 0 has no effect.</p>	R/W	0

Table 25. HC Port 3 Status and Control Register (HcRhPortStatus3) (Continued)

Bits	Field	Description	Type	Reset Value
0	CCS/CPE	<p>Port 3 current connection status/clear port enable</p> <p>When read as 1, indicates that port 3 currently has a USB device attached. When 0, indicates that no USB device is attached to port 3.</p> <p>This bit is set to 1 after root hub reset if the HcRhDescriptorB.DR[3] bit is set to indicate a nonremovable device on port 3.</p> <p>A write of 1 to this bit clears the port 3 port enable bit. A write of 0 to this bit has no effect.</p>	R/W	0

The host UE address register reports the local bus virtual address of the last local bus access which caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 26. Host UE Address Register (HostUEAddr)

Bits	Field	Description	Type	Reset Value
31–0	UE_ADDR	<p>Unrecoverable error address</p> <p>This register captures the local bus virtual address of any local bus operation that is started by the USB host controller that encounters an unrecoverable error condition. This information, along with the information in HostUEStatus, can help a developer determine why the USB host issued a local bus access that resulted in an unrecoverable error.</p> <p>This register is not affected by local bus timeouts occurring when the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral.</p>	R	0x0000 0000

The host UE status register reports the local bus cycle type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 27. Host UE Status Register (HostUEStatus)

Bits	Field	Description	Type	Reset Value
31–1	Reserved	Reserved	R	xxxxxxx
0	UEAccess	<p>Access type when unrecoverable error occurred</p> <p>When an unrecoverable error occurs due to timeout of a local bus write, this bit is set. When an unrecoverable error occurs due to timeout of a local bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs.</p> <p>This information, along with the information in HostUEAddr, can help a developer determine why the USB host issued a local bus access that resulted in an unrecoverable error.</p> <p>This register is not affected by local bus time-outs occurring when the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral.</p>		0

The host time-out control register controls the USB host controller local bus time-out mechanism. This register is not defined by the OHCI specification.

Table 28. Host Time-out Control Register (HostTimeoutCtrl)

Bits	Field	Description	Type	Reset Value
31–1	Reserved	Reserved		
0	TO_DIS	<p>Local bus time-out disable</p> <p>When 1, the USB host controller local bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory.</p> <p>When 0, the USB host controller waits indefinitely to access system memory. When cleared (the default state), the USB host controller waits no more than 4096 local bus clocks for completion of a local bus access to system memory. If the local bus cycle does not complete in that time, the USB host controller signals an unrecoverable error.</p> <p>This bit has no effect on MPU, DSP, or DMA controller accesses to local bus slave peripherals.</p>	R/W	0

The host revision register returns the revision number for the OMAP5910 USB host controller. This register is not defined by the OHCI specification.

Table 29. Host Revision Register (HostRevision)

Bits	Field	Description	Type	Reset Value
31–8	Reserved	Reserved	R	xxxxxx
7–4	MajorRev	Major revision number MajorRev indicates the major revision number of the USB host controller. The original OMAP5910 USB host controller version implements a major revision number of 0.	R	xx
3–0	MinorRev	Minor revision number MinorRev indicates the minor revision number of the USB host controller. The original OMAP5910 USB host controller version implements a minor revision number of 0.	R	xx

3.1 USB Host Controller Reserved Registers and Reserved Bit Fields

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

3.2 Endianism and USB Host Controller Registers

The OMAP5910 USB host controller assumes that all MPU accesses to its registers are 32-bit accesses. This restriction means that the host controller driver software may operate in either big-endian or little-endian without having to perform endian conversion on USB host controller register accesses. Software that uses 16-bit or 8-bit accesses to USB host controller registers does not work correctly, regardless of processor endianism mode.

The processor endianism does affect how the software must access the USB data structures and USB data buffers in system memory. See Section 6.5, *Endianism and USB Host Controller Access to System Memory* for details on endianism, data buffers, and data structures.

3.3 USB Host Controller Registers, USB Reset, and USB Clocking

When the USB host controller is not clocked (because the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit is 0), or when the ULPD does not provide 48 MHz to the USB host controller, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

The USB host controller completes its reset within about 72 clock cycles after CONF_MOD_USB_HOST_HHC_UHOST_EN_R is active and the ULPD begins providing clock to the USB host controller. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HcRevision register and the HcHCCA register return their correct reset default values.

4 USB Host Controller Interrupt Sources

4.1 OHCI Interrupts

The OMAP5910 USB host controller provides an interrupt output to the MPU level 2 interrupt handler on its IRQ_06 interrupt input. This is a level-sensitive interrupt signal, and the MPU level 2 interrupt handler IRQ_06 must be programmed as a level-sensitive input.

4.1.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

4.1.2 OHCI HcDoneHead Writeback Interrupt

The OHCI HcDoneHead writeback interrupt is supported as described in the *OHCI Specification for USB*.

4.1.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

4.1.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

4.1.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for USB*. This interrupt occurs if the USB host controller is unable to complete a local bus read or local bus write within 4096 local bus clocks when the USB host local bus timeout feature is enabled (see Table 28, *Host Timeout Control Register (HostTimeoutCtrl)*). When a local bus timeout causes an unrecoverable error, HostUEAddr and HostUEStatus are updated. When an isochronous TD is processed with an Offset/PSW field that is not set for *Not Accessed*, an unrecoverable error interrupt is generated but HostUEAddr and HostUeStatus are not updated.

4.1.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

4.1.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The OMAP5910 does not provide a connection between the USB host controller and USB port overcurrent detection hardware, so the root hub status change interrupt does not occur due to a port overcurrent event.

4.1.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

4.2 Local Bus MMU Interrupts

Interrupts from the local bus MMU to the MPU level 1 interrupt handler IRQ_17 input occur if the USB host controller attempts an access which the local bus MMU cannot perform. This interrupt can be an important tool when debugging a USB host controller driver and can be used to help identify operational faults in a final product.

This interrupt does not occur if the USB host controller attempts to access a local bus virtual address that is in the range 0x00000000 to 0x2FFFFFFF or 0x40000000 to 0xFFFFFFFF. For more detail on the local bus MMU, see Section 6, *USB Host Controller Access to System Memory*.

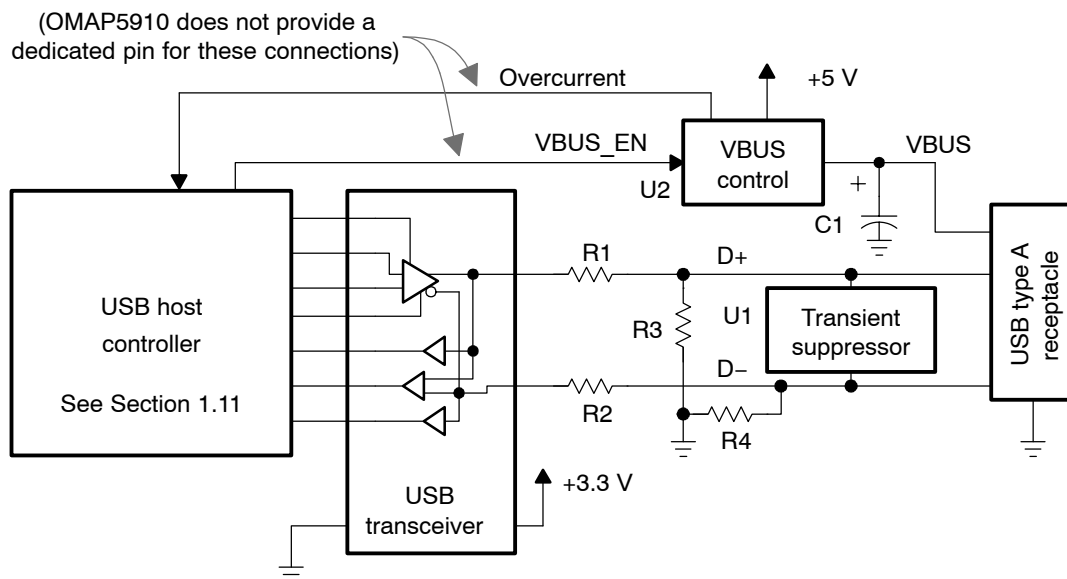
5 USB Pin Multiplexing

OMAP5910 USB signal multiplexing provides five main types of signaling: USB host and USB function with USB transceivers, USB host and USB function without USB transceivers, and UART1. This section describes first general external connectivity for these types of multiplexing and then the specific OMAP5910 USB multiplexing modes along with specific OMAP5910 connectivity for each mode.

5.1 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a USB-type A receptacle, power on the VBUS signal (may be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver. These elements are shown in Figure 3.

Figure 3. Typical USB Host Connections



R1, R2	Value depends on transceiver
R3, R4	15K Ohm +/- 5%
C1	Low ESR cap, minimum 120 uF
U1	Transient suppressor, such as SN65220, SN65240, or SN75240
U2	Power switch, such as TPS2014 or TPS2015

Because OMAP5910 does not provide a pin that connects to the USB host controller port power control registers, some other mechanism must be used if VBUS switching is required. Similarly, OMAP5910 does not provide any pins that connect to the USB host controller overcurrent status bits, so some other mechanism must be used if overcurrent sensing is required.

5.2 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB function controller must implement certain features. These features include a USB-type B receptacle, VBUS power detection, transient suppression, a controllable pullup resistor to the D+ or D- line, and USB-compatible upstream port transceiver. These elements are shown in Figure 4.

Figure 4. Typical USB Function Connections

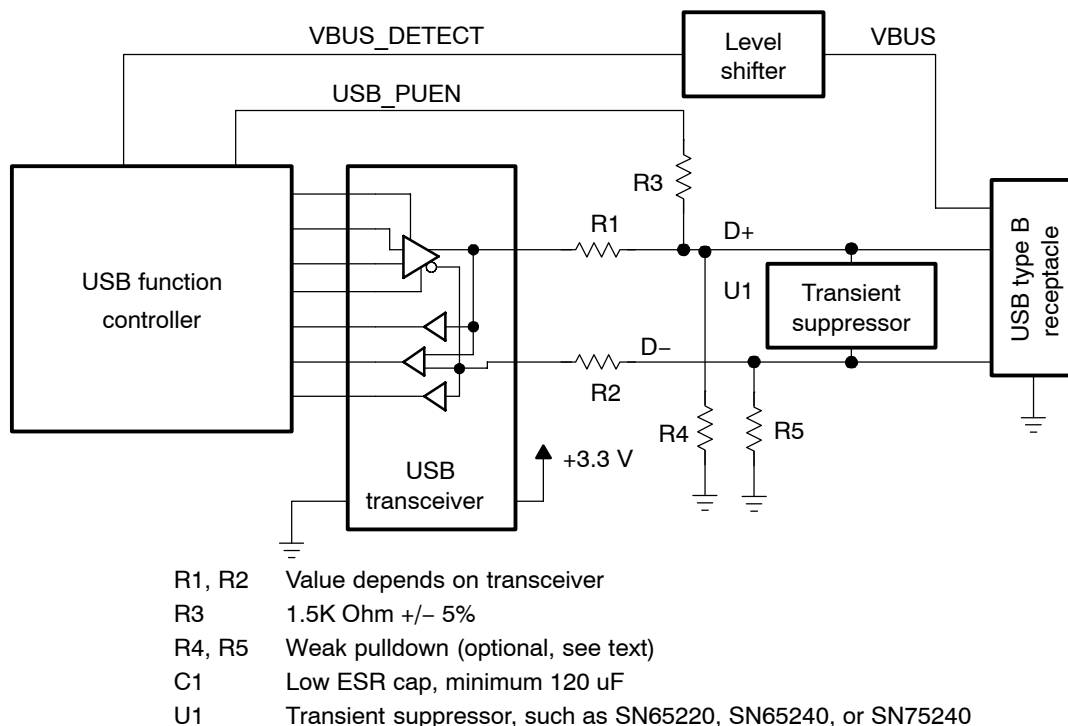


Figure 4 shows optional weak pulldown resistors R4 and R5. These can be used to hold the D+ and D- signals at voltages below the USB transceiver V_{IL} level when there is no USB host connected to the USB Type B connector. By keeping D+ and D- voltages below V_{IL} , the USB transceiver I_{DDQ} can be

reduced. Choice of value for R4 and R5 must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The OMAP5910 USB function controller only supports implementation as a full speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

5.3 On-Board Transceiverless Connection Using OMAP5910 Transceiverless Link Logic

The transceiverless link logic feature of the OMAP5910 USB signal multiplex-ing enables connection of the OMAP5910 device to an external, on-board USB host controller or external on-board USB function controller, without the use of USB transceivers or associated circuitry. When the transceiverless link logic is used, both of the USB transceivers, the series resistors, pullup and pull-down resistors, VBUS switching components, and USB connectors and cables that normally are used between a USB host controller and the downstream USB function controller are removed.

The transceiverless link logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP5910 device is used with an external USB integrated circuit which is on the same board.

When using the transceiverless link logic, six of the external USB integrated circuit pins that normally connect to a USB transceiver connect instead directly to OMAP5910 device pins. Signaling on these pins use CMOS levels.

Transceiverless link logic can be compared to a normal USB implementation as shown in Figure 5 and Figure 6. Figure 5 shows OMAP5910 being used as a USB host controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5910 transceiverless link logic. Figure 6 shows OMAP5910 used as a USB function controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5910 transceiverless link logic.

The transceiverless link logic function in the OMAP5910 device interprets the transmit control signals from the external USB integrated circuit and similar signals from the OMAP5910 USB host controller or OMAP5910 USB function controller and computes the equivalent USB differential pair state. The computed differential pair state is interpreted and the appropriate transceiver output signals are provided to the external USB integrated circuit and to the OMAP5910 USB host controller or OMAP5910 USB function controller.

Figure 5. OMAP5910 USB Host Controller Connection—With and Without the OMAP5910 Transceiverless Link Logic

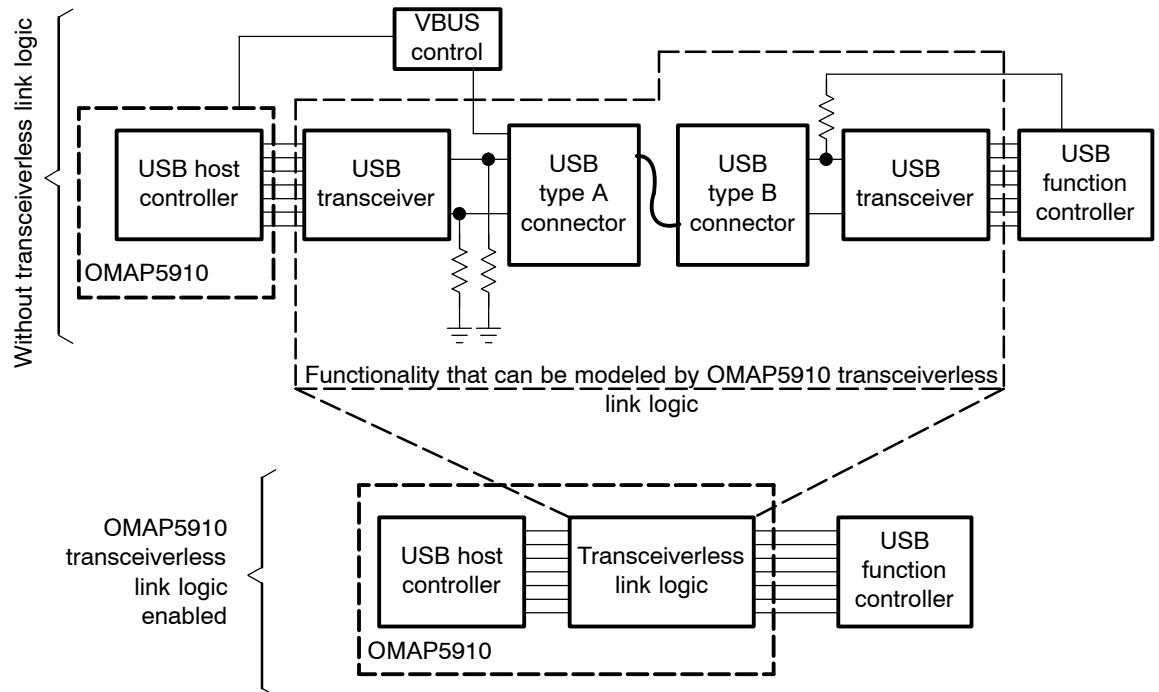
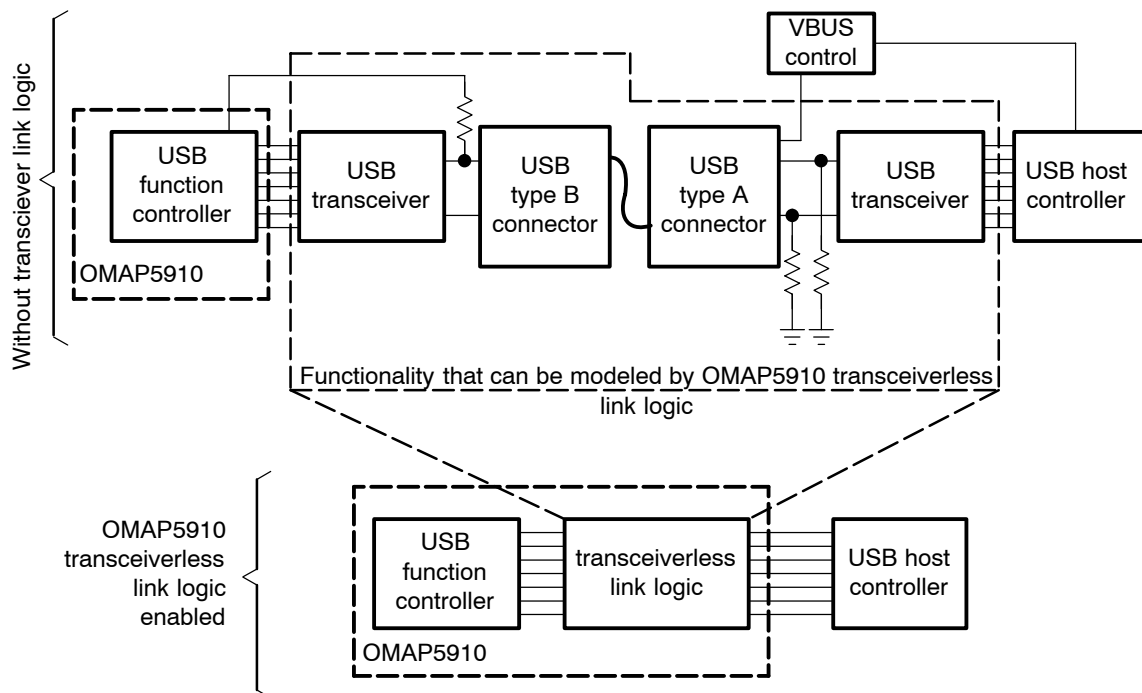


Figure 6. OMAP5910 USB Function Connection—With and Without the OMAP5910 Transceiverless Link Logic



5.4 USB Signal Multiplexing Mode Diagrams

The OMAP5910 USB signal multiplexing mechanisms provide a wide variety of options for bringing USB functionality to the OMAP5910 pins. These options are listed in Table 30 and are shown in Figure 7 through Figure 31. Each of these figures shows the external connectivity used to implement a typical system using one of the USB signal multiplexing modes. Each diagram assumes that OMAP5910 top-level signal multiplexing has been initialized to select the USB signal multiplexer as the source/destination for those signals shown as actively controlled by the USB signal multiplexing box. Top level pin muxing is configured via the OMAP5910 configuration registers described in section 6.8. In the figures, the items shown in gray do not receive or control OMAP5910 USB related pin signals for the HMC_MODE shown. For each configuration, appropriate external hardware is also required. This may include pullup or pulldown resistors, series resistors, USB transceiver devices, ESD protection devices, power switching circuitry, and USB connectors.

Table 30. USB Signal Multiplexing Modes

CONF_MOD_USB_HOST_HMC_MODE_R	USB Functions Available to OMAP5910		
	Integrated USB Transceiver Pins (USB.DM USB.DP)	Pin Group 1	Pin Group 2
0	USB function		
1	USB host port 1	USB host port 2	USB host port 3
2	USB host port 1	UART 1 [†]	USB host port 3
3	USB host port 1	USB function	USB host port 3
4	USB function	USB host port 2	USB host port 3
5	USB host port 1		USB host port 3
6	USB function		USB host port 3
7	USB host port 1	‡	‡
8Reserved. Do not use.	Reserved	Reserved	Reserved

[†] CONF_MOD_USB_HOST_HMC_MODE_R values that select UART 1 bring UART1 CTS, RX, and TX signals to pins that can, in other CONF_MOD_USB_HOST_HMC_MODE_R values, be used for USB.

[‡] CONF_MOD_USB_HOST_HMC_MODE_R 7 provides an internal signal path from six of the USB-related OMAP5910 input pins to six of the USB-related OMAP5910 output pins.

[§] CONF_MOD_USB_HOST_HMC_MODE_R 21 provides an internal signal path from USB host controller Port 3 to the OMAP5910 USB function via the transceiverless link logic.

Table 30. USB Signal Multiplexing Modes (Continued)

CONF_MOD_USB_HOST_HMC_MODE_R	USB Functions Available to OMAP5910		
	Integrated USB Transceiver Pins (USB.DM USB.DP)	Pin Group 1	Pin Group 2
9	USB host port 1	USB host port 2	USB host port 3 with TLL
10	USB host port 1	UART 1 [†]	USB host port 3 with TLL
11	USB host port 1	USB function	USB host port 3 with TLL
12	USB function	USB host port 2	USB host port 3 with TLL
13	USB host port 1		USB function with TLL
14	USB function		USB host port 3 with TLL
15	USB host port 1	USB host port 2	USB function with TLL
16	USB host port 1		
17	USB host port 1	USB host port 2	
18	USB host port 1	UART 1 [†]	
19	USB host port 1	USB function	
20	USB function	USB host port 2	
21 [§]	USB host port 1		
22			
23	USB host port 1	USB host port 2	USB host port 3 with TLL (TXD0–signaling)

[†] CONF_MOD_USB_HOST_HMC_MODE_R values that select UART 1 bring UART1 CTS, RX, and TX signals to pins that can, in other CONF_MOD_USB_HOST_HMC_MODE_R values, be used for USB.

[‡] CONF_MOD_USB_HOST_HMC_MODE_R 7 provides an internal signal path from six of the USB-related OMAP5910 input pins to six of the USB-related OMAP5910 output pins.

[§] CONF_MOD_USB_HOST_HMC_MODE_R 21 provides an internal signal path from USB host controller Port 3 to the OMAP5910 USB function via the transceiverless link logic.

Table 30. USB Signal Multiplexing Modes (Continued)

CONF_MOD_USB_HOST_HMC_MODE_R	USB Functions Available to OMAP5910		
	Integrated USB Transceiver Pins (USB.DM USB.DP)	Pin Group 1	Pin Group 2
24	USB host port 1	UART 1 [†]	USB host port 3 with TLL (TXD–signaling)
25	USB host port 1	USB function	USB host port 3 with TLL (TXD–signaling)
26–31			

[†] CONF_MOD_USB_HOST_HMC_MODE_R values that select UART 1 bring UART1 CTS, RX, and TX signals to pins that can, in other CONF_MOD_USB_HOST_HMC_MODE_R values, be used for USB.

[‡] CONF_MOD_USB_HOST_HMC_MODE_R 7 provides an internal signal path from six of the USB-related OMAP5910 input pins to six of the USB-related OMAP5910 output pins.

[§] CONF_MOD_USB_HOST_HMC_MODE_R 21 provides an internal signal path from USB host controller Port 3 to the OMAP5910 USB function via the transceiverless link logic.

Figure 7. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 0

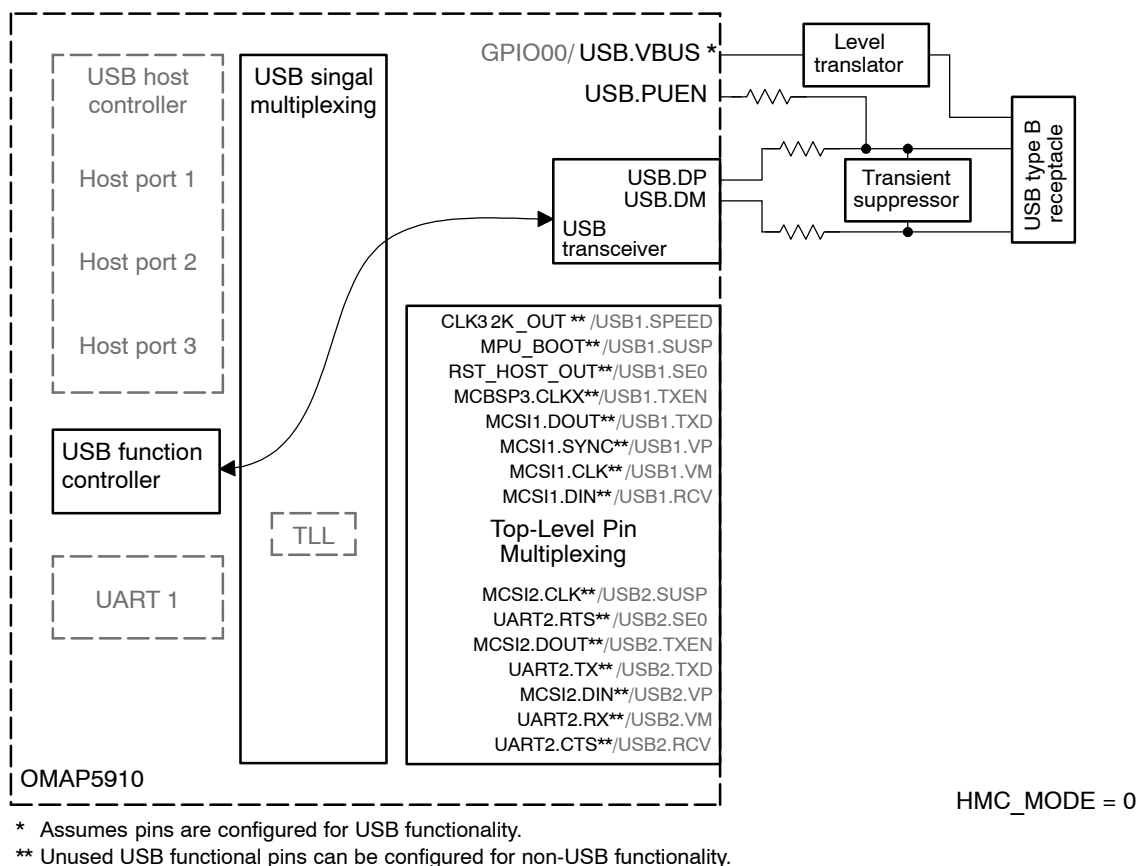
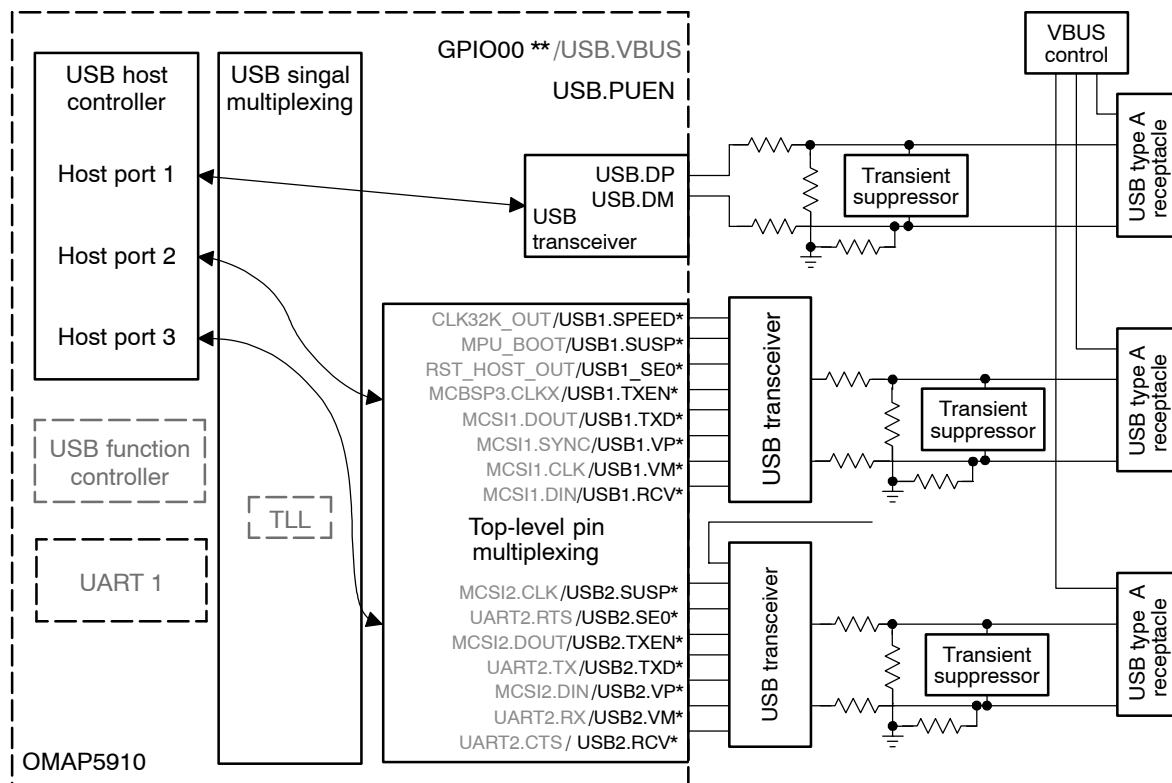


Figure 8. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 1

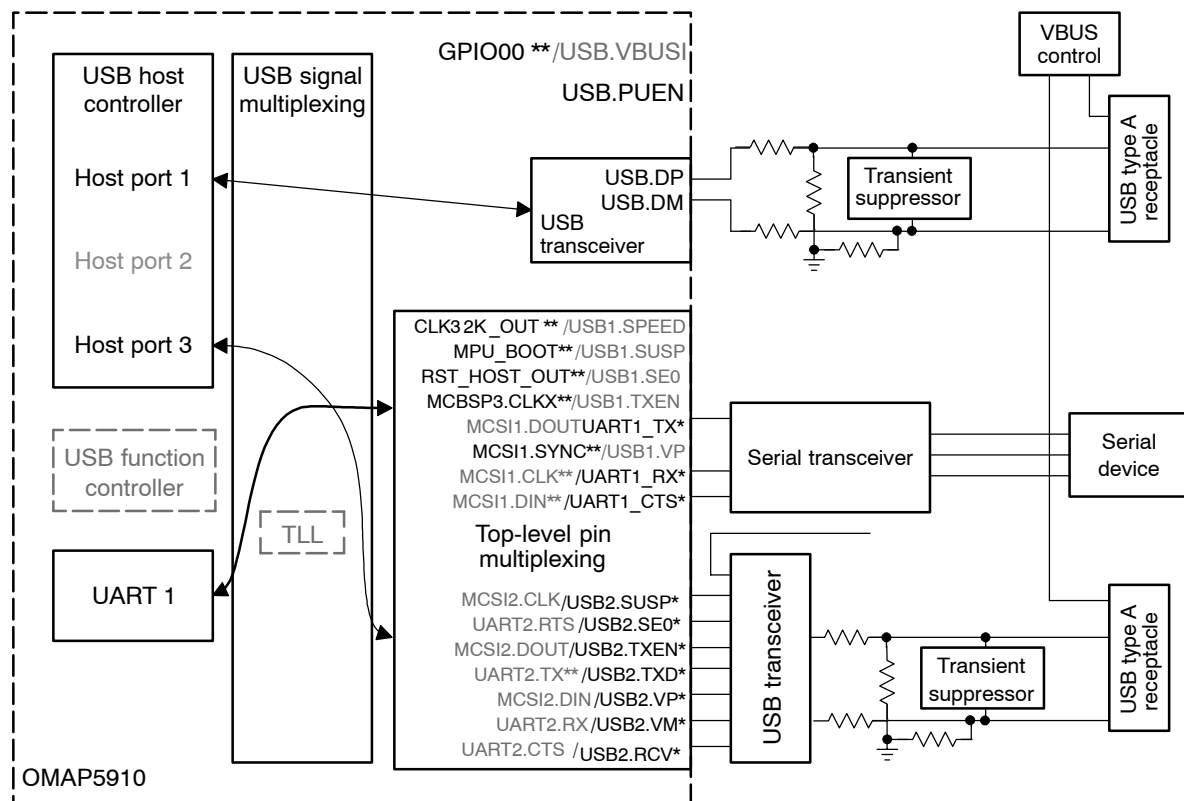


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 1

Figure 9. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 2

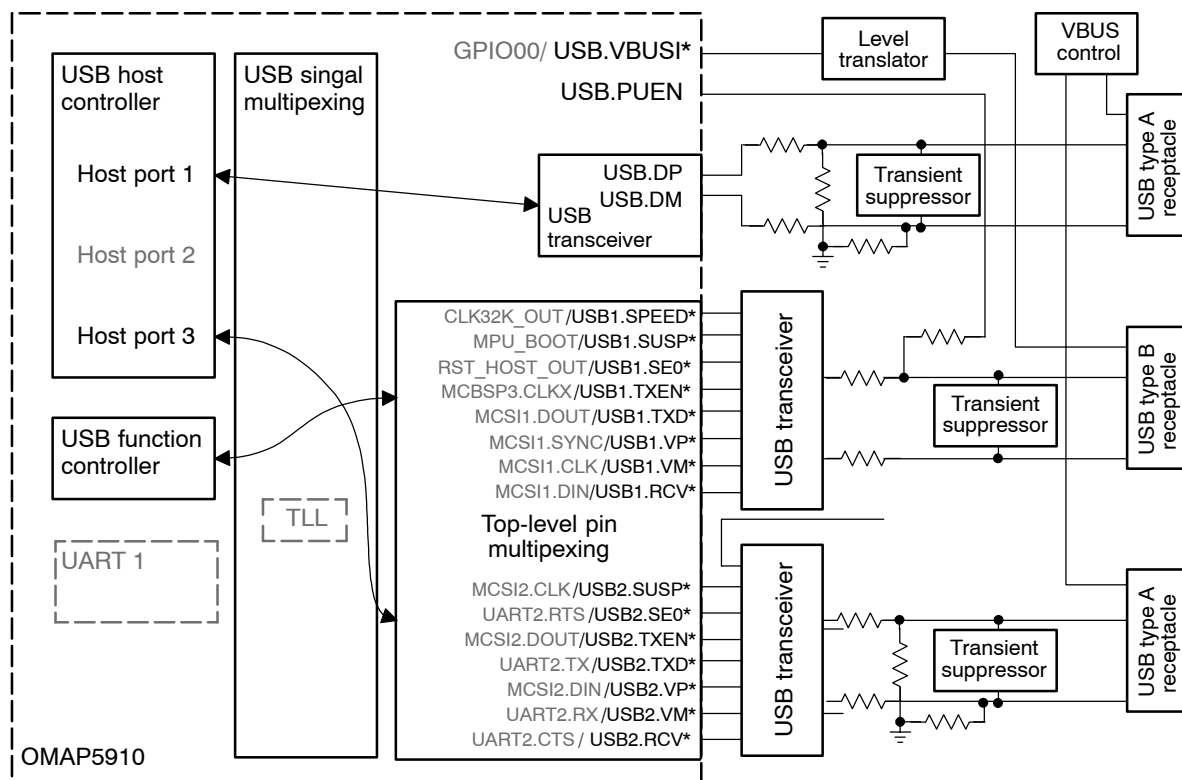


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 2

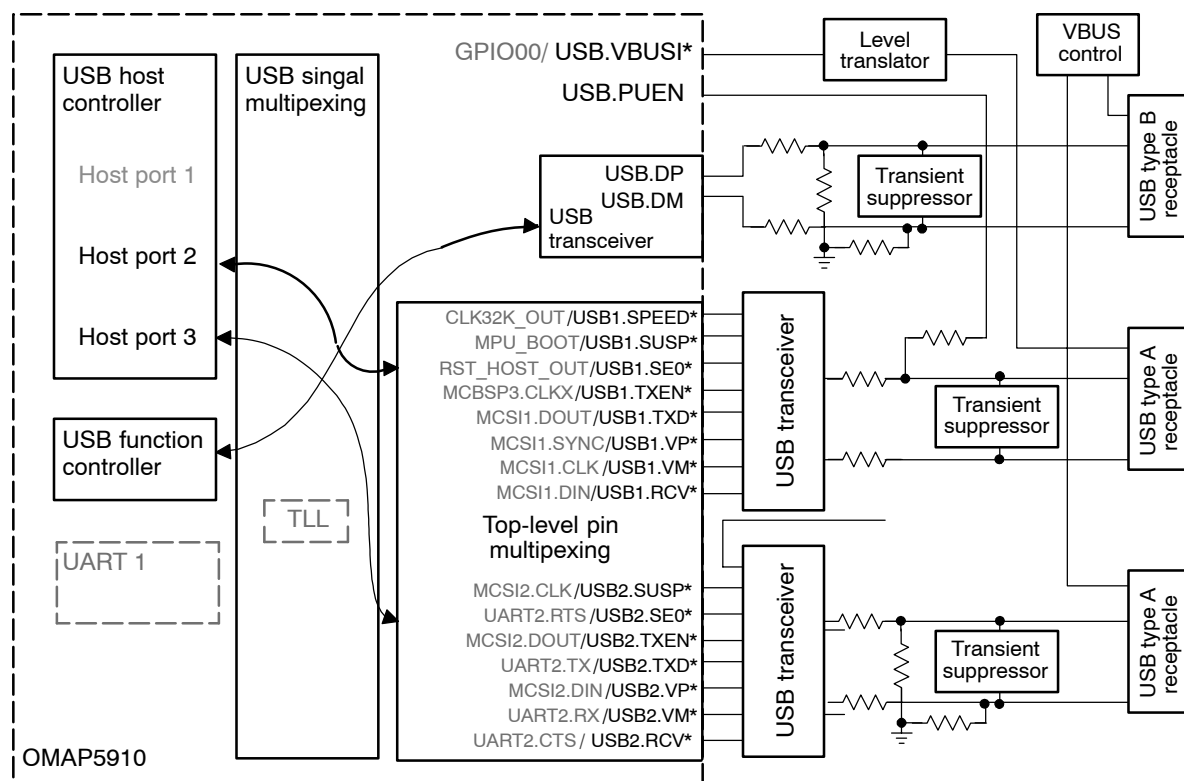
Figure 10. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 3



* Assumes pins are configured for USB functionality.

HMC_MODE = 3

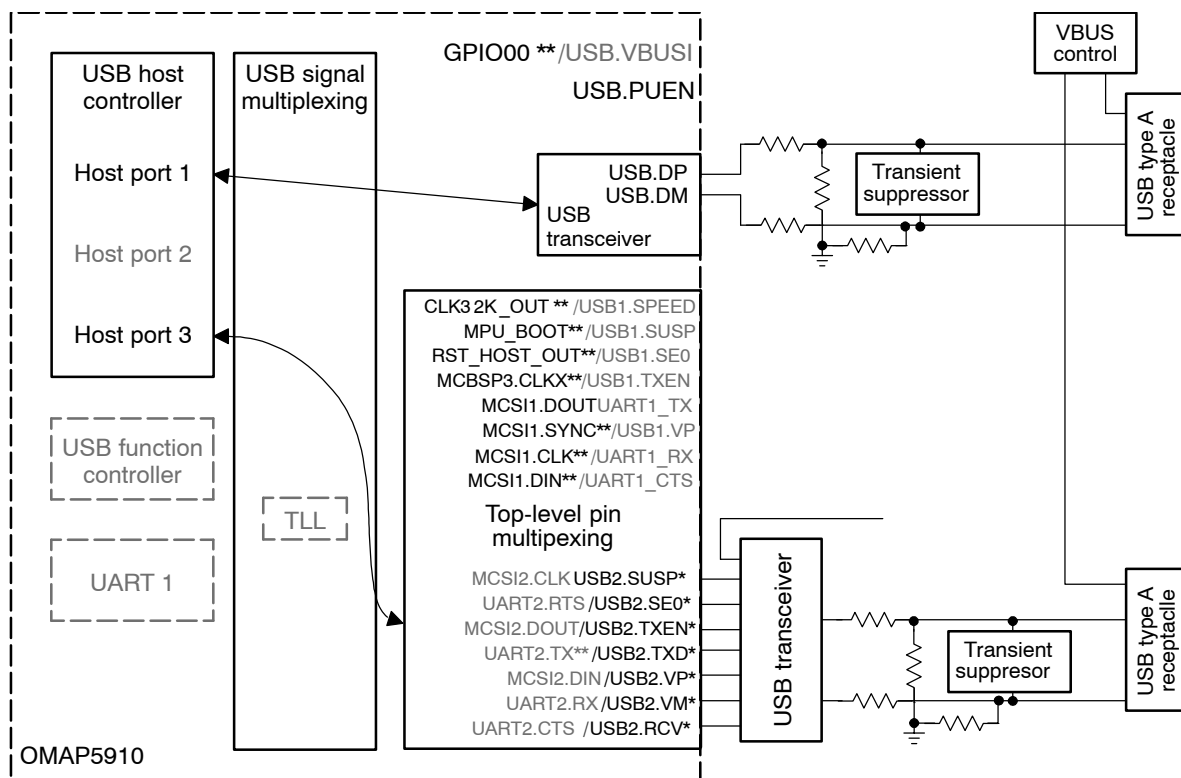
Figure 11. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 4



* Assumes pins are configured for USB functionality.

HMC_MODE = 4

Figure 12. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 5

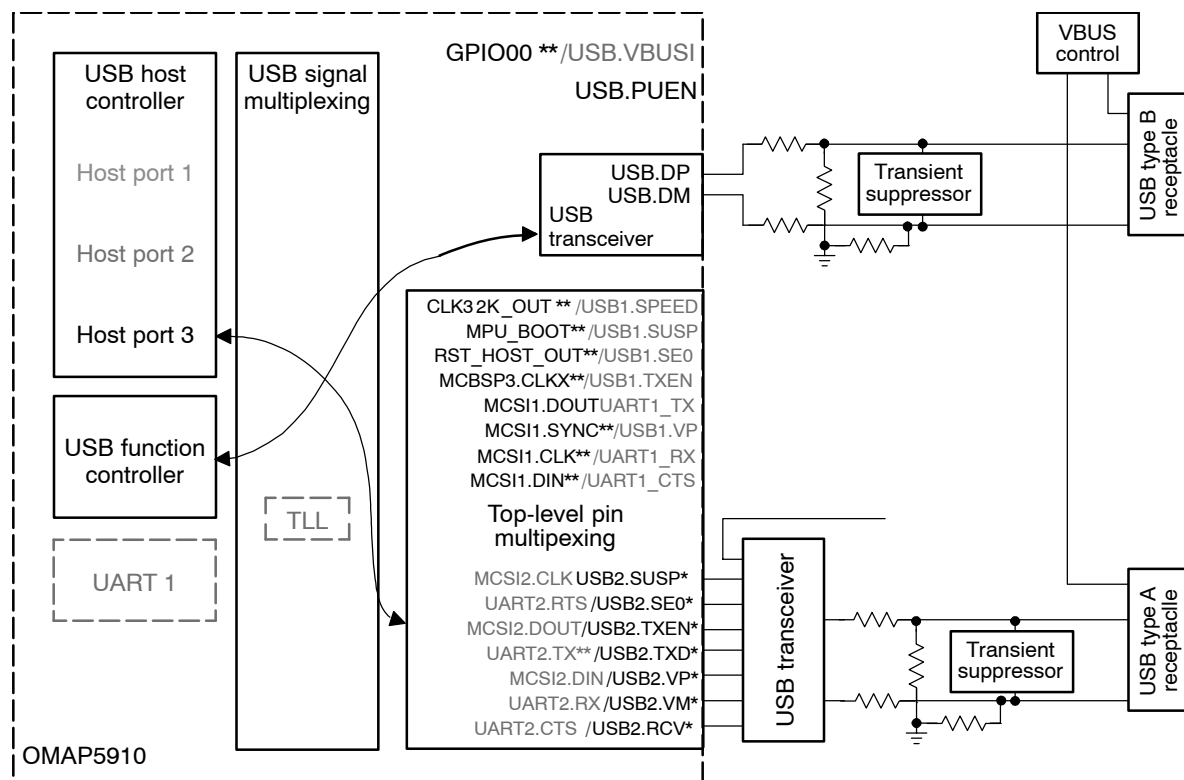


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 5

Figure 13. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 6

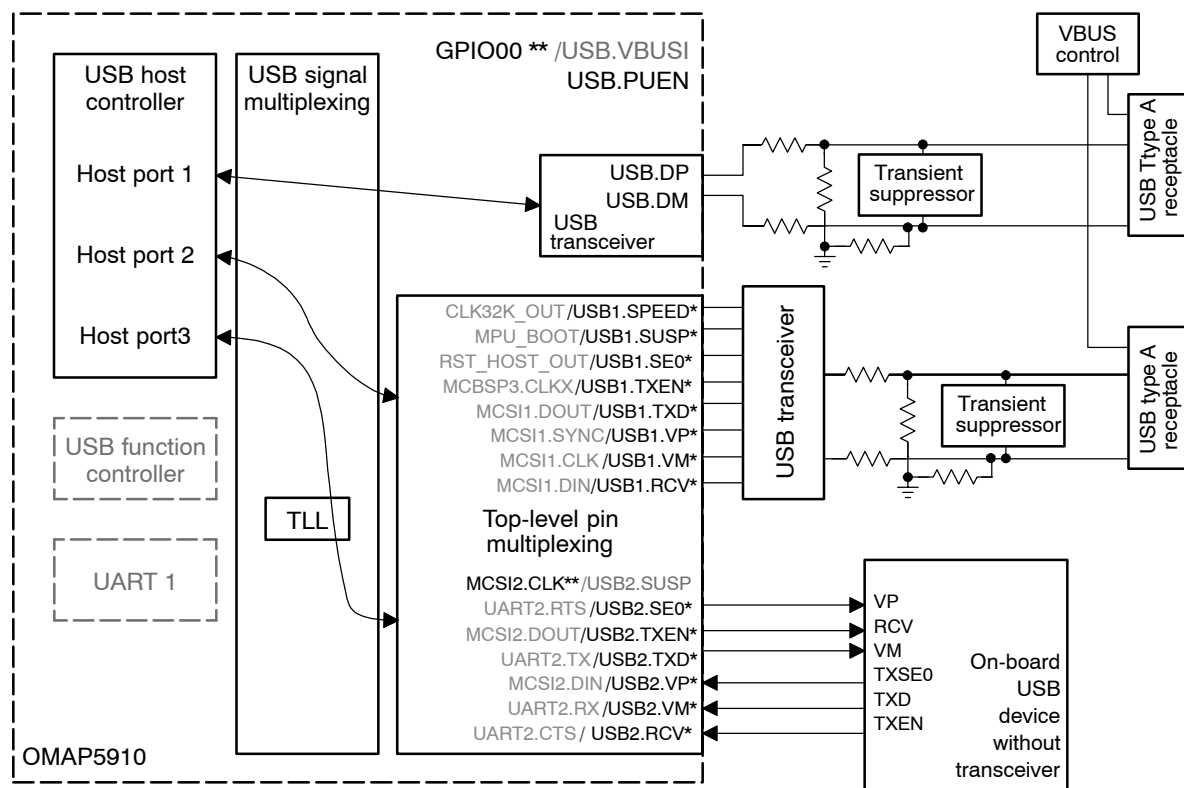


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 6

Figure 15. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 9

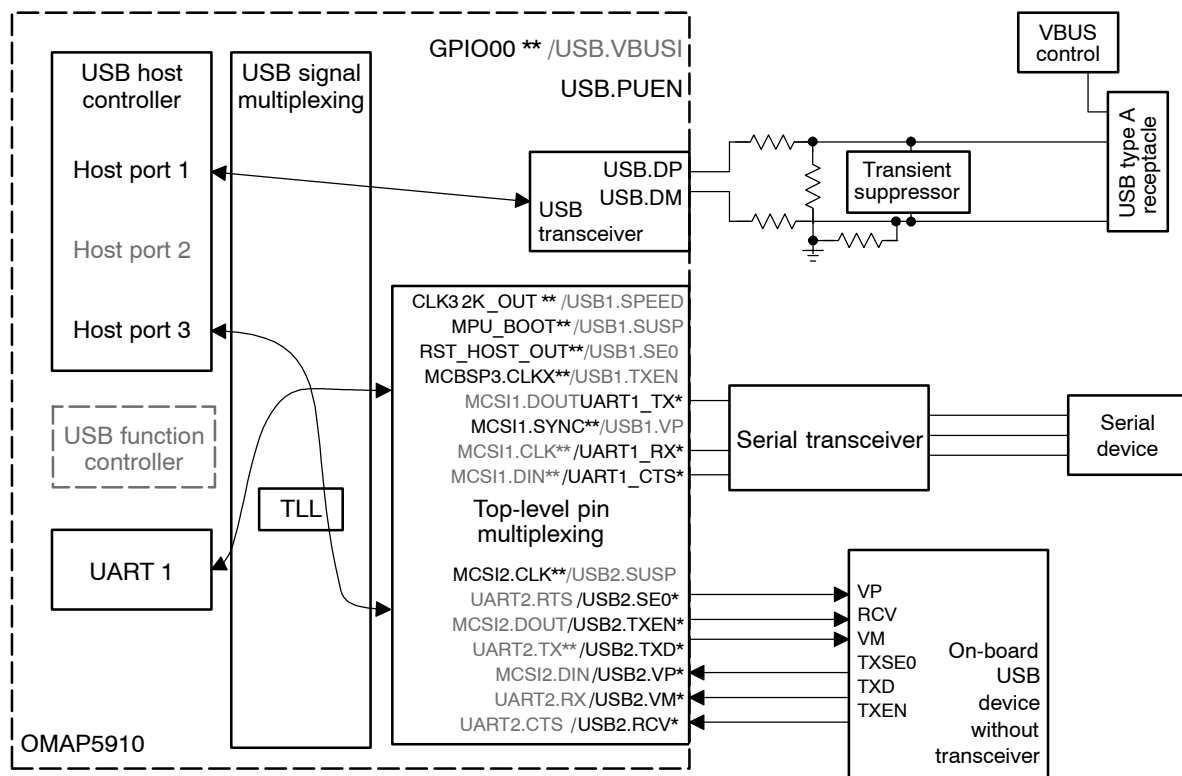


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 9

Figure 16. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 10

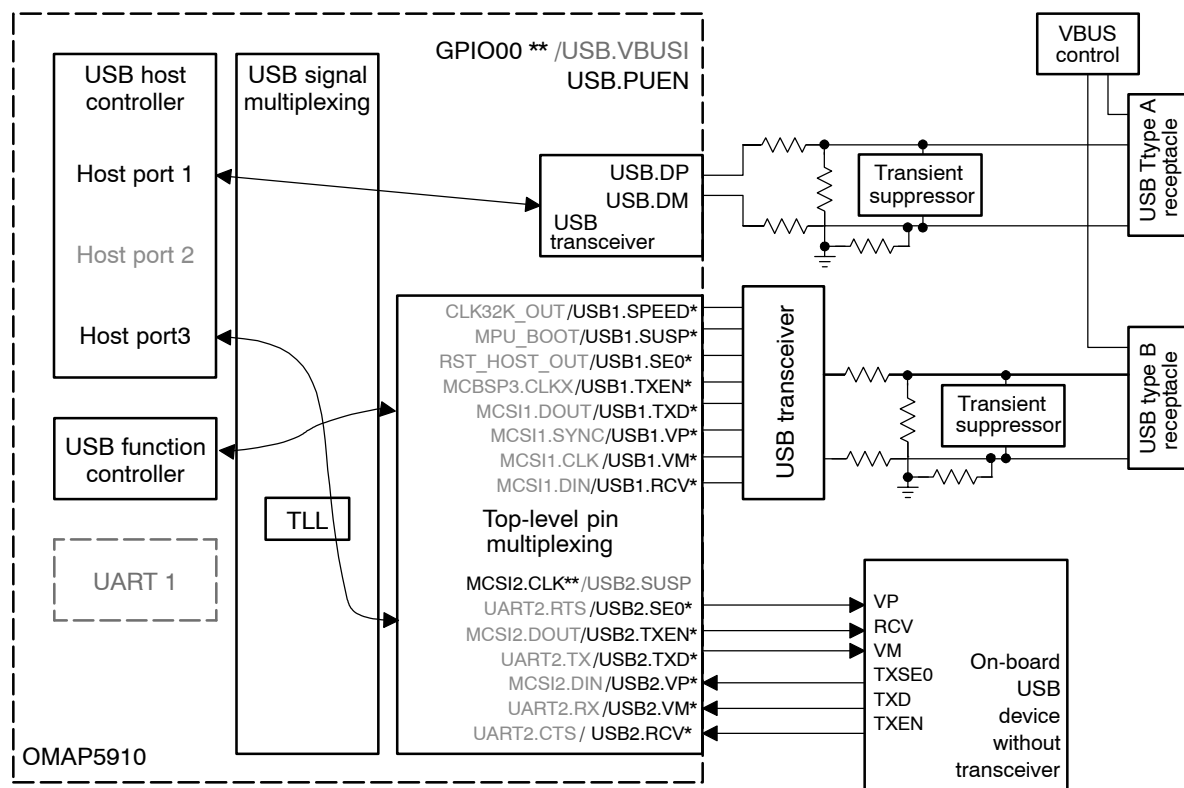


* Assumes pins are configured for USB functionality.

** used USB functional pins can be configured for non-USB functionality.

HMC_MODE = 10

Figure 17. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 11

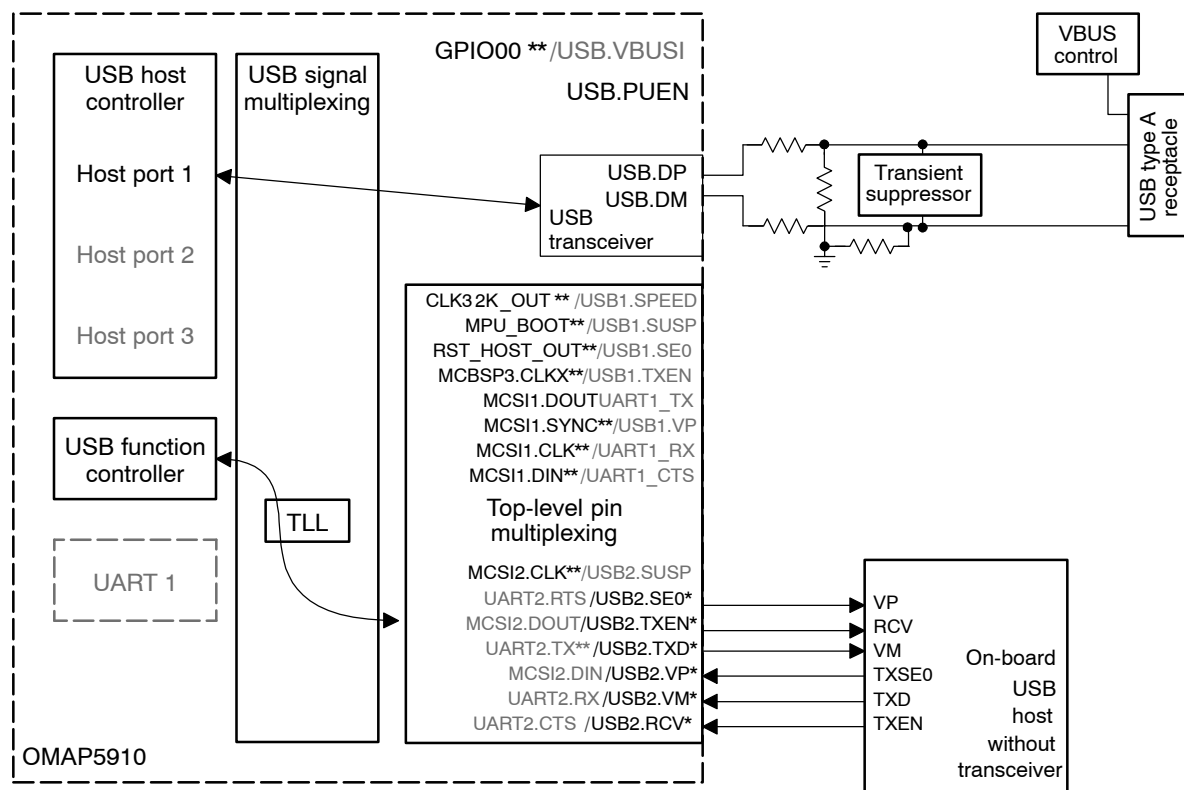


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 11

Figure 19. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 13

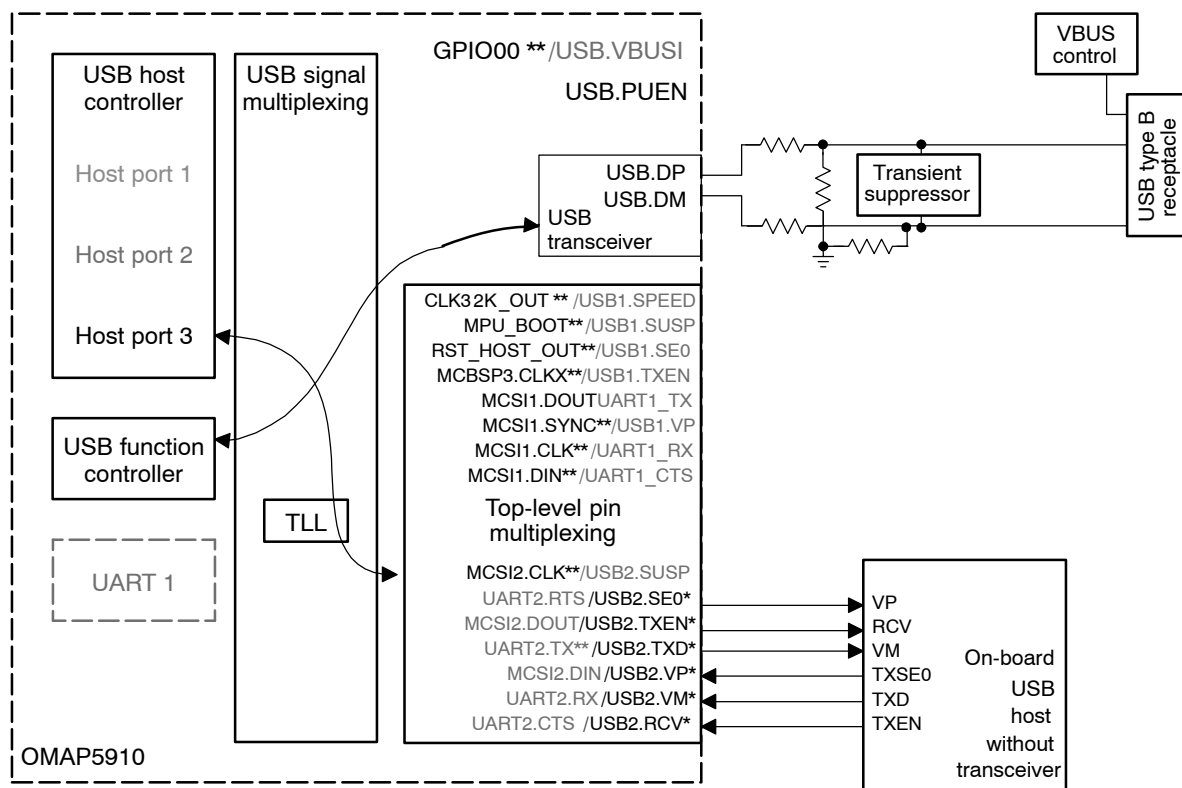


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 13

Figure 20. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 14

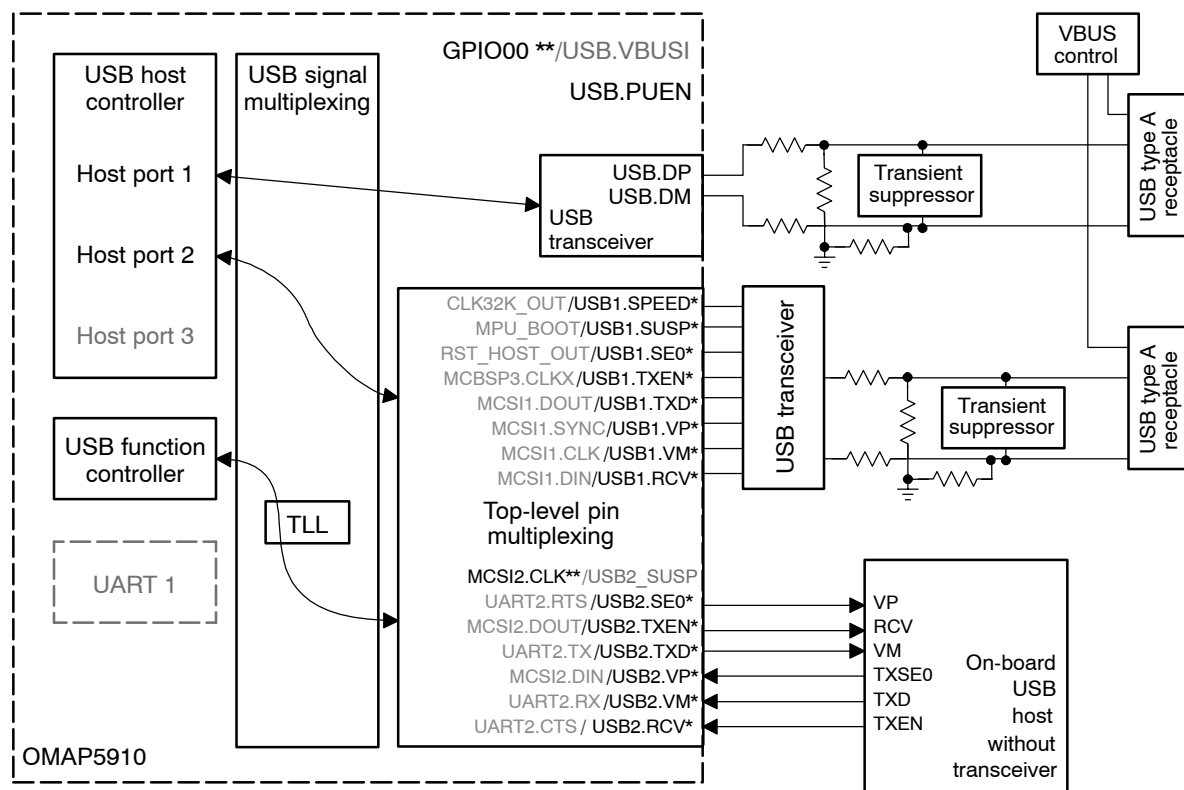


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 14

Figure 21. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 15

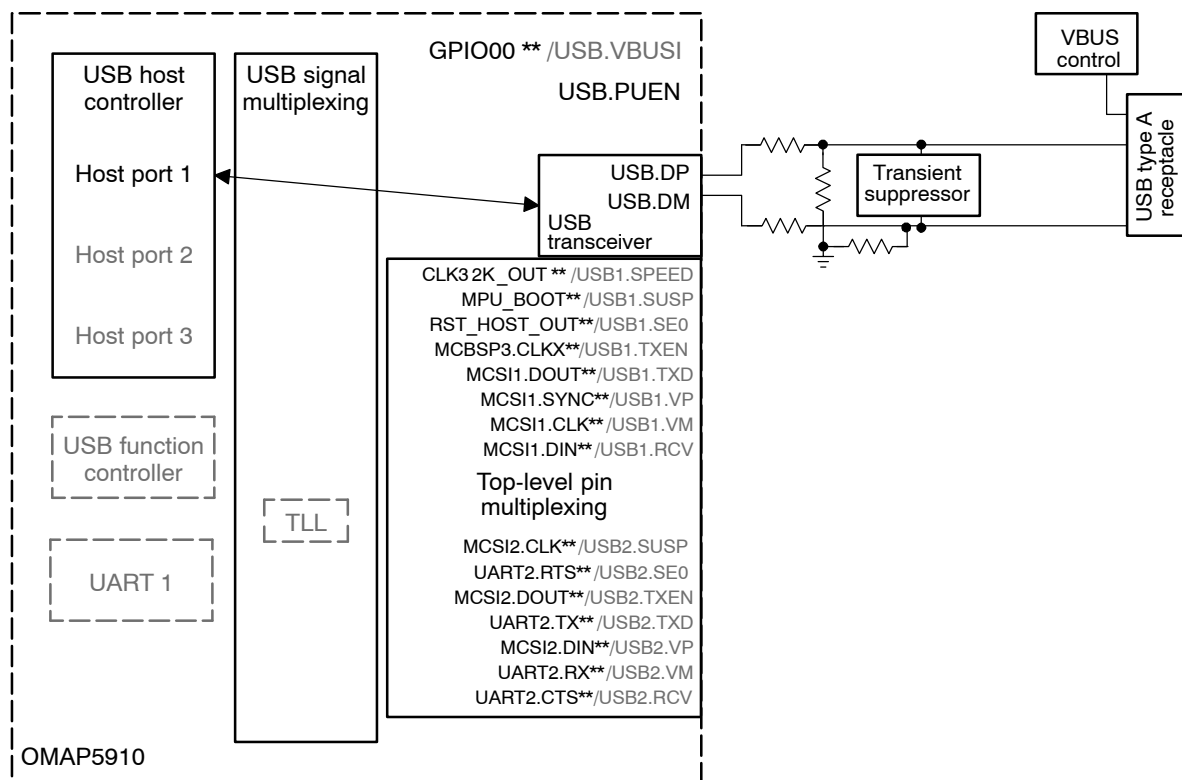


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 15

Figure 22. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 16

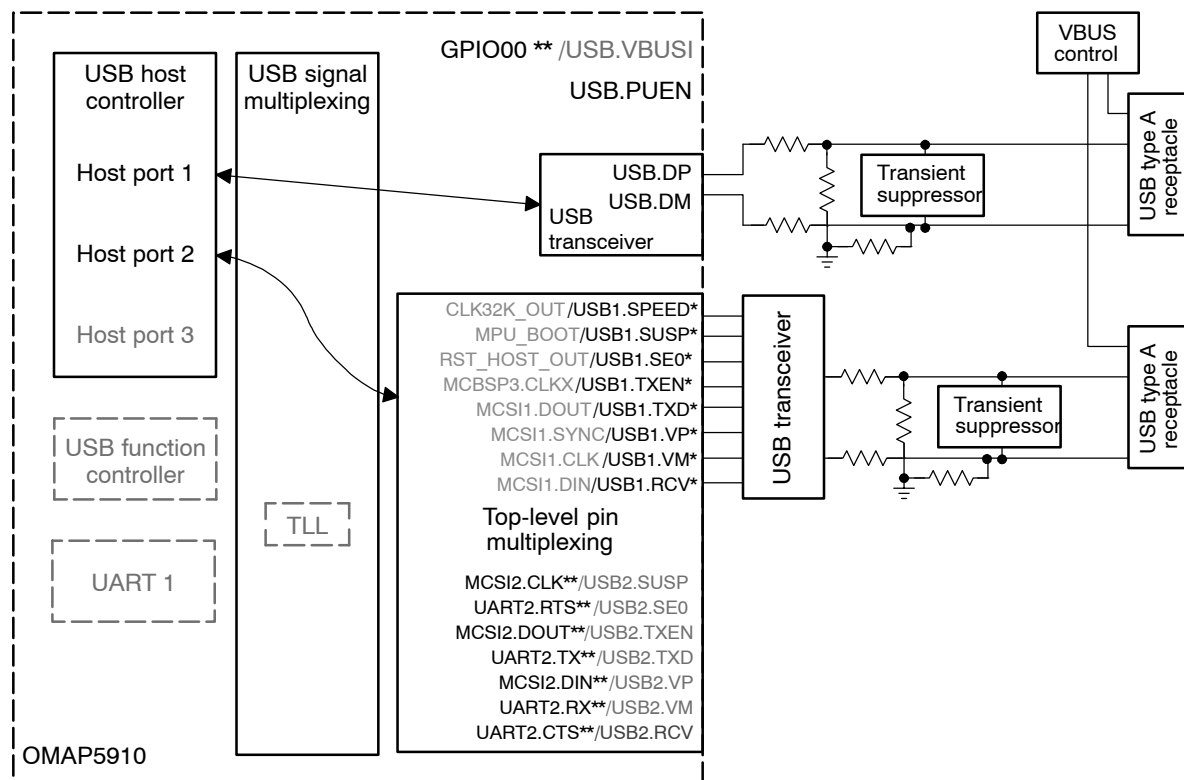


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 16

Figure 23. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 17

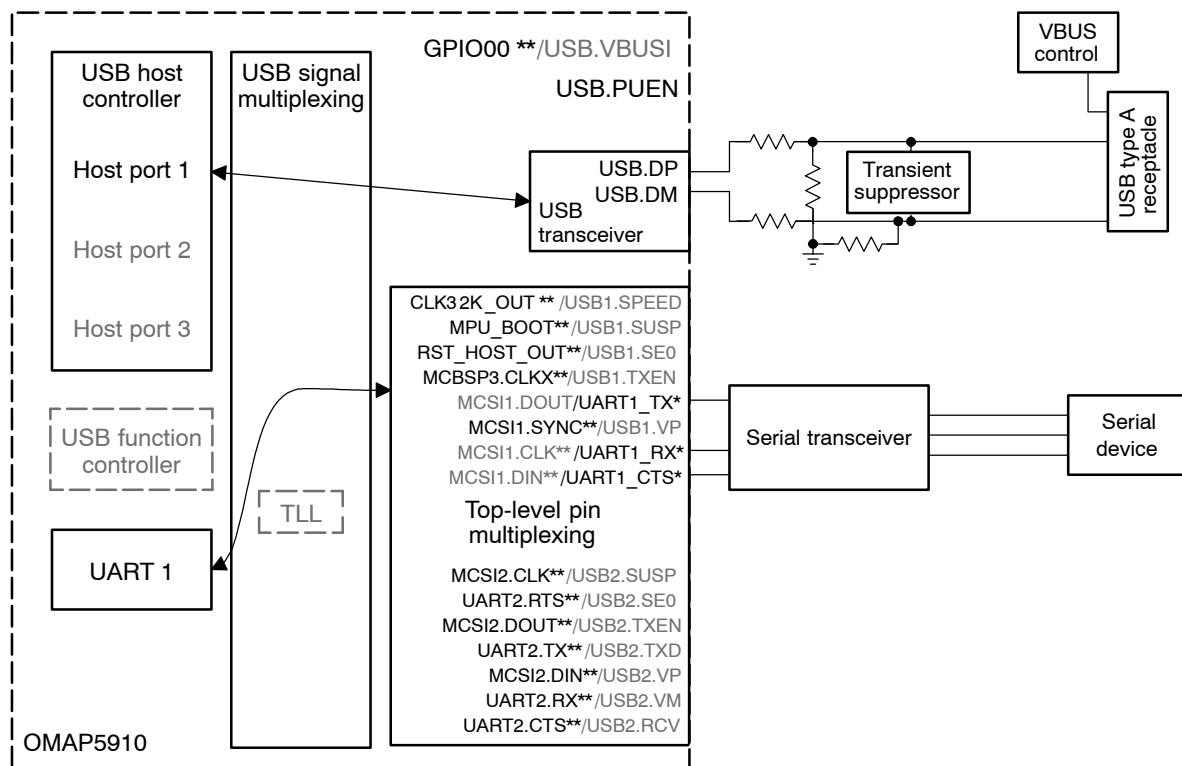


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 17

Figure 24. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 18

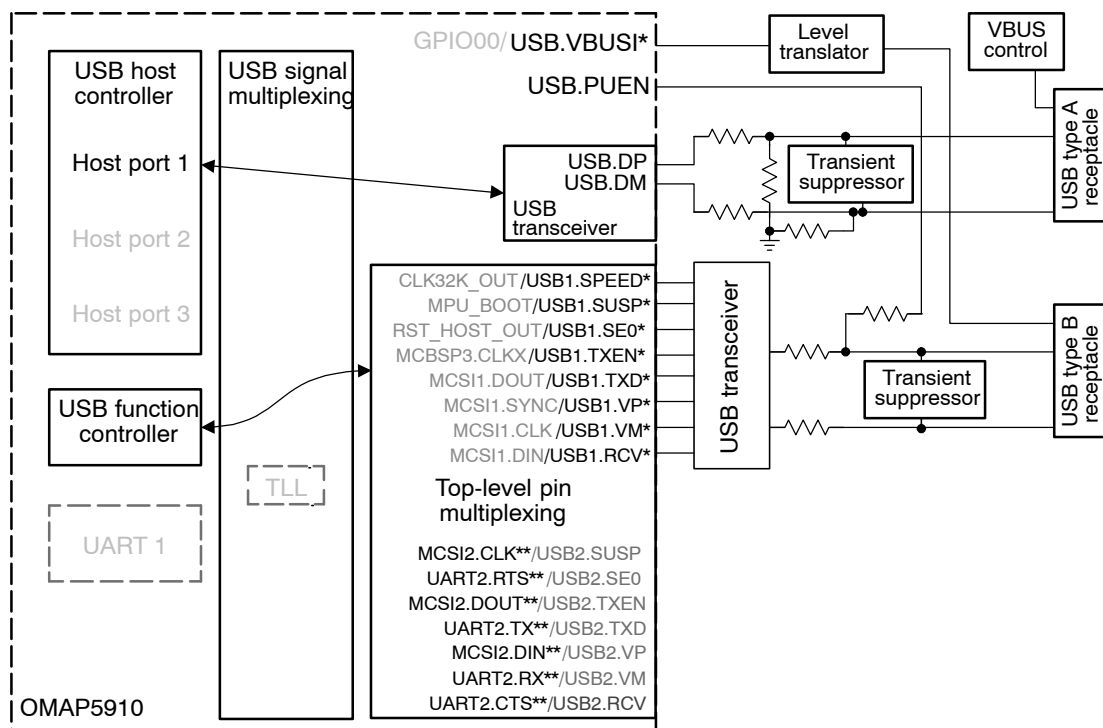


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 18

Figure 25. OMAP5910 With CONF MOD USB HOST HMC MODE R Set to 19

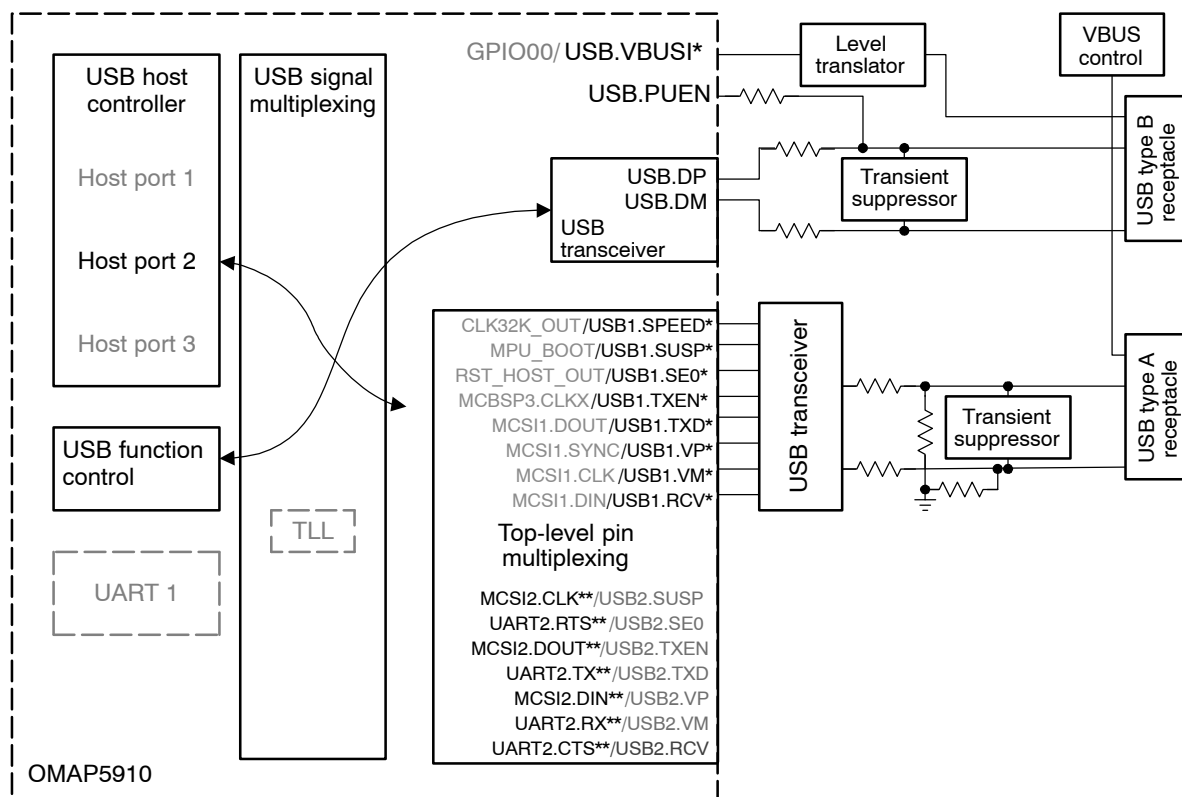


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 19

Figure 26. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 20

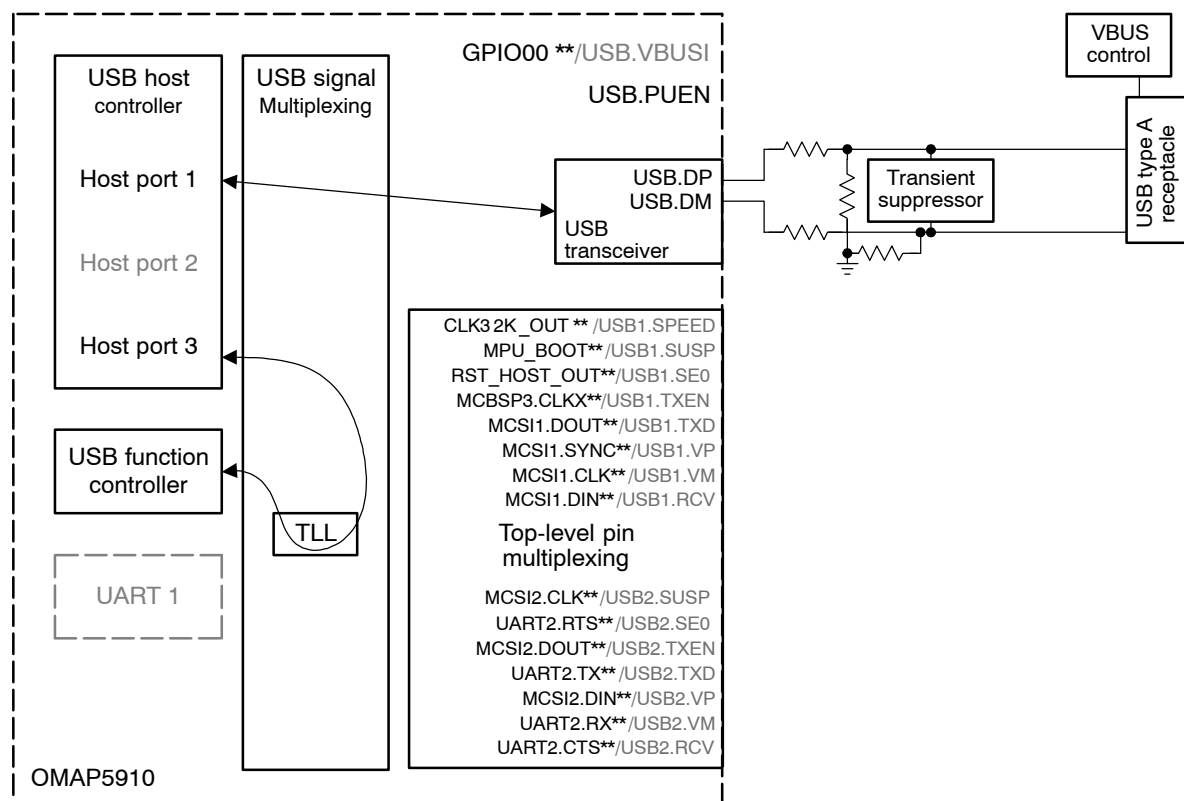


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 20

Figure 27. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 21

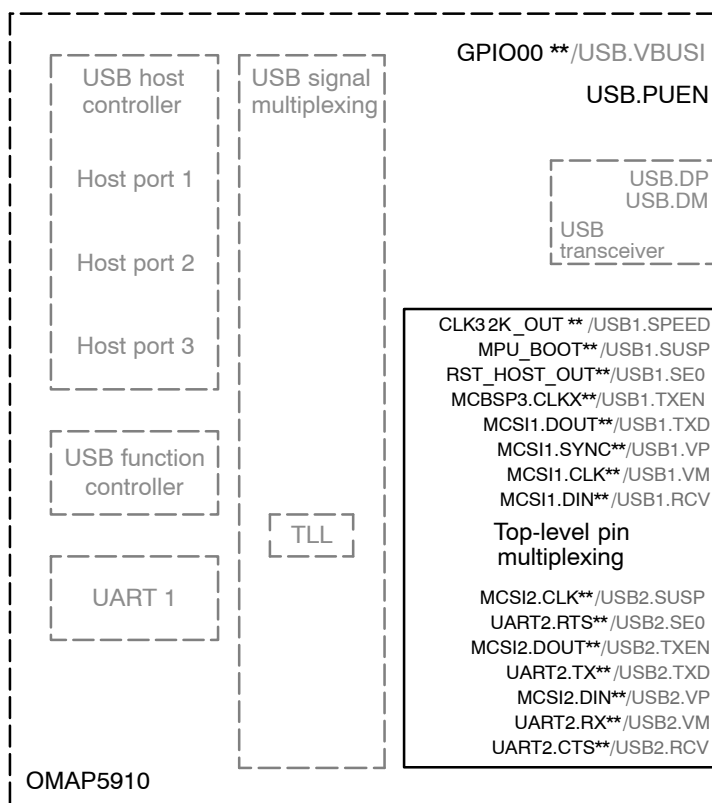


* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 21

Figure 28. OMAP5910 Configured for HMC_MODEs 22, 26-31



* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 22

HMC_MODE = 26

HMC_MODE = 27

HMC_MODE = 28

HMC_MODE = 29

HMC_MODE = 30

HMC_MODE = 31

Figure 29. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 23
(Transceiverless Connection Uses TXD+, TXD- Signaling)

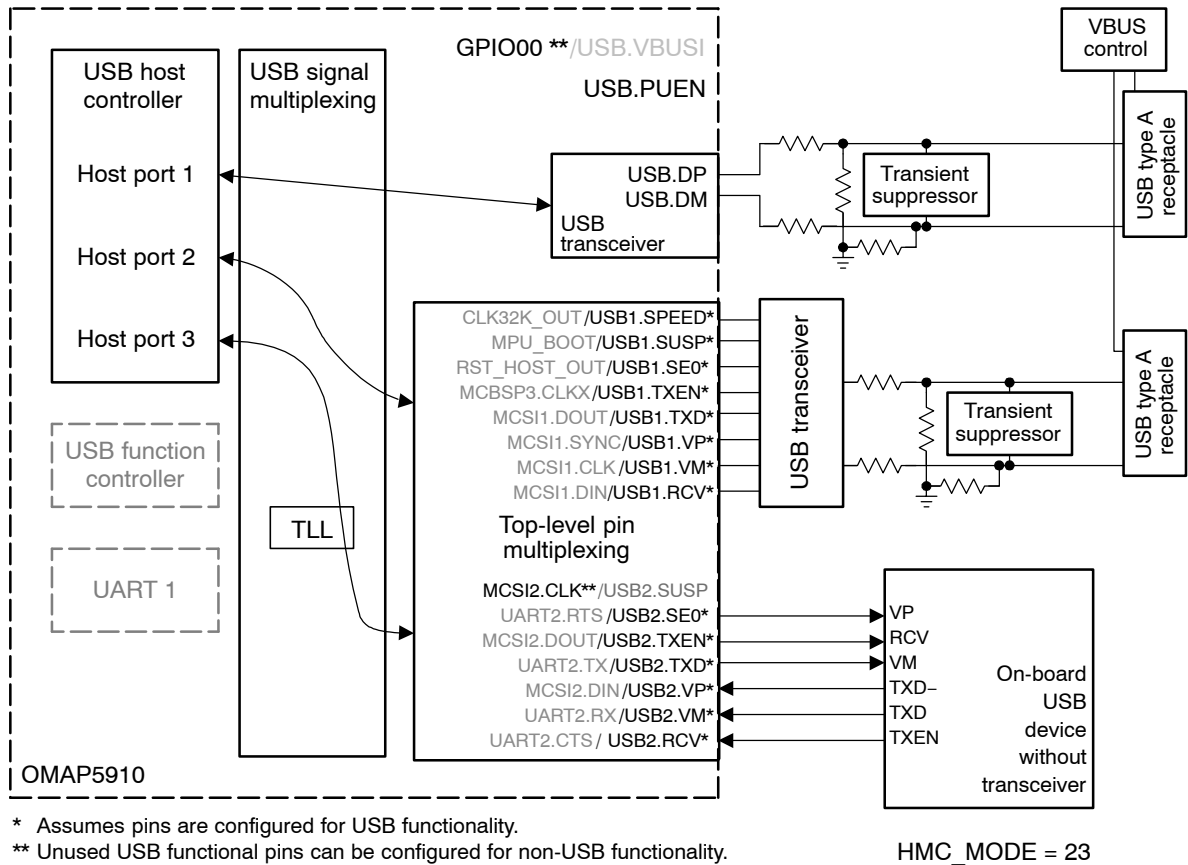


Figure 30. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 24
(Transceiverless Connection Uses TXD+, TXD- Signaling)

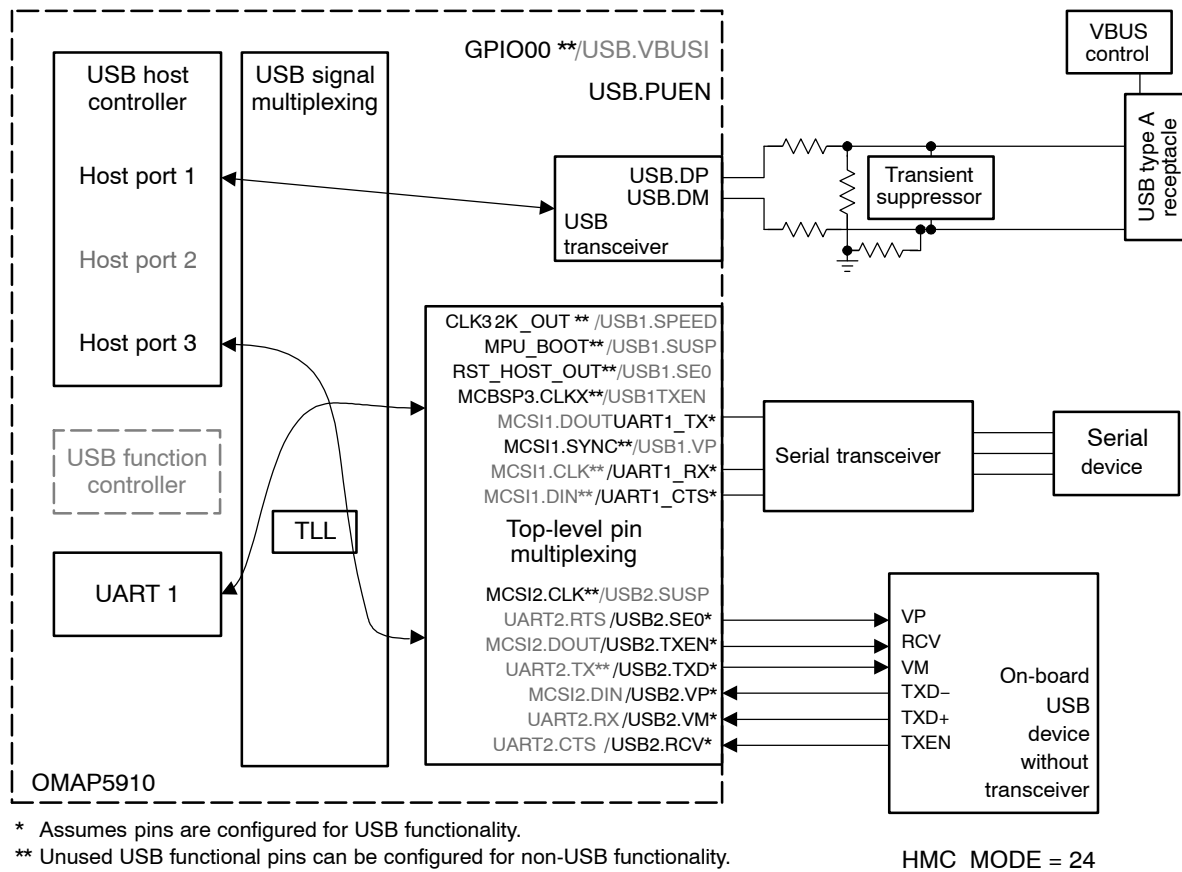
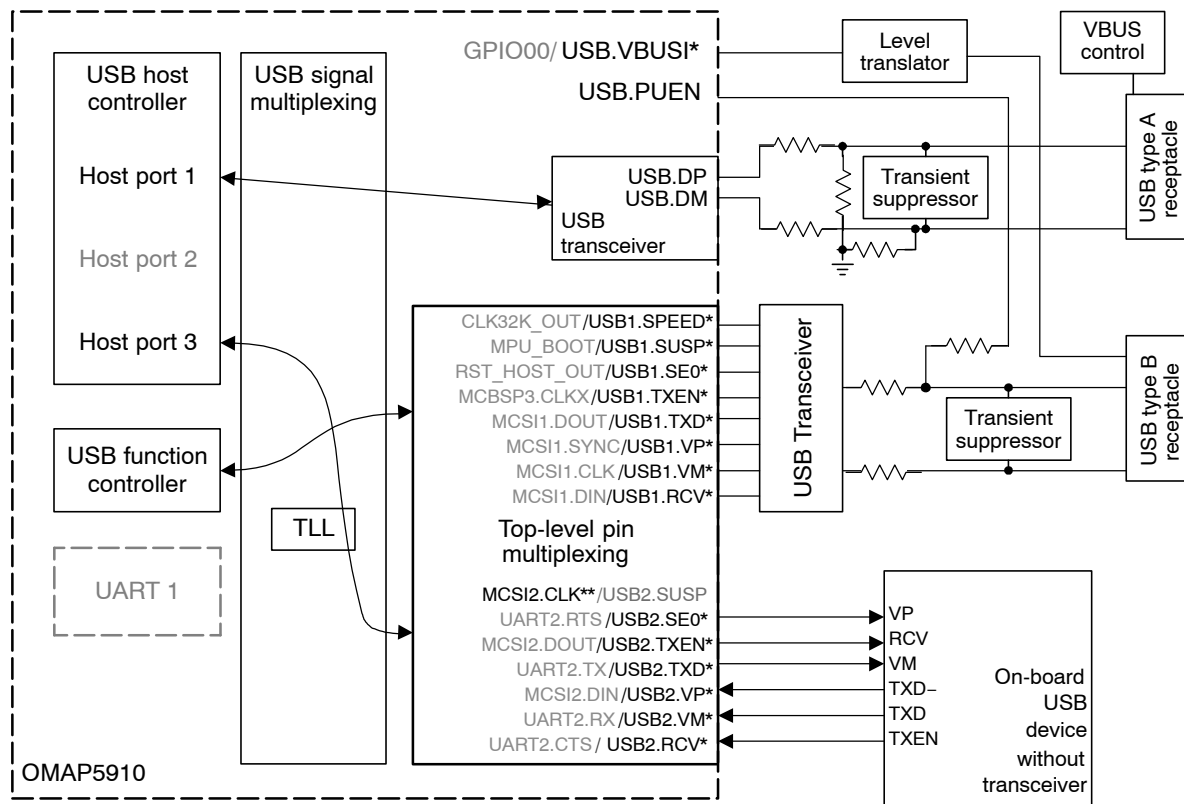


Figure 31. OMAP5910 With CONF_MOD_USB_HOST_HMC_MODE_R Set to 25
(Transceiverless Connection Uses TXD+, TXD- Signaling)



* Assumes pins are configured for USB functionality.

** Unused USB functional pins can be configured for non-USB functionality.

HMC_MODE = 25

5.5 Ports Shown as Unconnected

Many of the multiplexing modes show cases where a USB host port or the USB function controller are not connected to pins. When a USB signal multiplexing mode is selected that does not connect to a USB host controller port, that host controller port sees signaling that implies no external device is connected. When a USB signal multiplexing mode is selected that does not connect to the USB function controller port, the USB function controller port sees USB single-ended 0 signaling, which implies a USB reset.

5.6 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP5910 top-level signal multiplexing selects non-USB functionality for a pin but USB signal multiplexing is set to use that pin as an output, the signal from the USB signal multiplexing is ignored and the source selected by the OMAP5910 top-level signal multiplexing is used.

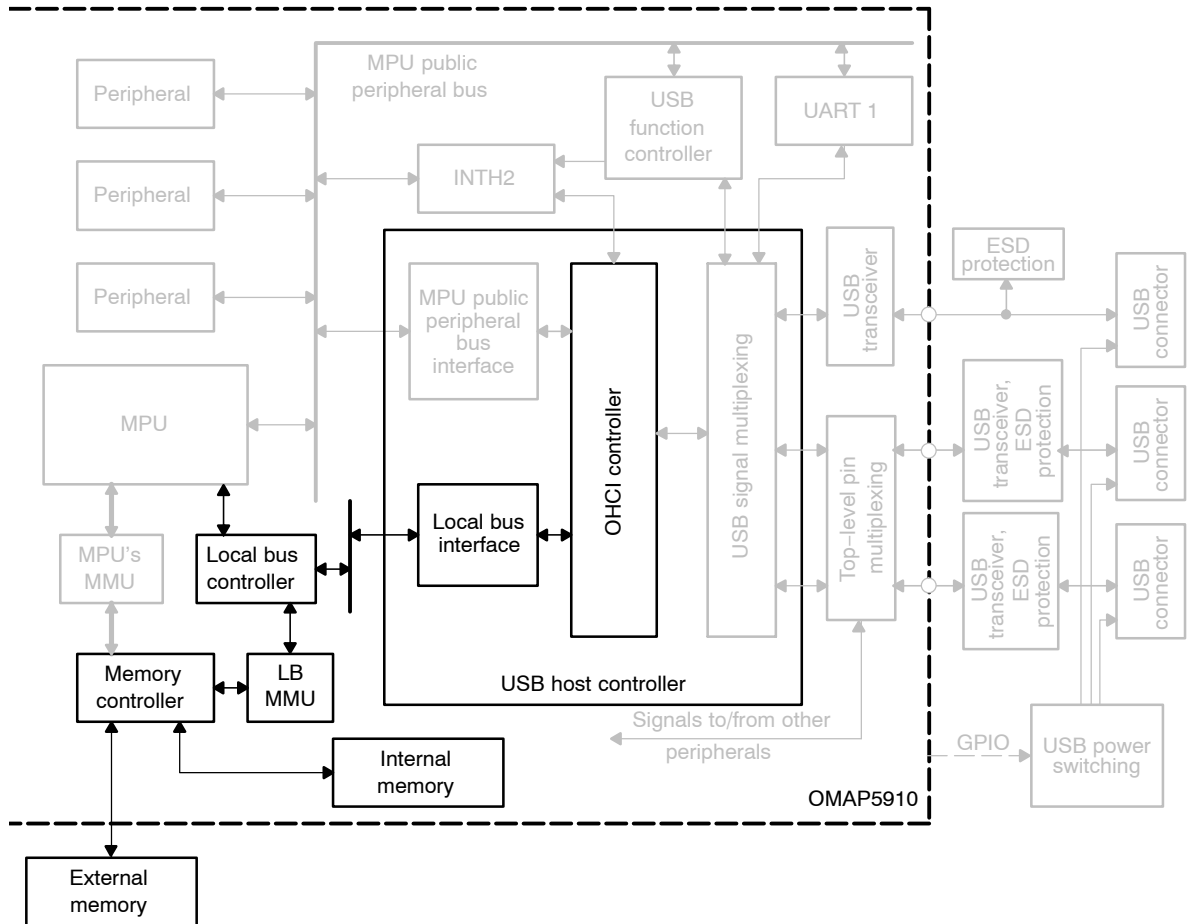
When OMAP5910 top-level signal multiplexing selects non-USB functionality for a pin but the USB signal multiplexing is set to use that pin as an input, the OMAP5910 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select a `CONF_MOD_HOST_HMC_MODE_R` value that brings some USB signals to the OMAP5910 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

6 USB Host Controller Access to System Memory

The USB host controller must have access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic. The OMAP5910 local bus controller allows the USB host controller to access OMAP5910 system memory, as shown in Figure 32.

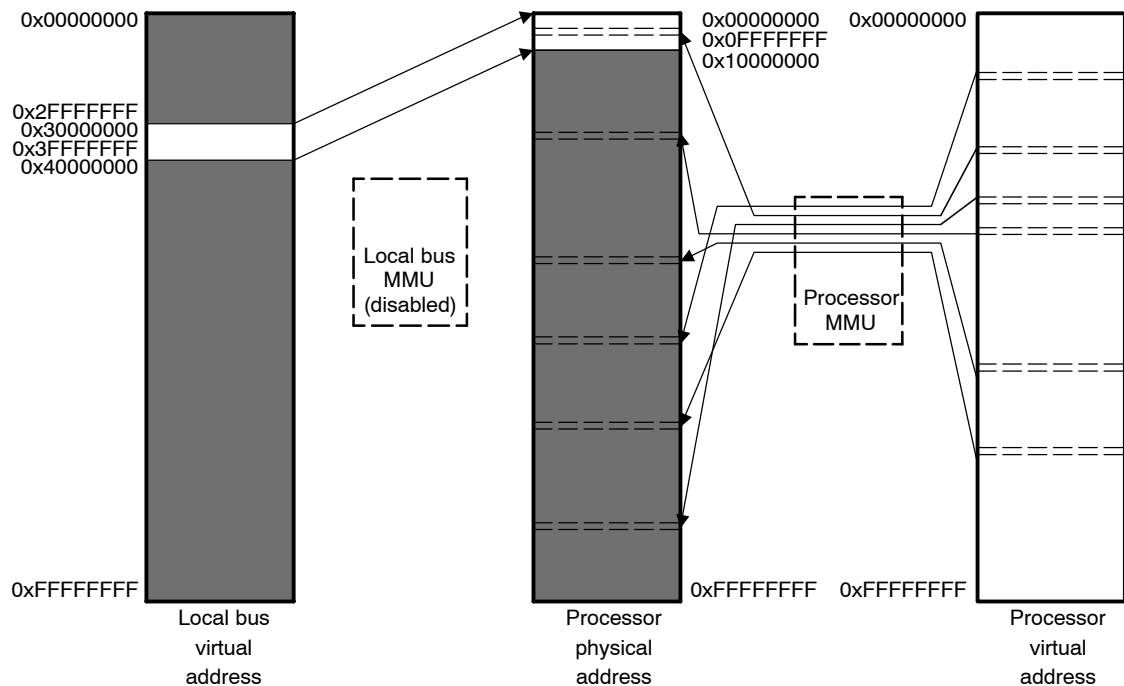
Figure 32. OMAP5910 USB Host Controller Data Path to System Memory



6.1 Local Bus Virtual Addressing

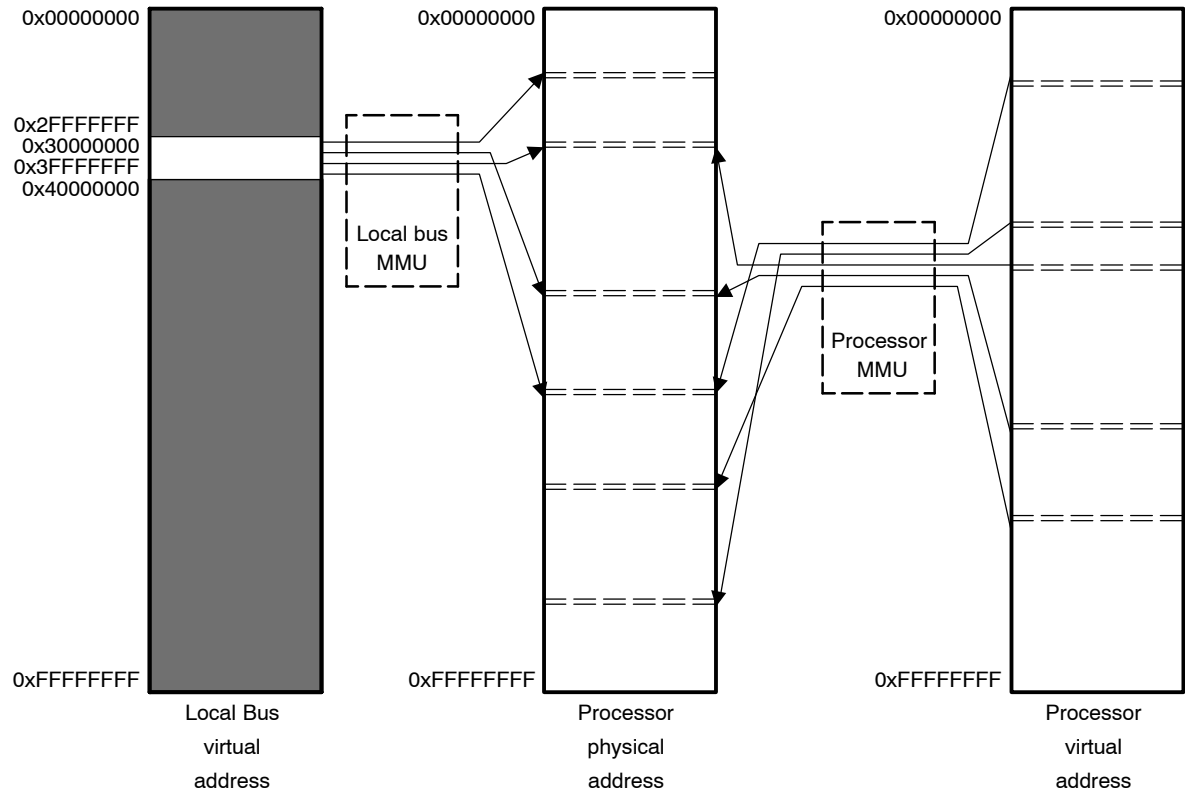
The OMAP5910 local bus implementation requires the USB host controller to use local bus virtual addresses with bits 31-28 set to 0011b, which gives the USB host controller a usable local bus virtual address range of 0x30000000 to 0x3FFFFFFF (a total range of 256 MBytes). If the local bus MMU is disabled, the local bus virtual address range from 0x30000000 to 0x3FFFFFFF automatically maps to processor address range 0x00000000 to 0x0FFFFFFF. With the local bus MMU disabled, it is not possible to access OMAP5910 physical addresses outside of this range. This is shown in Figure 33.

Figure 33. Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address With Local Bus MMU Disabled



When the OMAP5910 local bus MMU is enabled, the USB host controller is not limited to processor physical addresses between 0x00000000 and 0x0FFFFFFF. The local bus MMU can be programmed to devirtualize the local bus addresses in the local bus virtual address range 0x30000000-0x3FFFFFFF to any processor physical address. The relationships between local bus virtual addresses, processor virtual addresses, and processor physical addresses is shown in Figure 34.

Figure 34. Relationships Between Processor Virtual Address, Processor Physical Address, and Local Bus Virtual Address With Local Bus MMU Enabled



If any portion of the USB host controller data structures are stored in memory that is not in the physical address range 0x00000000 to 0x0FFFFFFF, then the local bus MMU must be programmed to provide a different mapping between local bus virtual addresses and physical addresses. A system implementation that does not implement any physical RAM in the physical address range 0x00000000 to 0x0FFFFFFF must enable the local bus MMU in order to use the OMAP5910 USB host controller.

Initialization of the local bus MMU is done using the same sorts of operations as are used for initializing the MPU MMU, except addressing the local bus MMU instead of the MPU MMU.

The processor local bus interface ignores any local bus activity in the local bus virtual address range from 0x0 to 0x2FFFFFFF, and the local bus virtual address range from 0x40000000 to 0xFFFFFFFF. Should the USB host request such an access (through improper setup of the USB host controller registers or improper initialization of endpoint descriptors, transfer descriptors, or the HCCA), the USB host local bus interface times out and signals an

unrecoverable error via the USB host interrupt mechanisms. If the USB host local bus time-out feature is disabled, the USB host controller instead waits indefinitely for completion of the local bus access and therefore locks up the local bus.

6.2 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP5910 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or local bus access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP5910 USB host controller can access the correct data from system memory, and that the MPU accesses that same data. This requires that any system memory accessed by the USB host controller be allocated in non-cached system memory.

If the OHCI data structures and/or data buffers are allocated in cached portions of system memory, a cache coherency problem can exist because the MPU can read from, and, if in writeback mode, write to the cache; but the USB host controller accesses are always directly to the physical system memory. If the data structures are in a cached portion of system memory and writeback mode is enabled, it is possible the USB host controller could read stale data that has not been updated by a cache writeback.

Similarly, if the data structure is in memory that is currently in the MPU cache (either writeback or writethrough mode) and the OHCI controller modifies the information in physical memory, the MPU can read stale data from the cache.

Cache coherency problems can be avoided by allocating the OHCI data structures (HCCA, EDs, and TDs) and the USB data buffers in noncacheable system memory. In this case, every MPU access directly accesses physical memory, so there is not a coherency issue. Configuration of cacheable portions of the MPU virtual address space is provided via the MPU memory management unit. See the description of the MPU MMU in Chapter 2, *MPU Subsystem*.

6.3 Local Bus Addressing and OHCI Data Structure Pointers

Because of the limitations described in Section 6.1, *Local Bus Virtual Addressing*, care is needed when programming the USB host controller OHCI registers that are pointers to data structures and when initializing the pointers inside those data structures. The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that are local bus virtual addresses that correspond to the physical addresses of the particular data structure. In most cases, the USB host controller OHCI registers that point to data structures in system memory are not programmed with the address that the MPU software uses to access those data structures.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. This list is pointed to by the HcDoneHead register, which contains a local bus virtual address that points to the most recent transfer descriptor that has been retired. As such, the host controller driver software must be able to convert from the local bus virtual address back to a MPU virtual address.

Several address conversion functions are helpful to enable proper addressing by the MPU software and the USB host controller. The functionality required for proper addressing depends on the settings used in the MPU MMU and local bus MMU; that is, functionality is system-dependent. The conversion functions are described in general terms in the next section.

6.3.1 MPUVAtoLBVA()—MPU Virtual Address to Local Bus Virtual Address Conversion Function

This function converts a MPU virtual address to the local bus virtual address that points to the same physical address in system memory. This output from this function is needed in programming the USB host controller registers that point to data structures in system memory and in programming pointers within the ED, TD, and HCCA data structures.

This function must understand the MPU MMU and local bus MMU programming and local bus virtual address restrictions. This routine is generally implemented in a two-step process—MPU virtual address to physical address, then physical address to local bus virtual address. Those two steps are described in section 6.3.3, *MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function* and section 6.3.5, *PAtoLBVA()—Physical Address to Local Bus Virtual Address Conversion Function*.

It is advisable to implement a checking function in this routine that verifies that the resulting local bus virtual address is in the usable local bus virtual address

range of 0x30000000 to 0x3FFFFFFF. If the USB host controller attempts to access a local bus virtual address outside of the valid range, an OHCI unrecoverable error occurs if the USB host controller local bus time-out feature is enabled. If the time-out feature is disabled and the USB host controller attempts an access outside of the valid range, the USB host controller local bus interface locks up and the USB host controller is not able to access the system memory data structures needed to issue USB packets.

Null pointers should be converted with caution. The OHCI USB host controller uses null pointers to indicate the end of lists. Some convention must be used to indicate an MPU null pointer and properly convert that MPU null pointer to a local bus virtual address of 0x00000000.

6.3.2 LBVAtoMPUVA()—Local Bus Virtual Address to MPU Virtual Address Conversion Function

This function converts a local bus virtual address to a MPU virtual address and is used when the host controller driver must traverse the ED and TD linked lists in system memory, or in traversing the done TD list.

This function must understand both the MPU MMU and local bus MMU programming and local bus virtual address restrictions; a routine generally implemented in a two step process—local bus virtual address to physical address, and physical address to MPU virtual address. Those two steps are described in section 6.3.4, *LBVAtoPA()—Local Bus Virtual Address to Physical Address Conversion Function* and section 6.3.6, *PAtoMPUVA()—Physical Address to MPU Virtual Address Conversion Function*.

Caution should be used when converting null pointers. The OHCI USB host controller uses null pointers to indicate the end of lists. Some convention must be used to indicate an MPU null pointer and properly convert a local bus virtual address of 0x000000 to that MPU null pointer value.

6.3.3 MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function

This function converts an MPU virtual address to the equivalent system physical address. This function must understand the way that the system software has configured the MPU MMU and must provide a conversion that has a result that is identical to the conversion done in hardware by the MPU MMU.

6.3.4 **LBVAtPA()—Local Bus Virtual Address to Physical Address Conversion Function**

This function converts a local bus virtual address to the equivalent system physical address. This function must determine the way that the system software has configured the local bus MMU and must provide a conversion that has a result that is identical to the conversion done in hardware by the local bus MMU.

This function must also determine the OMAP5910 local bus limitation that requires local bus virtual addresses to be in the range 0x30000000 to 0x3FFFFFFF.

6.3.5 **PAtLBVA()—Physical Address to Local Bus Virtual Address Conversion Function**

This function is a reverse version of the local bus virtual address to physical address conversion function. It accepts a processor physical address as its argument and returns the equivalent local bus virtual address.

It is possible to program the local bus MMU to allow two or more local bus virtual addresses to map to the same physical address – a situation that system software implementers must be careful to avoid.

It is advisable to implement a checking function in this routine that verifies that the resulting local bus virtual address is in the usable local bus virtual address range of 0x30000000 to 0x3FFFFFFF. If the USB host controller attempts to access a local bus virtual address outside of the valid range, an OHCI unrecoverable error occurs if the USB host controller local bus time-out feature is enabled. If the time-out feature is disabled and the USB host controller attempts an access outside of the valid range, the USB host controller local bus interface locks up and the USB host controller is not able to access the system memory data structures needed to issue USB packets.

6.3.6 **PAtMPUVA()—Physical Address to MPU Virtual Address Conversion Function**

This function is a reverse version of the MPU virtual address to physical address conversion. It accepts a physical address as an argument and returns the equivalent MPU virtual address.

It is possible to program the MPU MMU to allow two or more MPU virtual addresses to map to the same physical address. This is especially common in systems that use linear addressing within a task. System software implementers must be careful to avoid this situation or to perform the conversion in a way that accommodates the task-specific conversion requirements.

6.3.7 Physical, MPU Virtual, and Local Bus Virtual Addresses—an Example

For example, consider a system where the MPU places the HCCA and ED and TD lists in SRAM on OMAP5910 CS0 and where the USB data buffers are in external SDRAM. The MPU MMU is set up, in part, as shown in Table 31.

Table 31. MPU MMU Programming for Address Conversion Example

MPU Virtual Address	Physical Address	Page Size
0x00000000 to 0x000000FFF	0x0005E000 to 0x0005EFFF	Small (4K bytes)
0x00010000 to 0x0001FFFF	0x10170000 to 0x1017FFFF	Large (64K bytes)

For purposes of this example, assume that the system software has allocated the following MPU addresses for some OHCI data structures, as shown in Table 32.

Table 32. MPU Memory Allocations for Address Conversion Example

OHCI Structure	MPU Virtual Address
HHCA base address	0x00000700
First bulk ED on the ED list	0x00000140
First control ED on the ED list	0x00000150
First TD for first ED on bulk list	0x00000D90
First TD for first ED on control ED list	0x00000DA0
Data buffer start for first TD on first ED of bulk list	0x00010007
Data buffer start for first TD on first ED of control list	0x00013423

The corresponding physical addresses for these OHCI structures are shown in Table 33.

Table 33. Physical Addresses for Address Conversion Example

OHCI Structure	MPU Virtual Address	Physical Address
HHCA base address	0x00000700	0x0005E700
First bulk ED on the ED list	0x00000140	0x0005E140
First control ED on the ED list	0x00000150	0x0005E150
First TD for first ED on bulk list	0x00000D90	0x0005ED90

Table 33. Physical Addresses for Address Conversion Example (Continued)

OHCI Structure	MPU Virtual Address	Physical Address
First TD for first ED on control ED list	0x00000DA0	0x0005EDA0
Data buffer start for first TD on first ED of bulk list	0x00010007	0x10170007
Data buffer start for first TD on first ED of control list	0x00013423	0x10173423

Note: Assumes the MPU MMU initialization shown in Table 31 and the MPU memory allocations shown in Table 32.

Because the USB data buffers are to be stored outside of the physical address range that may be reached by the USB host controller when the local bus MMU is disabled, it is necessary to enable and program the local bus MMU. A possible way to program the local bus MMU is shown in Table 34.

Table 34. Local Bus MMU Programming for Address Conversion Example

Local Bus Virtual Address	Physical Address	Page Size
0x30F00000 to 0x30F0FFFF	0x0005E000 to 0x0005EFFF	Small (4 KBytes)
0x32100000 to 0x3210FFFF	0x10170000 to 0x1017FFFF	Large (64 KBytes)

Given this local bus MMU virtual address to physical address mapping, the local bus virtual addresses in Table 35 are needed to program the USB host controller OHCI registers and some of the pointers in the ED and TD structures.

Table 35. Local Bus Virtual Addresses for Address Conversion Example

OHCI Structure	MPU Virtual Address	Physical Address	Local Bus Virtual Address
HHCA base address	0x00000700	0x0005E700	0x30F00700
First bulk ED on the ED list	0x00000140	0x0005E140	0x30F00140
First control ED on the ED list	0x00000150	0x0005E150	0x30F00150
First TD for first ED on bulk list	0x00000D90	0x0005ED90	0x30F00D90
First TD for first ED on control ED list	0x00000DA0	0x0005EDA0	0x30F00DA0
Data buffer start (CBP) for first TD on first ED of bulk list	0x00010007	0x10170007	0x32100007
Data buffer start (CBP) for first TD on first ED of control list	0x00013423	0x10173423	0x32103423

Given all of the information in Table 35, the MPU performs the initializations described in Table 36.

Table 36. Some Data Structure Initializations for Address Conversion Example

Item initialized	Value	Type of Value
HcHCCA register	0x30F00700	Local bus virtual address
HcBulkHeadED register	0x30F00140	Local bus virtual address
HcControlHeadED register	0x30F00150	Local bus virtual address
TDQueueHeadPointer for first bulk ED	0x30F00D90	Local bus virtual address
TDQueueHeadPointer for first control ED	0x30F00DA0	Local bus virtual address
CBP for first TD of first bulk ED	0x32100007	Local bus virtual address
CBP for first TD of first control ED	0x32103423	Local bus virtual address

To properly initialize the EDs and TDs, the MPU software must allocate memory for the data structure and initialize the various fields. To initialize the address pointers in the structure, it begins by getting the MPU virtual address of the item that is pointed to by the data structure pointer. It converts that value to the corresponding local bus virtual address and places that value into the data structure pointer field.

For example, if the USB host controller driver must add a TD to the TD list in the first bulk ED, it must locate the last TD on the TD list and update it with the new TD information. It must then change the NextP pointer of the last TD to point to the local bus virtual address of a newly allocated empty TD. To traverse the TD list to find the last TD, the host controller driver starts by traversing the ED list. It starts by reading the HcBulkHeadED register, which returns a local bus virtual address. It converts this value to the corresponding MPU virtual address and reads the ED. If the ED is for a different function number or different endpoint number or direction, the driver reads the NextED pointer, converts the value from a local bus virtual address to a MPU virtual address, and checks this new ED for function number and endpoint number. Once it finds an ED that matches in function number and endpoint number (and perhaps endpoint direction), it begins traversing the TD list associated with the ED.

To traverse the TD list, the driver begins by saving a copy of the TailP value from the ED and by reading the HeadP value from the ED, which is a local bus virtual address. The driver converts this to a MPU virtual address, and reads

the TD. If the NextTD pointer is not the same as the saved copy of the TailP value, the TD is the not last TD in the list, and the driver converts the NextTD local bus virtual address to a MPU virtual address, fetches that TD, and compares the NextTD pointer to the saved TailP value. This process continues until the driver finds a TD with a NextTD value that matches the saved TD value.

Once the last TD on the correct list has been found, the driver copies the new TD information into the last TD, allocates memory for a new TD, converts the address of the new TD to a local bus virtual address, and updates the NextTD value in the last TD to point to the newly allocated TD. It also updates the TailP value in the ED to the local bus virtual address of the newly allocated TD.

6.4 NULL Pointers

The *OHCI Specification for USB* uses NULL pointers to indicate the end of a list. The OMAP USB host controller compares the ED and TD pointers against the value 0x00000000 to determine if the pointer is a null pointer. Address conversion routines must understand this usage and must not mistake 0x00000000, the null pointer, for local bus virtual address 0x30000000, which may point to a valid location in physical system memory.

6.5 Endianism and USB Host Controller Access to System Memory

The *OHCI Specification for USB* defines a little-endian controller. Since the OMAP5910 USB host controller is OHCI-compliant, it is defined for use in little-endian systems. The OMAP5910 MPU core and subsystem are also little-endian.

6.5.1 Endianism and OHCI Endpoint and Transfer Descriptors

OHCI endpoint and transfer descriptors are implemented as shown in the *OHCI Specification for USB*, with the bit positions in the endpoint and transfer descriptor data structures representing bit positions on the physical 32-bit data bus.

If using C structures for the EDs and TDs, be careful of data alignment of the structures and the implications of the MPU's little-endianism. The *OHCI Specification for USB* requires that EDs and TDs be aligned at 32-bit boundaries in system memory (local bus virtual address LS 4 bits must be 0). It may be useful to validate that your data structures are being properly initialized in memory by performing 32-bit reads to the initialized data structures and comparing the 32-bit read values to the intended values within the bit-fields. However, because the USB host controller and the MPU core

both use little-endian addressing, this should not arise any issues on the OMAP5910 device.

6.5.2 Endianism and OHCI Data Buffers

The OMAP5910 MPU subsystem only supports little-endian operation. The USB host controller assumes little-endian data addressing.

Little-endian addressing imposes the following data positions within a 32-bit aligned, 32-bit data value in memory, as shown in Table 37.

Table 37. Little-Endian Data Alignment Within 32-Bit Word

Address Offset	Data Size	Bit Position Within 32-Bit Word							
		31	24	23	16	15	8	7	0
0x0	1								First byte
0x0	2						Second byte		First byte
0x0	3				Third byte		Second byte		First byte
0x0	4		Fourth byte		Third byte		Second byte		First byte
0x1	1						First byte		
0x1	2				Second byte		First byte		
0x1	3		Third byte		Second byte		First byte		
0x2	1				First byte				
0x2	2		Second byte		First byte				
0x3	1		First byte						

7 OMAP5910 Local Bus

The OMAP5910 local bus supports both slave and master peripherals. This bus allows the MPU, DSP, and DMA controller to access local bus slave peripherals and allows local bus master peripherals to move data to and from system memory.

The OMAP5910 device does not implement any local bus slave peripherals. The local bus interface of the OMAP5910 USB host controller implements a local bus master peripheral which enables the USB host controller to access system memory via the OMAP5910 local bus and the OMAP5910 traffic controller.

7.1 LB Register Descriptions

Table 38 lists the registers associated with local bus (LB) control and status. Table 39 through Table 47 describe specific register bits.

The LB_CLOCK_DIV register has a direct impact on the ability of the USB host controller to access OMAP5910 system memory. The remaining OMAP5910 local bus control and status registers have no direct effect, because they control MPU, DSP, and DMA controller accesses to slave peripherals addressed via local bus and there are no slave peripherals on the OMAP5910 local bus.

The local bus memory management unit and its registers are discussed separately in section 8, *OMAP5910 Local Bus MMU*.

Table 38. Local Bus Control Registers

Field	Description	R/W	Size†	Address
LB_MPU_TIMEOUT	LB MPU time-out	R/W	32	FFFE:C100h
LB_HOLD_TIMER	LB hold timer	R/W	32	FFFE:C104h
LB_PRIORITY_REG	LB priority	R/W	32	FFFE:C108h
LB_CLOCK_DIV	LB clock divider	R/W	32	FFFE:C10Ch
LB_ABORT_ADD	LB abort address	R	32	FFFE:C110h
LB_ABORT_DATA	LB abort data	R	32	FFFE:C114h
LB_ABORT_STATUS	LB abort status	R	32	FFFE:C118h
LB_IRQ_OUTPUT	LB IRQ output	R/W	32	FFFE:C11Ch
LB_IRQ_INPUT	LB IRQ input	R	32	FFFE:C120h

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

7.2 LB MPU Time-out Register (LB_MPU_TIMEOUT)

This register controls the maximum number of local bus clocks allowed for an access by the OMAP5910 MPU, DSP, or DMA controller to a local bus slave device before signaling a local bus abort. This register has no effect on OMAP5910 USB host controller accesses to system memory.

Table 39. LB MPU Time-out Register (LB_MPU_TIMEOUT)

Bits	Field	Description	Type	Reset Value
31–8	Reserved	Reserved	R	0
7–0	TIMEOUT	<p>Local bus slave access time-out</p> <p>Number of local bus clocks to wait for completion of a local bus cycle initiated by the MPU, DSP, or DMA before signaling a local bus abort. This time-out does not apply to local bus cycles initiated by the OMAP5910 USB host controller.</p> <p>Because there are no local bus slave peripherals in the OMAP5910 device, this register can be set to a low value like 3. This causes the local bus controller to issue a local bus abort interrupt if the DSP, MPU, or DMA controller mistakenly attempts to access a physical address associated with the local bus slave memory space.</p>	R/W	0xFF

7.3 LB Hold Timer Register (LB_HOLD_TIMER)

This register configures the local bus controller bus request and bus hold functions. This register has no effect on OMAP5910 USB host controller accesses to system memory.

Table 40. LB Hold Timer Register (LB_HOLD_TIMER)

Bits	Field	Description	Type	Reset Value
31–10	Reserved	Reserved	R	0
9	BREQ_TIMER_EN	<p>Local bus request timer enable</p> <p>Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to 1 to enable the request timer. This helps prevent lockup of the local bus.</p>	R/W	0

Table 40. LB Hold Timer Register (LB_HOLD_TIMER) (Continued)

Bits	Field	Description	Type	Reset Value
8	HOLD_TIMER_EN	Local bus hold timer enable Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to 1 to enable the HOLD timer. This helps prevent lockup of the local bus.	R/W	0
7–0	HOLD_TIMER	Local bus hold timer value Because there are no local bus slave peripherals on OMAP5910, it is recommended that this register be set to a small number like 5 so that the hold timer does not need to wait a long time before signaling an abort. This speeds return of the local bus abort interrupt should the DSP, MPU, or DMA controller accidentally access the local bus.	R/W	0

7.4 LB Priority Register (LB_PRIORITY_REG)

The priority register is not used in OMAP5910.

Table 41. LB Priority Register (LB_PRIORITY_REG)

Bits	Field	Description	Type	Reset Value
31–0	Reserved	Reserved for future expansion. These bits should always be written as 0.	R	0

7.5 LB Clock Divider Register (LB_CLOCK_DIV)

This register controls local bus clocking. This clocking affects accesses by the USB host controller to system memory, as well as DMA controller, DSP, and MPU accesses to local bus slave peripherals. Because the OMAP5910 does not implement any local bus slave peripherals, the local bus clock rate mainly affects the USB host controller accesses to system memory.

This register also provides masks for external local bus interrupts and local bus abort interrupts that can be routed to the MPU level 1 interrupt controller.

The clock division register can be read in user and supervisor modes and can be written in supervisor mode only.

Table 42. LB Clock Divider Register (LB_CLOCK_DIV)

Bits	Field	Value	Description	Type	Reset Value
31–8	Reserved		Reserved	R	0
7	LB_ABORT_MASK		Local bus abort interrupt mask	R/W	1
		0	Local bus abort interrupt is enabled.		
		1	Local bus abort interrupt is disabled.		
			When enabled and the DSP, MPU, or DMA controller attempts to access a local bus slave peripheral, the local bus time-out counter counts down to zero and then signal a local bus abort. When disabled and the DSP, MPU, or DMA controller attempts to access a local bus slave peripheral, the local bus time-out counter does not count and the cycle locks up the local bus.		
			Since the OMAP5910 does not implement any local bus slave peripherals, it is recommended that this bit be cleared and that system software trap the local bus interrupt at MPU level 1 interrupt handler IRQ29 input.		
6–3	LB_IRQ_IN_MASK		Local bus interrupt input mask	R/W	0xF
			These bits enable (when 0) or disable (when 1) local bus interrupt inputs.		
			OMAP5910 does not provide interrupt sources for these interrupts, so it is recommended that these bits remain set to 1 to ensure future compatibility.		

Table 42. LB Clock Divider Register (LB_CLOCK_DIV) (Continued)

Bits	Field	Value	Description	Type	Reset Value
2–0	LB_CLK_DIV		Local bus clock divisor:	R/W	4
		010	Local bus clock is transfer controller clock divided by 2.		
		100	Local bus clock is transfer controller clock divided by 4.		
		110	Local bus clock is transfer controller clock divided by 6.		
		Other values: Reserved			
		These bits control the local bus clock rate. The clock for the local bus is derived from the OMAP5910 traffic controller clock. The local bus clock can be set to be transfer controller clock divided by 2, divided by 4, or divided by 6. All other values are reserved and must not be used.			
		This field must be set to a suitable value to allow USB host controller access to system memory. The USB host controller supports a maximum local bus clock frequency of 50 MHz. The divisor selected must not result in a local bus clock frequency greater than 50 MHz.			

7.6 LB Abort Address Register (LB_ABORT_ADD)

This register captures the local bus address if the MPU, DSP, or DMA controller attempts to access a local bus slave peripheral and a local bus time-out occurs.

This register is not affected by the OMAP5910 USB host controller accesses to system memory.

Table 43. LB Abort Address Register (LB_ABORT_ADD)

Bits	Field	Description	Type	Reset Value
31–0	LB_ABORT_ADD	Local bus address of last aborted local bus cycle When the DMA controller, MPU, or DSP attempts to access a local bus slave peripheral and a local bus abort occurs, the address of the cycle is captured to this register.	R	0xFFFF FFFF

7.7 LB Abort Data Register (LB_ABORT_DATA)

This register reports the data bus value for the last DMA controller, DSP, or MPU local bus cycle that was aborted.

This register is not affected by the OMAP5910 USB host controller accesses to system memory.

Table 44. LB Abort Data Register (LB_ABORT_DATA)

Bits	Field	Description	Type	Reset Value
31–0	LB_ABORT_DATA	Data for aborted local bus access attempt This register reflects the data seen on the local bus when a local bus access is aborted.	R	0xFFFF FFFF

7.8 LB Abort Status Register (LB_ABORT_STATUS)

This register provides the local bus cycle status for the last aborted local bus cycle that was issued by the DMA controller, DSP, or MPU. This register is not affected by OMAP5910 USB host controller accesses to system memory.

Table 45. LB Abort Status Register (LB_ABORT_STATUS)

Bits	Field	Description	Type	Reset Value
31–8	Reserved	Reserved	R	0
7–4	LB_BE	Byte enables from aborted local bus access Active-low byte enables seen at the last aborted local bus access that was issued by the DSP, DMA controller, or MPU.	R	0
3	LB_RD	Cycle type from aborted local bus access Indicates, when 1, that the last aborted local bus access was a read. When 0, that the last aborted local bus access was a write.	R	0
2	LB_DMA	DMA sourced aborted local bus access When 1, indicates that the last aborted local bus access was sourced by the DMA controller.	R	0
1	LB_DSP	DSP sourced aborted local bus access When high, indicates that the last aborted local bus access was sourced by the DSP.	R	0
0	LB_MPU	MPU sourced aborted local bus access When high, indicates that the last aborted local bus access was sourced by the MPU controller.	R	0

7.9 LB IRQ Output Register (LB_IRQ_OUTPUT)

This register is reserved for future expansion and should be left at 0.

Table 46. LB IRQ Output Register (LB_IRQ_OUTPUT)

Bits	Field	Description	Type	Reset Value
31–0	Reserved	Reserved	R	0

7.10 LB IRQ Input Register (LB_IRQ_INPUT)

This register reports the status of local bus interrupts, including interrupts from external sources (which are not used in the OMAP5910) and the status of the local bus abort interrupt.

This register has no effect on the OMAP5910 USB host controller accesses to system memory and is not affected by USB host controller accesses to system memory.

Table 47. LB IRQ Input Register (LB_IRQ_INPUT)

Bits	Field	Description	Type	Reset Value
31–5	Reserved	Reserved	R	0
4	ABORT_STAT	Local bus abort status 0: Local bus abort has occurred since last read of LB_IRQ_INPUT. 1: Local bus abort has not occurred since last read of LB_IRQ_INPUT. Reading this register sets this bit to 1. Writes have no effect. A local bus abort event causes an interrupt to the MPU level 1 interrupt handler if the LB_ABORT_MASK bit of the LB_CLOCK_DIV register is set to 0. Because the DMA controller, DSP, and MPU could mistakenly attempt to access a local bus slave peripheral address, it is recommended that system software trap the local bus abort interrupt and signal a system error should one occur.	R	1
3–0	IRQ_IN_STAT	Reserved for future expansion. These bits can be ignored.	R	0xF

7.11 Local Bus Initialization

Proper operation of the OMAP5910 local bus (for USB host controller access to system memory) requires that the FUNC_MUX_CTRL_0 register be properly initialized. FUNC_MUX_CTRL_0 bits 1:0 must be set either to 00b or to 11b in order to support correct local bus activity.

The local bus clock rate must be chosen to provide a suitable clock to the USB host controller. The local bus clock rate is controlled by the LB_CLOCK_DIV register LB_CLK_DIV field and is generated by dividing the OMAP5910 transfer controller clock by 2, 4, or 6. The local bus clock rate must be programmed to provide a local bus clock frequency that is 50 MHz or slower.

7.12 Local Bus Virtual Addressing

When an MPU, DSP, or DMA access to the local bus causes a local bus abort interrupt, the interrupt service routine for MPU level 1 interrupt IRQ29 must read the LB IRQ input register (LB_IRQ_INPUT) to clear the IRQ and then must clear the IRQ at the MPU level 1 interrupt handler input IRQ29.

8 OMAP5910 Local Bus MMU

The local bus memory management unit (MMU) is used by the local bus interface for address management of bus cycles initiated by the OMAP5910 USB host controller. The local bus MMU manages local bus virtual addresses, including conversion of local bus virtual addresses to physical addresses and monitoring of access permissions. The local bus MMU can be initialized by the MPU to allow the USB host controller to access the full range of OMAP5910 system memory. A detailed functional description of the MMU architecture can be found in Chapter 2, *MPU Subsystem*.

The local bus MMU includes the following basic blocks:

- ☐ A 32-entry translation look aside buffer (TLB)
- ☐ Walking table logic
- ☐ Registers for recording fault status and fault address

When properly configured and enabled, the walking table logic automatically performs the address conversion from local bus virtual address to physical address (for USB host controller accesses to system memory). Alternately, the walking table logic can be disabled, which allows the MPU to perform local bus virtual address translation under software control.

The local bus MMU is configured via registers which are on the MPU private peripheral bus. The MPU is responsible for correctly configuring the local bus MMU memory mapping functions.

The local bus MMU operates much like the MPU and DSP MMUs. The local bus MMU does not use the upper 4 bits of the local bus virtual address (they are ignored by the local bus MMU hardware) and does not support the prefetch feature.

8.1 OMAP5910 Local Bus MMU Registers

The OMAP5910 local bus MMU registers are listed in Table 48. Table 49 through Table 67 describe the specific register bits.

Table 48. Local Bus MMU Registers

Field	Description	User Access	Supervisor Access	Size	Address
Reserved	Reserved	R	R/W	16	FFFE:C200h
LB_MMU_WALKING_ST_REG	LB MMU walking status	R	R/W	16	FFFE:C204h
LB_MMU_CNTL_REG	LB MMU control	R	R/W	16	FFFE:C208h
LB_MMU_FAULT_AD_H_REG	LB MMU fault address high	R	R	16	FFFE:C20Ch
LB_MMU_FAULT_AD_L_REG	LB MMU fault address low	R	R	16	FFFE:C210h
LB_MMU_FAULT_ST_REG	LB MMU fault status	R	R	16	FFFE:C214h
LB_MMU_IT_ACK_REG	LB MMU interrupt acknowledge	W	W	16	FFFE:C218h
LB_MMU_TTB_H_REG	LB MMU TTB high	R	R/W	16	FFFE:C21Ch
LB_MMU_TTB_L_REG	LB MMU TTB low	R	R/W	16	FFFE:C220h
LB_MMU_LOCK_REG	LB MMU lock counter	R/W	R/W	16	FFFE:C224h
LB_MMU_LD_TLB_REG	LB MMU TLB load/read	R/W	R/W	16	FFFE:C228h
LB_MMU_CAM_H_REG	LB MMU CAM high	R/W	R/W	16	FFFE:C22Ch
LB_MMU_CAM_L_REG	LB MMU CAM low	R/W	R/W	16	FFFE:C230h
LB_MMU_RAM_H_REG	LB MMU RAM high	R/W	R/W	16	FFFE:C234h
LB_MMU_RAM_L_REG	LB MMU RAM low	R/W	R/W	16	FFFE:C238h
LB_MMU_GFLUSH_REG	LB MMU global flush	R/W	R/W	16	FFFE:C23Ch
LB_MMU_FLUSH_ENTRY_REG	LB MMU flush entry	R/W	R/W	16	FFFE:C240h
LB_MMU_READ_CAM_H_REG	LB MMU CAM read high	R/W	R/W	16	FFFE:C244h
LB_MMU_READ_CAM_L_REG	LB MMU CAM read low	R/W	R/W	16	FFFE:C248h

Table 48. Local Bus MMU Registers (Continued)

Field	Description	User Access	Supervisor Access	Size	Address
LB_MMU_READ_RAM_H_REG	LB MMU RAM read high	R/W	R/W	16	FFFE:C24Ch
LB_MMU_READ_RAM_L_REG	LB MMU RAM read low	R/W	R/W	16	FFFE:C250h

This register reports the local bus MMU walking table status.

Table 49. LB MMU Walking Status Register (LB_MMU_WALKING_ST_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–:2	Reserved	Reserved	-	-	-
1	Wtl_working	When 1, the walking table is active.	R	R/W	0
0	Reserved	Reserved	-	-	-

The LB MMU control register controls the local bus MMU reset and enable functions.

Table 50. LB MMU Control Register (LB_MMU_CNTL_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–6	Reserved	Reserved	-	-	-
5	Burst_16_mngt_en	When 1, enables 16 bit burst access management	R	R/W	0
4–3	Reserved	Reserved	R	R	00
2	Wtl_en	When 1, enables the walking table logic. When 0, the walking table is disabled and access to the TLB and lock counter are disabled.	R	R/W	0

Table 50. LB MMU Control Register (LB_MMU_CNTL_REG) (Continued)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
1	MMU_en	Local bus MMU enable 0: Local bus MMU is disabled. 1: Local bus MMU is enabled. When 0, the local bus MMU is disabled and local bus virtual addresses in the range 0000:0000h to 0FFF:FFFF are mapped directly to physical address range 0000:0000h to 00FF:FFFFh.	R	R/W	0
0	Reset_sw	When 0, holds the local bus MMU in reset. When 1, the local bus MMU is not held in reset. Software must set this bit to allow the MMU to function.	R	R/W	0

The LB MMU fault address registers report the local bus virtual address of the last local bus access which caused a local bus MMU fault.

Table 51. LB MMU Fault Address High Register (LB_MMU_FAULT_AD_H_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–0	Fault_address_MSB	Most significant 16 bits of the local bus address that caused a local bus MMU fault. The most significant 4 bits are always 0000.	R	R	0x0000

Table 52. LB MMU Fault Address Low Register (LB_MMU_FAULT_AD_L_REG)

Bits	Field	Description	User	Sup	Hardware Reset Value
15–0	Fault_address_LSB	Least significant 16 bits of the local bus address that caused a local bus MMU fault	R	R	0x0000

The LB MMU fault status register provides information on the type of fault encountered at the last local bus MMU fault.

Table 53. LB MMU Fault Status Register (LB_MMU_FAULT_ST_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–3	Reserved	Reserved	-	-	-
2	Perm_fault	Permission fault. Active high	R	R	0
1	Tlb_miss	TLB miss (when WTB disabled). Active high	R	R	0
0	Trans_fault	Translation fault (invalid descriptor). Active high	R	R	0

The LB MMU register is used to acknowledge the local bus MMU interrupt at the local bus MMU interrupt generator. Acknowledging the interrupt at the local bus MMU interrupt generator causes the generator to deassert its interrupt indication.

In response to a local bus MMU interrupt, the interrupt service routine must:

- 1) Read LB_MMU_FAULT_AD_L_REG and LB_MMU_FAULT_AD_H_REG.
- 2) Read LB_MMU_FAULT_ST_REG.
- 3) Determine how to respond to the faulty access.
- 4) Acknowledge the interrupt by writing a 1 to LB_MMU_IT_ACK_REG.
- 5) Acknowledge the interrupt at MPU level 1 interrupt handler IRQ17.

Table 54. LB MMU Interrupt Acknowledge Register (LB_MMU_IT_ACK_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	-	-	-
0	It_ack	Write a 1 to this bit to acknowledge the interrupt. A write of 0 has no effect; a write of 1 clears the bit automatically.	W	W	0

The LB TTB address registers define the physical address of the local bus MMU translation table base (TTB).

Table 55. LB MMU TTB Address High Register (LB_MMU_TTB_H_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–0	TTB_REG_H	Most significant 16 bits of physical address of translation table base address	R	R/W	0

Table 56. LB MMU TTB Address Low Register (LB_MMU_TTB_L_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–0	TTB_REG_L	Least significant 16 bits of physical address of translation table base address. Bits 9-0 must always be 0.	R	R/W	0

Table 57. LB MMU Lock Counter Register (LB_MMU_LOCK_REG)

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–10	Base_value	Locked entries base value	R/W	R/W	0
9–4	Current_victim	Current entry pointed to by the WTL. Base_value <= Current_victim <= 31	R/W	R/W	0
3–0	Unused				

Use the following procedures to write one entry into the local bus MMU TLB:

- 1) Disable the table walking logic (if not already disabled).
- 2) Write the appropriate value to LB_MMU_CAM_H_REG.
- 3) Write the appropriate value to LB_MMU_CAM_L_REG.
- 4) Write the appropriate value to LB_MMU_RAM_H_REG.
- 5) Write the appropriate value to LB_MMU_RAM_L_REG.
- 6) Write the TLB entry number to be modified into Current_victim: number must be equal to or greater than the current value of Base_value and must be less than or equal to 31.

- 7) Write the LD_TLB_REG register with the Ld_tlb_item bit set.
- 8) Enable the table walking logic (if necessary).

Use the following procedures to read one local bus MMU TLB entry:

- 1) Disable the table walking logic (if not already disabled).
- 2) Write the TLB entry number of the TLB entry to be read into Current_victim: number must be equal to or greater than the current value of Base_value and must be less than or equal to 31.
- 3) Write the LD_TLB_REG register with the Rd_tlb_item bit set.
- 4) Read the value from LB_MMU_READ_CAM_H.
- 5) Read the value from LB_MMU_READ_CAM_L.
- 6) Read the value from LB_MMU_READ_RAM_H.
- 7) Read the value from LB_MMU_READ_RAM_L.
- 8) Enable the table walking logic (if necessary).

This register controls reading and writing to the local bus MMU TLB.

Table 58. Local Bus MMU TLB Read/Write Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–2	Reserved	Reserved	–	–	–
1	Rd_tlb_item	When this bit is set, causes the TLB data pointed to by the lock counter to be read. Writing a 0 has no effect. Always reads as 0.	R/W	R/W	0
0	Ld_tlb_item	When this bit is set, causes the TLB to be written. Writing a 0 has no effect. Always reads as 0	R/W	R/W	0

The LB MMU content addressable memory (CAM) access registers can be used to access data in the local bus MMU content addressable memory. The VA_tag values are compared against the local bus virtual address when a local bus access occurs. Bits 31–28 of the local bus virtual address are ignored in the address comparison.

Note:

The USB host controller can malfunction if the local bus MMU is programmed with tiny pages. See Section 8.2.1, *Local Bus MMU Page Size and the USB Host Controller*, for more information.

Table 59. Local Bus MMU CAM High Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–6	Reserved	Reserved	–	–	–
5–0	VA_tag_l1_H	<p>The most significant six bits of the local bus virtual address, which are used for this entry level 1 table index.</p> <p>Because the local bus MMU ignores bits 31 -28 of the local bus virtual address, these 6 bits correspond to bits 27-22 of the 32-bit local bus virtual address used by the USB host controller.</p>	R/W	R/W	0

Table 60. Local Bus MMU CAM Low Register

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
15–14	VA_tag_l1_L		<p>Least significant two bits of the local bus virtual address to be used for this entry level 1 table index.</p> <p>Because the local bus MMU ignores bits 31 -28 of the local bus virtual address, VA_tag_l1_L correspond to bits 21 -20 of the 32-bit local bus virtual address used by the USB host controller.</p>	R/W	R/W	0

Table 60. Local Bus MMU CAM Low Register (Continued)

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
13–4	VA_tag_l2		<p>Bits of the local bus virtual address to be used for the level 2 table index lookup (depending on page size).</p> <p>For tiny pages, bits 13:4 of this register are compared against local bus virtual address bits 19-10.</p> <p>For small pages, bits 13:6 are compared against local bus virtual address bits 19-12.</p> <p>For large pages, bits 13:10 are compared against local bus virtual address bits 19-16.</p> <p>For sections, this field is ignored.</p>	R/W	R/W	0
3	P		Preserved bit. When 1, CAM entry is preserved. When 0, CAM entry is not preserved.	R/W	R/W	0
2	V		Valid bit.	R	R	0
		0	CAM entry is not valid			
		1	CAM entry is valid			
1–0	SLST		Page size associated with CAM entry:	R/W	R/W	0
		00	Section (1MB)			
		01	Large page (64KB)			
		10	Small page (4KB)			
		11	Tiny page (1KB)			

The LB MMU RAM registers provide information on the physical address associated with a CAM entry that defines a page of memory in the local bus virtual address space.

Table 61. Local Bus MMU RAM High Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–0	Ram_MSB	Most significant 16 bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0

Table 62. Local Bus MMU RAM Low Register

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
15–10	Ram_LSB		Least significant six bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0
9–8	AP		Access permission bits	R/W	R/W	0
		00	No access. Any local bus access to this page causes a permission fault.			
		01	No access. Any local bus access to this page causes a permission fault.			
		10	Read access only. Any local bus write access to this page causes a permission fault			
		11	Full access. Any local bus access to this page can complete without a permission fault.			
7–0	Unused					

The LB MMU global flush register allows flushing of all nonprotected TLB entries.

Note:

Flushing the whole TLB does not change the base_value or the victim_counter fields of the LB_MMU_LOCK_REG register.

Table 63. Local Bus MMU Global Flush Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	–	–	–
0	Global_flush	When written with a 1, all non-protected TLB entries are flushed. This has no effect when written with a 0. It always returns 0 on read.	R/W	R/W	0

The LB MMU entry flush register allows flushing of individual local bus MMU TLB entries.

Table 64. Local Bus MMU Entry Flush Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–1	Reserved	Reserved	–	–	–
0	flush_entry	When written with a 1, flushes the TLB entry pointed to by the virtual address in LB_MMU_CAM_H_REG and LB_MMU_CAM_L_REG, even if the entry is set as a protected entry. This has no effect when written with a 0. It always reads as 0.	R/W	R/W	0

The LB MMU CAM read registers allow reading of local bus MMU CAM entries. See the LB_MMU_LD_TLB_REG description.

Table 65. Local Bus MMU CAM Read High Register

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
15–6	Reserved		Reserved	–	–	–
5–0	VA_tag_l1_h		Most significant six bits of the local bus virtual address to be used for this entry level 1 table index	R/W	R/W	0

Table 65. Local Bus MMU CAM Read High Register (Continued)

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
15–14	VA_tag_l1_L		Least significant two bits of the local bus virtual address which are used for this entry level 1 table index	R/W	R/W	0
13–4	VA_tag_l2		Bits of the local bus virtual address which may be used for the level 2 table index lookup (depending on page size). For tiny pages, bits 13:4 are used. For small pages, bits 13:6 are used. For large pages, bits 13:10 are used. For sections, this field is ignored.	R/W	R/W	0
3	P		Preserved bit. When 1, CAM entry is preserved. When 0, CAM entry is not preserved.	R/W	R/W	0
2	V		Valid bit. When 1, CAM entry is valid. When 0, CAM entry is not valid	R	R	0
1–0	SLST		Page size associated with CAM entry:	R/W	R/W	0
		00	Section (1MB)			
		01	Large page (64KB)			
		10	Small page (4KB)			
		11	Tiny page (1KB)			

The LB MMU RAM read registers are used to read the local bus MMU TLB RAM. See the LB_MMU_LD_TLB_REG description.

Table 66. Local Bus MMU RAM Read High Register

Bits	Field	Description	Access		Hardware Reset Value
			User	Sup	
15–0	Ram_MSB	Most significant 16 bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0

Table 67. Local Bus MMU RAM Read Low Register

Bits	Field	Value	Description	Access		Hardware Reset Value
				User	Sup	
15–10	Ram_LSB		Least significant six bits of the physical address that corresponds to a local bus virtual address	R/W	R/W	0
9–8	AP		Access permission bits	R/W	R/W	0
		00	No access. Any local bus access to this page causes a permission fault.			
		01	No access. Any local bus access to this page causes a permission fault.			
		10	Read access only. Any local bus write access to this page causes a permission fault.			
		11	Full access. Any local bus access to this page can complete without a permission fault.			
7–0	Unused					

8.2 Local Bus MMU Programming for USB Host Controller Operation

8.2.1 Local Bus MMU Page Size and the USB Host Controller

The OHCI USB host controller assumes that page size is no smaller than 4K bytes. The OHCI USB host controller allows a transfer descriptor data buffer to split between two non-contiguous 4K byte pages in local bus virtual address space. If a transfer descriptor current buffer pointer is in a different 4K byte page of local bus virtual address space than the transfer descriptor buffer end, the OHCI USB host controller reads from or writes to that buffer starting at the current buffer pointer and increasing until it gets to the last byte of the 4K aligned block of local bus virtual address space. It then switches to a local bus virtual address, which has the same 20 upper bits as the transfer descriptor buffer end register, with 12 lower bits at 0.

Because of this behavior, it is strongly recommended that the mapping of local bus virtual addresses to processor physical addresses be done via 4K-byte pages (or larger). If tiny pages (1K byte per page) are used, they must be used in groups of four tiny pages that are contiguous both in local bus virtual address space and in processor physical address space, with the first of the tiny pages aligned on a 4K-byte boundary in both processor physical address space and local bus virtual address space. When using local bus MMU tiny pages, failure to allocate and align groups of four tiny pages as described here results in malfunction of the USB host controller memory access operations.

8.2.2 Local Bus MMU and Page Protection

The access protection features of the local bus MMU are similar to the protection mechanisms provided by the MPU MMU. If a local bus access attempts to access a local bus virtual address within a local bus MMU page that is marked as protected or attempts to write to a page that is marked as read-only, an interrupt is issued to the MPU level 1 interrupt handler.

8.2.3 Local Bus MMU Page Miss

A local bus access that has an address within the valid local bus address range but does not have a corresponding page mapping in the local bus MMU causes the local bus MMU to issue an interrupt to the MPU level 1 interrupt handler.

The MPU can make use of the local bus MMU page miss interrupt to allow software-based conversion between local bus virtual addresses and physical addresses. If using such a software mechanism, be aware that the software overhead must be considered in relation to the local bus time-out counter. If the software cannot provide the converted physical address to the local bus MMU quickly enough for the USB host controller local bus access to complete

before the local bus time-out counter completes its count, the unrecoverable error OHCI interrupt is signaled and the USB host controller stops performing USB packet transactions. It is possible to disable the USB host controller local bus time-out counter, but then there is no protection against the local bus lockup that occurs if the USB host controller attempts to access a local bus virtual address that is outside of the valid local bus virtual address range of 3000:0000h to 3FFF:FFFFh.

9 USB Host Controller Reset and Clock Control

9.1 USB Host Controller Clock Control

The OMAP5910 clock generation and system reset management module (ULPD) provides a 48-MHz clock to the USB host controller. This clock can be stopped by software to reduce USB host controller power consumption when USB host controller operation is not needed.

Clocking for the local bus is controlled by a different mechanism. When the USB host controller needs to access system memory, the local bus must be operating.

9.2 Initializing ULPD to Generate the 48-MHz Clock

The ULPD module generates 48 MHz for the USB host controller using either a digital PLL (DPLL) or an analog PLL (APLL). The USB host controller receives a clock from the ULPD module when the CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit is set. This register bit provides the clock request from the USB host controller to the ULPD.

The ULPD resets to a mode where the 48-MHz clock is generated by the DPLL. This sequence can be used to disable the DPLL and enable the APLL:

- 1) Clear the ULPD DPLL enable bit.
- 2) Wait until the DPLL lock bit goes to 0 to indicate that the DPLL is no longer locked (ULPDs DPLL_CTRL_REG register LOCK bit).
- 3) Set the APLL enable bit (ULPDs APLL_CTRL_REG register, APLL_NDPLL_SWITCH bit). (When this bit is 0, the DPLL output is selected rather than the APLL output).
- 4) Wait until the APLL global lock bit is 1 (ULPDs register, GLOBAL_LOCK bit).

When the ULPD module selects the APLL, the DPLL is shut off and all OMAP5910 modules that use 48 MHz receive 48 MHz from the APLL. When the ULPD module selects the DPLL, the APLL is shut off and all OMAP5910 modules that use 48 MHz receive 48 MHz from the DPLL.

9.3 USB Host Controller Hardware Reset

Reset of the USB host controller is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5910 peripherals, including the USB host controller. When held in reset, the USB host controller does not generate any USB activity on its USB ports. The USB host controller requires that its 48-MHz clock input (from the OMAP5910 ULPD module) be active and that the OMAP5910 configuration register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit be set in order to complete its reset sequence.

There is a delay of approximately 72 cycles of the ULPD USB host controller 48-MHz clock before the USB host controller is successfully reset. This delay starts at the latest of PER_EN bit set, CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit set, or 48-MHz clock start. When the USB host controller is in hardware reset, read or write accesses to its registers have no effect. It is recommended that USB host controller software read the HcRevision and HcHCCA registers after deasserting reset to verify the proper reset values. If the read values for both HcRevision and HcHCCA are not the correct, reset the values and continue reading until the proper reset values are seen.

The CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit, when cleared, also holds the USB host controller in a hardware reset. While the USB host controller is in reset, reads from the USB host controller registers do not return valid data and writes to the USB host controller registers have no effect.

Software that initializes the USB host controller must ensure that the reset is turned off, that the ULPD 48-MHz clock for the USB host controller is enabled, and that the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit is set. It must then wait until reads of both the HcRevision register and the HcHCCA register return their correct reset default values.

9.4 USB Host Controller OHCI Reset

The *OHCI Specification for USB* provides the HCR bit in the HCCCommand Status register, which resets the OHCI controller. This reset may be used to reset the OHCI functionality and has no effect on the USB host controller local bus and MPU public peripheral bus interfaces.

9.5 USB Host Controller Power Management

Power management of the OMAP5910 USB host controller is limited to disabling the clock to the USB host controller using the MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HHC_UHOST_EN_R bit. When this bit is

0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by CONF_MOD_USB_HOST_HMC_MODE_R is not affected, so a CONF_MOD_USB_HOST_HMC_MODE_R setting that multiplexes USB function controller and/or UART1 signals to OMAP5910 top-level multiplexing can still make use of the USB function controller and/or UART1.

When the OMAP5910 host controller's 48-MHz clock is disabled, all USB host controller OHCI registers and the HostUEAddr, HostUEStatus, HostTimeoutCtrl, and HostRevision registers are inaccessible.

9.6 Local Bus Clock

Proper operation of the USB host controller requires that the local bus clock be enabled. For details, see Section 7.11, *Local Bus Initialization*.

10 OMAP5910 USB Hardware Considerations

10.1 VBUS Power Switching For USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using Texas Instruments devices such as the TPS2014 and TSP2015 power distribution switch devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

10.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

10.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The OMAP5910 device provides the input pin USB_VBUSI, which is provided to the OMAP5910 USB function controller. This input is a CMOS input that is not rated for the full VBUS range specified by the USB specification. An external signal level converter is required to convert the VBUS signal range to a range suitable for the USB_VBUSI pin.

10.4 USB D+ Pullup Enable for USB Function Controller

When using a USB signal multiplexing mode that provides USB function controller signals to OMAP5910 pins, the OMAP5910 top-level pin multiplexing options lead to several possible USB pullup implementations.

When the USB.PUEN pin is set for top-level multiplexing configuration 0, the USB.PUEN pin is driven low when the pullup should be active and is driven high when the pullup should be inactive. In this mode of operation, an external inverter or an external 3-stateable device can be used to provide a nominal 3.3-V signal to the supply end of the USB D+ pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 1, the USB.PUEN pin provides a clock output and cannot be used to control the USB pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 2, the USB.PUEN pin is driven high when the pullup should be active and is driven low when the pullup should be inactive. In this mode of operation, the pullup resistor may be connected directly between the OMAP5910 USB.PUEN and the D+ signal.

When the USB.PUEN pin is set for top-level multiplexing configuration 3, the USB.PUEN pin is driven high when the pullup should be active and is placed in high impedance mode the pullup should be inactive. In this mode of operation, the pullup resistor can be connected directly between the OMAP5910 USB.PUEN and the D+ signal.

10.5 Port Passthrough Mode

When MOD_CONF_CTRL_0 register CONF_MOD_USB_HOST_HMC_MODE_R is 7 (with appropriate top level signal multiplexing settings), the signals from six OMAP5910 input pins are passed to six OMAP5910 output pins. This mode is described in Table 68.

Table 68. CONF_MOD_USB_HOST_HMC_MODE_R=7 Internal Connectivity

HMC_MODE 7		
Input Pin Name		Output Pin Name
MCSI1.DIN/USB2.VP	is logically connected to	RST_HOST_OUT/USB1_SE0
UART2.CTS/USB2.RCV	is logically connected to	MCBSP.CLK/USB1_TXEN
UART2.RX/USB2.VM	is logically connected to	MCSI1.DOUT/USB1.TXD
MCSI1.SYNC/USB1.VP	is logically connected to	UART2.RTS/USB2_SE0
MCSI1.DIN/USB1.RCV	is logically connected to	MCSI2.DOUT/USB2.TXEN
MCSI1.CLK/USB1.VM	is logically connected to	UART2.TX/USB2.TXD

10.6 UART1 Connectivity when CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24

The USB signal multiplexing provides modes that allow UART1 signals to be brought to OMAP5910 pins. This multiplexing is separate from UART1 multiplexing that is provided by OMAP5910 top-level pin multiplexing. When CONF_MOD_USB_HOST_HMC_MODE_R is set to 2, 10, 18, or 24, three signals to/from UART1 are provided to OMAP5910 pins when the OMAP5910

top-level pin multiplexing for those pins selects the USB signal multiplexer. These signal assignments are described in Table 69.

Table 69. CONF_MOD_USB_HOST_HMC_MODE_R = 2, 10, 18, and 24 UART Signal Assignments

OMAP5910 Pin Name	Signal Name	Direction
MCSI1.DOUT/USB1.TXD	UART1_TX	Output
MCSI1.BLK/USB1.VM	UART1_RX	Input
MCSI1.DIN/USB1.RCV	UART1_CTS	Input

10.7 MPU_BOOT Signal Sharing

The OMAP5910 device implements shared functionality on the MPU_BOOT/USB1.SUSP pin. The MPU_BOOT pin is sampled at hardware reset. If low, the MPU processor boots from memory connected to CS0 on the EMIFS; if high, the MPU processor enters the boot overlay mode, causing it to boot from memory connected to CS3 on the EMIFS. After reset, the pin may be configured for other functionality, such as the USB1.SUSP output. The MPU_BOOT signal has an internal pulldown resistor that is enabled by default. The boot overlay mode requires an external pullup resistor. The internal pulldown can be disabled in the OMAP configuration registers.

10.8 USB D+, D– Pulldown for USB Function Controller

System implementations which use the OMAP5910 USB function controller and are sensitive to supply current requirements may wish to implement weak pulldown resistors on the USB D+ and D– signals associated with the USB Type B receptacle. When there is no host controller attached upstream of the USB Type B receptacle, the undriven D+ and D– wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistor depends on transceiver characteristics, and board layout and must be designed to meet USB D+ and D– signal requirements.

11 Overview of the OMAP5910 USB Functional Module

This chapter describes the components and features of the OMAP5910 universal serial bus (USB) function module. The USB function module supports the implementation of a full-speed device fully compliant with the USB 1.1 standard. It provides an interface between the MPU core (TI925T) and the USB wire, and it handles USB transactions with minimal MPU intervention.

The module supports one control endpoint (EP0) (IN and OUT), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. For each endpoint, the specific parameters of a configuration are the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or isochronous), and the associated number.

The module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or isochronous transactions.

This chapter uses terminology defined in the USB1.1 Standard. Reader familiarity with this Standard is assumed. All references to local host (LH) in this chapter refer to the MPU processor.

Figure 35 shows the connection of the USB function module within the OMAP5910 in detail.

11.1 OMAP5910 Inputs/Outputs

Several configurations are possible for the USB function:

- ☐ USB function usable with internal transceiver (default configuration)
- ☐ USB function usable with external transceivers
- ☐ USB function not usable

11.2 USB Function Interrupts

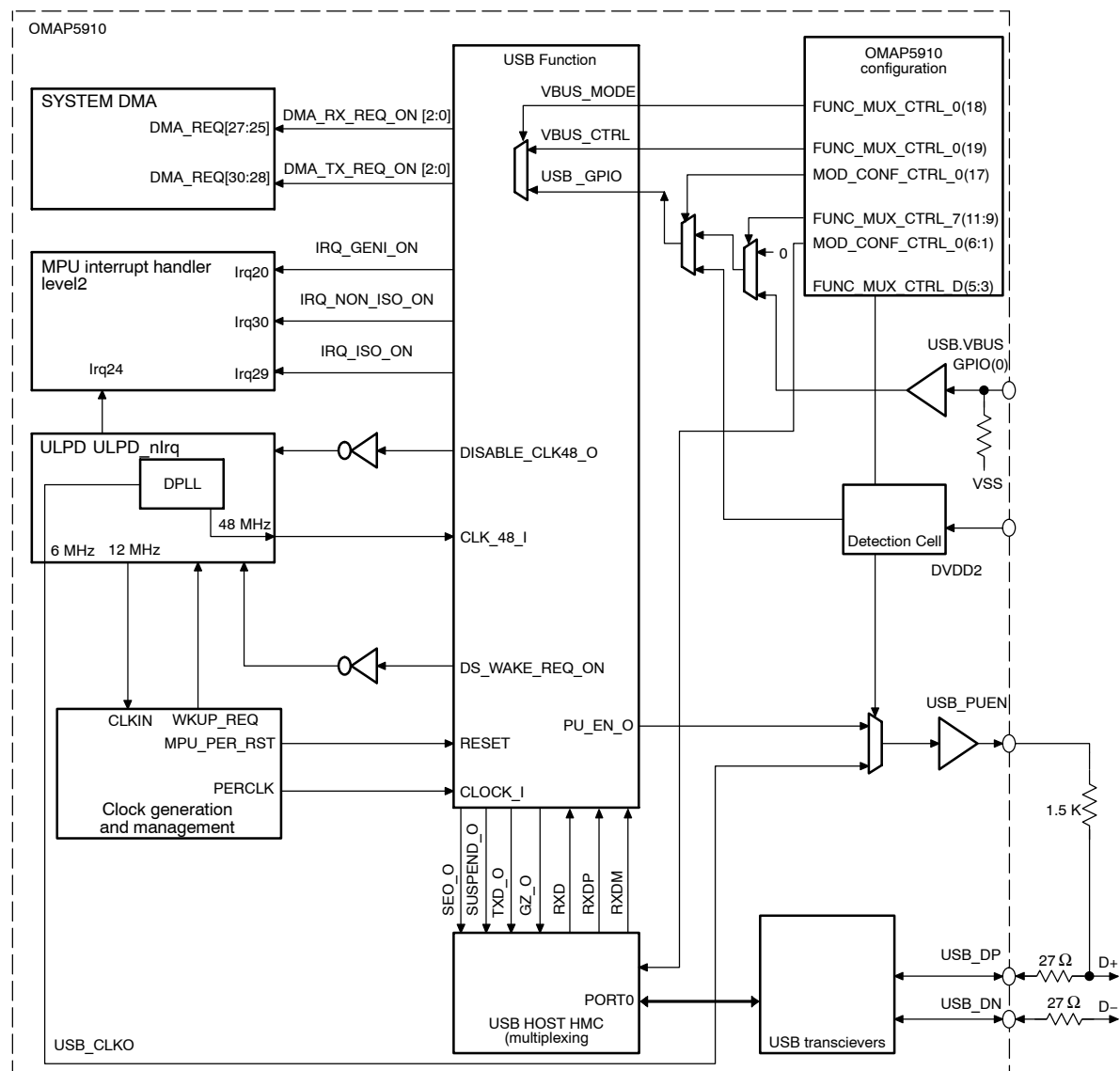
The USB function generates three interrupts:

- 1) General USB interrupt (including endpoint 0, DMA, and device states interrupts), `IRQ_GENI_ON`: Connected to the MPU level 2 interrupt handler, line 20 (level-sensitive)
- 2) Non-ISO endpoint-specific Interrupt, `IRQ_NON_ISO_ON`: Connected to the MPU level 2 interrupt handler, line 30 (level-sensitive)
- 3) Start of frame (SOF) interrupt for ISO transactions, `IRQ_ISO_ON`: Connected to the MPU level 2 interrupt handler, line 29 (level-sensitive)

The IRQ_ISO_ON interrupt is also connected to the frame adjustment counter module (FAC) to count the number of frame start.

This count value can then be used by system software to adjust the duration of the two time domains with respect to each other to reduce the overflow and underflow.

Figure 35. USB Function Environment



11.3 USB Function Clocks and Reset

The USB function has two clocks:

- An interface clock (CLOCK_I), used between the MPU TIPB and the USB function and connected to the MPU peripheral programmable clock (PERCLK), is derived by dividing CK_GEN1 (the output of DPLL1) by the value associated with the PERDIV field of the ARM_CKCTL register (0xFFFECE00).

This is a free-running clock when the system is awake.

- A 48-MHz functional clock (CLK_48_I), which is generated by the ULPD DPLL.

This clock can be shutdown by the USB function (DISABLE_CLK48_O) when:

- The USB is suspended (idle state).
- The USB is disconnected.

This shutdown must be enabled beforehand: Set SOFF_Dis (bit 1), of the SYSCON1 register to 0 (default value).

The MPU TIPB reset (MPU_PER_RST) resets the USB function.

11.4 USB Function DMA Requests

The USB function can use:

- Three receive DMA channels (DMA_RX_REQ_ON [2:0]): Any endpoint number (1:15) can be selected to be assigned to one receive DMA channel.

The DMA_RX_REQ_ON [2:0] are connected to the system DMA request [27:25].

- Three transmit DMA channels (DMA_TX_REQ_ON [2:0]): any endpoint number (1:15) can be selected to be assigned to one transmit DMA channel.

The DMA_TX_REQ_ON [2:0] are connected to the system DMA request [30:28].

11.5 USB Detection

When OMAP5910 is in deep sleep mode, the USB function can detect a bus connection to an external USB host or hub and generate a deep sleep wake request (DS_WAKE_REQ_ON) to wake up the system and to get the interface clock (CLOCK_I).

The USB function can also generate a DS_WAKE_REQ_ON request while the USB is connected: if the USB function is in idle (SUSPEND state) and a resume event occurs, the DS_WAKE_REQ_ON request is generated. This request does not wake up the system itself.

The ULPD module uses this request to generate an interrupt (ULPD_nlrq) to the MPU, which wakes up the system via its wake-up request (WKUP_REQ) (see *SPRU678, OMAP5910 Clock Generation and System Reset Management Reference Guide*).

Once the interface clock is present, the USB function can generate an attached/unattached interrupt when it detects that the OMAP5910 device is connected to an external USB host or hub or when it becomes disconnected.

This attached/unattached interrupt uses the general USB interrupt line (connected to the MPU level 2 interrupt handler, line 20) and the connection state is given by the ATT bit(0) of the DEVSTAT register.

The USB function can detect whether or not the USB is connected via either software or hardware.

This choice (soft/hard) is made by the VBUS_MODE bit (18) of the OMAP5910 FUNC_MUX_CTRL_0 register:

- ☐ VBUS_MODE = 1 (default value): Software detection is selected.
- ☐ VBUS_MODE = 0: Hardware detection is selected.

11.5.1 Software Detection

This detection depends on the VBUS_CTRL bit (19) of the OMAP5910 FUNC_MUX_CTRL_0 register:

- ☐ VBUS_CTRL=0 (default): USB not connected
- ☐ VBUS_CTRL=1: USB connected

11.5.2 Hardware Detection

This detection can have two sources:

- ☐ The GPIO(0) OMAP5910 input
- ☐ The OMAP5910 input of the USB function I/O power supply (DVDD2)

The selection between these two sources is made by the USB_W2FC_VBUS_MODE bit(17) of the MOD_CONF_CTRL_0 register (bit usable only in OMAP5910 configuration mode):

- ☐ USB_W2FC_VBUS_MODE = 0 (default): GPIO(0) is selected.
- ☐ USB_W2FC_VBUS_MODE = : I/O power supply detection is selected.

GPIO0 Detection

This detection must be enabled by software as follows:

- ☐ In the OMAP5910 configuration mode, when the VBUS_GPIO0_SELECT bit (12) (FUNC_MUX_CTRL_1 register) is set

Or

- ☐ In the OMAP5910 configuration mode, when the CONF_GPIO0_MODE bits (11:9) (FUNC_MUX_CTRL_7 register) are set to 010

The results of this detection are as follows:

- ☐ GPIO0 input = 0: USB not connected (default value via internal pulldown)
- ☐ GPIO0 input = 1: USB connected

I/O Power Supply Detection

An analog cell is used to detect the presence or not of the USB I/O power supply (DVDD2), which permits the detection of the presence or absence of the USB.

- ☐ DVDD2 present: USB connected
- ☐ DVDD2 not present: USB not connected

11.6 Software Disconnect

The PULLUP_EN (0) bit of the SYSCON1 register allows the device to disconnect itself from the USB.

This bit is by default directly connected to the OMAP5910 pad (USB.PUEN), which must be connected on the application board to a 1.5-k Ω resistor.

The other pin of the resistor must be connected to the positive differential line (OMAP5910 pad USB.DP) of the USB (D+).

Thus:

- ☐ PullUp_En bit=0 (by default): The external 1.5-k Ω is seen as a pulldown on the USB D+; the external USB host cannot detect the OMAP5910 USB function.
- ☐ PullUp_En bit=1: The external 1.5-k Ω is seen as a pullup on the USB D+, the USB host detects this level and, therefore, the presence of the OMAP5910 USB function. The USB host can then configure the USB function.

The USB.PUEN signal is multiplexed inside OMAP5910 with the USB.CLKO clock. Bits (5:3) of the FUNC_MUX_CTRL_D register control this multiplexing:

- ☐ FUNC_MUX_CTRL_D(5:3)= 000 (default): USB.PUEN signal is output.
- ☐ FUNC_MUX_CTRL_D(5:3)= 001: USB.CLKO clock is output.

The USB.CLKO clock comes from the ULPD DPLL after an internal dividing by 8 (6 MHz) (see *SPRU678, OMAP5910 Clock Generation and System Reset Management Reference Guide* for more details). This clock is dedicated to an external USB HUB.

12 Register Map

Table 70 lists the USB function registers. Table 71 through Table 93 describe the register bits. The MPU base address is FFFB:4000.

Table 70. USB Function Module Registers

Register	Description	Access	Offset Address
REV	Revision	R	0x00
Endpoint			
EP_NUM	Endpoint selection	R/W	0x04
DATA	Data	R/W	0x08
CTRL	Control	Set only	0x0C
STAT_FLG	Status flag	R	0x10
RXFSTAT	Receive FIFO status	R	0x14
SYSCON1	System configuration 1	R/W	0x18
SYSCON2	System configuration 2	Set only	0x1C
DEVSTAT	Device status	R	0x20
SOF	Start of frame	R	0x24
IRQ_EN	Interrupt enable	R/W	0x28
DMA_IRQ_EN	DMA interrupt enable	R/W/Clear	0x2C
IRQ_SRC	Interrupt source	R/Clear	0x30
EPN_STAT	Endpoint interrupt status	R	0x34
DMAN_STAT	DMA endpoint interrupt status	R	0x38
Reserved			0x3C
DMA Configuration			
RXDMA_CFG	Receive channels DMA configuration	R/W	0x40
TXDMA_CFG	Transmit channels DMA configuration	R/W	0x44
DATA_DMA	DMA FIFO data	R/W	0x48
Reserved			0x4C
TXDMA0	Transmit DMA control 0	R/W	0x50

Table 70. USB Function Module Registers (Continued)

Register	Description	Access	Offset Address
DMA Configuration (Continued)			
TXDMA1	Transmit DMA control 1	R/W	0x54
TXDMA2	Transmit DMA control 2	R/W	0x58
Reserved			0x5C
RXDMA0	Receive DMA control 0	R/W	0x60
RXDMA1	Receive DMA control 1	R/W	0x64
RXDMA2	Receive DMA control 2	R/W	0x68
Reserved			0x6C- 0x7C
Endpoint Configuration			
EP0	Endpoint configuration 0	R/W	0x80
EP1_RX	Receive endpoint configuration 1	R/W	0x84
EP2_RX	Receive endpoint configuration 2	R/W	0x88
...			...
EP15_RX	Receive endpoint configuration 15	R/W	0xBC
Reserved			0xC0
EP1_TX	Transmit endpoint configuration 1	R/W	0xC4
EP2_TX	Transmit endpoint configuration 2	R/W	0xC8
...			...
EP15_TX	Transmit endpoint configuration 15	R/W	0xFC

Note on register accesses:

- ☐ The local host may read from or write into registers using one of the following accesses:
 - 16-bit access: All bits of the register are accessed.
 - 8-LSB bit access: The 8 least significant bits are accessed.
 - 8-MSB bit access: The 8 most significant bits are accessed.

Local host actions are required in some particular cases when reading or writing data, depending on the access mode.

12.1 Revision Register (REV)

The read-only revision register (REV) contains the revision number of the module. A write to this register is forbidden.

Table 71. Revision Register (REV)

Bits	Field	Description
15–8	–	Reserved
7–0	Rev_nb	Revision number

12.1.1 REV_NB

This 8-bits field indicates the revision number of the current USB function module. This value is fixed by hardware.

0x01: Revision 0.1

0x02: Revision 0.2

0x21: Revision 2.1

....

Local host (LH) and universal serial bus (USB) reset have no effect on this register.

12.2 Endpoint Selection Register (EP_NUM)

The read/write endpoint selection register (EP_NUM) selects and enables the endpoint that can be accessed by the local host.

Table 72. Endpoint Selection Register (EP_NUM)

Bits	Field	Description
15–7	–	Reserved
6	Setup_Sel	Setup FIFO select
5	EP_Sel	TX / RX FIFO select
4	EP_Dir	Endpoint direction
3–0	EP_Num	Endpoint number

12.2.1 Setup FIFO Select (Setup_Sel)

Set by the local host in response to a setup general USB interrupt in order to access the EP0 read-only setup FIFO when reading the DATA register. Setting this bit clears the Setup interrupt bit. When this bit is set, other EP_NUM register bits must be 0.

CAUTION
After having read the setup FIFO, the local host must clear this bit by writing a 0.

0: No access

1: Access permitted

USB reset value: 0

Local host reset value: 0

12.2.2 TX/RX FIFO Select (EP_Sel)

Set by the local host to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If EP_Dir bit is set to 0, the local host can read data from endpoint RX FIFO by reading the DATA register; if EP_Dir bit is set to 1, the local host can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the local host must clear this bit.

CAUTION
Before the local host sets this bit, it must set Setup_Sel bit to 0. After having accessed the endpoint FIFO either for read or for write access the local host must clear this bit by writing a 0.

0: No access

1: Access permitted

Value after local host or USB reset is low.

12.2.3 Endpoint Direction (EP_Dir)

This bit gives the direction associated with the endpoint number selected in EP_Num.

0: OUT endpoint

1: IN endpoint

Value after local host or USB reset is low.

12.2.4 Endpoint Number (EP_Num)

The endpoint number binary encoded in these four bits, associated to the direction given by EP_Dir bit, is the current endpoint selected. All reads and writes to the endpoint status, control, and data locations are for this endpoint.

0000: EP0

0001: EP1

....

1111: EP15.

Value after local host or USB reset is low.

12.3 Data Register (DATA)

The data register (DATA) is the entry point to write into a selected TX endpoint, to read data from a selected RX endpoint, or to read data from the setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is forbidden. If selected endpoint direction is IN, this register is write-only and a read of this register is forbidden.

Table 73. Data Register (DATA)

Bits	Field	Description
15–0	DATA	Transmit/receive FIFO data

12.3.1 Transmit/Receive FIFO Data (DATA)

EP_Dir = 0: This register contains the data received by the USB core from USB host out or setup transactions. Data can only be read successfully if the EP_Sel bit is asserted, or if Setup_Sel bit is asserted (for setup data).

EP_Dir = 1: This register contains the data written by the local host to be sent to the USB host during the next IN transaction. Data can only be written successfully if the EP_Sel bit is asserted.

Note:

Writing the DATA register when EP_Dir = 0 and reading from DATA register when EP_Dir = 1 are denied.

12.4 Control Register (CTRL)

This set-only control register (CTRL) controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Note:

The endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register (CTRL) is implemented for this FIFO, because the local host cannot control it.

Table 74. Control Register (CTRL)

Bits	Field	Description
15–8	–	Reserved
7	Clr_Halt	Clear halt endpoint (non-isochronous)
6	Set_Halt	Set halt endpoint (non-isochronous)
5–3	–	Reserved
2	Set_FIFO_En	Set FIFO enable (non-isochronous)
1	Clr_EP	Clear endpoint
0	Reset_EP	Endpoint reset (non-Ctrl)

12.4.1 Clear Halt Endpoint (Clr_Halt)

Only concerns non-isochronous endpoints.

Used by the local host to clear an endpoint halt condition.

0: No action

1: Clear halt condition

Always read 0.

Note:

It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Clr_Halt bit, in order to avoid possible impacts on interrupts.

12.4.2 Set Halt Endpoint (Set_Halt)

Only concerns non-isochronous endpoints.

Used by the local host to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The local host can disable the endpoint interrupt if it does not wish to be informed of STALL handshakes.

CAUTION
If the endpoint to halt is used by a DMA channel, the local host must disable the DMA channel before setting the halt conditions for this endpoint.

0: No action

1: Halt endpoint

Always read 0.

Note:

It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Set_Halt bit, in order to avoid possible impacts on interrupts.

The local host must check that FIFO is empty before setting the halt feature for the endpoint. A stalled transaction has no effect in clearing the FIFO.

12.4.3 Set FIFO Enable (Set_FIFO_En)

Only concerns non-isochronous endpoints.

If the selected endpoint direction is IN, this bit is used by the local host to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, this bit is used by the local host to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not set, the device returns a NAK handshake.

Isochronous endpoint FIFOs are always enabled.

The local host must never enable the endpoint 0 FIFO if not performing a control transfer. For bulk and interrupt endpoints, the FIFO must never be enabled when Set_Halt = 1 (halt feature enabled) or when RX FIFO is not empty. Furthermore, during endpoint interrupt handling, the local host must have cleared the interrupt bit before setting the Set_FIFO_En bit (to avoid masked ACK interrupts).

0: No action

1: FIFO enabled

Always read 0.

Note:

It is not required to set the EP_Sel bit before setting this bit. Except when this bit is set during the handling of an interrupt to the endpoint, the local host must not set the EP_Sel bit before setting the Set_FIFO_En bit, in order to avoid possible impacts on interrupts.

12.4.4 Clear Endpoint (Clr_EP)

This bit is set by the local host to clear the selected endpoint FIFO pointers and flags. This resets the FIFO pointers and the FIFO empty status bit. The FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. Previous transaction handshake status is also cleared. For isochronous endpoints or non-isochronous double-buffered endpoints, both foreground and background FIFO are cleared.

0: No action

1: Clear endpoint

Always read 0.

12.4.5 Endpoint Reset (Reset_EP)

Only concerns non-control endpoints.

Set by the local host to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears halt condition (HALT = 0), and clears FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an isochronous endpoint, it only clears the FIFO (both foreground and background).

0: No action

1: Reset endpoint

Always read 0.

12.5 Status Register (STAT_FLG)

The read-only status flag register provides a status of the FIFO and the results of the transaction handshakes for the selected endpoint. The eight MSB are reserved for isochronous endpoints, while the eight LSB are reserved for non-isochronous endpoints. This register cannot be read if EP_Sel bit is not asserted for the endpoint. No status flag exists for the read-only setup FIFO, which is always enabled.

The updates for non-isochronous transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no interrupt is pending on the endpoint.

Note:

Non-transparent, non-isochronous IN transactions are those transactions responding with an ACK handshake, a STALL handshake, or optionally a NAK handshake if the Nak_En bit is asserted to 1. An ERR handshake or a NAK handshake when the Nak_En bit is 0 is considered transparent.

A write to this register has no effect.

Table 75. Status Register (STAT_FLG)

Bits	Field	Description
15	–	Reserved
14	Miss_In	Isochronous missed IN token for the previous frame (isochronous IN)
13	Data_Flush	Isochronous receive data flush (isochronous OUT)
12	ISO_Err	Isochronous receive data error (isochronous OUT)
11–10	–	Reserved
9	ISO_FIFO_Empty	Isochronous FIFO empty
8	ISO_FIFO_Full	Isochronous FIFO full
7	–	Reserved
6	EP_Halted	Endpoint halted flag (non-isochronous)
5	STALL	Transaction stall (non-isochronous)

Table 75. Status Register (STAT_FLG) (Continued)

Bits	Field	Description
4	NAK	Transaction non-acknowledge (non-isochronous)
3	ACK	Transaction acknowledge (non-isochronous)
2	FIFO_En	FIFO enable status (non-isochronous)
1	Non_ISO_FIFO_Empty	Non-isochronous FIFO empty
0	Non_ISO_FIFO_Full	Non-isochronous FIFO full

12.5.1 Isochronous Missed IN Token (Miss_In)

Only concerns isochronous IN endpoints.

Notifies the local host that the core missed a valid isochronous IN token during previous frame and that TX data was flushed from the FIFO instead of being transmitted to the USB host. This bit is updated on a start of frame (SOF).

0: The endpoint received an IN token the previous frame.

1: The endpoint did not receive an IN token the previous frame and TX data was flushed.

Value after local host or USB reset is low.

12.5.2 Isochronous Receive Data Flush (Data_Flush)

Only concerns isochronous OUT endpoints.

When set, this bit indicates that data was flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the local host does not read all of the data from the foreground FIFO in a frame.

This bit is updated every frame.

0: Not significant

1: Data was flushed

Value after local host or USB reset is low.

12.5.3 Isochronous Receive Data Error (ISO_Err)

Only concerns isochronous OUT endpoints.

When set, this bit indicates that the isochronous data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty.

This bit is updated every frame.

0: Not significant

1: Isochronous packet received with errors

Value after local host or USB reset is low.

12.5.4 Isochronous FIFO Empty (ISO_FIFO_Empty)

Only concerns isochronous endpoints.

Set when the FIFO for the selected isochronous endpoint is empty, either via an appropriate write to the Clr_EP bit or the Reset_EP bit, or after successful reads from the selected FIFO.

0: Isochronous FIFO not empty

1: Isochronous FIFO empty

Value after local host or USB reset is high (FIFO empty).

12.5.5 Isochronous FIFO Full (ISO_FIFO_Full)

Only concerns isochronous endpoints.

Set when the FIFO for the selected isochronous endpoint is full. This condition is cleared by setting the Clr_EP bit or the Reset_EP bit, or after one successful read (by the local host or the USB host).

0: Isochronous FIFO not full

1: Isochronous FIFO full

Value after local host or USB reset is low (FIFO empty).

12.5.6 Endpoint Halted Flag (EP_Halted)

Only concerns non-isochronous endpoints.

This bit, when set to 1, indicates the selected endpoint is halted. The endpoint can be put into the halt state only by the local host writing the endpoint halt control bit (in response to a Set_Feature request, for instance).

0: The selected endpoint is not halted.

1: The selected endpoint is halted.

Value after local host or USB reset is low.

12.5.7 Transaction Stall (STALL)

Only concerns non-isochronous endpoints.

This status bit is set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no interrupt is pending on current buffer. The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by an halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No STALL handshake was returned.

1: A STALL handshake packet was returned.

Value after local host or USB reset is low.

12.5.8 Transmit Non-Acknowledge (NAK)

Only concerns non-isochronous endpoints with the Nak_En bit asserted.

This status bit is set at the end of a transaction if a NAK handshake is returned to the USB host, and if no interrupt is pending on current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint and the FIFO_En bit is not set for the endpoint. The bit is cleared when the local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No NAK handshake was returned (the Nak_En bit is set).

1: A NAK handshake packet was returned and the Nak_En bit is set.

Value after local host or USB reset is low.

12.5.9 Transaction Acknowledge (ACK)

Only concerns non-isochronous endpoints.

Set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host, and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device, and the ACK handshake was returned. The bit is cleared when the local host has finished handling the corresponding interrupt (at EP_Sel bit deselection).

0: No ACK handshake packet was returned.

1: An ACK handshake packet was returned.

Value after local host or USB reset is low.

12.5.10 FIFO Enable (FIFO_En)

Only concerns non-isochronous endpoints.

This bit is asserted when the Set_FIFO_En bit is set to 1 and is cleared automatically after a transaction completes with an ACK or STALL.

0: The non-isochronous endpoint FIFO is disabled.

1: The non-isochronous endpoint FIFO is enabled.

Value after local host or USB reset is low.

12.5.11 Non-Isochronous FIFO Empty (Non_ISO_FIFO_Empty)

Only concerns non-isochronous endpoints.

Set when the FIFO for the selected non-isochronous endpoint is empty, either via an appropriate Clr_EP bit or the Reset_EP bit or after successful reads from the selected FIFO.

0: Non-isochronous FIFO not empty

1: Non-isochronous FIFO empty

Value after local host or USB reset is high (FIFO empty).

12.5.12 Non-Isochronous FIFO Full (Non_ISO_FIFO_Full)

Only concerns non-isochronous endpoints.

Set when the FIFO for the selected non-isochronous endpoint is full. This condition is cleared by setting the Clr_EP bit or the Reset_EP bit, or after one successful read (by the local host or the USB host).

0: Non-isochronous FIFO not full

1: Non-isochronous FIFO full

Value after local host or USB reset is low (FIFO empty).

12.6 Receive FIFO Status Register (RXFSTAT)

The read-only receive FIFO status register (RXSTAT) tells how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The local host cannot read this register if EP_Sel bit is not set for the endpoint.

Note:

No receive FIFO status exists for the setup FIFO, because 8 bytes always are expected.

Table 76. Receive FIFO Status Register (RXSTAT)

Bits	Field	Description
15–10	–	Reserved
9–0	RXF_Count	Receive FIFO byte count

12.6.1 Receive FIFO Byte Count (RXF_Count)

This 10-bit field indicates the number of bytes currently in the receive FIFO.

Value after local host or USB reset is low (all 10 bits).

12.7 System Configuration Register 1 (SYSCON1)

The read/write system configuration 1 register (SYSCON1) provides control functions for power management and miscellaneous control for the core.

Table 77. System Configuration Register 1(SYSCON1)

Bits	Field	Description
15–9	–	Reserved
8	Cfg_Lock	Device configuration locked
7–5	–	Reserved
4	Nak_En	NAK enable
3	–	Reserved
2	Self_Pwr	Self-powered
1	SOFF_Dis	Shutoff disable
0	Pullup_En	External pullup enable

12.7.1 Device Configuration Locked (Cfg_lock)

After the local host has entered the device configuration (registers 0x20 to 0x3F), it must set this bit so that the device can be used. If the device configuration is not locked, the device is not ready to be used.

0: Device configuration is not locked. Device is not ready.

1: Device configuration is locked.

Value after local host reset is low, after USB reset is unchanged (keep previous configuration).

12.7.2 NAK Enable (Nak_En:)

This bit can be set by the local host so that it will be signaled for NAK transaction handshake response. When this bit is set, the NAK bit is set on a NAK handshake if no interrupt is pending on the endpoint and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the local host and no interrupt is asserted.

0: NAK disabled

1: NAK enabled

Value after local host or USB reset is low.

Note:

If the local host sets this bit, it must wait for a NAK interrupt before selecting the TX endpoint to write TX data.

12.7.3 Self-Powered (Self_Pwr)

Indicates to the USB host whether the device is bus-powered or self-powered. This is needed for a GET_DEVICE_STATUS auto-decoded request. The local host must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see USB 1.1 specification chapter 9).

0: Bus-powered

1: Self-powered

Value after local host reset is low, after USB reset is unchanged.

12.7.4 Shutoff Disable (SOFF_Dis)

When this bit is set, it disables the power shutoff circuitry.

0: Power shutoff circuitry enabled

1: Power shutoff circuitry disabled

Value after local host reset is low, after USB reset is unchanged.

12.7.5 External Pullup Enable (Pullup_En)

Allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent USB traffic when the device is not ready.

0: Pullup disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off.

1: Pullup enabled.

Value after local host reset is low, after USB reset is high, after detach is unchanged.

12.8 System Configuration Register 2 (SYSCON2)

The set-only system configuration 2 register (SYSCON2) provides miscellaneous controls for the function. A read of this register always returns 0.

Table 78. SYSCON2 – System Configuration Register 2 (SYSCON2)

Bits	Field	Description
15–7	–	Reserved
6	Rmt_Wkp	Remote wakeup
5	Stall_Cmd	Stall command
4	–	Reserved
3	Dev_Cfg	Device configured
2	Clr_Cfg	Clear configured
1–0	–	Reserved

12.8.1 Remote Wakeup (Rmt_Wkp)

This set-only bit when written with a 1 initiates the remote wakeup sequence, regardless of whether or not the R_Wk_OK bit has been previously set to 1 by the USB host. So the MPU must make sure that remote wakeup has been enabled by the USB host by reading the DEVSTAT register to check that the R_Wk_OK bit [6] is set to 1 before generating a remote wakeup by writing a 1 to the Rmt_Wkp bit [6]. Reading the Rmt_Wkp bit always returns 0. Writing 0 into this bit has no effect.

0: No action

1: Initiates the remote wakeup sequence

Always read 0.

12.8.2 Stall Command (Stall_Cmd)

Only concerns non-autodecoded requests on control endpoint (EP0).

This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. This set-only bit always reads 0.

0: No action

1: Stall current USB command

Always read 0.

12.8.3 Device Configured (Dev_Cfg)

If the local host receives a SET_CONFIGURATION with a valid configuration value and the device is in addressed state, it must write a 1 to this bit to inform the command decodes that the device has moved to the configured state. The CFG bit is set to 1 by the core.

If the device is already configured when the SET_CONFIGURATION request is received, the local host must not set this bit. If the new configuration value is 0, it must set the Clr_Cfg bit in order to move to the addressed state.

Reading this bit always returns 0. Writing 0 to this bit has no effect.

0: No action

1: Allows CFG to be set

Always read 0.

12.8.4 Clear Configured (Clr_Cfg)

If the local host receives a SET_CONFIGURATION with a configuration value of 0 and if device is configured, it must write a 1 to this bit to inform the command decoder that the device is now deconfigured (has moved to the addressed state). The CFG bit is cleared by the core.

Reading this bit always returns 0. Writing 0 to this bit has no effect.

0: No action

1: Allows CFG to be cleared

Always read 0.

12.9 Device Status Register (DEVSTAT)

The read-only device status register (DEVSTAT) provides a status reflecting the visible device states as defined in USB1.1 chapter 9. A write to this register has no effect.

This register is double buffered. If the DS_Chg_IE bit is set (interrupt enabled), the background register is moved to foreground position only after clearing any pending DS_Chg interrupts. So if there is a state change and there is still a pending DS_Chg interrupt, then recent state change is not visible because the background register was updated and not moved into foreground position.

Table 79. Device Status Register (DEVSTAT)

Bits	Field	Description
15–7	–	Reserved
6	R_WK_OK	Remote wakeup granted
5	USB_Reset	USB reset signaling is active
4	SUS	Suspended state
3	CFG	Configured state
2	ADD	Addressed state
1	DEF	Default state
0	ATT	Attached state

12.9.1 Remote Wakeup Enabled (R_WK_OK)

This bit is automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host has granted the function the ability to assert remote wakeup.

0: Not significant

1: Remote wakeup granted

Value after local host or USB reset is low.

12.9.2 USB Reset Signaling (USB_Reset)

This bit returns 1 when the USB host is resetting the USB bus.

A valid USB reset resets all the endpoint FIFOs, all other control register bits except Cfg_Lock, all associated configuration registers (0x20 to 0x3F), and bits DS_Chg_IE and DS_Chg. This register (DEVSTAT) forces the device to the default state. This bit is cleared at the end of reset.

This bit is double buffered just as the other DEVSTAT bits are. If there is a pending interrupt that is not handle when a USB reset occurs, and if that interrupt is handled only when USB reset is finished, the local host does not see the USB_Reset bit going high and then low.

0: Device not being reset by USB host

1: Device is being reset by USB host

Value after local host reset is low and during USB reset is high (low after USB reset).

12.9.3 Suspended State (SUS)

A device is, at minimum, attached to the USB and is powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It may also have a unique address and be configured for use. However, because the device is suspended, the host can not use the device function. This bit returns 1 when the USB device is in suspend state.

0: Not suspended

1: Suspended

Value after local host or USB reset is low.

12.9.4 Configured State (CFG)

A device is attached to the USB and powered, has been reset, has a unique address, and is configured. The host can now use the function provided by the device. This bit returns 1 when the USB device has been configured after a set `Dev_Cfg = 1`. This bit remains set to 1 until the device becomes deconfigured.

This bit is cleared when the core receives a valid `SET_CONFIGURATION` request and the local host sets `Clr_Cfg` bit. While this bit is not set to 1, any transaction not for control EP0 is ignored. A `GET_ENDPOINT_STATUS` to a non-control endpoint is stalled.

0: Not configured

1: Configured

Value after local host or USB reset is low.

12.9.5 Addressed State (ADD)

A device is attached to the USB and powered, has been reset, and a unique device address has been assigned. This bit returns 1 after a `SET_ADDRESS` standard request. This bit remains set to 1 until the device becomes de-addressed.

0: Not addressed

1: Addressed

Value after local host or USB reset is low.

12.9.6 Default State (DEF)

This bit returns 1 when the USB device is attached to the USB and powered and has been reset. This bit remains set to 1 until the device becomes depowered. Device moves into default state as soon as the USB reset is effective.

0: Not in default

1: Default

Value after local host is low and after USB reset is high.

12.9.7 Attached State (ATT)

This bit returns 1 when the device is attached to the USB and powered. This bit remains set to 1 until the device becomes depowered.

0: Not attached

1: Attached

Value after local host reset is low (unattached) or high (attached). After USB reset, value is high.

12.10 Start of Frame Register (SOF)

The read-only start of frame register (SOF) provides a frame timer status for use in isochronous communications. A write to this register is forbidden.

Table 80. Start of Frame Register (SOF)

Bits	Field	Description
15–13	–	Reserved
12	FT_Lock	Frame timer locked
11	TS_OK	Time stamp OK
10–0	TS	Time stamp number

12.10.1 Frame Timer Locked (FT_Lock)

The USB host sends out a start of frame (SOF) packet every millisecond. When a SOF packet is not received by the device due to a bus error, a local start-of-frame is generated for use by the isochronous FIFO switch.

Once the core receives two valid SOFs separated by time-frame (TF), it sets the FT_LOCK to 1 only if TF is the frame interval (IF) allowed by USB 1.1 specification ($IF = [11964:12036]$ USB bit time). If TF is out of this interval, the FT_Lock value remains 0, and a local SOF is generated by the core.

When the FT_Lock bit is set and the frame timer is locked to the timing TF, a local SOF is generated if no valid SOF has been received in an interval of TF since the last valid the The FT_Lock bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame-time. If the core does not receive a valid SOF, the frame timer remains lock to TF.

When the FT_Lock is cleared, a local SOF is generated after 12036 USB bit times if no valid SOF has been received, and the FT_Lock remains 0.

0: Frame timer is not locked.

1: Frame timer is locked.

Value after local host or USB reset is low.

Each time a valid SOF is received by the core out of allowed interval IF, a local SOF is generated and isochronous FIFO switch.

12.10.2 Time Stamp OK(TS_OK)

This bit indicates that the time stamp in the TS field is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise.

0: Time stamp is invalid.

1: Time stamp is valid.

Value after local host or USB reset is low.

12.10.3 Time Stamp Number(TS)

This field returns the time stamp from last USB host valid SOF packet. The frame number is valid if the TS_OK is 1. In case of a SOF miss, this value is not updated and TS_OK is cleared.

Value after local host or USB reset is low (all 11 bits).

12.11 Interrupt Enable Register (IRQ_EN)

The read/write interrupt enable register (IRQ_EN) enables all non-DMA interrupts (control, state changed, isochronous, non-isochronous).

Table 81. Interrupt Enable Register (IRQ_EN)

Bits	Field	Description
15–8	–	Reserved
7	SOF_IE	Start-of-frame interrupt enable
6	–	Reserved
5	EPn_RX_IE	Receive endpoint n interrupt enable (non-isochronous)
4	EPn_TX_IE	Transmit endpoint n interrupt enable (non-isochronous)
3	DS_Chg_IE	Device state changed interrupt enable
2–1	–	Reserved
0	EP0_IE	EP0 transactions interrupt enable

When a bit position is set to 1 by the local host, an interrupt is signaled to the local host if the corresponding IRQ_SRC bit is asserted to 1 by the core, for any IRQ_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the local host.

0: Interrupt disabled

1: Interrupt enabled

Value after local host or USB reset is low for all bits except the DS_Chg_IE bit, which remains unchanged after a reset from USB host.

12.12 Interrupt Source Register (IRQ_SRC)

The read/clear-only interrupt source register (IRQ_SRC) has for function to identify and clear the source of the interrupt signaled by a set flag.

Table 82. Interrupt Source Register (IRQ_SRC)

Bits	Field	Description
15–11	–	Reserved
10	TXn_Done	Transmit DMA channel n done interrupt flag (non-isochronous)
9	RXn_Cnt	Receive DMA channel n transactions count interrupt flag (non-isochronous)
8	RXn_EOT	Receive DMA channel n end of transfer interrupt flag (non-isochronous)
7	SOF	Start-of-frame interrupt flag
6	–	Reserved
5	EPn_RX	EPn OUT transactions interrupt flag
4	EPn_TX	EPn IN transactions interrupt flag
3	DS_Chg	Device state changed interrupt flag
2	Setup	Setup transaction interrupt flag
1	EP0_RX	EP0 OUT transactions interrupt flag
0	EP0_TX	EP0 IN transactions interrupt flag

Common to all bits:

The local host can only clear a set bit location by writing a 1 into the bit location (except for Setup bit, which is automatically cleared by the core). A write of 0 has no effect.

When a bit location is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.

0: No interrupt

1: Interrupt signaled

Value after local host or USB reset is low, except for the DS_Chg bit, which is high after a USB reset.

12.12.1 Transmit DMA CH.n Done Interrupt Flag (TXn_Done)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXn_TSC (transfer size counter) equals 0 and the last IN transaction completes with an ACK. When this bit is asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed.

The endpoint interrupt the EPn_TX is never set for the assigned endpoint to TX DMA channel n.

0: No action

1: Non-isochronous transmit DMA transfer for a channel has ended.

Value after local host or USB reset is low.

12.12.2 RX DMA CH.n Transactions Count Interrupt Flag (RXn_Cnt)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core during an active receive DMA transfer each time RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after RX DMA data has been read (end of DMA request). When this bit is asserted, the local host must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_Cnt interrupt is asserted also if the RXn_Stop bit is set; in this case, both the RXn_EOT and the RXn_Cnt are asserted.

0: No action

1: Non-isochronous receive DMA transfer for a channel has reached transactions count level.

Value after local host or USB reset is low.

12.12.3 Receive DMA CH.n EOT Interrupt Flag (RXn_EOT)

Only for non-isochronous DMA transfer. This bit is never set for isochronous DMA transfer.

This bit is set automatically by the core when a receive DMA channel has detected an end of transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after RX DMA data has been read (end of DMA request). When this happens, the DMA-assigned endpoint FIFO is kept disabled (the FIFO_En = 0) to avoid receiving a new packet data from the USB host. The local host can grant another DMA transfer to the same endpoint by simply enabling the FIFO again (the FIFO_En = 1).

An end of transfer is detected when the core receives a data packet whose size is less than the configured endpoint FIFO size (or empty) or when RXn_TC equals 0 after an OUT transaction with ACK status and the RXn_Stop bit is set.

When this bit is asserted, the local host must read the DMA_n_RX_IT_src to identify the endpoint number for which the transfer completed and must read the DMA_n_RX_SB to be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from DATA_DMA register).

The endpoint interrupt EP_n_RX bit is never set for the assigned endpoint to RX DMA channel.

0: No action

1: Non-isochronous receive DMA transfer for a channel has ended.

Value after local host or USB reset is low.

12.12.4 Start Of Frame Interrupt Flag (SOF)

Every millisecond, the USB host outputs a start of frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 to the SOF bit location clears the flag. Writing a 0 has no effect.

In accordance with the USB1.1 spec, if an SOF is not received or is corrupted, the core still sets this flag at the same rate (if bit FT_Lock = 1) or after 12043 USB bit times (if bit FT_lock = 0).

0: No action

1: Start of frame packet received (or internal SOF)

Value after local host or USB reset is low.

12.12.5 OUT Transaction Endpoint n Interrupt Flag (EPn_RX)

Only concerns non-isochronous endpoints.

This bit is automatically set by the core when a handshake sequence occurs for an OUT transaction to an interrupt of bulk endpoint (NAK with the Nak_En bit set, ACK, or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt.

0: No action

1: OUT transaction detected on an endpoint.

Value after local host or USB reset is low.

12.12.6 IN Transaction Endpoint n Interrupt Flag (EPn_TX)

Only concerns non-isochronous endpoints.

This bit is automatically set by the core when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with the Nak_En bit set, ACK or STALL). The local host must read EPN_STAT register to identify the endpoint causing the interrupt.

0: No action

1: IN transaction detected on an endpoint.

Value after local host or USB reset is low.

12.12.7 Device State Changed Interrupt Flag (DS_Chg)

This bit is automatically set by the core when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position.

0: No action

1: Device state change detected

Value after local host reset is low and after USB reset is high.

12.12.8 Setup Transaction Interrupt Flag (Setup)

This bit is automatically set by the core when a valid setup transaction completes one control endpoint for a non-autodecoded control request and is cleared automatically by the core when the local host sets the Setup_Sel bit when reading setup data. A write of 1 to it has no effect.

0: No action

1: Valid setup transaction occurred on endpoint 0.

Value after local host or USB reset is low.

12.12.9 OUT Transaction Endpoint 0 Interrupt Flag (EP0_RX)

This bit is set automatically by the core when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with the Nak_En bit set, ACK, or STALL).

0: No action

1: OUT transaction on EP0

Value after local host or USB reset is low.

12.12.10 IRQ_SRC[0].EP0_TX: IN Transaction Endpoint 0 Interrupt Flag

This bit is set automatically by the core when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with the Nak_En bit set, ACK, or STALL).

0: No action

1: IN transaction on EP0

Value after local host or USB reset is low.

12.13 Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)

The read-only non-isochronous endpoint interrupt status register (ENP_STAT) identifies the non-isochronous endpoint causing an EPn interrupt. A write into it is forbidden.

If a non-transparent transaction occurs before a previous one on another endpoint in the same direction has been handled by the local host, the second interrupt is asserted only after first one has been cleared by the local host and EPN_STAT is updated with the corresponding interrupt assertion.

Table 83. Non-Isochronous Endpoint Interrupt Status Register (EPN_STAT)

Bits	Field	Description
15–12	–	Reserved
11–8	EPn_RX_IT_src	Receive endpoint interrupt source (non-isochronous)
7–4	–	Reserved
3–0	EPn_TX_IT_src	Transmit endpoint interrupt source (non-isochronous)

12.13.1 Receive Endpoint Interrupt Source (EPn_RX_IT_src)

Only concerns non-isochronous endpoints. When the EPn_RX flag bit is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the EPn_RX flag bit is cleared, the four bits read as 0.

0000: No receive endpoint interrupt is pending.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.13.2 Transmit Endpoint Interrupt Source (EPn_TX_IT_src)

Only concerns non-isochronous endpoints.

When the EPn_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EPn_TX flag is cleared, the four bits read as 0.

0000: No transmit endpoint interrupt is pending.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.14 Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)

The read-only non-isochronous DMA interrupt status register (DMAN_STAT) identifies the endpoint causing a DMA interrupt. A write into it is forbidden.

If a DMA interrupt occurs before a previous one on another endpoint in the same direction has been handled by the local host, the second interrupt is asserted only after first one has been cleared by the local host and DMAN_STAT is updated when the corresponding interrupt is asserted.

Table 84. Non-Isochronous DMA Interrupt Status Register (DMAN_STAT)

Bits	Field	Description
15–11	–	Reserved
12	DMAn_RX_SB	DMA receive single byte (non-isochronous)
11–8	DMAn_RX_IT_src	DMA receive interrupt source (non-isochronous)
7–4	–	Reserved
3–0	DMAn_TX_IT_src	DMA transmit interrupt source (non-isochronous)

12.14.1 DMA Receive Single Byte (DMAn_RX_SB)

Only concerns non-isochronous endpoints (isochronous endpoints receive a constant number of bytes).

This bit is set when the RXn_EOT interrupt is asserted and the core receives an odd number of bytes during the last transaction. This bit determines the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the RXn_EOT flag is cleared, this bit read as 0.

0: No EOT DMA interrupt is pending, or core received an even number of bytes during last transaction.

1: An EOT DMA interrupt is pending, and an odd number of bytes was received during last transaction.

Value after local host or USB reset is low.

12.14.2 DMA Receive Interrupt Source (DMA_n_RX_IT_src)

Only concerns non-isochronous endpoints.

When the EP_n_RX flag bit is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EP_n_RX flag bit is cleared, the four bits read as 0.

0000: No receive DMA interrupt is pending.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.14.3 DMA Transmit Interrupt Source (DMA_n_TX_IT_src)

Only concerns non-isochronous endpoints.

When the EP_n_TX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the EP_n_TX flag is cleared, the four bits read as 0.

0000: No transmit DMA interrupt is pending.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.15 Receive DMA Channels Configuration Register (RXDMA_CFG)

The read/write receive DMA channels configuration register (RXDMA_CFG) enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured through register EPn_RX. The RXDMA_CFG register can be filled when the Cfg_Lock bit is set.

There is no hardware mechanism to protect against setting invalid endpoints.

Table 85. Receive DMA Channels Configuration Register (RXDMA_CFG)

Bits	Field	Description
15–12	–	Reserved
11–8	RXDMA2_EP	Receive endpoint number for DMA channel 2
7–4	RXDMA1_EP	Receive endpoint number for DMA channel 1
3–0	RXDMA0_EP	Receive endpoint number for DMA channel 0

12.15.1 Receive Endpoint Number for DMA Channel 2 (RXDMA2_EP)

The endpoint number binary-encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.

0000: Receive DMA channel 2 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.15.2 Receive Endpoint Number for DMA Channel 1 (RXDMA1_EP)

The endpoint number binary-encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.

0000: Receive DMA channel 1 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.15.3 Receive Endpoint Number for DMA Channel 0 (RXDMA0_EP)

The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.

0000: Receive DMA channel 0 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.16 Transmit DMA Channels Configuration Register (TXDMA_CFG)

The read/write transmit DMA channels configuration register (TXDMA_CFG) enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by a TX DMA channel must have been configured through register EPn_TX. TXDMA_CFG register can be filled when the Cfg_Lock bit is set.

There is no hardware mechanism to protect against setting invalid endpoints.

Table 86. Transmit DMA Channels Configuration Register (TXDMA_CFG)

Bits	Field	Description
15–12	–	Reserved
11–8	TXDMA2_EP	Transmit endpoint number for DMA channel 2
7–4	TXDMA1_EP	Transmit endpoint number for DMA channel 1
3–0	TXDMA0_EP	Transmit endpoint number for DMA channel 0

12.16.1 Transmit Endpoint Number for DMA Channel 2 (TXDMA2_EP)

The endpoint number binary-encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.

0000: Transmit DMA channel 2 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.16.2 Transmit Endpoint Number for DMA Channel 1 (TXDMA1_EP)

The endpoint number binary-encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.

0000: Transmit DMA channel 1 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.16.3 Transmit Endpoint Number for DMA Channel 0 (TXDMA0_EP)

The endpoint number binary-encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.

0000: Transmit DMA channel 0 is deactivated.

0001: EP1

....

1111: EP15

Value after local host or USB reset is low (all 4 bits).

12.17 DMA FIFO Data Register (DATA_DMA)

The DMA FIFO data register (DATA_DMA) is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1, or 2.

Table 87. DMA FIFO Data Register (DATA_DMA)

Bits	Field	Description
15–0	DATA_DMA	DMA FIFO data

12.17.1 DMA FIFO Data(DATA_DMA)

When an RX DMA request is active for a channel (only one active at a given time), this register contains the data received by the core from USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel.

When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction.

It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active.

The local host must not access this register directly; however, there is no hardware mechanism to protect from such access. Do not attempt access into this register during DMA request handling.

12.18 Transmit DMA Control Registers (TXDMA0...TXDMA2)

The read/write transmit DMA control registers (TXDMA...TXDMA2) control the operation of the transmit DMA channel n ($n = 0, 1, 2$).

Table 88. Transmit DMA Control Registers (TXDMA...TXDMA2)

Bits	Field	Description
15	TXn_EOT	Transmit DMA channel n end of transfer
14	TXn_Start	Transmit DMA channel n start
13–10	–	Reserved
9:0	TXn_TSC	Transmit DMA channel n transfer size counter

12.18.1 Transmit DMA Ch.n End of Transfer (TXn_EOT)

This bit can be either 0 or 1 for BULK DMA transfer.

When set to 1 by the local host, it signals to the core that the transfer size set in TXn_TSC is in bytes. A TX done interrupt (the TXn_Done) is asserted with the last IN transaction. If the number of bytes set in TXn_TSC is a multiple of the endpoint buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet.

When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes.

0: DMA transfer size is in buffers.

1: DMA transfer size is in bytes.

Value after local host or USB reset is low.

12.18.2 Transmit DMA Ch.n Start (TXn_Start)

Set by the local host to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, except if the local host clears endpoint in TXDMA_CFG register (see part 8.8). Writing 0 to this bit has no effect and a read of this bit always returns 0. The TXn_Done interrupt bit is asserted when the DMA transfer ends.

0: No action

1: DMA transfer start

Always reads 0.

12.18.3 Transmit DMA Ch.n Transfer Size Counter (TXn_TSC)

The binary-encoded value from 0 to 1023, which is written by the local host into this register, corresponds to the number of bytes or number of buffer transfers (function of TXn_EOT), which is transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has left to transmit. This read mode is only provided for software debug purposes.

For isochronous transfer, the user must verify that the set value does not exceed the isochronous FIFO size for the endpoint. There is no hardware mechanism to protect from this situation. If it happens, results are unpredictable.

For bulk transfer, when TXn_EOT = 0 a set value of TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, ...001, 000, stop. When TXn_EOT = 1, a set value of TXn_TSC = 0 a NULL packet is sent in response to the next IN token.

Value after local host or USB reset is low (all 10 bits).

12.19 Receive DMA Control Registers (RXDMA...RXDMA2)

These read/write receive DMA control registers enable monitoring of incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Table 89. Receive DMA Control Registers (RXDMA0...RXDMA2)

Bits	Field	Description
15	RXn_Stop	Receive DMA channel n transfer stop
14–8	–	Reserved
7–0	RXn_TC	Receive DMA channel n transactions count

12.19.1 Receive DMA Ch.n Transfer Stop (RXn_Stop)

When this bit is set, an RXn_EOT interrupt is asserted to the local host after n OUT transactions where n is the encoded binary value + 1 programmed into RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT) and the local host expects a given amount of data for the transfer.

At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The local host must set the Set_FIFO_En for the endpoint to reenables the channel.

Value after local host or USB reset is low.

12.19.2 Receive DMA Ch.n Transactions Count (RXn_TC)

The local host can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXn_TC field. This register must be programmed to the desired transactions watermark limit prior to enabling the DMA transfer for the receive DMA channel n.

A reached watermark does not disable an active DMA transfer if RXn_Stop was not set. If RXn_Stop was set for the transfer, both RXn_Cnt and RXn_EOT interrupts are asserted.

A read of this register returns the number of transactions remaining before the RXn_Cnt interrupt flag is asserted. This read mode is only provided for software debug purposes.

Value after local host or USB reset is low (all 8 bits).

12.20 Endpoint 0 Configuration Register (EP0)

The read/write endpoint 0 configuration register (EP0) gives the device configuration for control endpoint 0.

Table 90. Endpoint 0 Configuration Register (EP0)

Bits	Field	Description
15–14	–	Reserved
13–12	EP0_Size	Endpoint 0 FIFO size
11	–	Reserved
10–0	EP0_ptr	Endpoint 0 pointer

12.20.1 Endpoint 0 FIFO Size (EP0_Size)

This field contains the endpoint 0 FIFO size value and must match the value sent by the local host to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (the Non_ISO_FIFO_Empty, the Non_ISO_FIFO_Full) and overrun/underrun conditions are based on this value for all IN and OUT transactions to endpoint 0.

The local host must fill this field before setting the Cfg_Lock bit.

00: 8 bytes

01: 16 bytes

10: 32 bytes

11: 64 bytes

Value after local host reset is low (both bits), after USB reset is unchanged.

12.20.2 Endpoint 0 Pointer (EP0_ptr)

This field contains the address of the endpoint 0 pointer. Value 0x000 is forbidden (reserved for setup FIFO).

0x000: address = BASE (forbidden)

0x001: address = BASE + 8 bytes

0x002: address = BASE + 16 bytes

0x003: address = BASE + 24 bytes

...

0x0FF: address = BASE + 2040 bytes

Value after local host reset is low (all bits), after USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 14, *Device Initialization*).

12.21 Receive Endpoint Configuration Registers (EP1_RX...EP15_RX)

The read/write receive endpoint configuration registers (EP1_RX...EP15_RX) give the device configuration for non-control receive endpoint n (n: 115). The endpoints size fields must match values sent by the local host to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The local host must fill this field before setting the Cfg_Lock bit and must not change the values once Cfg_Lock bit is set.

Table 91. Receive Endpoint n Configuration Registers (EP1_RX...EP15_RX)

Bits	Field	Description
15	EPn_RX_Valid	Receive endpoint n valid
14	EPn_RX_Size/Db	Receive non-isochronous endpoint n double-buffer (Db) Or receive isochronous endpoint n size[2]
13–12	EPn_RX_Size	Receive endpoint n size
11	EPn_RX_Iso	Receive isochronous endpoint n
10–0	EPn_RX_ptr	Receive endpoint n pointer

12.21.1 Receive Endpoint n Valid (EPn_RX_Valid)

This bit must be set by the local host to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.

1: Receive endpoint n is part of the device configuration.

0: Receive endpoint n does not exist for this configuration.

The value after the local host reset is low, after USB reset is unchanged.

12.21.2 Receive Endpoint n Double-Buffer (EPn_RX_Db)

This bit is only for non-isochronous endpoints. For isochronous endpoints, which are always double-buffered, this bit is endpoint size MSB.

This bit must be set by the local host to allow double buffering for receive non-isochronous endpoint n. This is used to reduce number of transactions resulting in NAK handshake.

1: Double buffer used for non-isochronous receive endpoint n.

0: No double buffer for non-isochronous receive endpoint n.

Value after local host reset or USB reset is unchanged.

12.21.3 Receive Endpoint n Size (EPn_RX_Size)

This paragraph includes description of EPn_RX.[14] bit for isochronous endpoints.

This field contains the endpoint n FIFO size value. Status flags (the Non_ISO_FIFO_Empty, the Non_ISO_FIFO_Full, the ISO_FIFO_Empty, the ISO_FIFO_Full) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n (see Table 92).

Table 92. Endpoint n Size Values

Non-Isochronous [13:12]	Isochronous [14:12]
00: 8 bytes	000: 8 bytes
01: 16 bytes	001: 16 bytes
10: 32 bytes	010: 32 bytes
11: 64 bytes	011: 64 bytes
	100: 128 bytes
	101: 256 bytes
	110: 512 bytes
	Reserved

Value after local host reset or USB reset is unchanged.

12.21.4 Receive Isochronous Endpoint n(EPn_RX_Iso)

This field must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt).

0: Receive endpoint n type is isochronous.

1: Receive endpoint n type is bulk or interrupt.

Value after local host reset or USB reset is unchanged.

12.21.5 Receive Endpoint n Pointer (EPn_RX_ptr)

This field contains the address of the receive endpoint n pointer. Value 0x000 is forbidden (reserved for setup FIFO).

CAUTION
For isochronous endpoints or for non-isochronous endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong.

0x000: address = BASE (forbidden)

0x001: address = BASE + 8 bytes

0x002: address = BASE + 16 bytes

0x003: address = BASE + 24 bytes

...

0x0FF: address = BASE + 2040 bytes

Value after local host reset or USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 14, *Device Initialization*).

12.22 Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)

The read/write transmit endpoint configuration registers (EP1_TX...EP15_TX) configure the device for noncontrol transmit endpoint n (n: 115). The endpoint size fields must match the values sent by the local host to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The local host must fill this field before setting the Cfg_Lock bit and must not change the values once the Cfg_Lock bit is set.

Table 93. Transmit Endpoint Configuration Registers (EP1_TX...EP15_TX)

Bits	Field	Description
15	EPn_TX_Valid	Transmit endpoint n valid
14	EPn_TX_Size/Db	Transmit non-isochronous endpoint n double-buffer or transmit isochronous endpoint n size[2]
13–12	EPn_TX_Size	Transmit endpoint n size
11	EPn_TX_Iso	Transmit isochronous endpoint n
10–0	EPn_TX_ptr	Transmit endpoint n pointer

12.22.1 EPn_TX[15].EPn_TX_Valid: Transmit Endpoint n Valid

This bit must be set by the local host to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.

1: Transmit endpoint n is part of the device configuration.

0: Transmit endpoint n does not exist for this configuration.

Value after local host reset is low, after USB reset is unchanged.

12.22.2 Transmit Endpoint n Double-Buffer(EPn_TX_Db)

This bit is only for non-isochronous endpoints used in DMA mode. For isochronous endpoints, which are always double buffered, this bit is the endpoint size MSB. For non-isochronous endpoints which are not used in a DMA transfer, double-buffering is not provided.

This bit must be set by the local host to allow double buffering for transmit non-isochronous endpoint n, when used in a DMA transfer. This is used to reduce number of transactions resulting in NAK handshake.

1: Double buffer used for non-isochronous transmit endpoint n.

0: No double buffer for non-isochronous transmit endpoint n.

Value after local host or USB reset is unchanged.

12.22.3 Transmit Endpoint n Size (EPn_TX_Size)

EPn_TX.[14] bit description only applies for isochronous endpoints.

This field contains the endpoint n FIFO size value. Status flags (FIFO_Empty, FIFO_Full) and underrun condition are based on this value for all IN transactions to endpoint n (see Table 92, *Endpoint n Size Values*).

Value after local host reset or USB reset is unchanged.

12.22.4 Transmit Isochronous Endpoint n (EPn_TX_Iso)

This field must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt).

0: Transmit endpoint n type is isochronous.

1: Transmit endpoint n type is bulk or interrupt.

Value after local host or USB reset is unchanged.

12.22.5 Transmit Endpoint n Pointer (EPn_TX_ptr)

This field contains the address of the transmit endpoint n pointer.

CAUTION
For isochronous endpoints or for non-isochronous endpoints that allow double-buffering, 2*TX buffer size must be reserved for ping-pong.

0x000: address = BASE

0x001: address = BASE + 8 bytes

0x002: address = BASE + 16 bytes

0x003: address = BASE + 24 bytes

...

0x0FF: address = BASE + 2040 bytes

Value after local host reset or USB reset is unchanged.

Set the pointer value higher than 0xFF, because the memory size is 2K bytes. A pointer value equal to 0xFF corresponds to 2040 bytes: addressing upper bytes results in memory overlap (see Section 14, *Device Initialization*).

13 USB Transactions

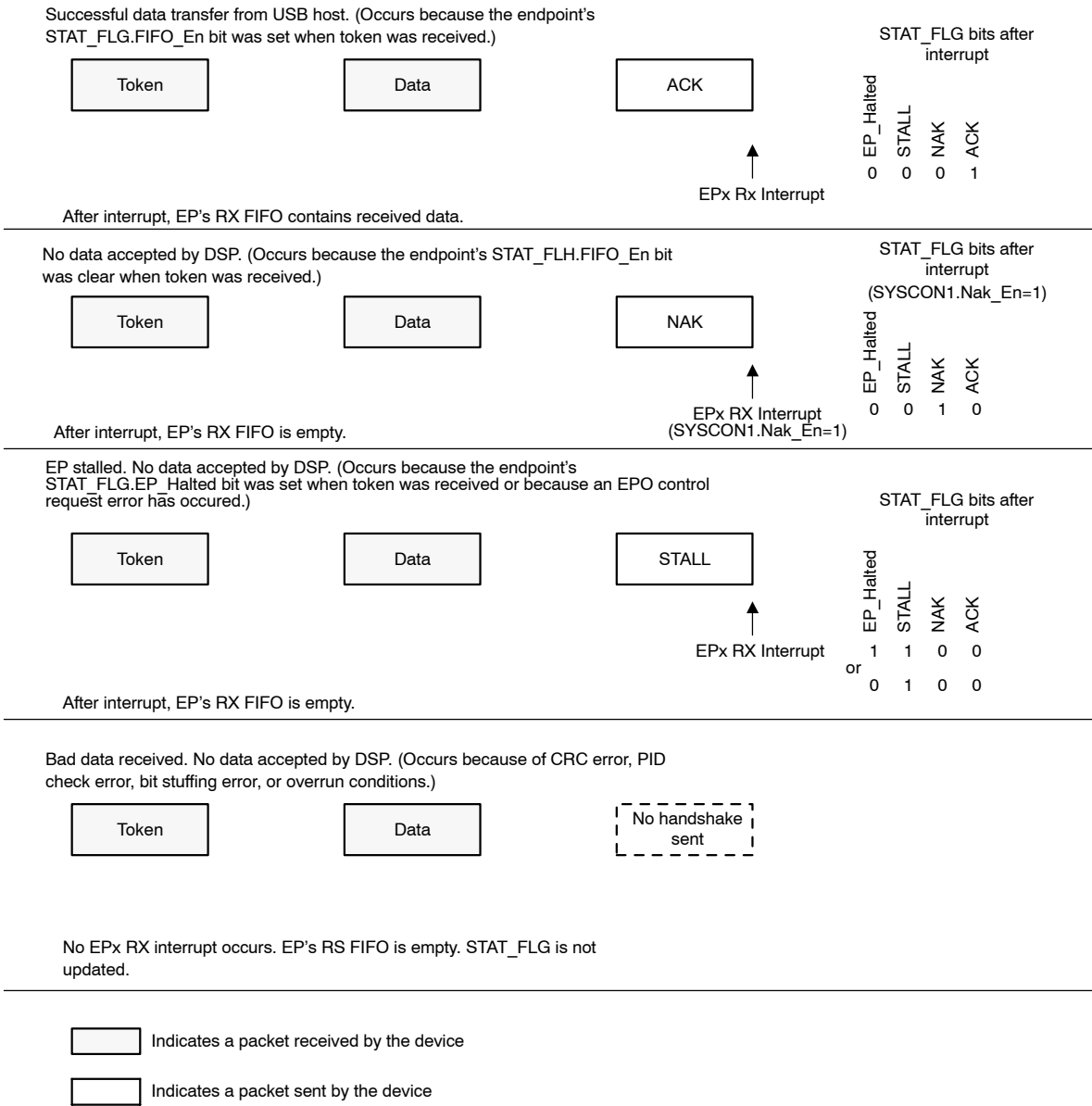
There is an interrupt to the local host at the end of a USB transaction if the local host has actions to perform. Isochronous transactions are an exception, because isochronous interrupt information is available at start of frame interrupts. The local host ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following sections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the local host are not taken into account in this section. The information is organized so that each section deals with one type and direction of transaction, such as non-isochronous, non-setup OUT transactions, non-isochronous IN transactions, isochronous OUT transactions, isochronous IN transactions, etc. This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

13.1 Non-Isochronous, Non-Setup OUT (USB HOST → LH) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the local host and where the USB handshaking protocols are in effect and data transmission is guaranteed. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

Figure 36 shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: ACK (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case shows an abnormal case where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first, ACK, case.

Figure 36. Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions



13.1.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

The Set_FIFO_En bit provides the main control for the ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint the FIFO_En bit is 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO and, when the transaction

completes, the USB module can return ACK to the USB host to indicate that the data was received correctly (this is the top case shown in Figure 36). If, however, the FIFO_En bit was 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (the second case shown in Figure 36).

It is important to note that the USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The local host code must be careful not to read too much data. Local host code must read the RXF_Count value before reading data from the RX FIFO.

After a USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint's FIFO_En bit. Once the local host software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the Set_FIFO_En bit. Local host software can use the Set_FIFO_En bit as a receive flow control mechanism.

Acknowledged Transactions (ACK)

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the local host and the STAT_FLG is updated. In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1, then read the endpoint status. The ACK bit is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If the FIFO_Empty is cleared, the host sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO) and the data is in the endpoint RX FIFO. The local host knows the number of bytes to read from RX FIFO by reading the RXF_Count value. The local host can then read RX data from DATA register. Once the local host has read the data from the FIFO, it sets the Set_FIFO_En bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO and then clears the EP_Sel bit. This clears the ACK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

The device can be configured via the Nak_En bit, either to inform the local host of a NAKed transaction or not. If the NAK_EN bit is cleared, no interrupt is asserted to the local host if an OUT transaction completes with a NAK handshake and the NAK bit not set. If the Nak_En bit is set, the USB module issues an endpoint-specific interrupt to the local host at completion of an OUT transaction to an endpoint and the NAK bit is set. In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1 then read the endpoint status. The NAK bit is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The local host must set the Set_FIFO_En bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO and then clear the EP_Sel bit. This clears the NAK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

13.1.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions

STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint either if the endpoint's EP_Halted bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint's STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. The FIFO_En bit is of lower priority than the EP_Halted; when the EP_Halted bit is set, transactions to the RX endpoint are stalled, regardless of the FIFO_En value. If the FIFO_En bit is set, the FIFO_En bit is automatically cleared at the end of the STALLed transaction, and RX FIFO is cleared.

In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number and EP_Sel to 1, then read the endpoint status. The STALL bit is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If the EP_Halted has been set by the local host and can be removed, the local host must set the Clr_Halt to clear the condition and set the Set_FIFO_En to allow the next USB OUT transaction to the endpoint to be placed into the RX

FIFO. If the EP_Halted has been set in response to a SET_FEATURE request sent by the USB host or if the bit is cleared (control transaction only), the local host has no action to perform and must clear the EP_Sel bit. This clears the STALL bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Packet Errors

In case of a receive data error during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the local host (the fourth case shown in Figure 36). Additionally, the endpoint RX FIFO is not filled and the FIFO_En bit is not cleared. If the local host clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

Sequence Bit Errors

If the core does not receive expected DATA PID during an OUT transaction, the module automatically returns an ACK handshake to the USB host, regardless of the FIFO_En bit (per USB spec). Data is ignored, and no interrupt is asserted to the local host.

This error occurs if an ACK handshake from the previous OUT transaction is received corrupted by the USB host.

13.1.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the local host. Additionally, the endpoint RX FIFO is not filled, and the FIFO_En bit is not cleared.

The local host must not read more data from the RX FIFO than the amount indicated by RXF_Count.

13.2 Non-Isochronous IN (LH->USB HOST) Transactions

Non-isochronous IN transactions refer to USB transactions where data is moved from the local host to the USB host where the USB handshaking protocols are in effect and data transmission is guaranteed. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not guarantee USB bandwidth.

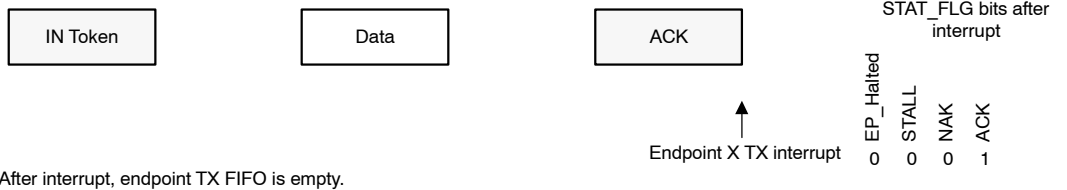
To provide data for an endpoint IN transaction, the local host code writes the transmit data into the endpoint transmit FIFO. Local host code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint's FIFO_En bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and the FIFO_En bit is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint's previous transmit activity is taken care of, the local host code gains access to endpoint's FIFO and status by setting EP_Sel bit. Then the local host can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data has been written to the endpoint FIFO, local host code sets the Set_FIFO_En bit to allow the USB to use the endpoint's TX FIFO and then clears the EP_Sel bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

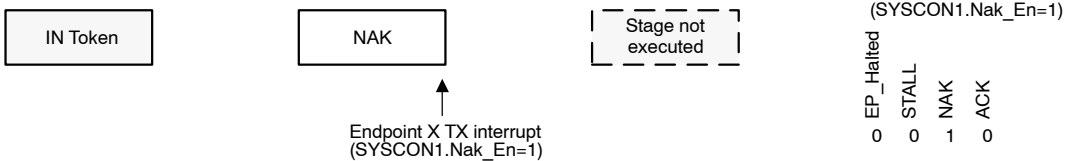
Figure 37 shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: ACK (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case where there is an error either in the token packet received by the core, or in the data packet received by the USB host.

Figure 37. Non-Isochronous IN Transaction Phases and Interrupts

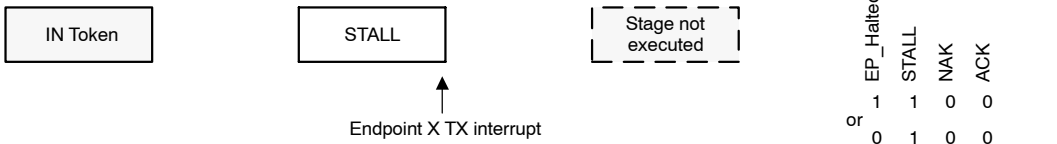
Successful data transfer to USB host (endpoint STAT_FLG.FIFO_En bit was set when token was received).



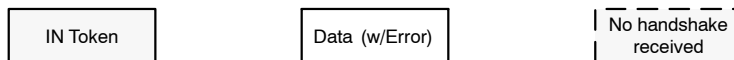
No data transmitted by LH (endpoint STAT_FLG>FIFO_En bit was clear when token was received).



Endpoint stalled. No data transmitted by LH (endpoint STAT_FLG>EP_Halted bit was set when token was received or an EP0 control request error has occurred).




Endpoint TX data error during transmission.



Endpoint TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT_FLG is unchanged.

 Indicates a packet received by the device

 Indicates a packet sent by the device

13.2.1 Non-Isochronous IN Endpoint Handshaking

Per the USB spec for IN transactions, the USB host may only provide one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates successful transfer (first case shown in

Figure 37), and the second indicates that the host received a garbled data packet (last case shown in Figure 37).

Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the local host (see first case in Figure 37). In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The ACK bit is set to indicate that the endpoint received an ACK handshake from the USB host and that the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the local host has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear the EP_Sel bit. This clears the ACK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register.

Non-must Transactions (NAK)

For the case where the local host is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint (as shown in the second case in Figure 37). Readiness to transmit data is signaled via the endpoint's FIFO_En bit; when 1, it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint's FIFO_En bit is 0 and an IN transaction to the endpoint occurs, a NAK handshake is sent, indicating that the local host is not ready to handle the request.

If the Nak_En bit is cleared when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the local host is generated. If the Nak_En bit is set when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the NAK bit is set and an endpoint-specific interrupt to the local host is generated.

In response to the endpoint interrupt, the local host must read the EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The NAK bit is set to indicate that the endpoint sent a NAK handshake to the USB host. If the local host has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The local host must then clear the EP_Sel bit. This clears the NAK bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register. Signaling NAK does not cause the endpoint's TX FIFO to be cleared (since the local host still retains control of the FIFO).

Signaling NAK handshakes for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK may not be a good mechanism in cases where the local host is not able to service a request for long periods of time.

13.2.2 Non-Isochronous IN Transaction Error Conditions

STALLED Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint's EP_Halted flag bit is set (as shown in the third case in Figure 37) or if a request error occurs (control transaction only). A STALL handshake indicates that the device endpoint is in a condition where it is not able to transfer data and that the USB host must not retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, typically a control transfer via endpoint 0. The local host can set the endpoint EP_Halted bit by writing the appropriate value in the EP_NUM register to select it. It can then set the endpoint's Set_Halt bit and clear it by selecting the endpoint and setting the endpoint's Clr_Halt bit. When the endpoint EP_Halt bit is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STALL bit is set, and an endpoint-specific interrupt to the local host is generated.

In response to the endpoint interrupt, the local host must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The local host must set EP_Num to the endpoint number, EP_Dir to 1 (to signal an IN endpoint), and EP_Sel to 1, then read the endpoint status. The STALL bit is set to indicate that the endpoint sent a STALL handshake to the USB host. The local host must then clear the EP_Sel bit. This clears the STALL bit for this endpoint and allows the next transaction status to be written to the STAT_FLG register

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; so for a given endpoint number, the TX can be halted when the RX is not.

Packet Errors

If an error (CRC, bit stuffing or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the local host occurs for transactions with corrupted packets. If the local host clears the TX FIFO during the data packet of an IN transaction, a bit stuffing error is forced.

If the USB host returns no handshake after an IN transaction (case of error during transmission), the USB function module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO, and can be resent during next IN transaction, the FIFO_En bit is not cleared, and no interrupt is asserted to the local host.

13.2.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The local host cannot write more data into the TX FIFO than the configured FIFO size.

13.3 Isochronous OUT (USB HOST→ LH) Transactions

Isochronous OUT transactions are USB transactions where a given amount of data is transferred from the USB host to the USB function module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the local host is generated at completion of an isochronous OUT transaction. The local host is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

At every SOF interrupt, the local host code must select the endpoint for each isochronous OUT endpoint by writing the appropriate value into the EP_NUM register and checking the ISO_FIFO_Empty bit. If the RX FIFO contains data, code must read the RXF_Count value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the DATA register, then clear the EP_Sel bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the 1-ms USB frame, the USB interface implements double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the local host is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, while the foreground FIFOs are enabled to the local host.

Figure 38 shows the two phases (isochronous OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the local host is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the local host at each start of frame (SOF) interrupt, which is shown as the second case in Figure 37.

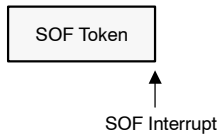
Figure 38. Isochronous OUT Transaction Phases and Interrupts

Successful data transfer from USB host




No handshake occurs. Endpoint RX FIFO contains receive data after data packet completes. No interrupt occurs.

Reception of SOF causes SOF interrupt.
An SOF interrupt is generated even if the SOF packet is corrupted.



LH code for SOF interrupt service routine must fill all isochronous IN endpoint TX FIFOs with new transmit data and pull new receive data from all isochronous OUT endpoint RX FIFOs.

 Indicates a packet received by the device

 Indicates a packet sent by the device

13.3.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL, the NAK, and the ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

13.3.2 Isochronous OUT Transaction Error Conditions

If the local host fails to read all of the data in the isochronous OUT endpoint foreground FIFO by the time that the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed and the Data_Flush bit is asserted for the duration of the next frame.

There is no special indication for the case where the USB host does not provide a transaction to an isochronous OUT endpoint during a frame; but once the FIFO that was background in that frame is foreground, the FIFO is empty. A 0-length data isochronous OUT transaction also results in an empty FIFO and can not be distinguished from a missed isochronous OUT transaction.

If an isochronous OUT transaction occurs with data error (CRC, PID check, bit stuffing), the RX FIFO is empty at the next SOF interrupt and the ISO_Err bit is asserted for the duration of the next frame.

13.3.3 Isochronous OUT Endpoint FIFO Error Conditions

The local host must never read more data than value given by the RXF_Count.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the ISO_Err bit is set at the next SOF interrupt. A properly configured USB system does not do this.

Both foreground and background isochronous FIFOs are cleared when the Clr_EP bit is set.

13.4 Isochronous IN (LH->USB HOST) Transactions

Isochronous IN transactions are USB transactions where a given amount of data is transferred from the USB function module device to the USB host every 1-ms USB frame. No handshaking is provided.

The USB module provides double buffering of data for isochronous IN endpoints; the background FIFO is used as the source of data for IN transactions to the isochronous endpoint, and the foreground FIFO can be written to by the local host. When an IN transaction to an isochronous endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The local host is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

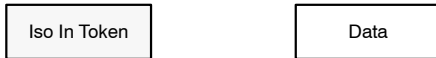
In response to the SOF interrupt, for each isochronous IN endpoint, local host code selects the endpoint (via EP_NUM register), then fills the endpoint TX FIFO (via DATA register). Once all the transmit data has been written to the FIFO, the local host code must clear the EP_Sel bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the 1-ms USB frame, the USB interface implements a double buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the local host is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each start of frame (SOF). Because isochronous endpoints implement double buffering, isochronous endpoints do not control access to the FIFOs via the Set_FIFO_En bit; the Set_FIFO_En and the FIFO_En bits are not implemented for isochronous IN endpoints.

Figure 39 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the local host is generated as a result of an isochronous IN transaction. There is no handshake phase. The SOF transaction causes an SOF interrupt to the local host; it is assumed that the local host refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

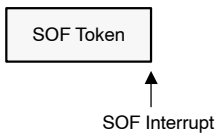
Figure 39. Isochronous IN Transaction Phases and Interrupts

Successful data transfer to PC host

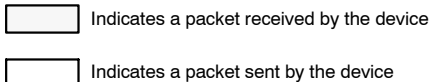


No handshake occurs. Endpoint TX FIFO is empty after data sent. No endpoint interrupt occurs. STAT_FLG is unchanged.

Reception of SOF causes SOF interrupt.
An SOF interrupt is generated even if the SOF packet is corrupted.



LH code for SOF ISR must fill all isochronous IN endpoint TX FIFOs with new transmit data and pull new receive data from all isochronous OUT endpoint RX FIFOs.



13.4.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STALL, the NAK, and the ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the local host to report handshake results for isochronous endpoints.

13.4.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an isochronous IN transaction in the previous frame and if data were present in the TX FIFO to be sent at the IN transaction, the Miss_In bit is asserted for the duration of the following frame. If the isochronous IN endpoint is cleared in the middle of a USB transaction to the background FIFO, a bit-stuffing error is forced for the isochronous transaction.

13.4.3 Isochronous IN Endpoint FIFO Error Conditions

If the local host attempts to overfill the configured endpoint FIFO, data written to the DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background

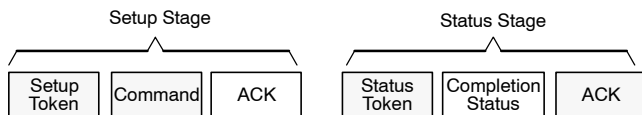
FIFO and an IN transaction for that endpoint occurs. Since an isochronous TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the local host does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

13.5 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB function module is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 40 and Figure 41. If an IN or OUT transaction is received in a control request, this transaction is automatically stalled by the core.

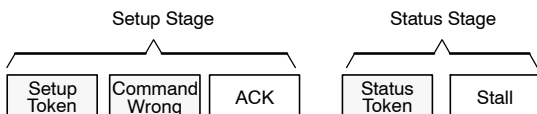
Figure 40. Stages and Transaction Phases of Autodecoded Control Transfers

Autodecode control write transfer--correct status
(set address, clear/set device/interface feature).



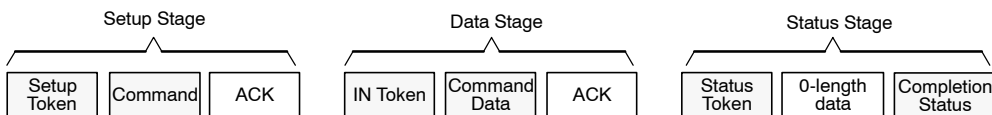
Interrupt occurs/status flags are not updated.

Autodecode control read transfers--request error
(due to wrong setup data).



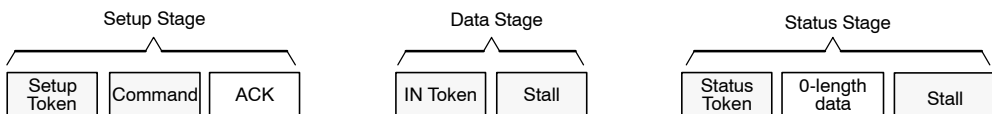
Interrupt occurs/status flags are not updated.

Autodecode control read transfers--correct status
(get device/endpoint status)



Interrupt occurs/status flags are not updated.

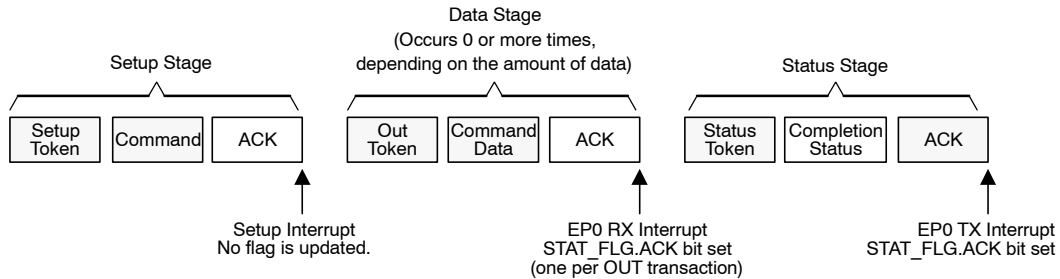
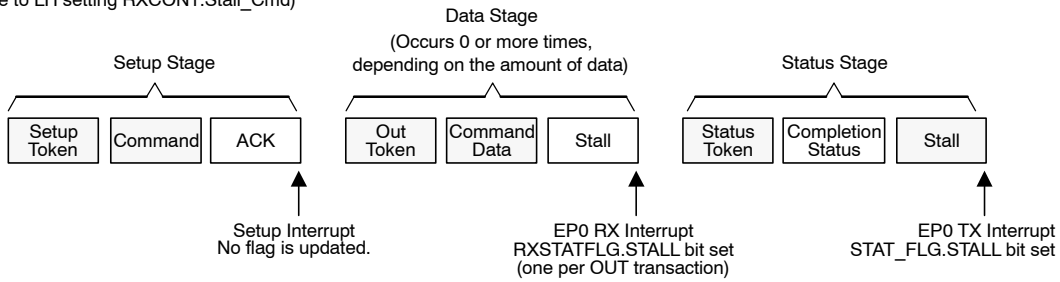
Autodecode control read transfers--request error
(due to wrong setup or command data)



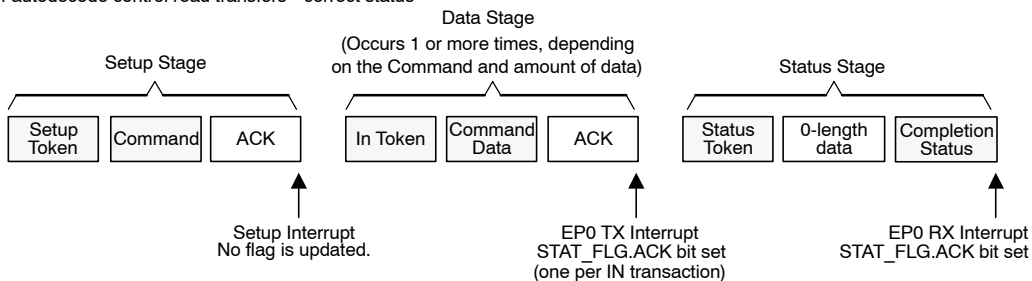
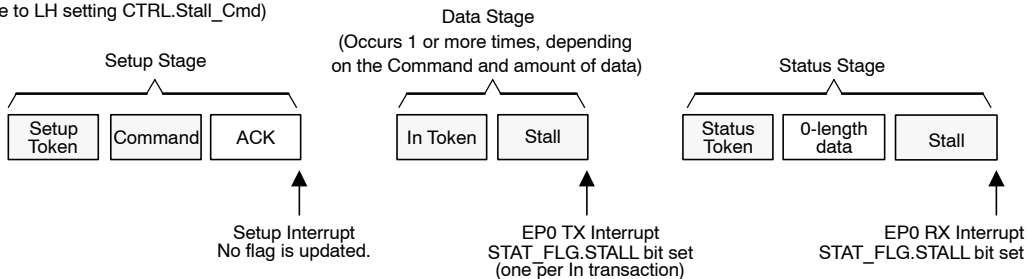
Interrupt occurs/status flags are not updated.

Figure 41. Stages and Transaction Phases of Non-Autodecoded Control Transfers

Non-autodecode control write transfers--correct status

Non-autodecode control write transfers--request error
(due to LH setting RXCON1.Stall_Cmd)

Non-autodecode control read transfers--correct status

Non-autodecode control read transfers--request error
(due to LH setting CTRL.Stall_Cmd)

Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0, have specific USB protocol meaning, and are not handled automatically by the core. The USB function block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the normal RX and TX control bits that affect transaction handshaking. A general USB interrupt to the local host occurs at the end of each transaction of each stage of a control transfer. The local host must perform the following actions to act on non-autodecoded control transfers:

- ☐ Process the setup phase USB interrupt. The local host reads the control transfer command from the setup FIFO and decodes the command. For control reads, the local host fetches the requested read data and places it (or as much of the read data as will fit) into the endpoint 0 FIFO, then enables the endpoint 0 FIFO. For control writes, the local host code only enables the endpoint 0 FIFO. Local host code also sets any flags needed for processing endpoint 0 USB interrupts during the control transfer.
- ☐ Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data has been sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and, when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the local host must set the endpoint 0 control bits to signal the desired stage status to the host.
- ☐ Process the status stage endpoint 0 general USB interrupt. The local host provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0, have specific USB protocol meaning, and are handled automatically by the USB function block without any intervention by the local host. The USB function block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the local host during autodecoded control transfers.

If no request defined by the USB1.1 specification is associated with the data of the setup phase, the request is stalled by the core and the local host is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine if it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the local host (if the local host had control of the FIFO). New setup data is placed into the setup FIFO, and the setup interrupt flag is set (Setup bit).

In response to the setup interrupt, the local host must select the setup FIFO by setting Setup_sel bit. This clears the setup flag. The local host must then read 8 bytes from the setup FIFO, clear the Setup_Sel bit, and check that the setup bit has not been reset by a new setup transaction. If the setup flag is asserted, the local host must discard the previously read data and handle the new setup packet as explained above. Thus the local host never misses a new occurring setup transaction (this is per USB 1.1 specification).

13.5.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If the new address is different from 0, the device moves into addressed state (the ADD bit set) if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device's remote-wakeup feature, the R_WK_OK bit is set or cleared as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see USB 1.1 specification).

Per USB 1.1 specification, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after the setup stage, even if no status stage occurs.

Autodecoded Control Write Transfer Handshaking

The USB function module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which is ignored by the USB module. The Set_FIFO_En and Stall_Cmd bits have no effect on the handshaking.

Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit-stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

13.5.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device request to get the endpoint and device status. These control read transfers access information that is kept in registers inside the USB module, so local host code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected endpoint's status information (depending on the index value in the setup stage data packet) during the data phase of the single IN transaction of the data stage and provides ACK as the handshake for the status stage handshake phase. The local host receives no interrupt.

Autodecoded Control Read Transfer Handshaking

The USB function module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, except if a corrupted token packet is received, which is ignored by the USB module. The FIFO_En and Stall_Cmd bits have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLED and no interrupt is asserted to the local host. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way—any data stage transaction from the host in which a data error occurs is ignored.

It is possible that the USB host sends a get endpoint/device status request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

13.5.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the set/clear endpoint feature, set configuration, set interface, set descriptor, and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages (setup, data (optional), and status).

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a local host general USB interrupt with the setup flag set. The local host must respond to this general USB interrupt by setting the Setup_Sel bit, which clears the setup interrupt flag. The local host must then read 8 bytes from the setup FIFO via the DATA register, clear EP_Sel bit, and check the setup flag. If the setup flag is set, the local host must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the setup flag is cleared, the local host code must interpret this request information and performs any application-specific activity needed due to the setup stage request (see Figure 41). If there is one or more data stages for the transfer, the local host must set the Set_FIFO_en bit for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation apply as for non-isochronous, non-control OUT endpoints. The local host can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the local host must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control bit FIFO_En. The local host may delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding the FIFO_En bit equal to 0), or by causing ACK handshaking by setting the Set_FIFO_En bit to 1 (with an empty endpoint 0 FIFO). An endpoint 0 TX general USB interrupt is sent to the local host at completion of the status stage.

After a SET_CONFIGURATION request, the device moves into addressed or configured state as soon as the local host sets the Dev_Cfg or the Clr_Cfg bits.

Specific Local Host Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate Set_Halt control bit.

If the device receives a valid clear endpoint halt feature request, it must set the appropriate Reset_EP bit to clear halt condition, set FIFO flags, and reset data PID to DATA0 for the endpoint. If specified endpoint number is 0, the local host has only to set the Clr_Halt bit to clear halt condition.

If the device receives a valid set configuration request, it must reset all endpoints by setting the Reset_EP control bits, set the Self_Pwr bit to the appropriate value, and set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the set configuration was received, the local host must write 1 to the Dev_Cfg bit to allow the device to move into the configured state (the CFG bit set). If the device was configured when the set configuration was received, and new configuration value is 0, the local host must write 1 to the Clr_Cfg bit to allow the device to move back into the addressed state (the CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set by setting the Reset_EP control bits and must set the halt conditions for endpoints not used by this interface.

Other local-host-required-actions are specific to the request and not detailed in this document.

Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module. No interrupt is generated.

Data stage handshaking for non-autodecoded control write transfers is dependant on the endpoint 0's FIFO_En, EP_Halted, and Stall_Cmd bits. The local host may delay completion of any transaction of the data stage by signaling NAK (via the Set_FIFO_En bit not set). The USB specification requires that once a STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the Stall_Cmd or the Set_Halt (reflected in the EP_Halted register bit) register bits provide this functionality. Also note that the EP_Halted bit does not reflect the forced STALL caused by the Stall_Cmd bit; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0's FIFO_En and Stall_Cmd bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB function module returning a zero-length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. While NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host may choose to abort the control write after some number of NAKs.

Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the local host cannot deal with itself, the local host must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting the endpoint 0 Stall_Cmd bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

13.5.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a local host general USB interrupt with the setup flag set. The local host must respond to this general USB interrupt by setting the Setup_Sel bit, which clears the setup interrupt flag. The local host must then read 8 bytes from the setup FIFO via the DATA register, clear EP_Sel bit, and check the setup flag. If the setup flag is set, the local host must discard the setup data it has just read, and handle the new setup data packet following the same scheme. If the setup flag is cleared, the local host code interprets this request information and then prepares data for the IN transaction that follows. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO and setting the Set_FIFO_En bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation apply as for non-isochronous, non-control IN endpoints; the local host can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, local host code must move more data to the endpoint 0 FIFO until the last bytes of the requested data has been provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The local host code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a 0-length data packet, and the local host code must return its completion status for the control read standard request via standard handshaking mechanisms.

In the case of returning exactly what the host requested when the request was a multiple of the maximum packet size, no zero length packet is required. A zero-length packet is required only when the amount of data the device has to return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size.

Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependant on the endpoint 0 FIFO_En, EP_Halted, and Stall_Cmd bits. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The Stall_Cmd and the Set_Halt (reflected through the EP_Halted register bit) register bits provide this functionality. The EP_Halted does not reflect the forced STALL caused by the Stall_Cmd bit; it retains its previous value.

Status stage is controlled by the FIFO_En and the Stall_Cmd bits.

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB function module responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALL is returned by the core, and an interrupt is asserted.

Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs that the local host cannot handle itself while handling the control read, the local host must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0's Stall_Cmd bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for bulk/interrupt transactions. The USB function module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the local host.

13.5.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 94. Autodecoded Versus Non-Autodecoded Control Requests

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
GET_STATUS	Device	Autodecoded	None	Device status is returned (the Self_Pwr and R_WK_OK bits).
	Interface	Non-autodecoded	The local host must stall the command (via the Stall_Cmd bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the local host.
	Endpoint	Autodecoded	None	The core automatically stalls the command if end-point number is different from 0.

Table 94. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
CLEAR/SET FEATURE	Device	Autodecoded	None (DS_Chg interrupt is asserted to the local host after any the R_WK_OK bit modification).	The core handles the request.
	Interface	Autodecoded	None (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled.
	Endpoint	Non-autodecoded	The local host must stall the command (via the Stall_Cmd bit) if endpoint number/type/direction is not correct. The local host must reset the endpoint after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, local host only has to clear or set halt condition: FIFO and data PID are always correct for next setup.	Command is passed to the local host.
SET_ADDRESS	Device	Autodecoded	None (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_Chg interrupt to the local host).	<input type="checkbox"/> Default: device moves into the addressed state if address number is different from 0. <input type="checkbox"/> Addressed: device takes the new address value or moves in default state if address number is 0. <input type="checkbox"/> Configured: request is STALLED.
GET_DESCRIPTOR	All	Non-autodecoded	The local host must write descriptor data into endpoint 0 FIFO.	Command is passed to the local host.

Table 94. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
SET_DESCRIPTOR	All	Non-autodecoded	The local host must stall the command (via the Stall_Cmd bit) if it does not support set descriptor requests.	Command is passed to the local host.
GET/SET CONFIGURATION	Device	Non-autodecoded	<p>The local host must stall the command (via the Stall_Cmd bit) if configuration number is not correct.</p> <p>If the request is SET_CONFIG, the local host must reset all endpoints, halt endpoints not used by the default interface setting, set the Self_Pwr value if device is self-powered for the configuration set, and then set the Dev_Cfg bit (if config nb is not 0), or set the Clr_Cfg bit (if config nb is 0) before allowing status stage to complete.</p> <p>The device goes to configured state (if Dev_Cfg set), or moves to addressed state (if Clr_Cfg set) and a DS_Chg interrupt is asserted to the local host.</p>	Command is passed to the local host.

Table 94. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	LH Required Action	Device Behavior if Device not Configured
GET/SET INTERFACE	Interface	Non-autodecoded	The local host must stall the command (via the Stall_Cmd bit) if interface/setting number is not correct. If the request is SET_INTERFACE, the local host must reset endpoints used by the interface, and then halt endpoints not used by the interface setting, before allowing status stage to complete.	Command is passed to the local host.
SYNCH_FRAME	Endpoint	Non-autodecoded	The local host must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the local host.

- ☐ Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
- ☐ If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the local host must set the Halt feature for the endpoint. This does not happen if the USB host works correctly.
- ☐ If endpoint 0 is halted, per USB 1.1 specification (see USB 1.1 specification, section 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
- ☐ Requests are handled per specification USB 1.1, when specified in this specification, but many device reactions are not specified by USB 1.1.

13.5.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in setup data packet.

During control reads, if the USB host requests more data than indicated in setup packet, an unexpected IN transaction is STALLED, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLED. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because local host will not have enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in setup packet, the transfer is STALLed. If the USB host sends less bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the local host will not have enabled the TX FIFO.

14 Device Initialization

To allow communication between the device and a USB host, the local host must configure the device by filling the configuration registers.

For each endpoint, the local host must write to dedicated register:

- ☐ Endpoint size
- ☐ Whether or not double buffering is allowed for endpoint
- ☐ Endpoint type (isochronous or non-isochronous)
- ☐ Address of the pointer

The RAM has a specified size (2048 bytes), and the local host can choose its configuration by setting appropriate value. Figure 42 shows an example of RAM organization.

Once the endpoints are configured, the local host must set the Cfg_Lock bit. If this bit is not set, all transactions are ignored by the core. Then, when the local host is ready to communicate with the USB host, it must set the Pullup_En bit. The local host can wait until the DS_Chg Attach interrupt has been detected and handled before setting the Pullup_En bit. The USB host does not detect the device until this bit is set.

Figure 43 and Figure 44 are flowcharts of the configuration phase.

Figure 42. Example of RAM Organization

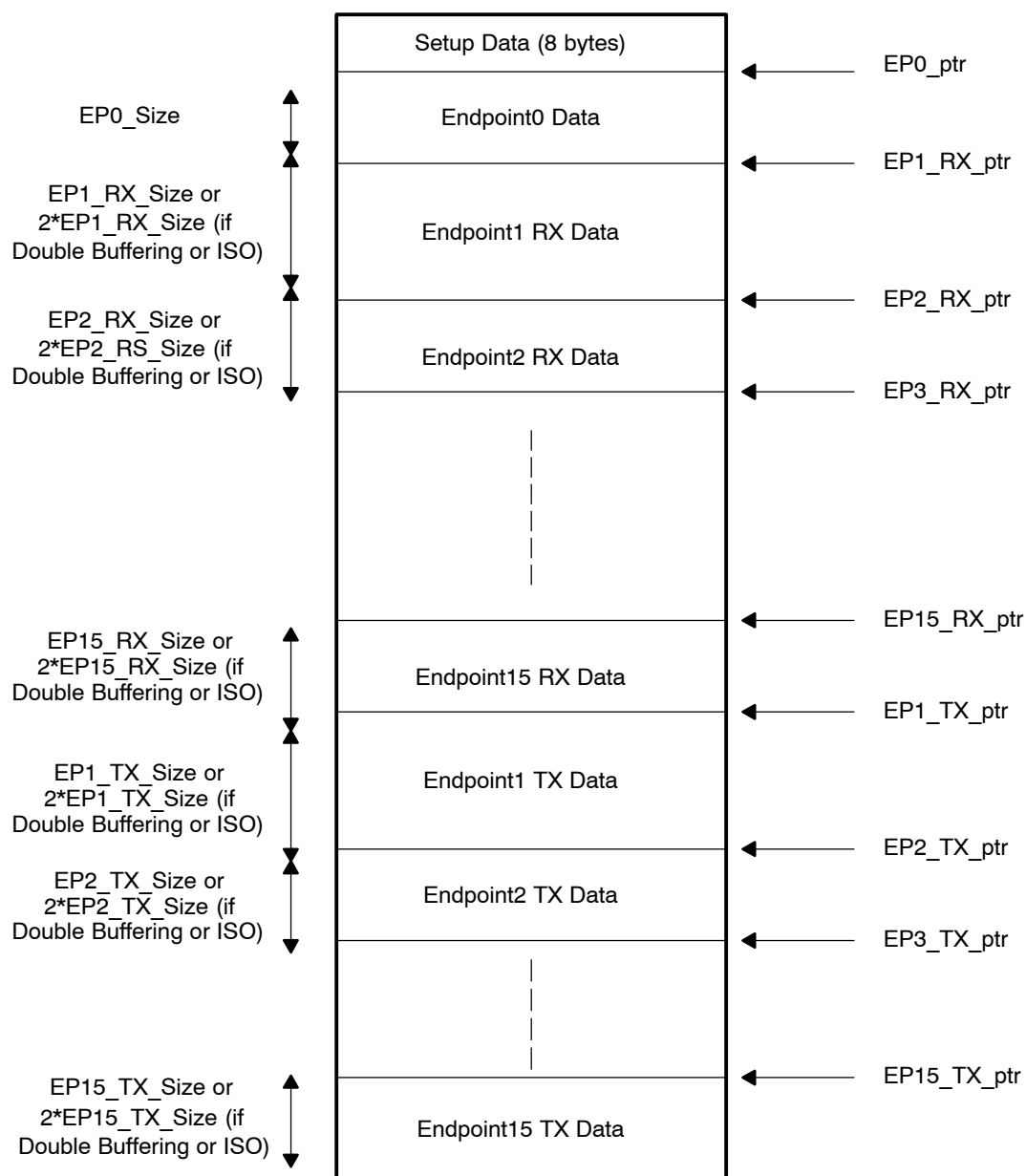


Figure 43. Device Configuration Routine

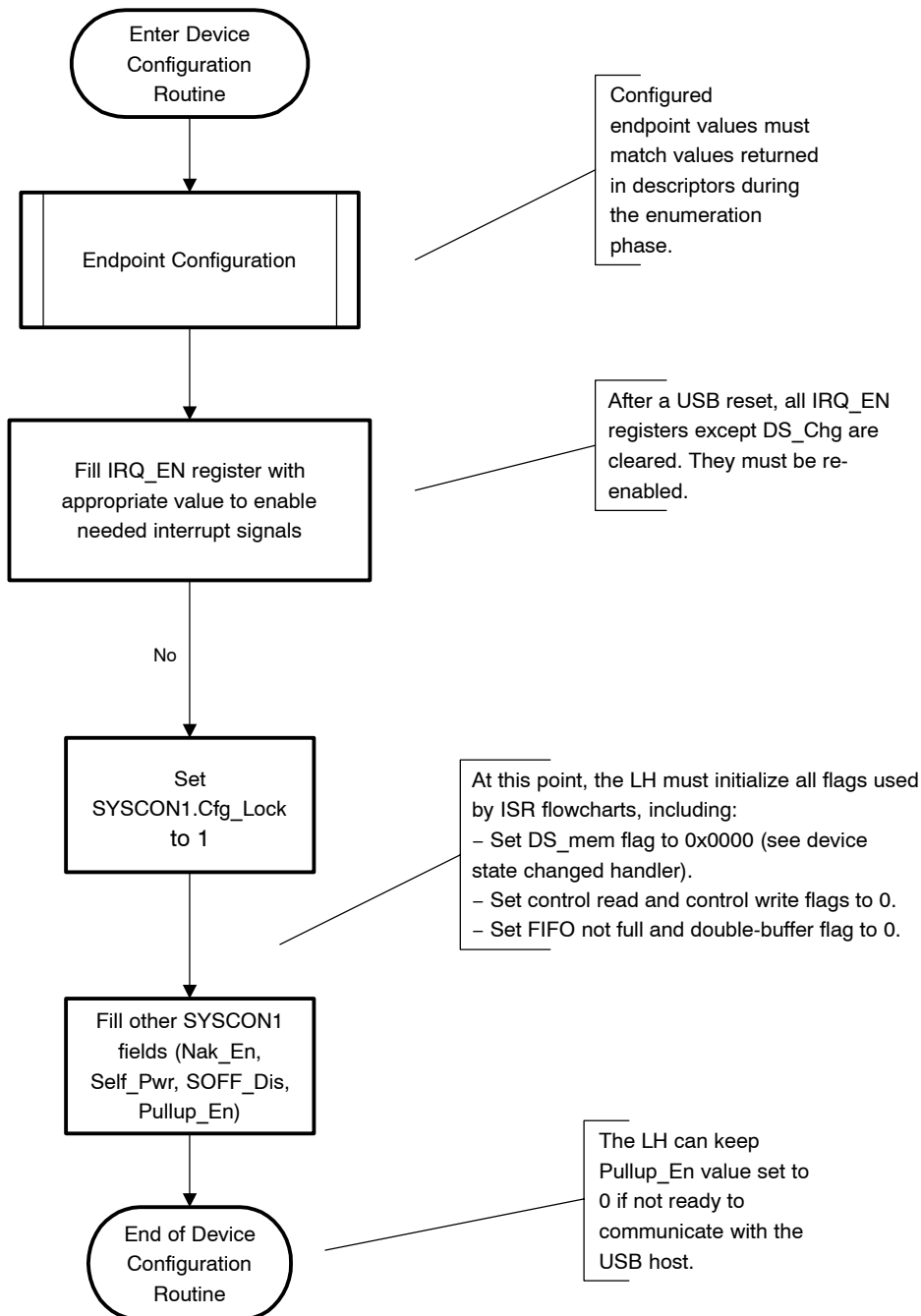
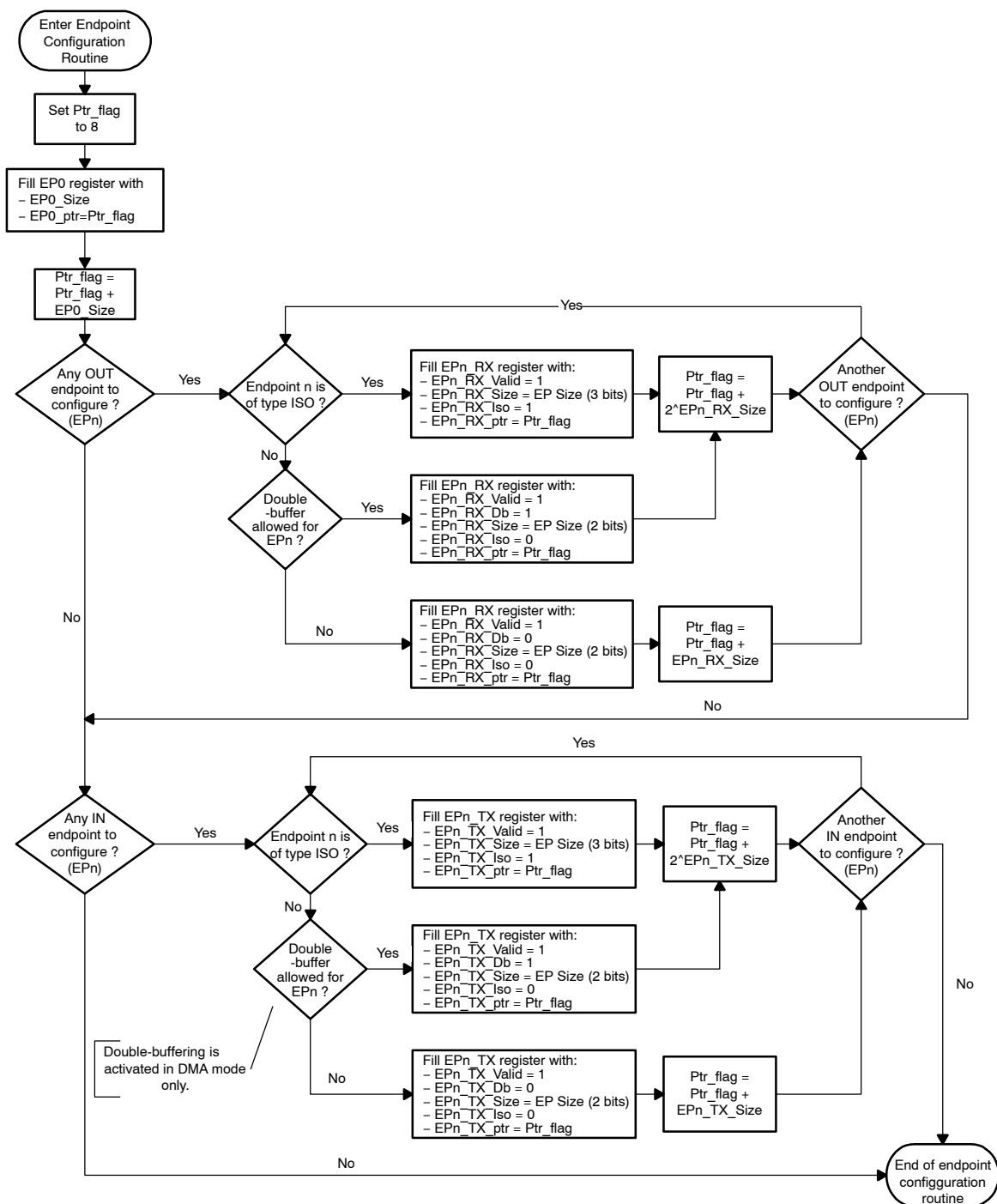


Figure 44. Endpoint Configuration Routine



15 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the local host must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling.

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the `Set_FIFO_En` bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The local host enters the Prepare for USB RX Transfers routine, shown in Figure 45, after the enumeration phase and then properly reacts to endpoint interrupts. Whether double buffering is allowed or not is transparent to the local host, unless both FIFOs are cleared through the `Clr_EP` or the `Reset_EP` bits. In that case, and in the case where the local host finishes handling an interrupt without having set the `Set_FIFO_En` bit, the local host must reenter the Prepare for USB RX Transfers routine.

For transmit endpoints, the local host enters the Prepare for USB Transfer on endpoint *n* routine each time a new file must be transmitted from endpoint *n* to USB host. The local host must not enter this routine until data written into TX FIFO from previous transfer has been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the `Clr_EP` or the `Reset_EP` bits.

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the local host reacts appropriately and enables the EP0 FIFO only if necessary.

Figure 45. Prepare for USB RX Transfer Routine

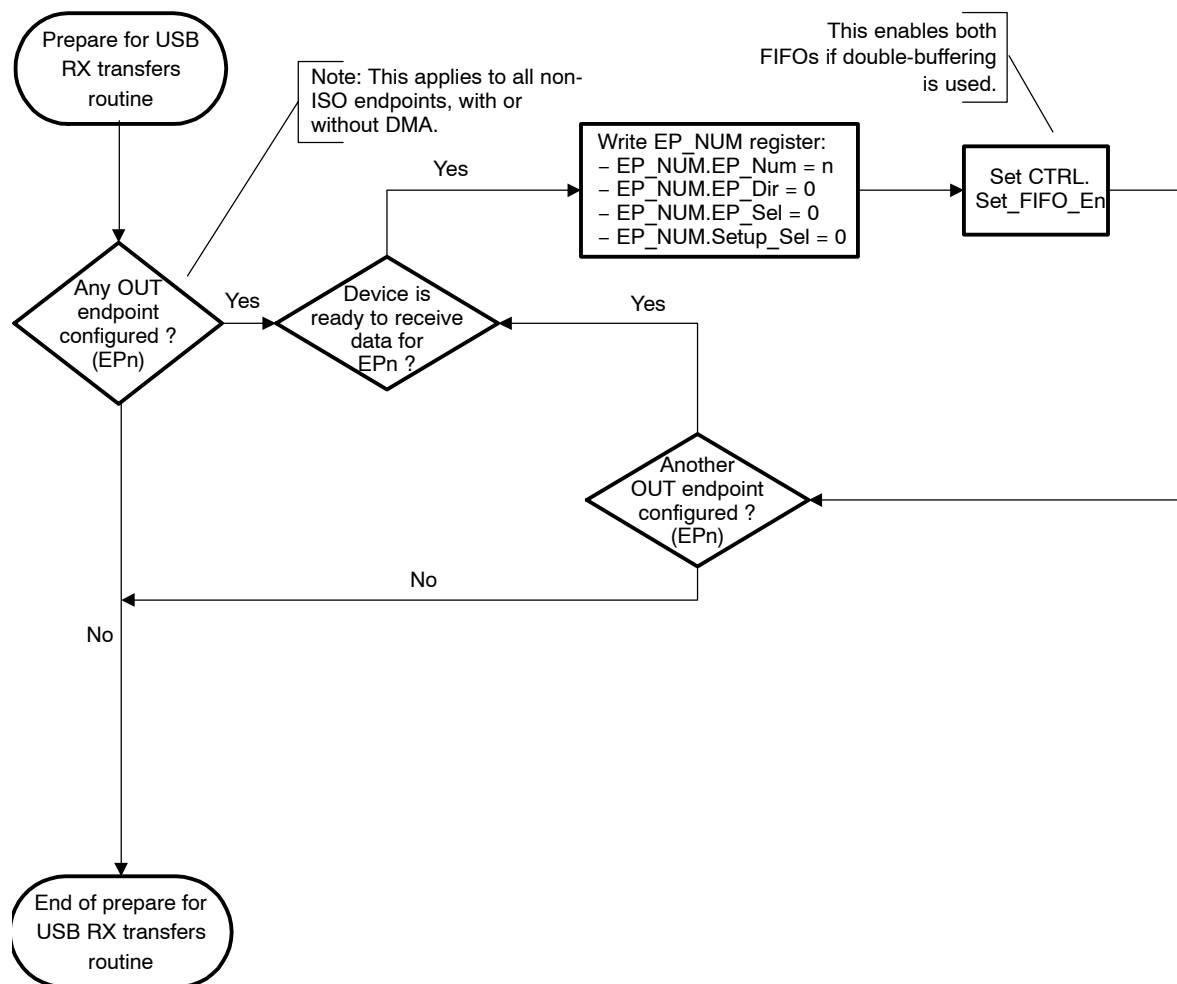
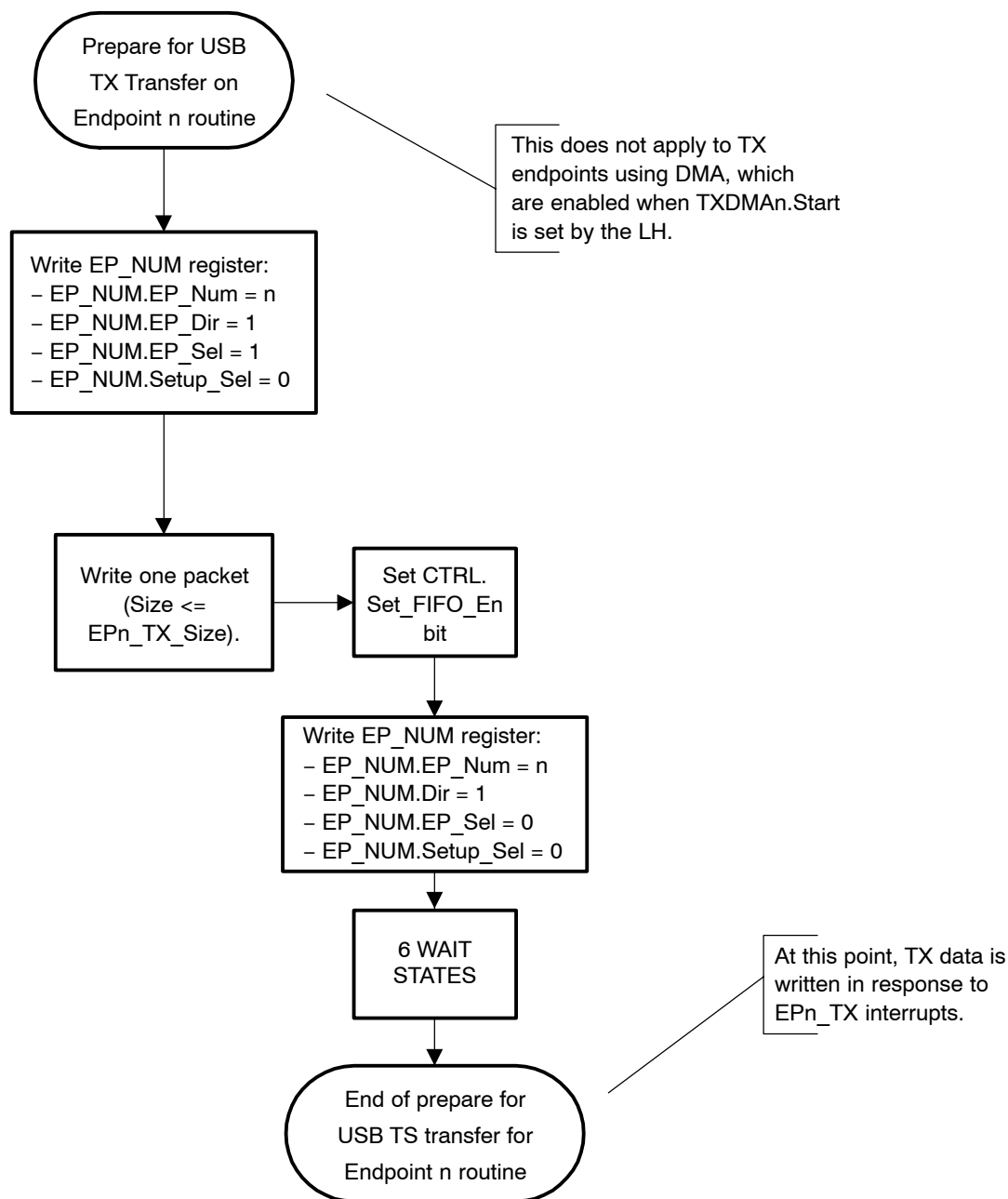


Figure 46. Prepare for TX Transfer on Endpoint n Routine



Note: No double buffering in non-DMA TX mode.

16 Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB ISR processing. System-architecture-specific details are left to the engineers who write the local host and USB host code. One USB-specific interrupt register is provided (IRQ_SRC) to keep track of the interrupts to handle. These interrupts can be of the following types:

- ☐ General USB interrupts (including endpoint 0, DMA, and device states interrupts)
- ☐ Non-isochronous endpoint-specific interrupt
- ☐ Start of frame (SOF) interrupt for isochronous transactions

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some special interrupts generated due to USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated due to USB activity for non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented below to give a guideline for how to handle the interrupts related to the USB function module. The flowcharts in this section assume that the Nak_En bit is cleared.

A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of endpoint, except for endpoint 0. The flowcharts show reads from these application buffers for IN transactions on TX endpoints, and they show writes to these application buffers for OUT transactions on RX endpoints.

16.1 Important Note on USB Interrupts

When an endpoint interrupt is asserted, the local host sets the EP_Sel bit to 1. The local host must finish the interrupt handling before clearing EP_Sel bit, because clearing this bit clears the corresponding status bit in the status flag register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the local host must not select then deselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The local host does not need to set EP_Sel to 1 when setting the Set_FIFO_En, the Set_Halt and the Clr_Halt bits.

The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed, if endpoint halt feature was set), so that the local host never misses

an ACKed transaction. If double buffering is used for an endpoint, the status flag register is updated if there is zero or one interrupt pending for the endpoint and is not updated if there are already two interrupts pending on the endpoint.

The local host does not need to set the Nak_En bit during normal operation. However, this bit must be set when the local host finishes handling an endpoint interrupt without having set the corresponding Set_FIFO_End bit. During TX transaction, if the Nak_En bit is set, the local host must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by reception of a NAK interrupt while the local host is writing the TX data.

16.2 Parsing the General USB Interrupt

The general USB interrupt ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume during enumeration phase) and control transfers on endpoint 0 or non-isochronous DMA transfers in either receive or transmit mode. Multiple interrupts may be active at any time, and all interrupts must be dealt with by the ISR before returning from the ISR. Figure 47 shows an appropriate flowchart for parsing the general USB interrupts.

16.3 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions, so that the local host cannot miss a setup transaction, even if it occurs during data or status phase of another transfer (case of aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and controlling how the local host must generate or respond to the control transfer. This information includes:

- ☐ bmRequestType
- ☐ bmRequest
- ☐ wValue
- ☐ wIndex
- ☐ wLength

The setup interrupt handler shown in Figure 48 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 49 to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- ☐ Control read flag
- ☐ Control write flag

Figure 47. General USB Interrupt ISR Source Parsing Flowchart

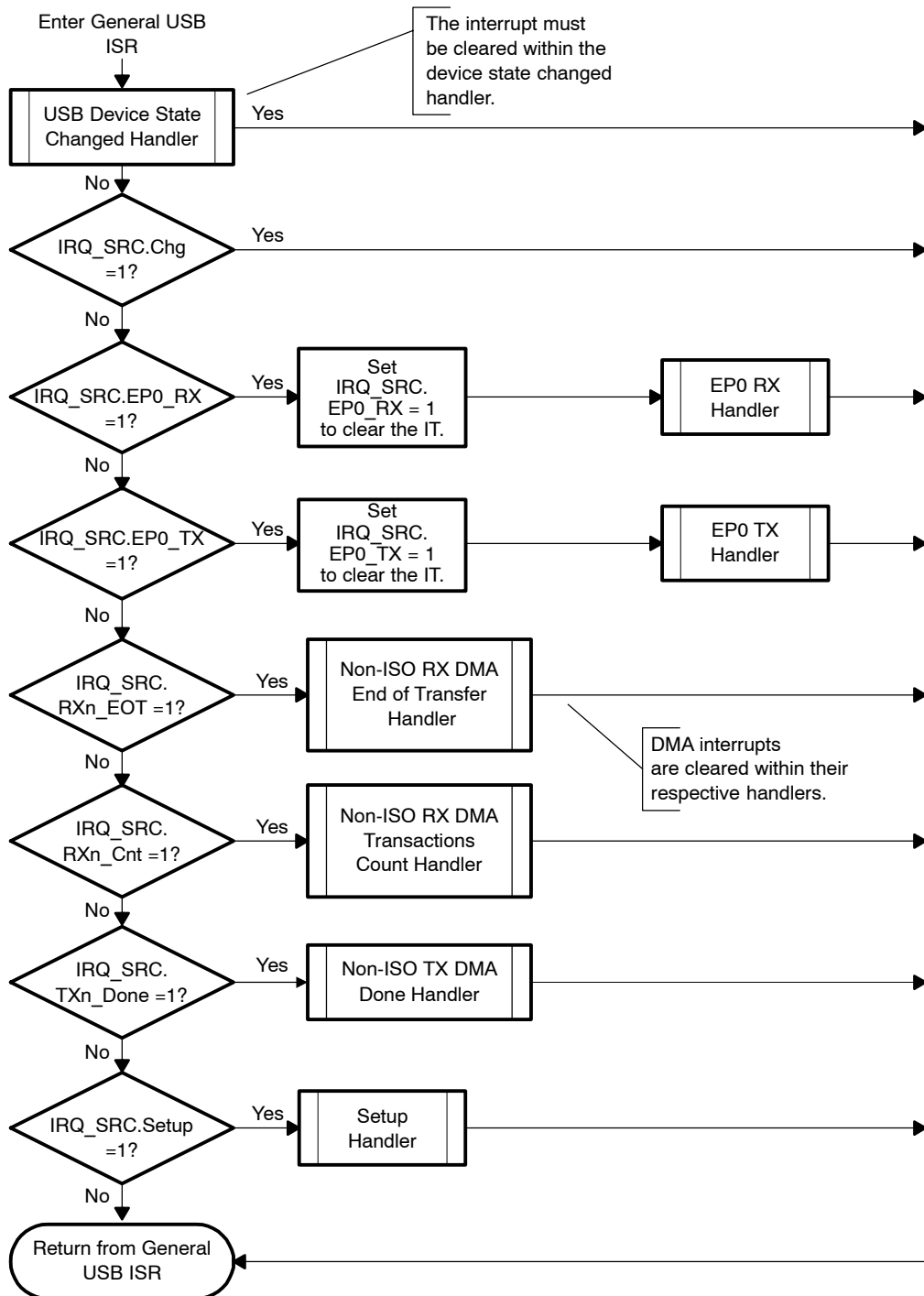


Figure 48. Setup Interrupt Handler

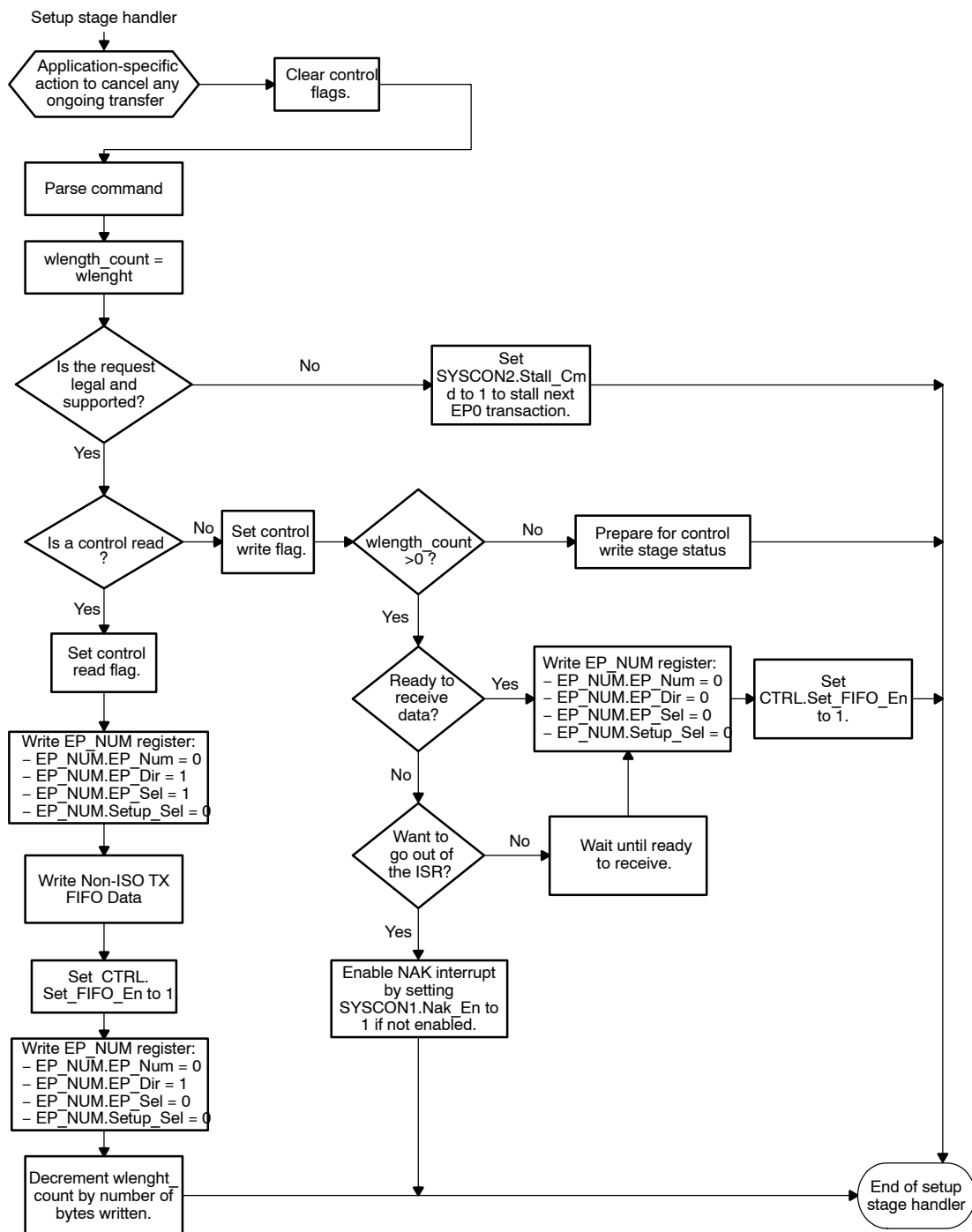
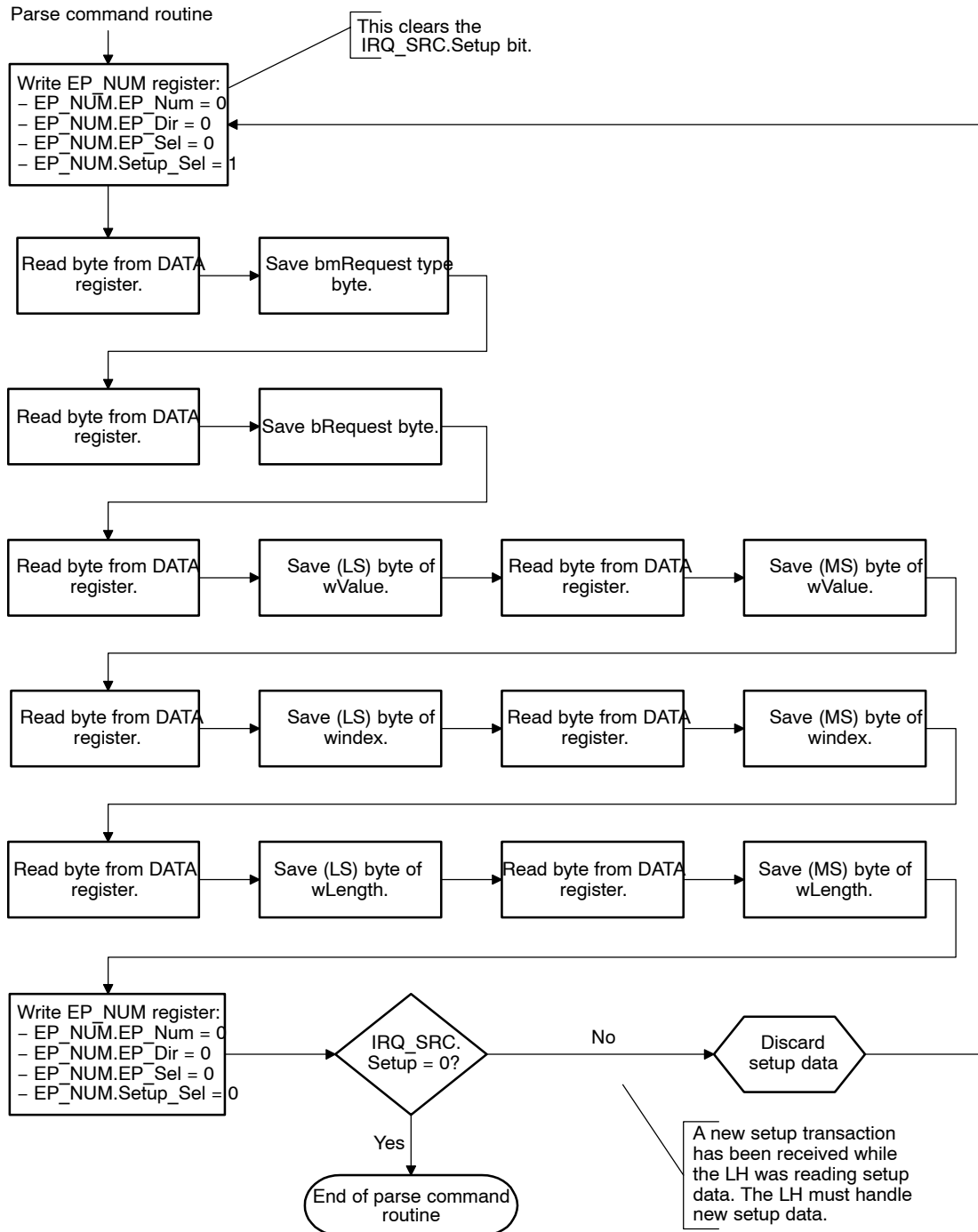


Figure 49. Parse Command Routine (Setup Stage Control Transfer Request)



16.4 Endpoint 0 RX Interrupt Handler

The endpoint 0 RX portion of the general USB interrupt handler, shown in Figure 50, must handle general USB interrupts related to control OUT transactions on endpoint 0. Notice that no EP0 interrupt is generated for autodecoded control transfers.

16.5 Endpoint 0 TX Interrupt Handler

The endpoint 0 TX portion of the general USB interrupt handler, shown in Figure 52, must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information.

Figure 50. Endpoint 0 RX Interrupt Handler

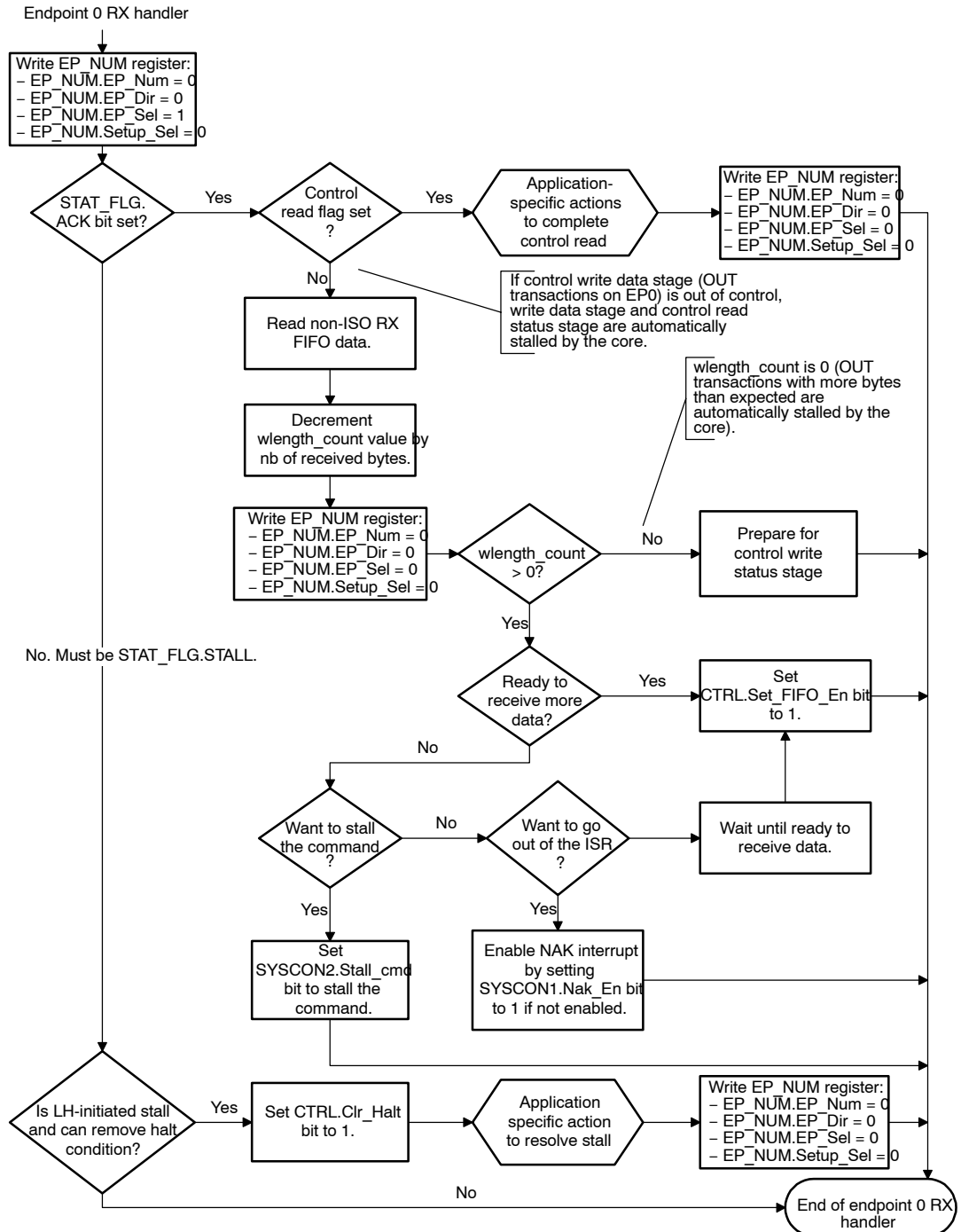


Figure 51. Prepare for Control Write Status Stage Routine

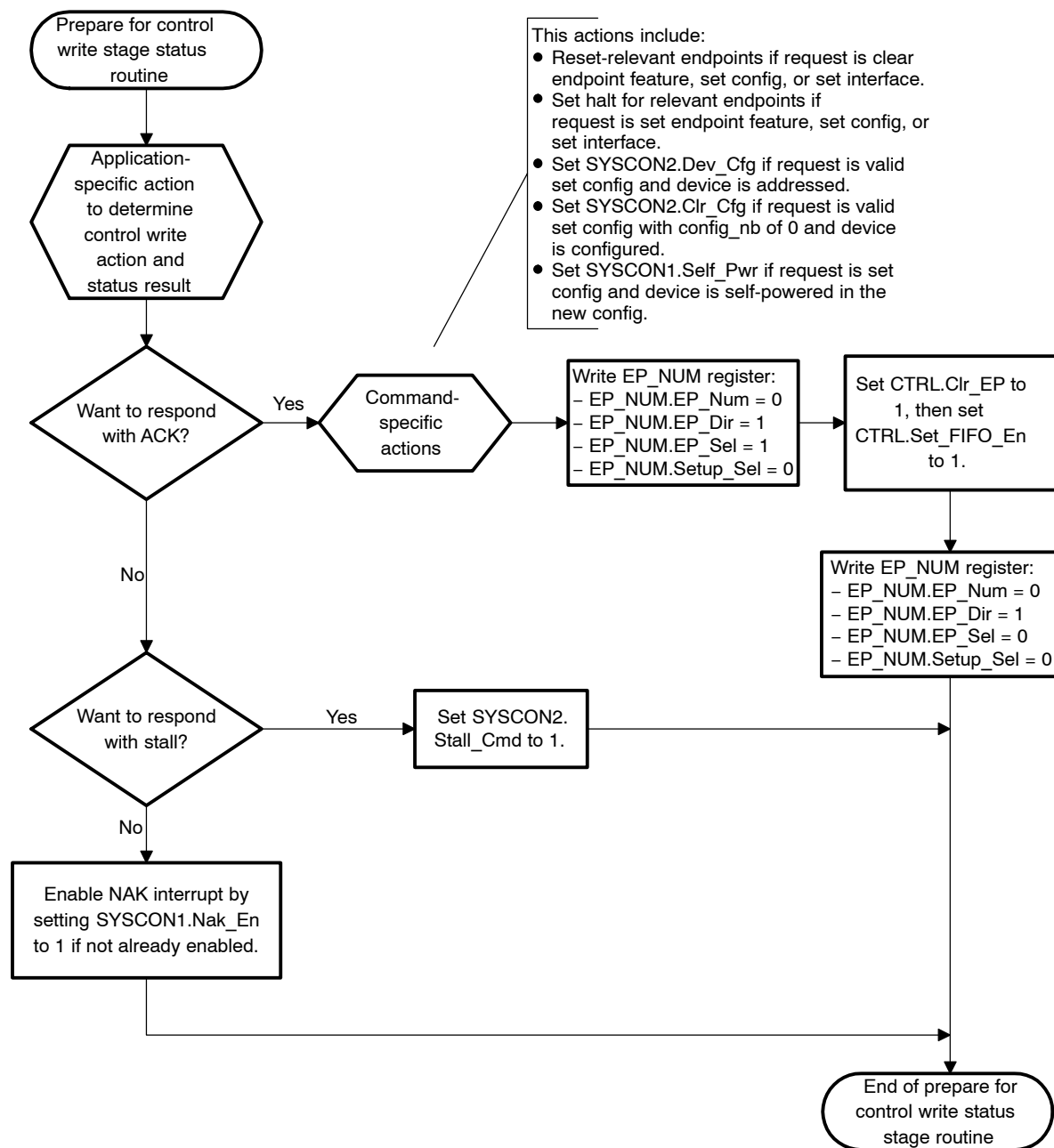


Figure 52. Endpoint 0 TX Interrupt Handler

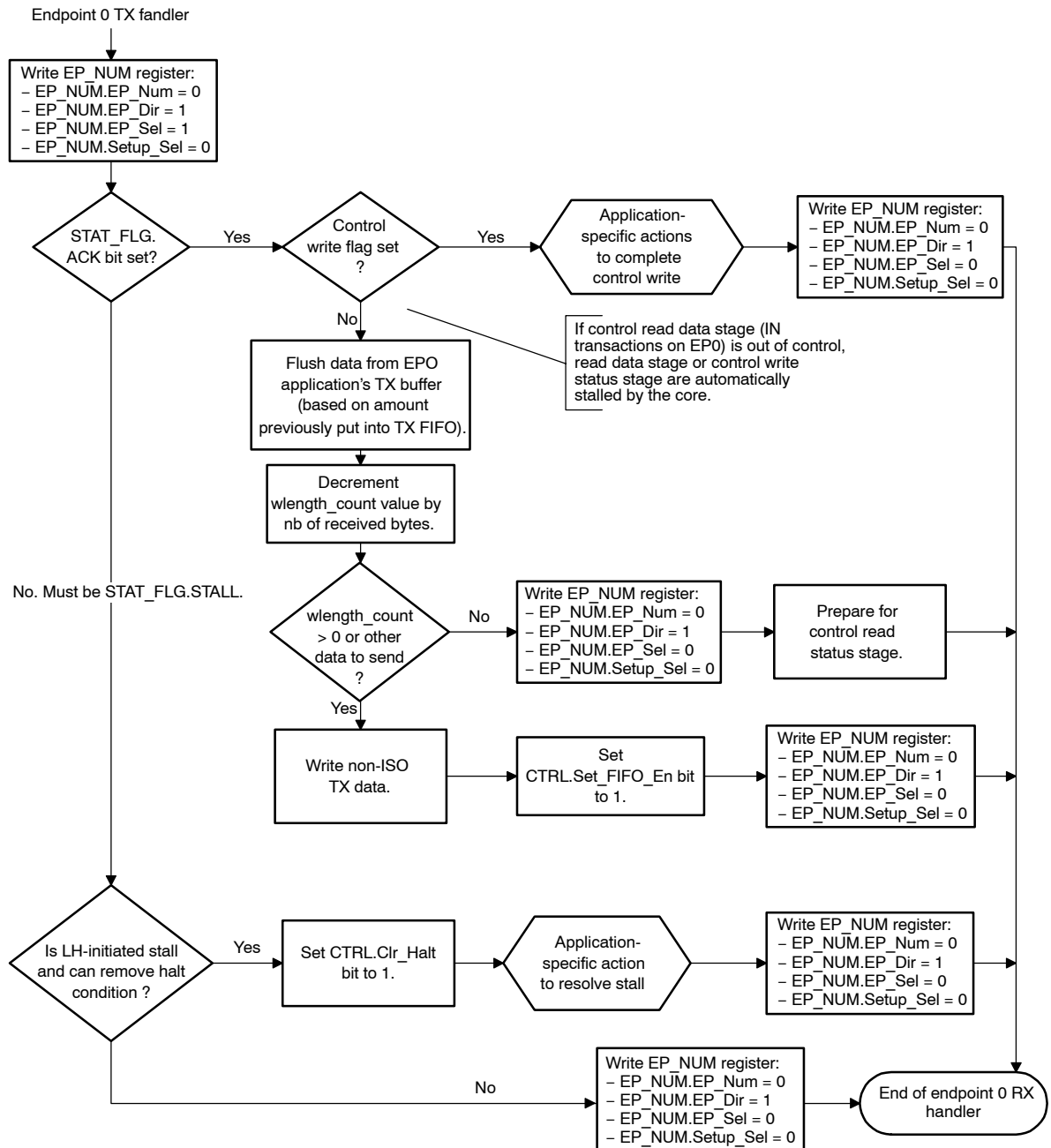
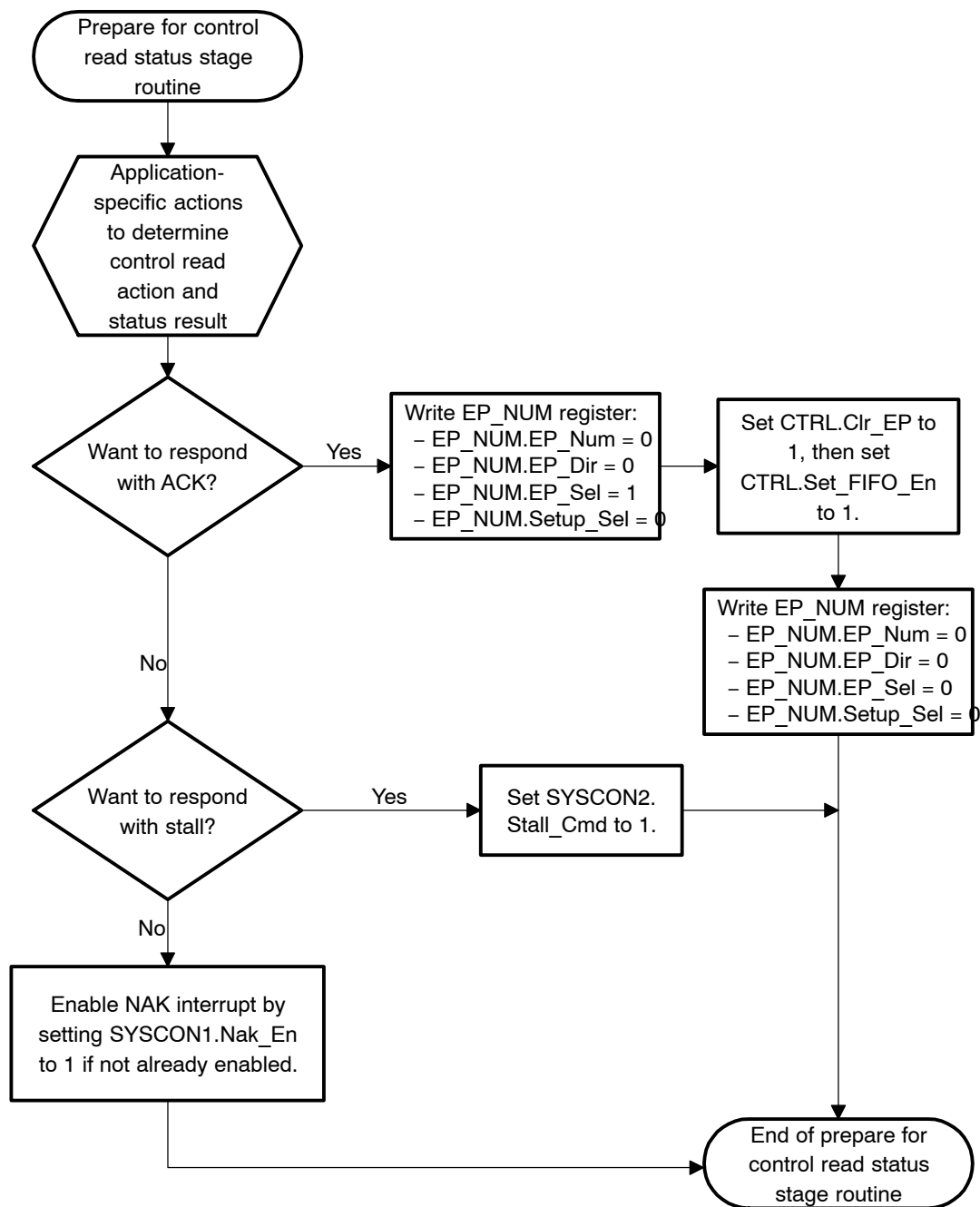


Figure 53. Prepare for Control Read Status Stage Routine



16.6 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB function and how they can be handled.

The state machine (see Figure 54) moves the USB function device from one state to another state with respect to USB1.1 specification. The attach/unattach transition is not shown in the transition flow.

Since the SET_CONFIGURATION is not decoded by the core, the local host has the responsibility to distinguish a SET_CONFIGURATION with a non-valid configuration value from other SET_CONFIGURATION requests and to set the Dev_Cfg only if configuration value is valid (value 0 is non-valid), when device is in addressed state. When device is in configured state, the local host has the responsibility to set the Clr_Cfg if configuration number is 0 so that the device moves to addressed state.

Device states are visible in DEVSTAT register and are decoded as follows:

- ☐ Attached: The device is attached to the USB and powered.
- ☐ Default: The device is attached to the USB, powered, and reset.
- ☐ Addressed: The device is attached to the USB, powered, reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number different of 0.
- ☐ Configured: The device is attached to the USB, powered, reset, has an address different from 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request only if the local HOST has set the Dev_Cfg bit (meaning the configuration is valid).
- ☐ Suspended: Device is at minimum default and has not seen bus activity for 5 ms.
- ☐ Reset: When set, the device is receiving a valid USB host reset.
- ☐ R_WK_OK: This bit is set/cleared automatically after a valid SET_DEVICE_FEATURE/CLEAR_DEVICE_FEATURE request, respectively.

Any change in the DEVSTAT register bits triggers a device change interrupt (the DS_Chg) if enabled.

The device moves to addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the local host sets the Clr_Cfg or the DEV_Cfg bit.

Figure 54. USB Function Device State Transitions

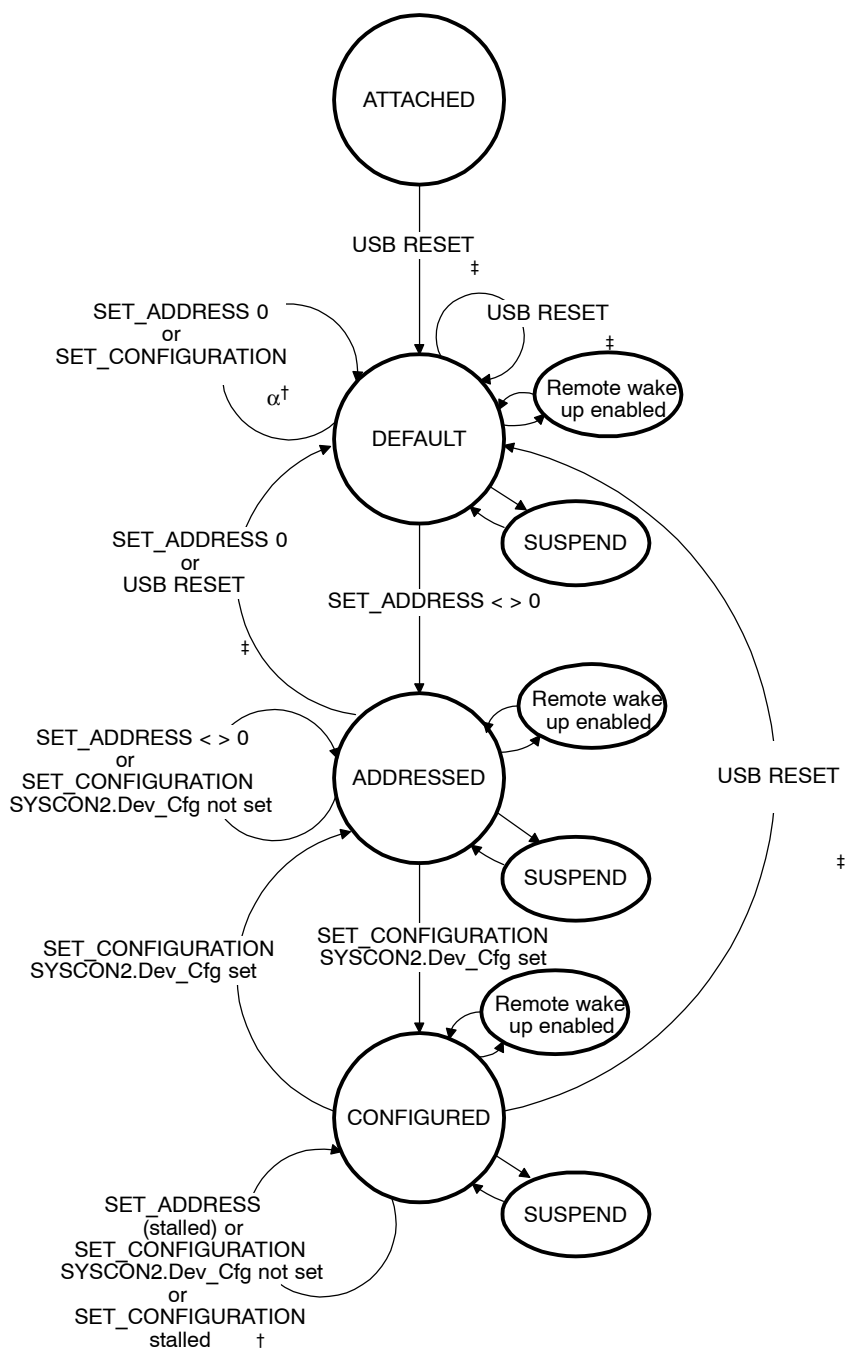
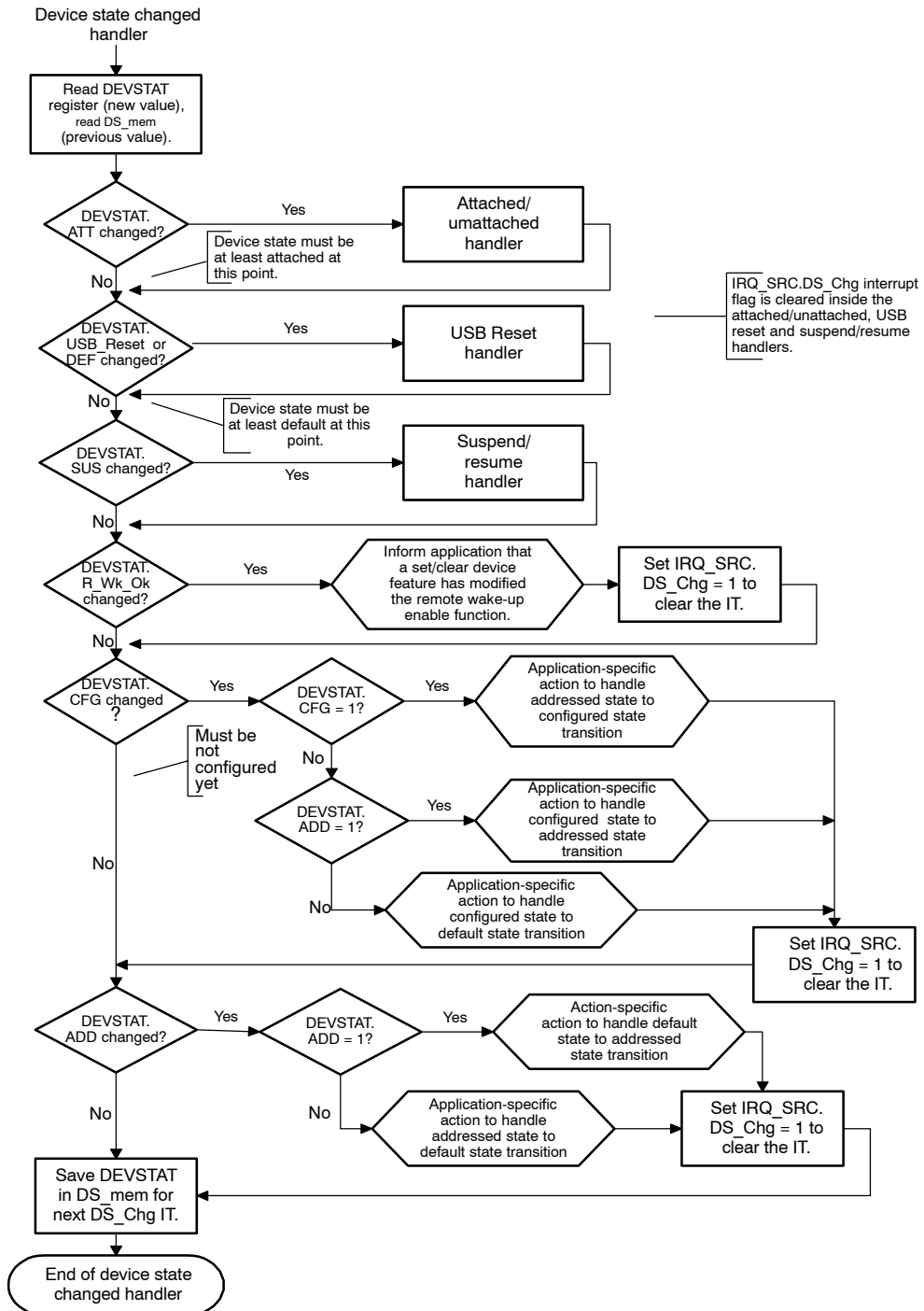


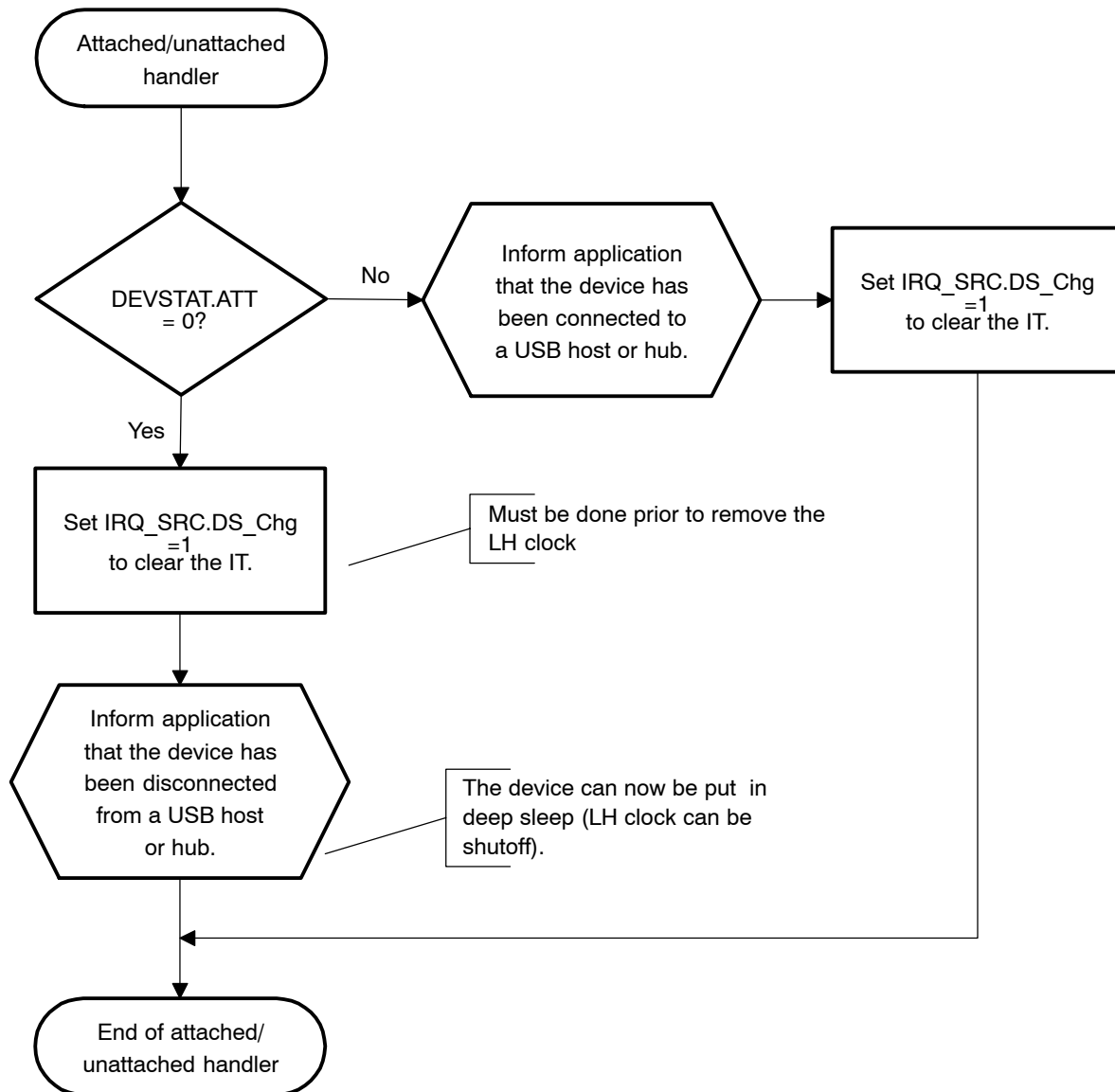
Figure 55. Typical Operation for USB Device State Changed Interrupt Handler



16.7 Device States Attached/Unattached Handler

The device attached/unattached interrupt (Figure 56) occurs either when the device detects it is connected to the USB host or Hub (VBUS is on) or when it becomes disconnected (VBUS is off). The local host can use this interrupt to put the device into deep sleep or to initialize any application-specific information relating to the USB device.

Figure 56. Attached/Unattached Handler



16.8 USB Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the local host (see Figure 57 and Figure 58). The local host responds to this interrupt by performing the following operations:

- ☐ Cancels any ongoing USB transaction and/or control transfer handling
- ☐ Clears any copies that the application has of configuration number or of alternate interface numbers
- ☐ Clears any application-specific information relating to halted endpoints
- ☐ Clears any application-specific information relating to the remote wake enable flag
- ☐ Clears any application-specific information relating to the suspend mode flag
- ☐ Clears any application-specific copy of the frame number

16.9 Suspend/Resume Interrupt Handler

When a USB suspend/resume general USB interrupt occurs (see Figure 59), the USB module has either entered or left suspend mode. The local host code must determine which and react appropriately.

The suspend sense hardware is implemented to trigger only after 5 ms of bus IDLE. This forces compliance with the USB Spec Version 1.1 t_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before remote device can signal resume).

If the local host wants to wake the device from suspend mode and remote wakeup enable is set (bit `R_WK_OK` = 1), it must first turn its clock on (if stopped) then set the `Rmt_Wkp`. The device then resumes.

If shutoff is enabled (the `SOFF_Dis` = 0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or local host driven). Setting the `SOFF_Dis` bit is part of the device configuration; however, the local host can modify its value at suspend interrupt time if necessary.

16.10 Parsing the Non-Isochronous Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (Figure 60) must parse the interrupt identifier registers `IRQ_SRC` to determine the interrupts that are active (`EPn_RX`, `EPn_TX`, or both). The two interrupts can be active at any time, and must be dealt with by the ISR before returning from the ISR. The ISR must then read `EPN_STAT` register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time.

Figure 57. USB Reset Handler Flowchart I

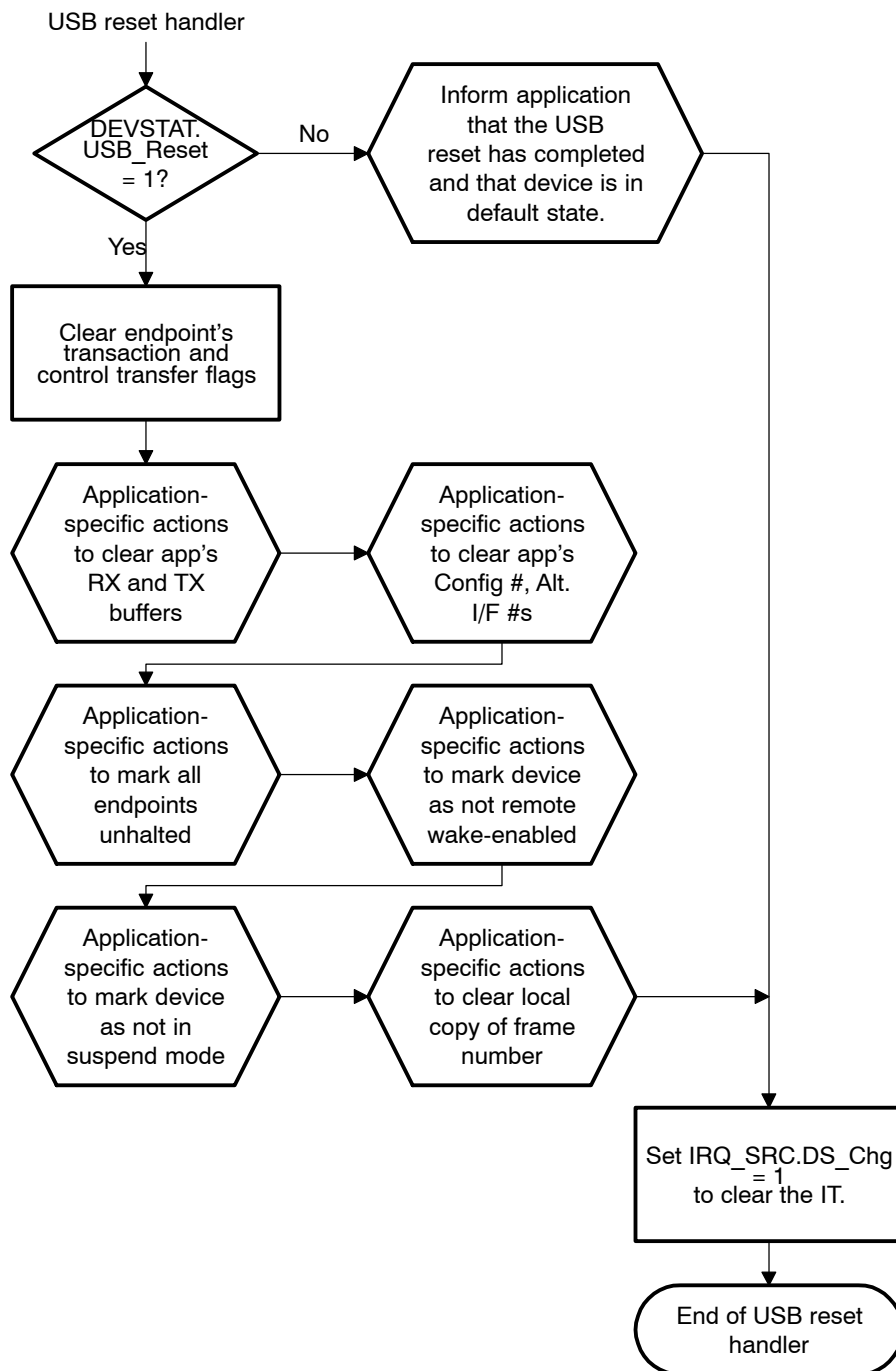


Figure 58. USB Reset Handler Flowchart II

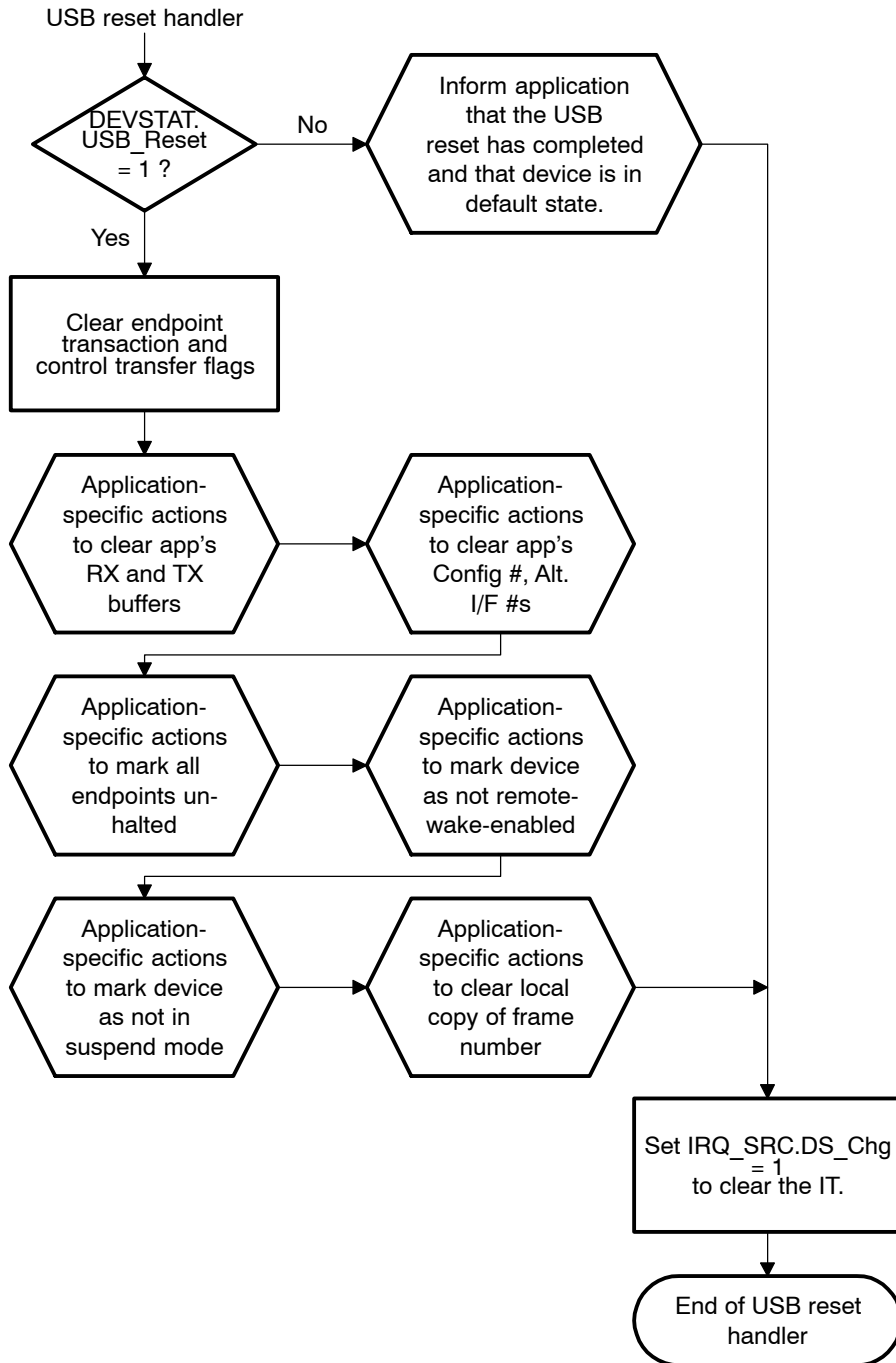


Figure 59. Typical Operation for USB Suspend/Resume General USB Interrupt Handler

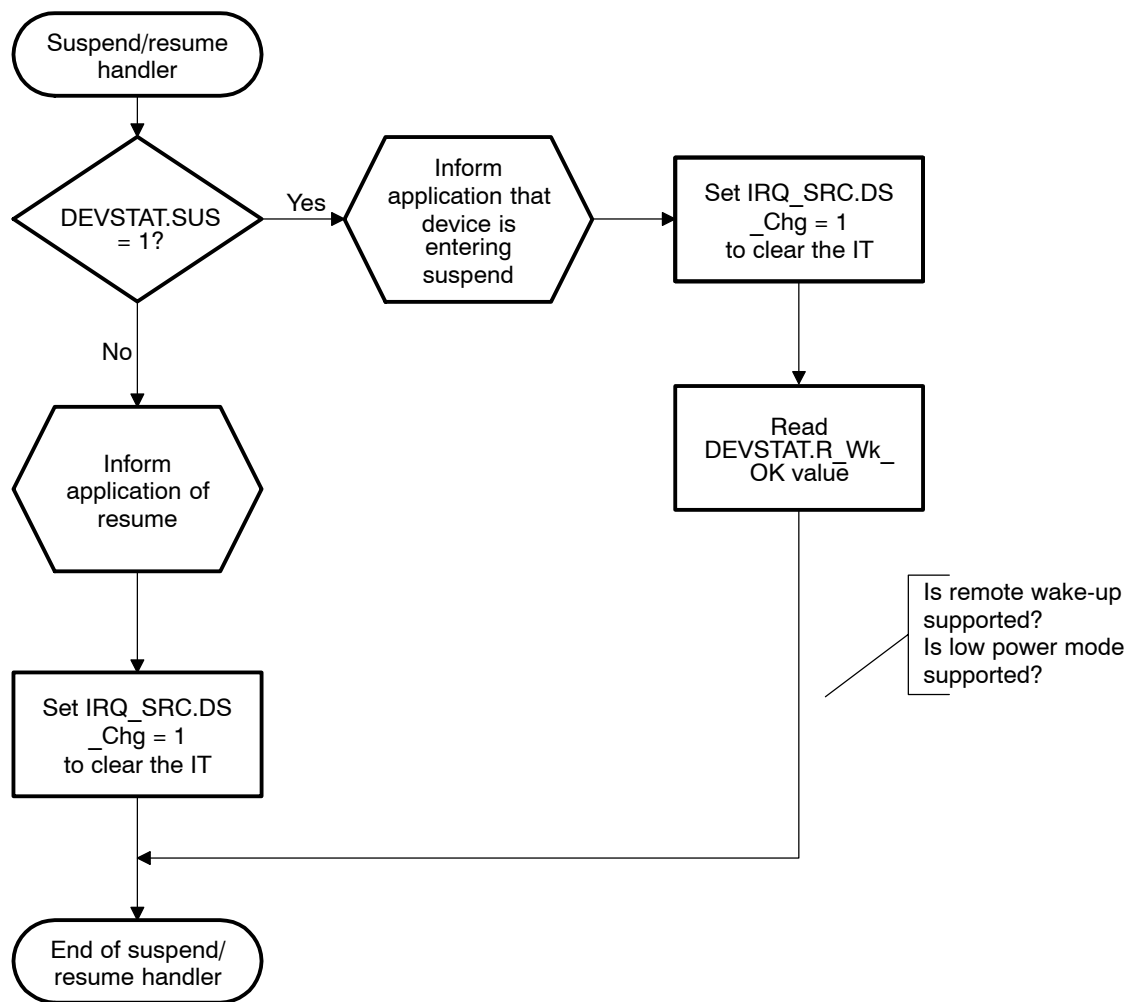
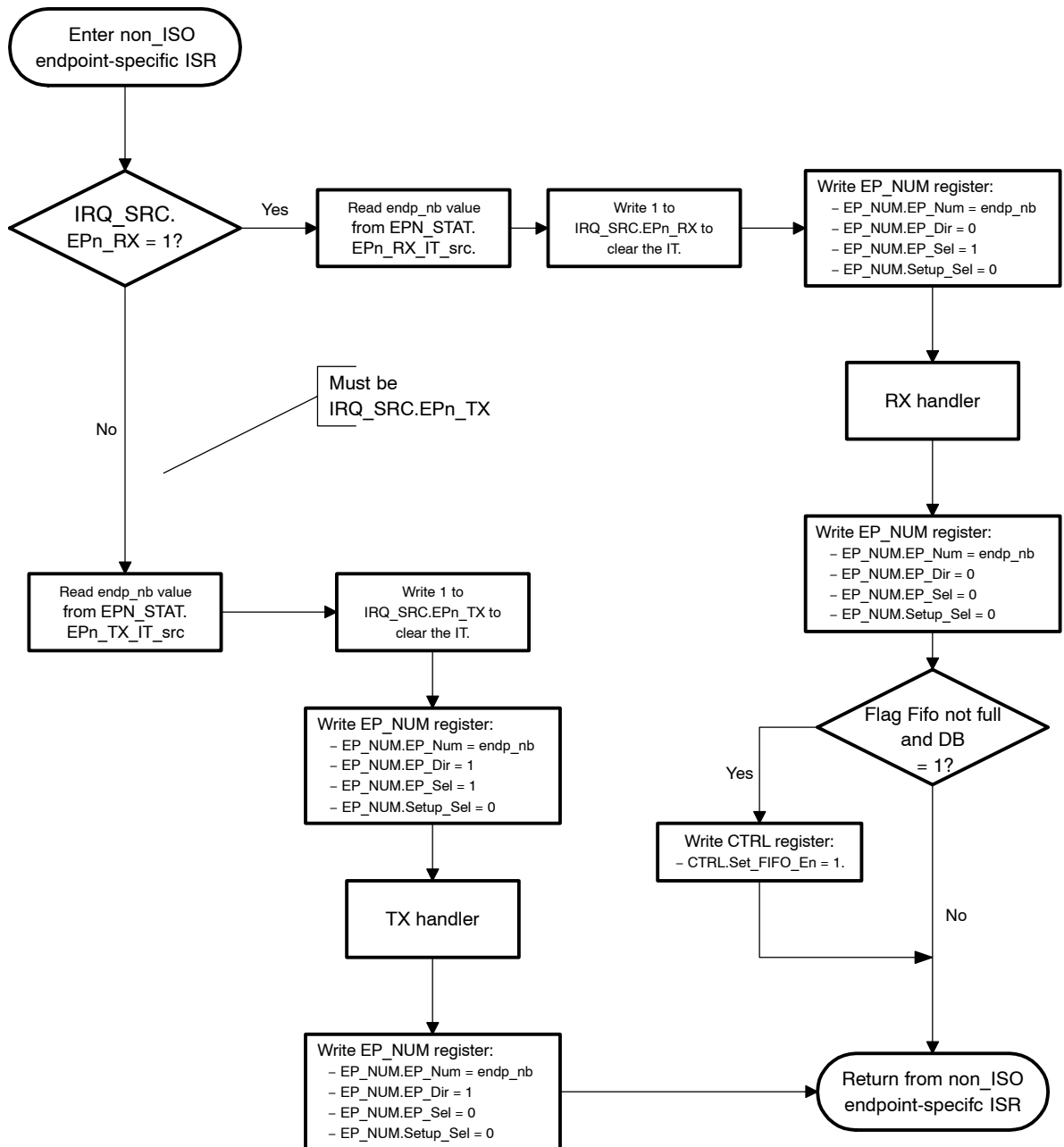


Figure 60. Non-Isochronous Endpoint-Specific (Except ER 0) ISR Flowchart



16.11 Non-Isochronous, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 61 shows the operations necessary to handle non-isochronous, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-isochronous, non-control receive endpoints, where the only differences are in EP_NUM register value set for the selection of the proper application RX data buffer in the read non-isochronous RX FIFO data routine (see Figure 62).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately due to the more complex three-stage transfer mechanism used for control writes.

16.12 Non-Isochronous, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 63 shows the operations necessary to handle non-isochronous, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-isochronous, non-control transmit endpoints, where the only differences are in EP_NUM register value set for the routine selection of the application TX buffer (see Figure 64).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately due to the more complex three-stage transfer mechanism used for control reads.

16.13 SOF Interrupt Handler

SOF interrupts to the local host occur once per USB frame. The local host must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 65 shows the SOF ISR flowchart. The read isochronous RX FIFO data and write isochronous TX FIFO data procedures are shown in Figure 66 and Figure 67, respectively.

Figure 61. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

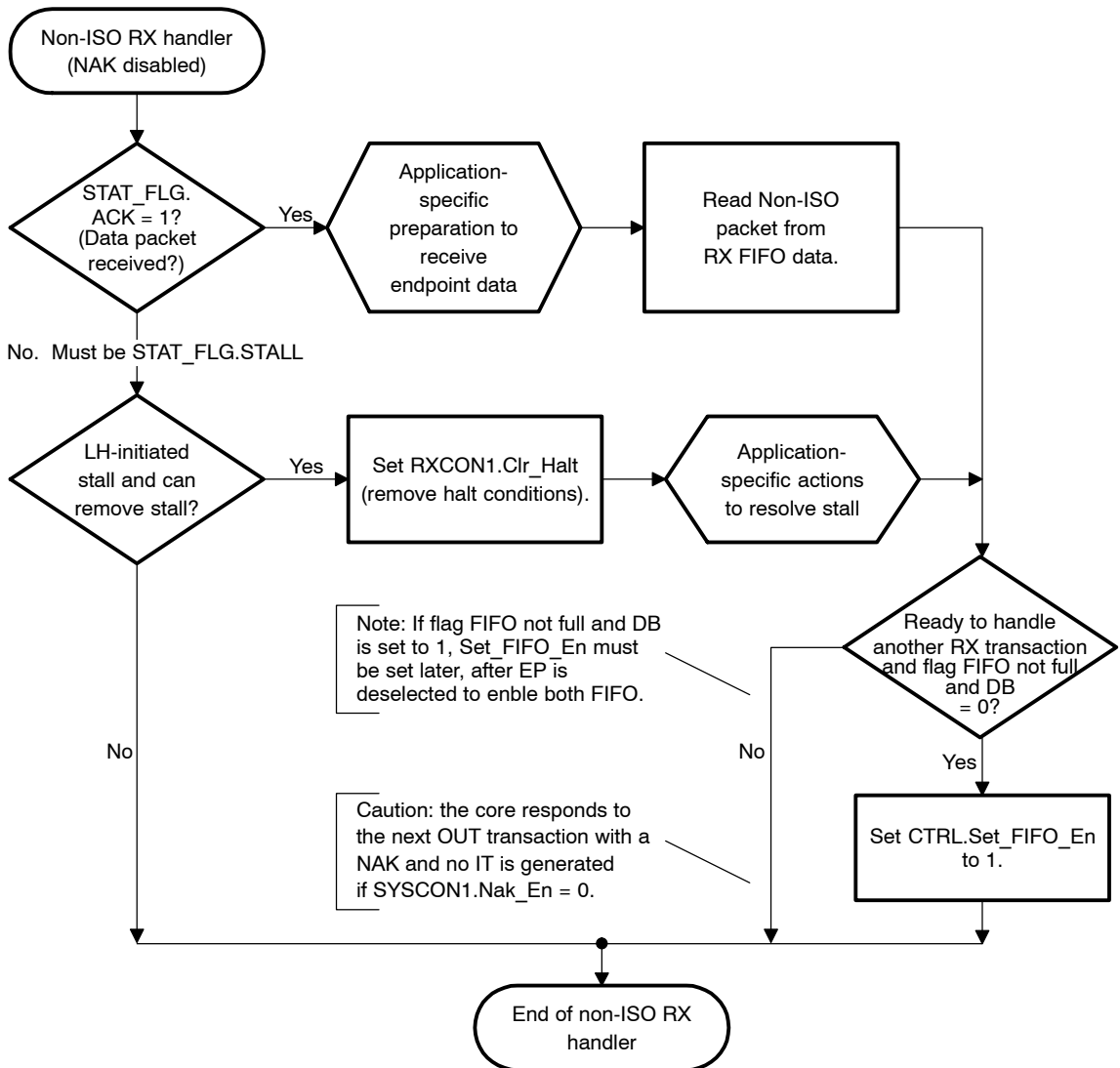


Figure 62. Read Non-Isochronous RX FIFO Data Flowchart

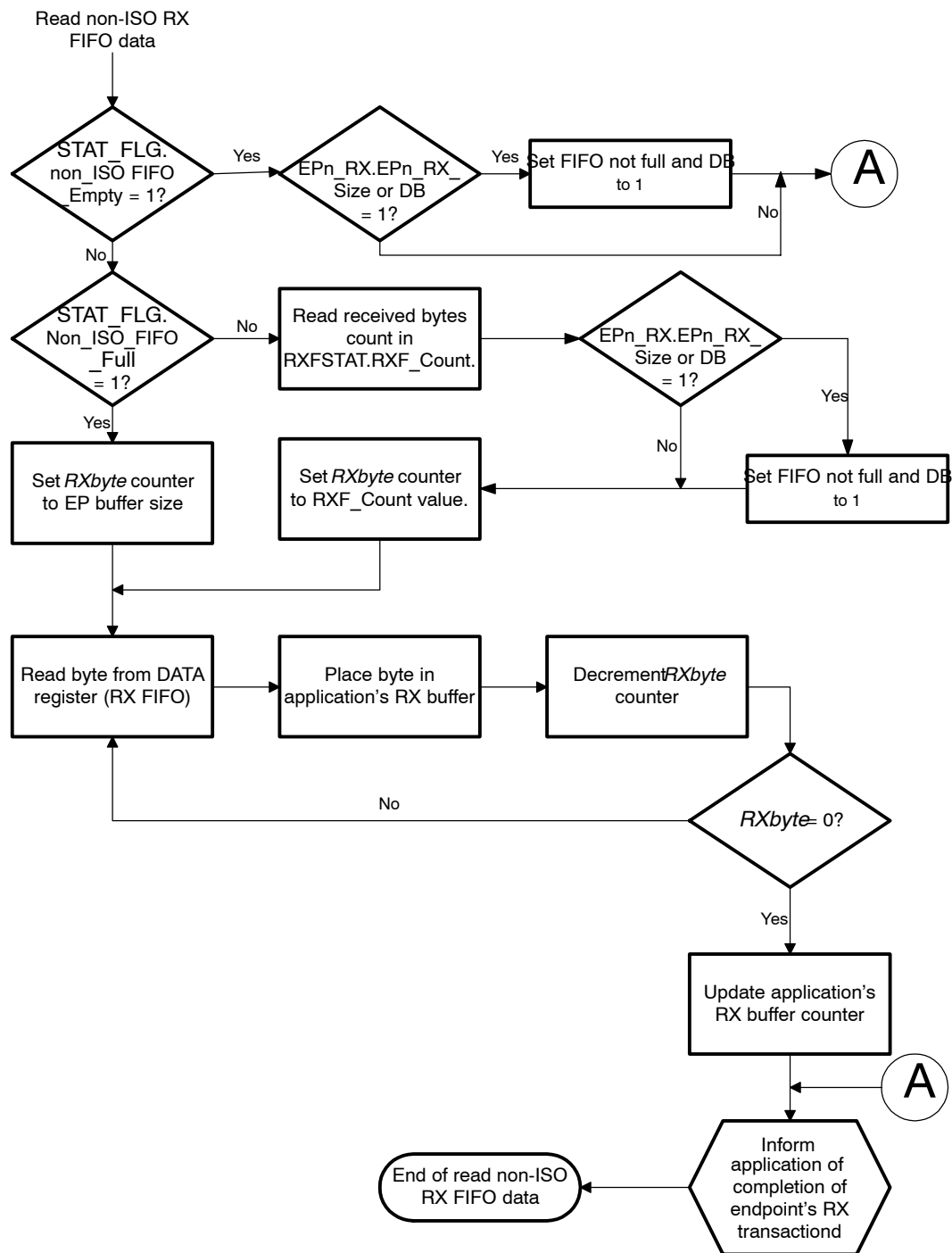


Figure 63. Non-Isochronous Non-control Endpoint Transmit Interrupt Handler

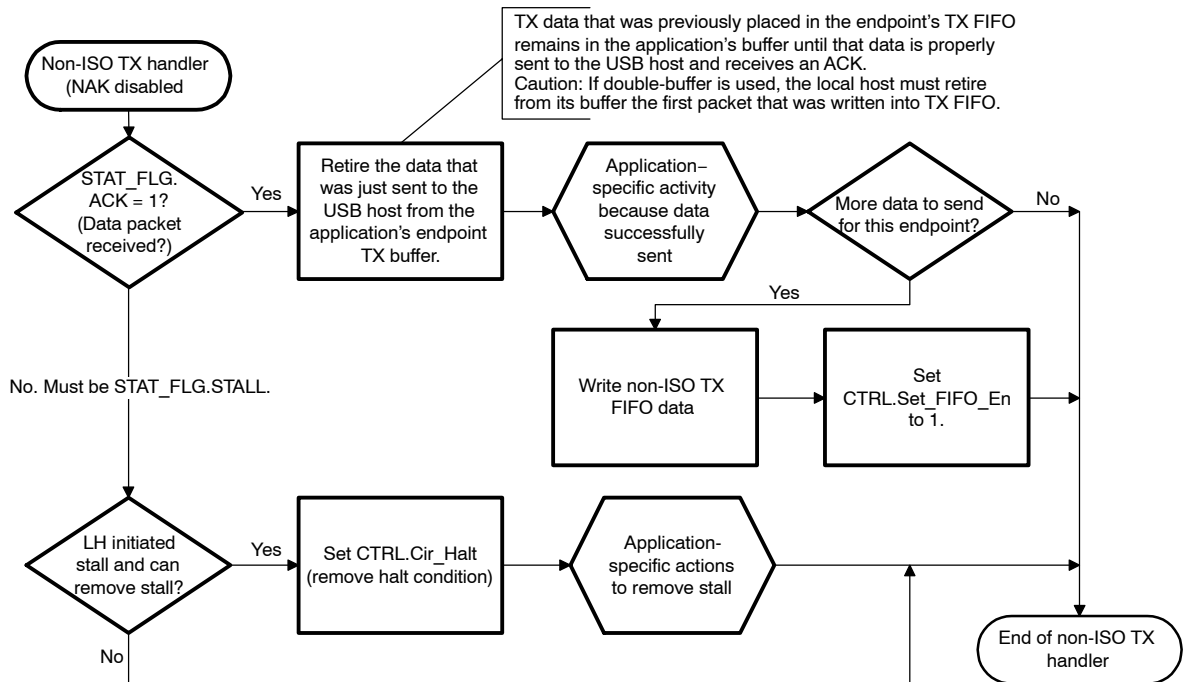


Figure 64. Write Non-Isochronous TX FIFO Data Flowchart

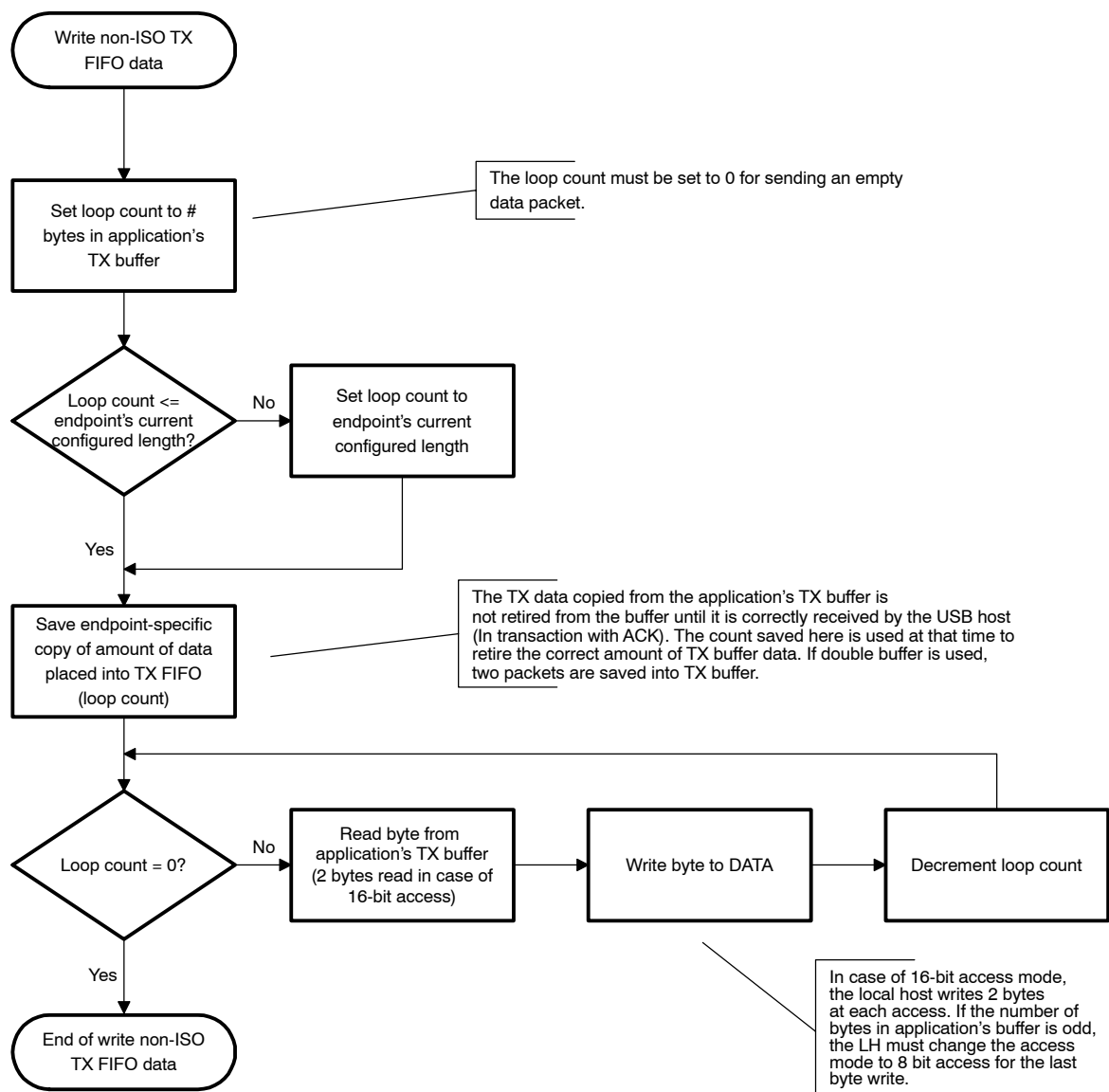


Figure 65. SOF Interrupt Handler Flowchart

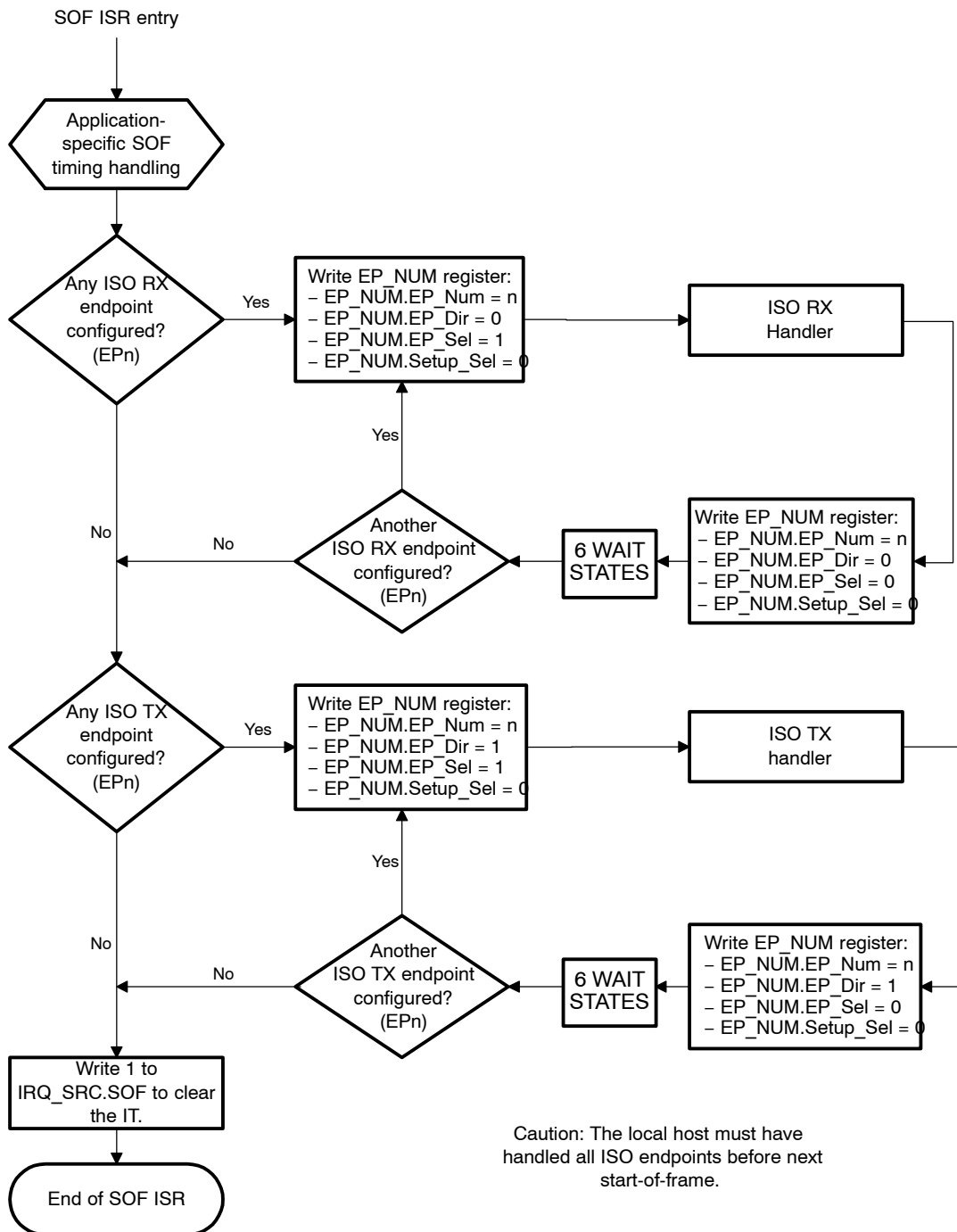


Figure 66. Read Isochronous RX FIFO Data Flowchart

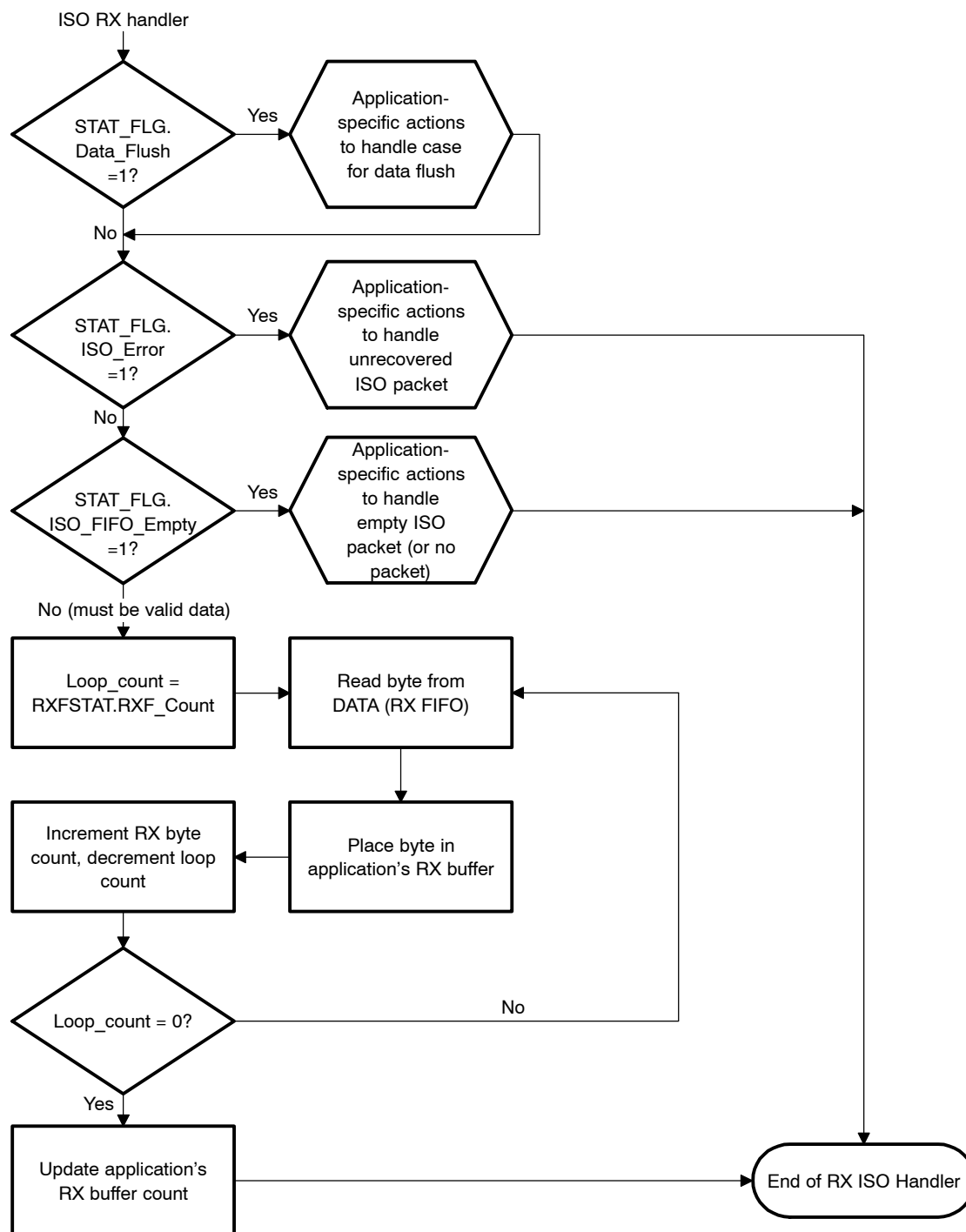
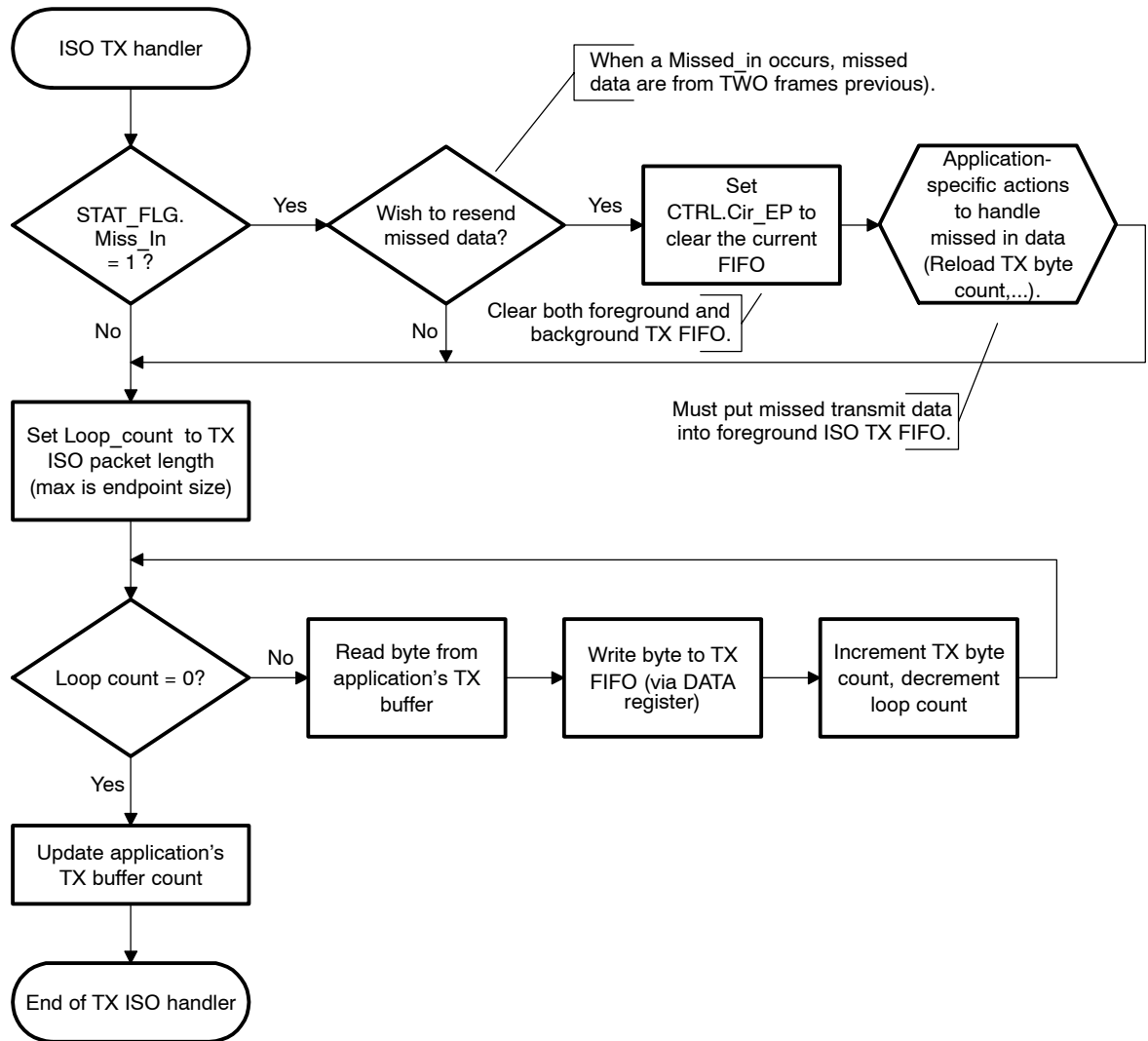


Figure 67. Write Isochronous TX FIFO Data Flowchart



16.14 Summary of USB-Related Interrupts

Table 95. USB Interrupt Type by Endpoint Type

Interrupt Type	General USB IRQs			EP-Specific IRQs		SOF	
	Setup (EP0)	Control (EP0) Out	Control (EP0) In	Other	Bulk or Interrupt Out	Bulk or Interrupt In	(Isochronous) SOF
Transaction ACKd		√	√		√	√	
Transaction NAKed (if enabled)		√	√		√	√	
Transaction STALLed		√	√		√	√	
Setup	√						
SOF							√
Device State Changed				√			
RX DMA EOT (non_ISO)				√			
RX DMA Trans Count (non_ISO)				√			
TX DMA Done (non_ISO)				√			

17 DMA Operation

The USB function module provides support for six DMA channels. Three receive DMA channels are reserved for OUT transfers (isochronous or non-isochronous) and three transmit DMA channels are reserved for IN transfers (isochronous or non-isochronous). It is not possible to operate DMA transactions on control EP0.

The local host must not access an endpoint used in a DMA transfer through EP_NUM, CTRL and STAT_FLG registers (in DMA, this remark applies after the local host has set the Set_FIFO_En bit to enable the RX DMA transfer). In particular, the local host must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

17.1 Receive DMA Channels Overview

Receive DMA channels are programmed via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA_n_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when a RX DMA request is active, through the register DATA_DMA. The RX FIFO accessed is that of the endpoint for which DMA request is active (only one RX DMA request active at a time).

17.2 Non-Isochronous OUT (USB HOST → LH) DMA Transactions

During non-isochronous transfers to a DMA-operated OUT endpoint, a request to the local host DMA controller is generated when data has been placed into the endpoint FIFO and must be read. Notice that ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-isochronous OUT transfers.

☐ End of transfer interrupt (the RX_n_EOT)

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full).

- When the number of received transactions has reached the programmed value in RXn_TC field, if Rxn_Stop bit has been set by the local host.

After an end of transfer interrupt, the local host must set the Set_FIFO_En for the endpoint to reenable the channel.

CAUTION

The local host must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

- Transactions count interrupt (the RXn_Cnt)

This interrupt performs watermark control. It can be used by the local host to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

CAUTION

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) has reached the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-isochronous endpoint. An RXn_Count interrupt is asserted even if RXn_Stop has been set: in that case, both RXn_Count and RXn_EOT interrupts are asserted.

The transaction count watermark is programmed in the RXDMA_n register.

Figure 68. Non-Isochronous RX DMA Transaction Example ($RX_TC = 2$)

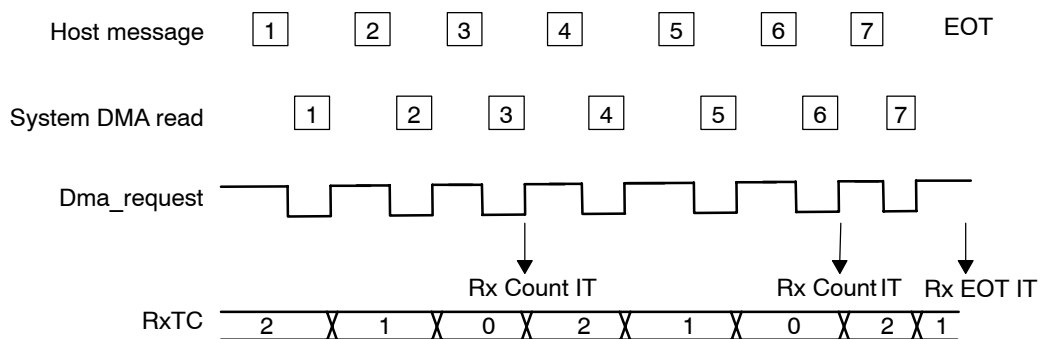


Figure 69. Non-Isochronous RX DMA Start Routine

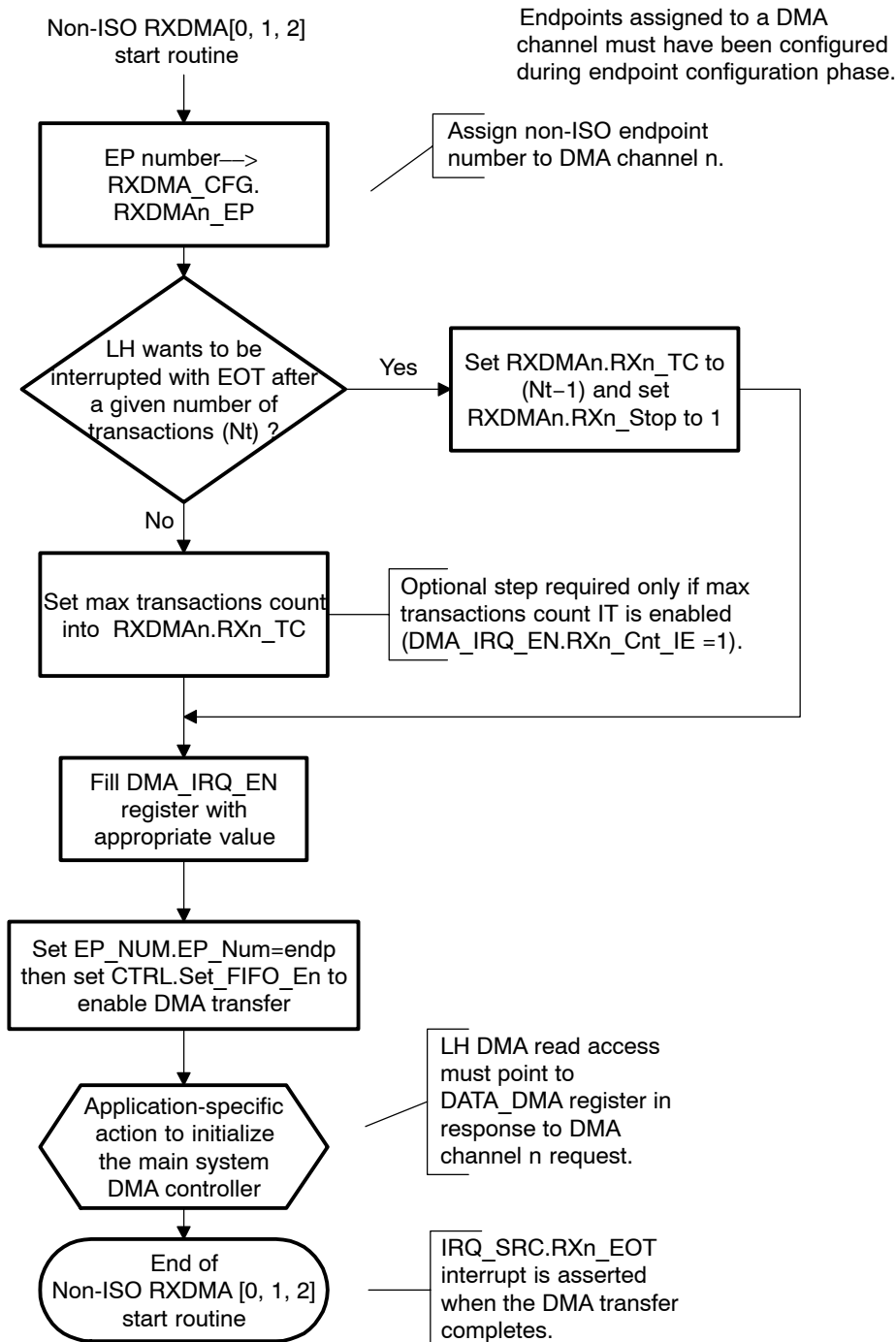


Figure 70. Non-Isochronous RX DMA EOT Interrupt Handler

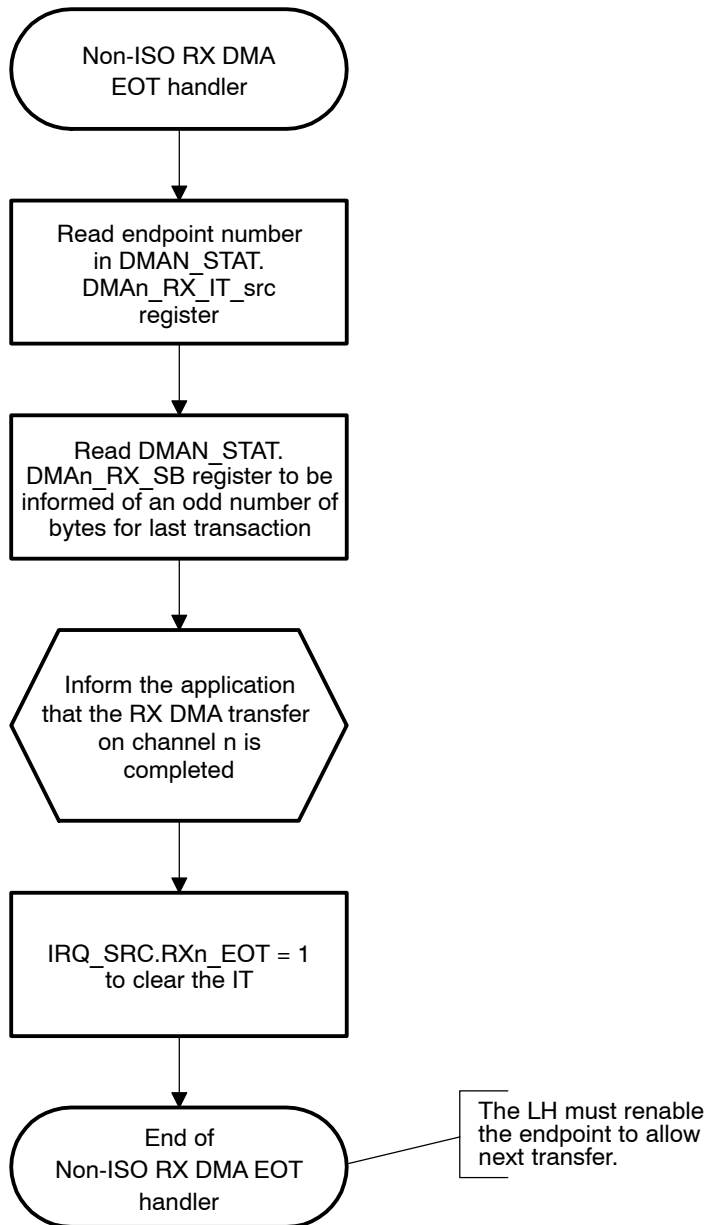
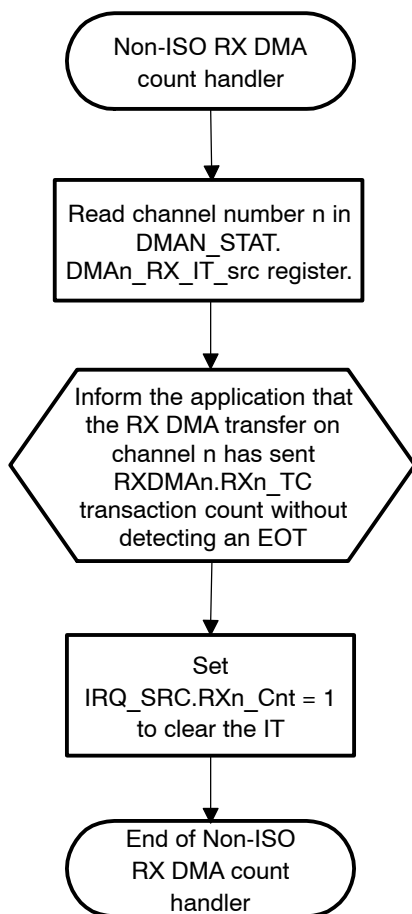


Figure 71. Non-Isochronous RX DMA Transaction Count Interrupt Handler



17.3 Isochronous OUT (USB HOST → LH) DMA Transactions

During isochronous transfers to a DMA-operated OUT endpoint, a request to the local host DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to isochronous OUT endpoints.

Figure 72. Isochronous RX DMA Transaction

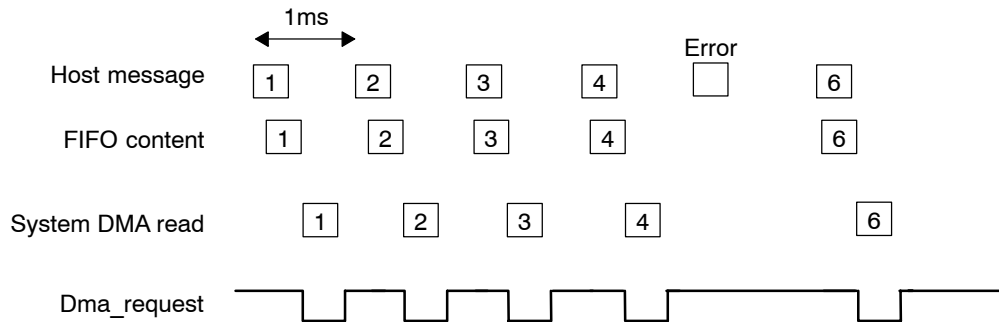
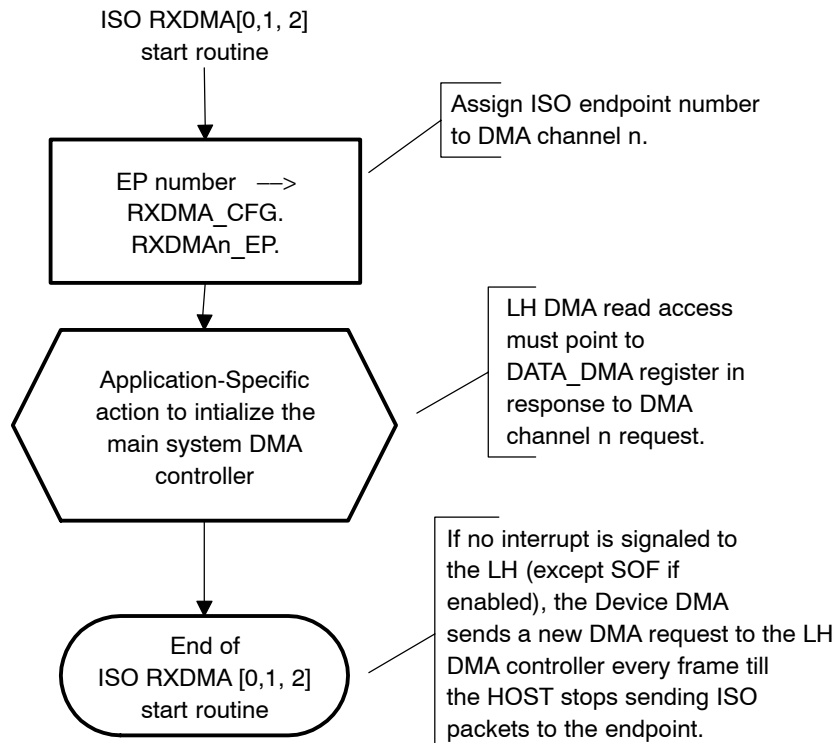


Figure 73. Isochronous RX DMA Start Routine



17.4 Transmit DMA Channels Overview

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning a nonzero value in TXDMA_n_EP (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for isochronous or non-isochronous endpoints. Transmitted data must be written into the DATA_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

17.5 Non-Isochronous IN (LH → USB HOST) DMA Transactions

Non-isochronous (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 75 and Figure 76 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-isochronous endpoints in the following manner. The transfer size counter (TX_n_TSC) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the local host main DMA controller is generated when the endpoint buffer is empty initially after that the START bit is set and then each time there is space free in TX FIFO for other TX packets to be written, until TX_n_TSC counts down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

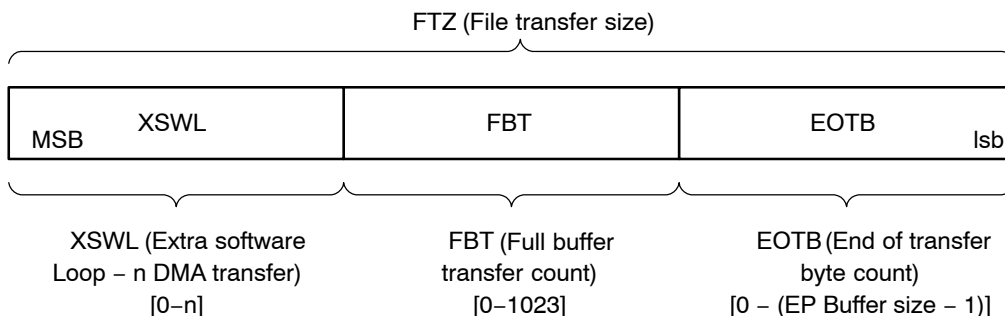
A DMA transmit transfer done interrupt is signaled to the local host after the last IN transaction completes successfully. This is after the START bit was set and after TX_n_TSC equals 0 for the selected DMA channel.

CAUTION
The local host must not initiate a new TX DMA transfer until it receives a TX_Done interrupt.

Small file transfer less than 1024 bytes can be achieved in a single DMA pass signaled by a single interrupt completion. File size equal or greater than 1024 bytes needs two or more DMA passes signaled by an interrupt completion after each pass.

The size of the file to transfer (FTZ) can be assimilated conceptually as a concatenation of three arguments as shown in Figure 74.

Figure 74. File Transfer Size



The flowchart in Figure 75 shows the necessary steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to local host via a DONE interrupt, as depicted in Figure 76. The start routine and the associated interrupt handler are tightly coupled.

An example would be 100603 bytes to transfer via 32 bytes IN bulk endpoint, which gives XSWL = 0x3, FBT = 0x47, EOTB = 0x1b. In this example, five passes of DMA transfer, signaled by five TXn_Done interrupts, are required, as follows:

- 1) EOT = 0, FBT = 0, loop = 3 32768 bytes transferred (1024 x 32 bytes)
- 2) EOT = 0, FBT = 0, loop = 2 32768 bytes
- 3) EOT = 0, FBT = 0, loop = 1 32768 bytes
- 4) EOT = 0, FBT = 0x47, loop = 0 2272 bytes (71 x 32 bytes)
- 5) EOT = 1, FBT = 0x1B, loop = 0 27 bytes

Figure 75. Non-Isochronous TX DMA DMA Start Routine

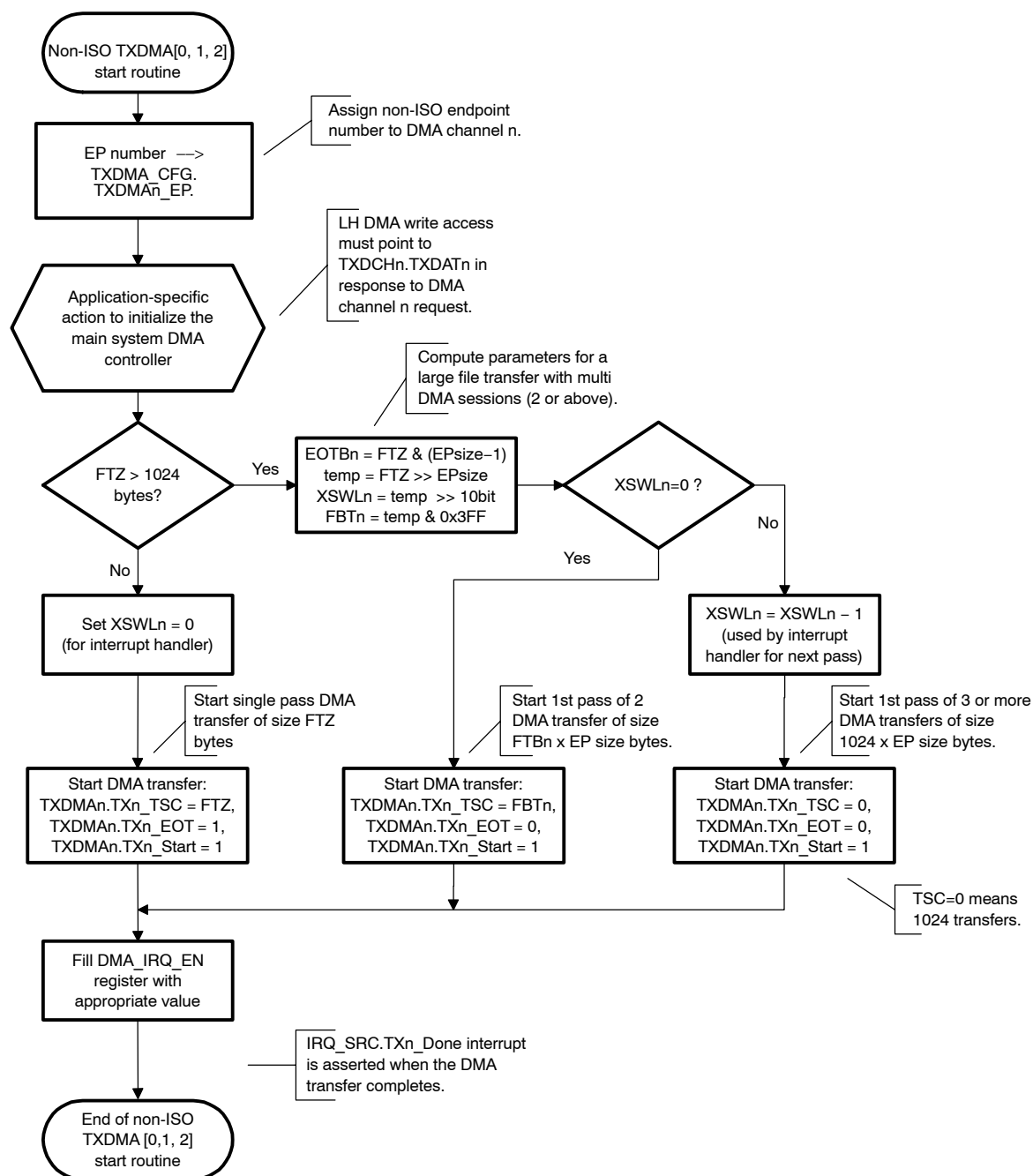
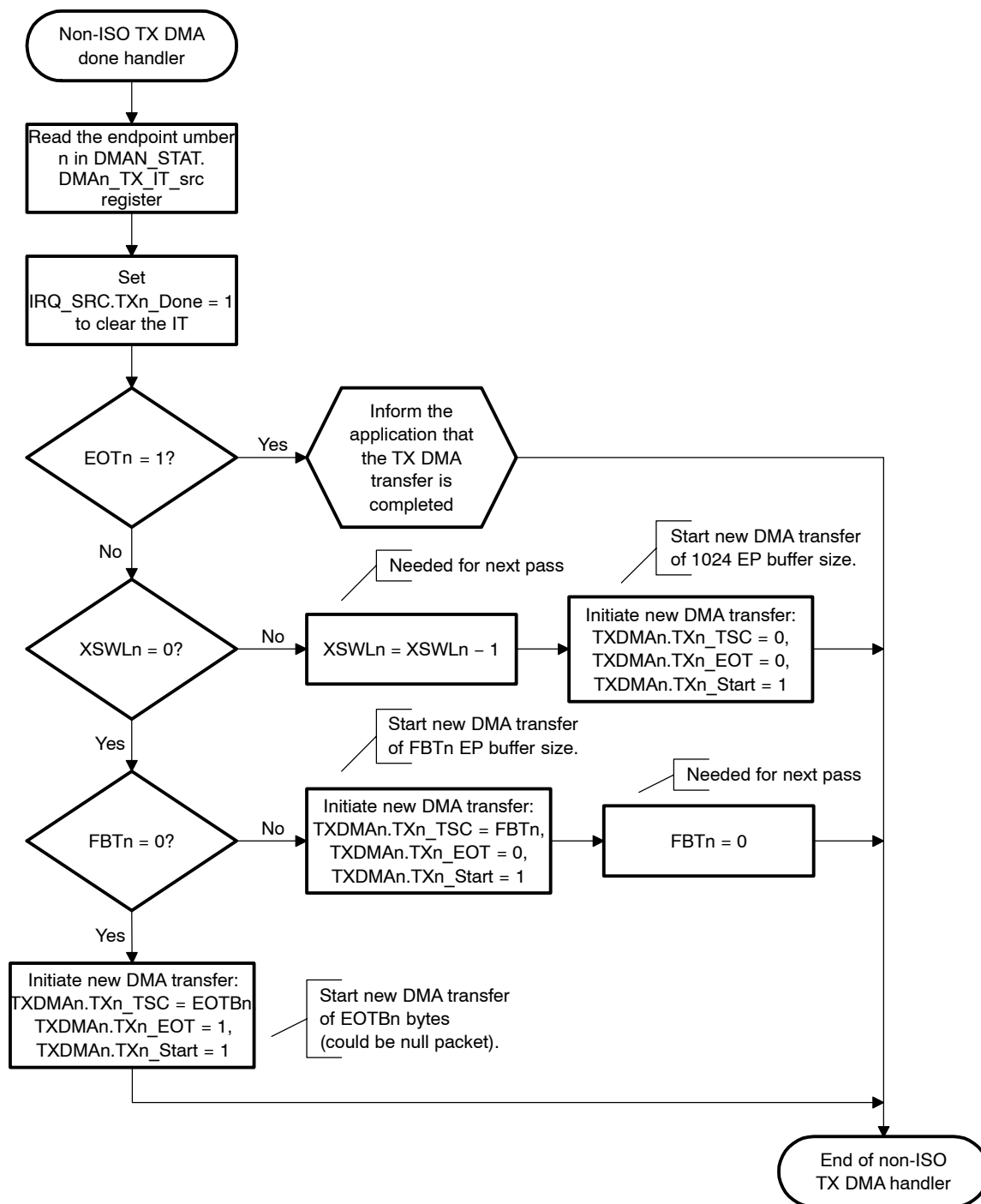


Figure 76. Non-Isochronous TX DMA Done Interrupt Handler



17.6 Isochronous IN (USB HOST → LH) DMA Transactions

For isochronous endpoints (Figure 77), the transfer size counter (TXn_TSC) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint; otherwise the results are unpredictable.

A request to the local host main DMA controller is generated when the endpoint buffer is empty initially; after that, the START bit is set and then set again after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches TXn_TSC value.

During isochronous transfers to a DMA-operated IN endpoint, a request to the local host system DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with DMA transfer.

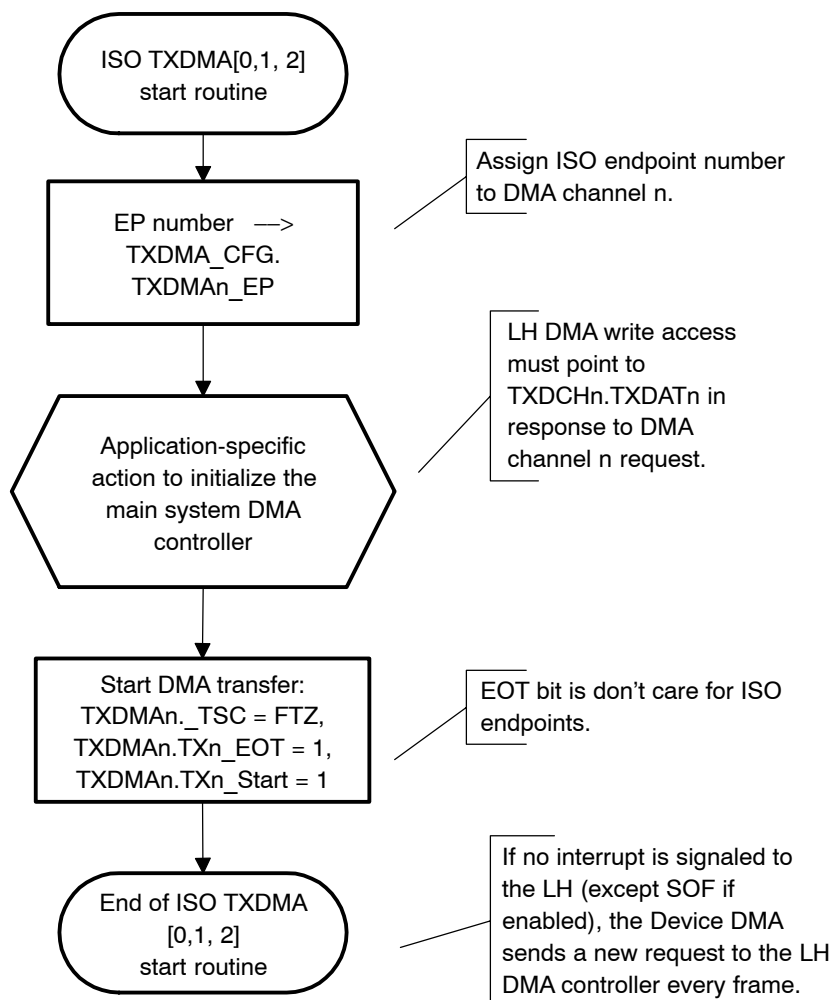
No interrupt is signaled to the local host during DMA operation to isochronous IN endpoints.

17.7 Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each start-of-frame if a TX DMA channel is configured for an isochronous endpoint; request must be serviced imperatively.

- ☐ For a USB TX DMA transfer, a DMA synchronization event is sent for each frame (DMA_CCR register FS field set to 1). Here the software decides the size of the USB transfer, so the DMA transfer is known in advance. The software must program the system DMA and the USB function to this transfer size.
- ☐ For a USB RX DMA transfer, a DMA synchronization event is sent for each element (DMA_CCR register FS field set to 0). Here the transfer size is unknown, and the DMA transfer is undetermined. Even if the USB function is programmed to transfer n elements, a RX DMA request is generated for each element.

Figure 77. Isochronous TX DMA Start Routine



17.8 Note on DMA Channel Deconfiguration

It is recommended that the local host wait for an EOT (RX) or a done (TX) interrupt before disabling the channel by writing a value 0 in TX/RXDMA_CFG register. However, if needed by the application, the local host can deselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The behavior is as follows:

□ For RX transfer:

If an RX DMA request is active for the endpoint when endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data has been read). When double buff-

ering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at deselection). An EOT interrupt is asserted if an end-of-transfer is detected.

If RX DMA request is not active when deselection occurs, the effect is immediate.

□ For TX transfer:

If request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data has been sent through an IN transaction (both buffers in case of double buffering with both buffers full). No TX_Done interrupt is asserted even if TSC bit value is 0 after the transaction.

If TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) have been sent through IN transaction(s). If TXDMA_n_TSC value is 0 at this point, no TX_Done interrupt is asserted. If TX_Done interrupt had already been asserted when endpoint is deselected, configuration is effective only after the TX_Done interrupt handling.

TX/RXDMA_n_EP reflects endpoint value until deconfiguration is effective. The local host must read this register to know if channel has been disabled or not yet. It must wait until read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generates an endpoint-specific interrupt (if non-transparent).

If the selected endpoint is isochronous, deconfiguration is effective after the TX/RX request has been serviced and the following isochronous transactions are handled at SOF interrupt through endpoint registers (EP_NUM and STAT_FLG).

18 Power Management

The flowchart in Figure 78 shows the values assigned to USB function signals concerned with power management in the functioning of the device state. These signals are:

☐ PUEN_O

Pullup enable signal, which always reflects the Pullup_En register bit.

☐ SHUTOFF_O

Power circuitry shutoff signal, controlled by the core and the SOFF_Dis bit.

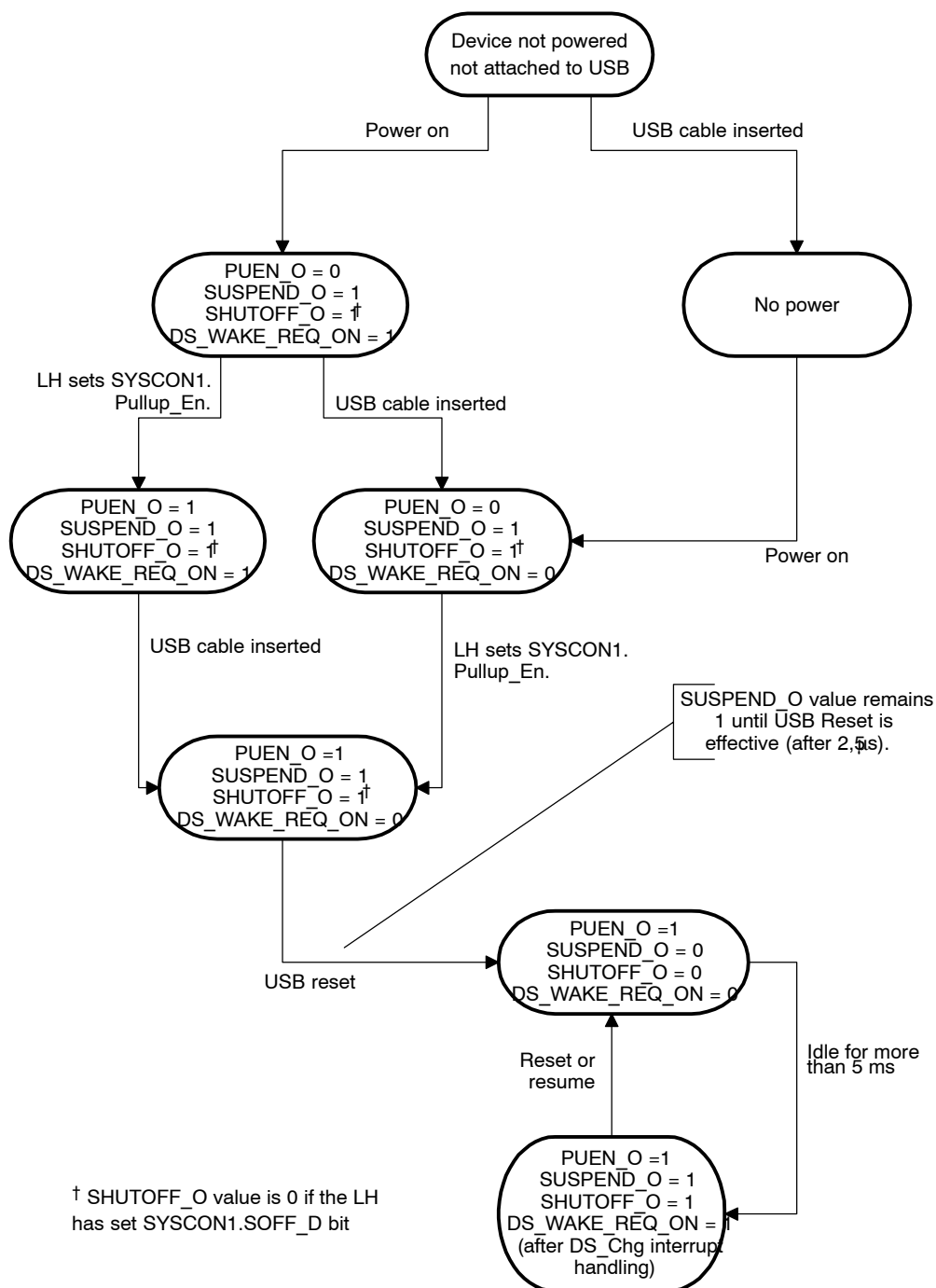
☐ DS_WAKE_REQ_ON

Deep sleep wake request, asserted low when interface clock is needed.

☐ SUSPEND_O

Suspend signal, asserted high when the device is in suspend mode.

Figure 78. Power Management Signal Values



19 Frame Adjustment Counter

The function of the frame adjustment counter module is to count the number of rising edges of one signal, the start of the frame interrupt of the USB function, during a programmable number of rising edges of a second signal, the transit-frame synchronization of the McBSP2. This count value can then be used by the system-level software to adjust the duration of the two time domains, with respect to each other, to reduce overflow and underflow. For example, if the data being transferred is audio data, this module can be part of a solution that helps to reduce pops and clicks.

20 Features

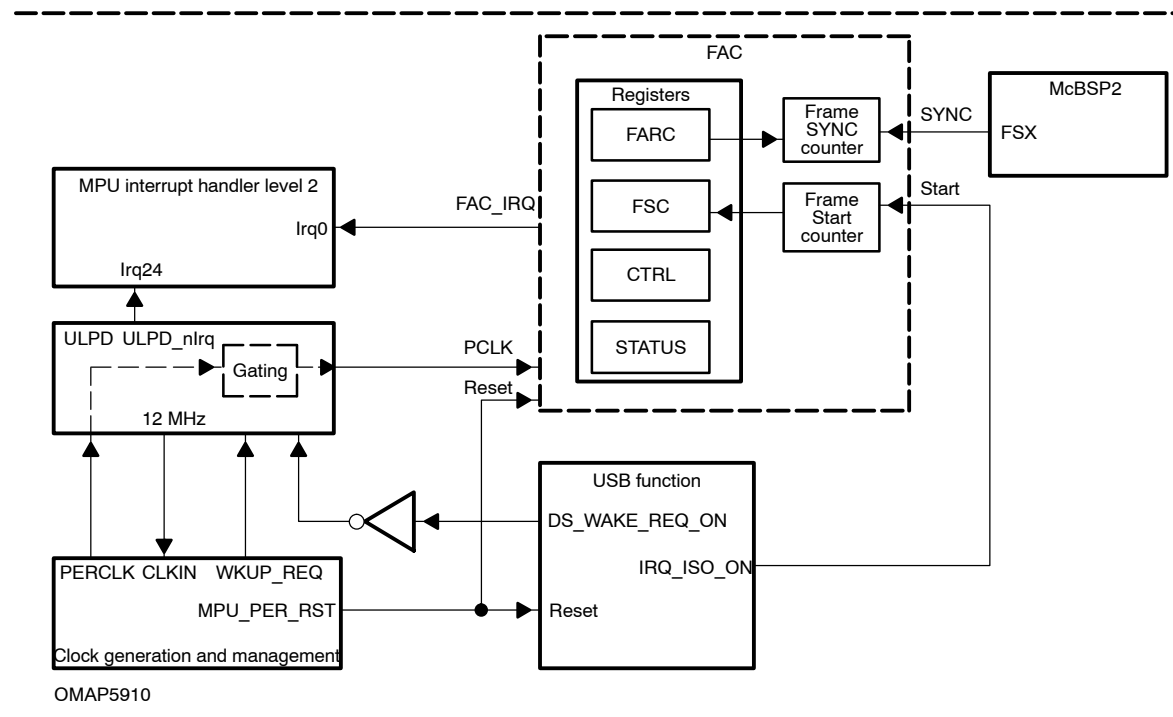
The frame adjustment counter (FAC) module consists of a frame synchronization capture pin and a frame start capture pin. A frame adjustment reference count register (FARC) is programmed with the number of frame-synchronization pulses over which the frame start pulses are to be counted. A frame start count register (FSC) is updated with the number of frame-start rising edges that occur during the programmable FARC period.

A control and configuration register (CTRL) allows the module to be put into either continuous or halt modes. In the continuous mode, the FSC register is periodically updated with a new value each time the FARC register value is met, and a new count is automatically initiated. In the halt mode, the FSC register is updated with a new value when the FARC register value is met, counting halts, and an interrupt is generated. In the halt mode, a new count is initiated upon software servicing the interrupt by reading the FSC register. The RUN bit in the control and configuration register can enable and disable the counters. If the RUN bit is set to zero, the counters are reset immediately even though the count is not finished. The software can use this bit as a software reset. Additionally, there is a FAC status register (STATUS) containing a FSC_FULL bit that indicates to the system software if the FSC has been read subsequent to the last FSC update.

The main FAC features are:

- ☐ Frame-synchronization capture pin (SYNC)
- ☐ Frame-start capture pin (START)
- ☐ Programmable frame-adjustment reference count register (FARC)
- ☐ Read-only frame-start count register (FSC)
- ☐ Interrupt-generation logic
- ☐ Configuration and control register (CTRL)
- ☐ Status register (STATUS)

Figure 79. FAC Top-Level Diagram



21 Synchronization and Counter Control

Because the frame-start and frame-synchronization signals are from different time domains, the FAC module synchronizes these two signals to the system clock domain and uses the synchronized signals as the count enables. The actual counters for the frame synchronization and frame start are clocked by the system clock.

The synchronization mechanism is based on the assumption that the system clock is running at least eight times faster than the frame synchronization and frame start. Figure 80 and Figure 81 show the synchronization logic and the counter hookup.

Figure 80. FAC-Module Counters and Clock Synchronization

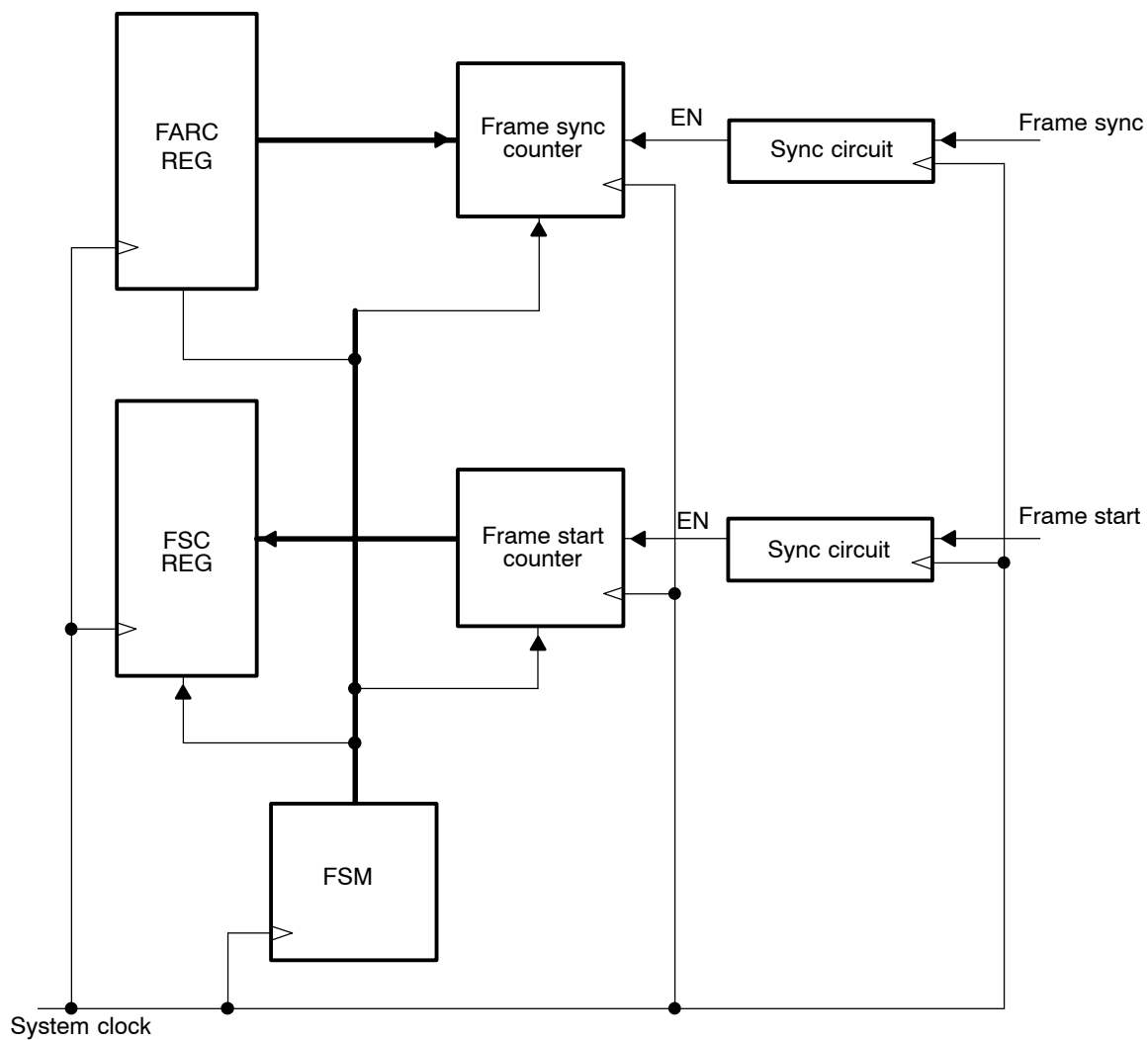


Figure 81. Synchronization Circuit for Frame-Synchronization and Frame-Start Signals

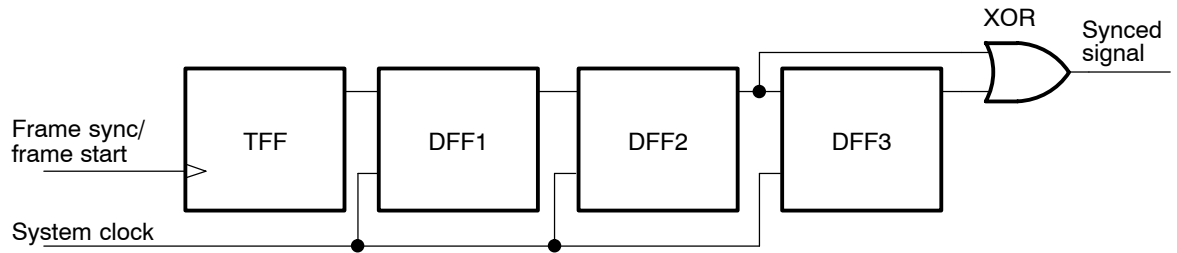
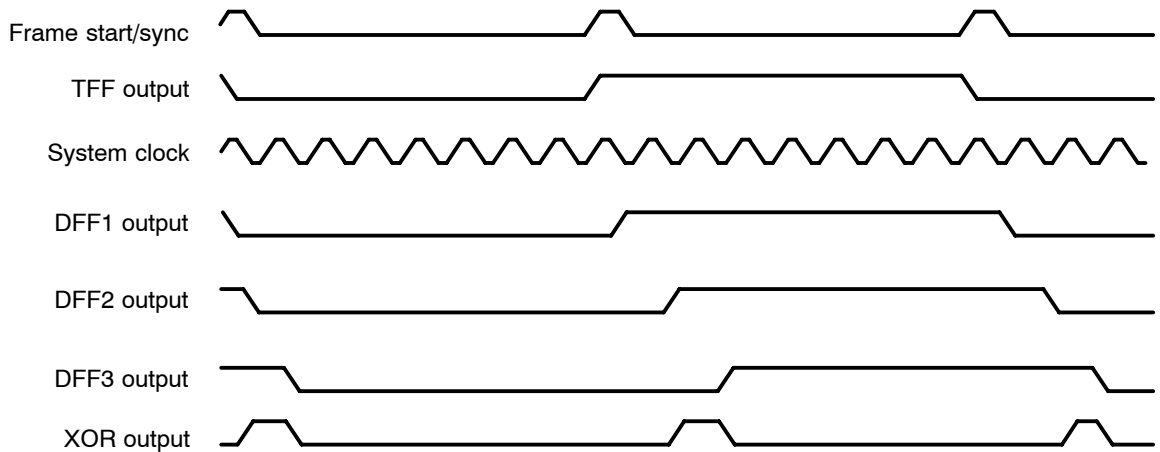


Figure 82 shows the actual waveforms of the output of each flip-flop and the XOR output.

Figure 82. Synchronization Circuit Waveforms



22 FAC Interrupt

The FAC generates 1 interrupt, FAC_IRQ (in halt mode when the FARC value is met), connected to the MPU level 2 interrupt handler, line 0 (level-sensitive)

23 FAC Clocks and Reset

The FAC works with a clock (PCLK), which is provided by the ULPD from a request generated by the USB function (DS_WAKE_REQ_ON).

The DS_WAKE_REQ_ON request does not wake-up the system itself.

The ULPD module uses this request to generate an interrupt (ULPD_nlrq) to the MPU, which wakes up the system via its wake-up request (WKUP_REQ).

Once the system is awakened (i.e., 12-MHz provided to the MPU), the MPU-programmable-peripheral clock (MPUPER_CK) is used as the source clock for the FAC clock (for more details, see SPRU678, *OMAP5910 Clock Generation and System Reset Management*).

The MPU TIPB reset (MPU_PER_RST) resets the FAC.

24 Software Interface

Table 96 lists the FAC registers. Table 97 through Table 100 describe the register bits.

Table 96. FAC Registers

Register	Description	Type	Address
FARC	Frame adjustment reference count	R/W	FFFB:A800
FSC	Frame start count	R	FFFB:A804
CTRL	Control and configuration	R/W	FFFB:A808
STATUS	Status	R	FFFB:A80C
SYNC CNT	Frame synchronization counter	R	FFFB:A810
START CNT	Frame start counter	R	FFFB:A814

The FAC module is a word16 module with 32-bit aligned addresses.

The FARC is programmed with the number of frame-synchronization counts over which the frame-start pulses are counted. This is a 16-bit programmable fixed reference in the range of 0-65536. A value of zero disables the count operation.

Table 97. Frame-Adjustment-Reference-Count Register (FARC)

Bits	Field	Description	Reset Value
15–0	FARC	16-bit value in the range 0-65536: 0 = disable counting	0

The FSC is a 16-bit read-only register that contains the number of frame-start rising edges that occur during the programmable FARC period. The frame-start counting can be in two modes. When the CNT bit in the CTRL register is set to one, the counting is in the continuous mode and this register is periodically updated (every time the frame-adjustment reference count is met) with the new count value. If the CNT bit is zero, the counting is in the halt mode. The FSC register is updated when the frame-adjustment reference count is met, and the counting halts until the software reads the FSC register.

A level-sensitive interrupt can be generated to indicate that the frame-start counting is finished, and the FSC register is loaded with a new count value.

The interrupt is controlled by the INT_ENABLE bit in the CTRL register. If this bit is set to one, an interrupt is generated when the FSC register is updated. Since the interrupt is level-sensitive, the interrupt signal is kept low until the software reads the FSC register or the RUN bit in the control register is set to zero. When the FSC register is read or the RUN bit in the control register is set to zero, the interrupt signal is reset to one. When the INT_ENABLE bit is set to zero, no interrupt is generated. The interrupt can be enabled or disabled for both the continuous and halt modes.

Table 98. *Frame-Start-Count Register (FSC)*

Bits	Field	Description	Reset Value
15–0	FS	16-bit value	0

The CTRL is a read/write register used to configure the module. The RUN bit is used to enable the frame-start counter. If this bit is set to 0, the frame-start counting is disabled immediately. The software can use this bit as a software reset for the FAC module by setting the RUN bit to zero. When the RUN bit is set to zero, the frame-start counter, the frame-synchronization counter and the FSC register are reset to zero. The software reset also clears the STATUS register FSC_FULL bit to zero. If an interrupt has been generated and the FAC module is waiting for an FSC register read, a software reset puts the counter control back to idle state. This means that after the software reset the counter starts counting again, regardless of whether the FSC register has been read or not.

Table 99. FAC-Control-and-Configuration Register (CTRL)

Bits	Field	Description	Reset Value
15–3	Reserved		0
2	INT_ENABLE	When this bit is set to a 1, an interrupt is generated when the FSC is updated. If this bit is set to a 0, no interrupt is generated. The INT_ENABLE bit is independent of the CNT bit. The interrupt can be enabled or disabled in either the continuous mode or halt mode.	0
1	RUN	Enables operation of the counter. When this bit is set to zero, the frame-start counter, the frame-synchronization counter, and the FSC are reset to zero. Any pending interrupt also is cleared when the RUN bit is set to zero.	0
0	CNT	1: Continuous mode: Periodically updates the FSC value each time the frame-adjustment reference count is met. 0: Halt mode: Updates the FSC value when the frame-adjustment reference count is met and halts operation until the FSC is read.	0

The status register (STATUS) is a read/write register that contains an interrupt-status bit.

Table 100. FAC-Status Register (STATUS)

Bits	Field	Description	Reset Value
15–1	Reserved		0
0	FSC_FULL	This bit is set to a 1 when the FSC is updated. This bit is set back to a 0 when the FSC has been read or the RUN bit in control and configuration register is zero.	0

A

ACK, See acknowledged 201
acknowledged, transaction
 OUT 201
 USB IN 206

C

cache, coherency
 in data buffers 106
 in OHCI data structures 106
clock, USB function 147
configuration, DMA channel 277
control
 read transfer
 autodecoded 219
 non-autodecoded 222
 transfer
 data stage length 227
 endpoint 0 214
 write transfer
 autodecoded 218
 non-autodecoded 219
 required local host actions for
 non-autodecoded 221

D

D+ pulldown, USB function controller 144
D+ pullup enable, USB function controller 142
D- pulldown, USB function controller 144
data, stage length, control transfer 227
detection, USB 148
DMA
 channel, deconfiguration 277

operation, USB 265
receive, channels (USB) 265
request
 limit on active requests (USB) 276
 USB function 147
USB
 isochronous IN transactions 276
 isochronous OUT transactions 271
 non-isochronous IN transactions 272
 non-isochronous OUT transactions 265
 transmit channels 272

E

endianism
 and OHCI data buffers 114
 and USB host controller access to system
 memory 113
endpoint 0
 control, transfer 214
 interrupt handler
 receive 242
 transmit 242
error, conditions
 autodecoded control read 219
 autodecoded control write 218
 isochronous IN 213
 isochronous IN endpoint FIFO 213
 isochronous OUT 210
 isochronous OUT endpoint FIFO 211
 non-autodecoded control read transfer 224
 non-autodecoded control write transfer 222
 non-isochronous IN 207
 non-isochronous IN endpoint FIFO 208
 non-isochronous, non-control OUT 202
 non-isochronous, non-control OUT endpoint
 FIFO 203

F

frame, synchronization, MPU public
peripherals 283

H

handshaking
 autodecoded control write 218
 control read transfer, autodecoded 219
 isochronous IN endpoint 213
 isochronous OUT endpoint 210
 non-autodecoded control read 223
 non-autodecoded control write transfers 221
 non-isochronous IN endpoint 205
 USB, non-isochronous, non-control OUT
 endpoint 200
hardware, reset, USB host controller 140
host, controller
 clock control 139
 connectivity with USB transceivers 67
 OHCI reset 140
 power management 140
 USB 21
 USB access to system memory 103
 USB hardware reset 140
 USB reset 139

I

I/O, configuration, USB function 145
initialization, USB 229
interrupt
 handler
 endpoint 0 receive 242
 endpoint 0 transmit 242
 *non-isochronous, non-control IN endpoint
 transmit* 256
 *non-isochronous, non-control OUT endpoint
 receive* 256
 SOF 256
 USB reset 251
 USB resume 251
 USB setup 237
 USB suspend 251
MMU, local bus 66
OHCI 65

parsing
 non-isochronous endpoint-specific 251
 USB 237
sources, USB host controller 65
USB
 function 145
 operation 236
 summary 264

L

local bus
 addressing, and data structure pointers 107
 MMU
 interrupts 66
 programming 137
 overview 115
 virtual addressing, USB 104
local host, required actions, non-autodecoded
control write transfers 221

M

MMU
 interrupts, local bus 66
 programming, USB local bus 137
MPU, public peripherals
 frame adjustment counter 281
 frame synchronization 283
multiplexing, conflicts 102

N

NAK, See non-acknowledged 202
non-acknowledged, transaction
 OUT 202
 USB IN 206
null pointers 113

O

OHCI
 controller, overview 24
 data buffers, and endianism 114
 differences from OMAP1510 24
 interrupts 65
 null pointers 113
 OMAP1510 implementation 25
 reset, USB host controller 140
OMAP1510, transceiverless link logic 70

P

- packet error
 - USB IN 208
 - USB OUT 203
- parsing
 - interrupts, non-isochronous
 - endpoint-specific 251
 - USB, interrupt 237
- pin multiplexing, USB 67
- port, passthrough mode, USB 143
- power, management
 - USB 279
 - USB host controller 140
- public peripherals, MPU
 - frame adjustment counter 281
 - frame synchronization 283

R

- request, USB, autodecoded vs.
 - non-autodecoded 224
- reset
 - hardware, USB host controller 140
 - interrupt handler, USB 251
 - OHCI, USB host controller 140
 - USB
 - function* 147
 - host controller* 139
- resume, interrupt handler, USB 251

S

- sequence bit error, USB OUT 203
- signal sharing, ARM_BOOT 144
- SOF, interrupt handler, USB 256
- software, USB disconnect 150
- stalled, transaction
 - USB IN 207
 - USB OUT 202
- states
 - attached handler 250
 - changed handler 247
 - unattached handler 250
- suspend, interrupt handler, USB 251

- synchronization, frame, MPU public
 - peripherals 283

T

- transaction
 - acknowledged
 - USB IN 206
 - USB OUT 201
 - non-acknowledged
 - USB IN 206
 - USB OUT 202
 - stalled
 - USB IN 207
 - USB OUT 202
 - USB
 - isochronous IN* 212
 - isochronous OUT* 209
 - non-isochronous IN* 203
 - non-isochronous, non-setup OUT* 199
 - overview* 199
- transceiverless, connection, OMAP1510 link 70
- transfer
 - autodecoded control read, USB 219
 - autodecoded control write, USB 218
 - control
 - data stage length* 227
 - endpoint 0* 214
 - non-autodecoded read* 222
 - non-autodecoded write* 219
 - required local host actions for*
 - non-autodecoded write* 221
 - USB, preparation 233
- transient suppression, USB connectors 142

U

- ULPD, initializing, to generate 48-MHz clock 139
- USB
 - connectors, transient suppression 142
 - detection 148
 - DMA
 - isochronous IN transactions* 276
 - isochronous OUT transactions* 271
 - non-isochronous IN transactions* 272
 - non-isochronous OUT* 265
 - operation* 265
 - receive channels* 265
 - transmit channels* 272

- DMA requests, limit on active requests 276
- function
 - clocks* 147
 - DMA requests* 147
 - I/O configuration* 145
 - interrupts* 145
 - module* 145
 - reset* 147
- function controller
 - connectivity with USB transceivers* 68
 - D+ pulldown* 144
 - D+ pullup enable* 142
 - D- pulldown* 144
 - VBUS monitoring* 142
- host controller 21
 - interrupt sources* 65
 - access to system memory* 103
 - clock control* 139
 - hardware reset* 140
 - null pointers* 113
 - OHCI reset* 140
 - power management* 140
 - reset* 139
- host controller access to system memory, and
 - endianism 113
- interrupt
 - operation* 236
 - parsing* 237
 - parsing non-isochronous endpoint-specific* 251
 - summary* 264
- interrupt handler
 - setup* 237
 - SOF* 256
- initialization 229
- local bus, virtual addressing 104
- local host, MMU programming 137
- pin multiplexing 67
- port passthrough mode 143
- power management 279
- requests, autodecoded vs.
 - non-autodecoded 224
- reset, interrupt handler 251
- resume, interrupt handler 251
- signal multiplexing 73
- software disconnect 150
- suspend, interrupt handler 251
- transactions
 - isochronous IN* 212
 - isochronous OUT* 209
 - non-isochronous IN* 203
 - Non-isochronous, non-setup OUT* 199
 - overview* 199
- type A host, VBUS power switching 142

V

VBUS

- monitoring, USB function controller 142
- power switching, for USB type A host 142