TMS320TCI6616

Communications Infrastructure KeyStone SoC

Silicon Revisions 1.0, 1.0A, 2.0

Silicon Errata

Literature Number: SPRZ331G
May 2014

Release History

Release	Date	Description/Comments
SPRZ331G	May 2014	• Updated Advisory 9 - HyperLink Temporary Blocking Issue (Page 24) • Added Advisory 43 - False DDR3 Write ECC Error Reported Under Certain Conditions (Page 77)
SPRZ331F	December 2013	• Added Step 8 in workaround 2 of Adivosry 32 (Page 57) • Added Usage Note 33: Incorrect Output from TAC for Certain Channels When Configuring TAC_Gn_DATA_FETCH and TAC_Gn_HEAD_FETCH Registers with Values Having a Difference of 8 or 16 Usage Note (Page 113)
SPRZ331E	July 2013	• Updated the description of Usage Note 1: AIF2 Protocol Encoder Registers Usage Note (Page 76) • Added Usage Note 32: DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note (Page 110)
SPRZ331D	June 2013	• Added Advisory 39: Single MFENCE Issue (Page 63) • Added Advisory 40: Read Exception and Data Corruption Issue (Page 65) • Added Advisory 41: Incorrect Output from TAC for HS-PDSCH and P-CCPCH Channels in Certain Spreader Allocation Sequences (Page 69) • Added Advisory 42: Incorrect output from TAC for a Fetching Spreader on a Collision between Input Header's Fetch Write to the Spreader's SPM and Software's Write to SPM Issue (Page 71) • Added Usage Note 27 Revised PLL Programming Sequence Usage Note (Page 101) • Added Usage Note 28 Core Wake Up on RESETz Usage Note (Page 102) • Added Usage Note 29 BSDL Testing Support Usage Note (Page 103) • Added Usage Note 30: AIF2 CPRI FastC&M Restrictions and Usage Note (Page 105) • Added Usage Note 31: Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note (Page 107)
SPRZ331C	May 2012	• Added Usage Note 26: IDMA1 Performance Limitation Usage Note (Page 89) • Added Advisory 38: DDR3 Incremental Write Leveling Issue (Page 62) • Added Usage Note 25: CAS Write Latency at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note (Page 88) • Added Usage Note 24: PLL Boot Configuration Setting and DEVSPEED Register Usage Note (Page 87) • Added Usage Note 23: Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note (Page 86) • Added Workaround 3 (Page 28) • Added Usage Note 22: The Clock Input To PASS Usage Note (Page 85) • Added Usage Note 21: Sticky Bits in PCIe MMRs Usage Note (Page 84) • Added Usage Note 20: PCIe BAR5 Window Sixe Configuration in Boot ROM (Page 83)
SPRZ331B	January 2012	• Added Advisory 37: Power Domains Hang During Powering Up at The Same Time As a RESETZ (Hard Reset) Received Issue (Page 61) • Added Advisory 36: System Reset Operation Disconnects SoC from CCS Issue (Page 60) • Added Advisory 35: SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue (Page 57) • Added Advisory 34: AIF2 Line Code Violation Error from B8 SerDes to B4 SerDes Lanes Cross Connection Issue (Page 56) • Added Advisory 33: Data Corruption on C66x CorePAC Initiated PCIe MMRs Read in Big Endian Mode Issue (Page 55) • Added Advisory 32: L2 Cache Corruption During Block and Global Coherence Operations Issue (Page 53) • Added Advisory 31: AIF2 CPRI 8x UL Peak Bandwidth Issue (Page 52) • Usage Note19: AIF2 LTE 3MHz and 1.4MHz Support Usage Note (Page 81)

Contents

List of Figures

List of Tables

TMS320TCI6616 Communications Infrastructure KeyStone SoC Silicon Revisions 1.0, 1.0A, 2.0

Introduction

This document describes the silicon updates to the functional specifications for the TMS320TCI6616 fixed-/floating-point digital signal processor. See the device-specific data manual, *TMS320TCI6616 Communications Infrastructure Keystone SoC* data manual (literature number SPRS624) for more information.

Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320TCI6616CYP). Texas Instruments recommends one of two possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully-qualified production device

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing

TMDS Fully-qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

Developmental product is intended for internal evaluation purposes.

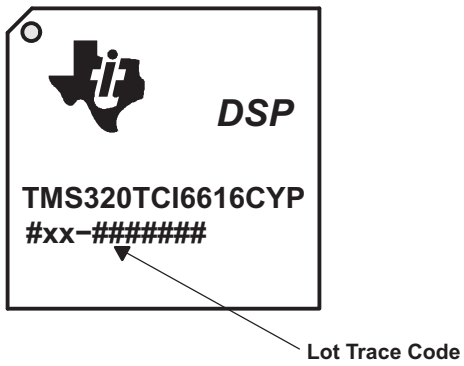
TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

Package Symbolization and Revision Identification

The device revision can be determined by the lot trace code marked on the top of the package. The location of the lot trace code for the CYP package is shown in Figure 1. Figure 1 also shows an example of TCI6616 package symbolization.

Figure 1 Lot Trace Code Example for TMS320TCI6616 (CYP Package)



Silicon revision correlates to the lot trace code marked on the package. This code is of the format #xx-#######. If xx is ***10***, then the silicon is revision 1.0. Table 1 lists the silicon revisions associated with each lot trace code for the TCI6616 devices.

Table 1 Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision
10A	1.0A	Silicon revision 1.0A
20	2.0	Silicon revision 2.0
End of Table 1		

The TCI6616 device contains multiple read-only register fields that report revision values. The JTAG ID (JTAGID), C66x CorePac Revision ID (MM_REVID) and CPU Control Status (CSR) registers allow the customer to read the current device and CPU level revision of the TCI6616.

The JTAG ID register (JTAGID) is a read-only register that identifies to the customer the JTAG/Device ID. Usually, the value in the VARIANT field of the JTAG ID Register changes based on the revision of the silicon being used. However, for this device the entire JTAG ID Register may change based on the silicon revision.

The C66x CorePac Revision ID register (MM_REVID) is a read-only register that identifies to the customer the revision of the C66x CorePac. The value in the VERSION field of the C66x CorePac Revision ID Register changes based on the version of the C66x CorePac implemented on the device. More details on the C66x CorePac Revision ID register can be found in the TMS320TCI6616 *Communications Infrastructure Keystone SoC* data manual (literature number SPRS624).

The CPU Control Status Register (CSR) contains a read-only REVISION_ID field that identifies to the customer the revision of the CPU being used. More information about the CPU Control Status Register can be found in the *C66x CPU and Instruction Set Reference Guide* (literature number SPRUGH7).

Table 2 shows the contents of the CPU Control Status Register CPU_ID and REVISION_ID fields, C66x CorePac MM_REVID Register REVISION field, and the JTAGID register for each silicon revision of the TCI6616 device.

Table 2 Silicon Revision Variables

Silicon Revision	CPU CSR Register	C66x CorePac MM_REVID Register	TCI6616 JTAGID Register
1.0	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 1.0 MM_REVID[REVISION] = 0000h	JTAGID = 0009D02Fh
1.0A	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 1.0 MM_REVID[REVISION] = 0000h	JTAGID = 0B94102Fh
2.0	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 2.0 MM_REVID[REVISION] = 0001h	JTAGID = 1009D02Fh
End of Table 2			

Note—Because silicon revision 1.0A uses the same JTAGID value as another device, the user must additionally read bit-10 of the register at address 02350678h to identify the device. That bit will be equal to **1** for the TCI6616 device.

For more information on the JTAG ID and CorePac Revision ID Registers, see the TMS320TCI6616 *Communications Infrastructure Keystone SoC* data manual (literature number SPRS624).

Silicon Updates

Table 3 lists the silicon updates applicable to each silicon revision. For details on each advisory, click on the link below.

Table 3 Silicon Revision Updates (Part 1 of 3)

Silicon Update Advisory	See	Applies To Silicon Revision		
		1.0	1.0A	2.0
SRIO Deadlock Issue	Advisory 1	X		
Queue Manager and Infrastructure Packet DMA Stall Issue	Advisory 2	X		
SPI Boot Issue	Advisory 3	X		
Delayed SRIO PLL Lock Results In Incomplete SRIO Configuration Issue	Advisory 4	X		
Potential E-Fuse Autoload Errors During Device Initialization Issue	Advisory 5	X		
SRIO Potential Message Header Corruption Issue	Advisory 6	X		
Boot ROM CM Setting For SGMII Issue	Advisory 7	X		
Main PLL Clock Divider Not Properly Configured Issue	Advisory 8	X		
HyperLink Temporary Blocking Issue	Advisory 9	X	X	X
SRIO Reset Isolation Not Functional Issue	Advisory 10	X	X	
SRIO Type 9 / Type 11 RX Message Uses Wrong Context Issue	Advisory 11	X		
Total Number of SRIO RXU Contexts Decreases Due to Timeout or Teardown Issue	Advisory 12	X		
DDR3 Automatic Leveling Issue	Advisory 13	X	X	X
Potential L2 Cache Corruption During Block Coherence Operations Issue	Advisory 14	X	X	
Multiple PLLs May Not Lock After Power-on Reset Issue	Advisory 15	X		
Queue Manager External Linking RAM Location Issue	Advisory 16	X	X	X
AIF2 AIF_TX_MONO_MODE Not Compatible With Other Navigator Modules Issue	Advisory 17	X	X	
DDR3 Excessive Refresh Issue	Advisory 18	X	X	X
SRIO Register Aliasing Issue	Advisory 19	X	X	
AIF2 CPRI LTE Ingress Antenna Carrier Packing Issue	Advisory 20	X	X	
SRIO Packet Forwarding Issue	Advisory 21	X	X	
SRIO Messaging in Highly Oversubscribed System Issue	Advisory 22	X	X	
SRIO Direct I/O NREAD Data Corruption Issue	Advisory 23	X	X	
TAC P-CCPCH QPSK Symbol Data Mode with STTD Issue	Advisory 24	X	X	X
AIF2 GSM T1, T2, T3 Mis-calculation Issue	Advisory 25	X	X	
AIF2 Multiple DBMR Operation Issue	Advisory 26	X	X	
AIF2 SerDes Comma Alignment Failure Issue	Advisory 27	X	X	
SRIO Control Symbols Are Sent More Often Than Required Issue	Advisory 28	X	X	X
Corruption of Control Characters in SRIO Line Loopback Mode Issue	Advisory 29	X	X	X
SerDes Transit Signals Pass ESD-CDM up to +/-150V Issue	Advisory 30	X	X	X
AIF2 CPRI 8x UL Peak Bandwidth Issue	Advisory 31	X	X	X
L2 Cache Corruption due to Block and Global Coherence Operations Issue	Advisory 32			X
Data Corruption on C66x CorePAC Initiated PCIe MMRs Read in Big Endian Mode Issue	Advisory 33	X	X	
AIF2 Line Code Violation Error from Serdes B8 to Serdes B4 Lanes Cross Connection Issue	Advisory 34	X	X	
Serdes AC-JTAG (1149.6) Receiver Sensitivity Issue	Advisory 35	X	X	
System Reset Operation Disconnects SoC from CCS Issue	Advisory 36			X
Power Domains Hang on Powering Them up at The Same Time as a $\overline{\text{RESET}}$ Received Issue	Advisory 37	X	X	X

Table 3 Silicon Revision Updates (Part 2 of 3)

Silicon Update Advisory	See	Applies To Silicon Revision		
		1.0	1.0A	2.0
DDR3 Incremental Write Leveling Issue	Advisory 38	X	X	X
Single MFENCE Issue	Advisory 39	X	X	X
Read Exception and Data Corruption Issue	Advisory 40	X	X	X
Incorrect Output from TAC for HS-PDSCH and P-CCPCH Channels in Certain Spreader Allocation Sequences Issue	Advisory 41	X	X	X
Incorrect Output from TAC for a Fetching Spreader on a Collision between Input Header's Fetch Write to the Spreader's SPM and Software's Write to SPM Issue	Advisory 42	X	X	X
False DDR3 Write ECC Error Reported Under Certain Conditions	Advisory 43	x	x	x
AIF2 Protocol Encoder Registers Usage Note	Usage Note 1	X	X	
TAC DL TPC Timing Usage Note	Usage Note 2	X	X	X
PCIe Device ID Field Reset Value Usage Note	Usage Note 3	X		
DDR3 Turn-Around Time For DQ & DQS Usage Note	Usage Note 4	X		
Packet DMA Does Not Update RX PS Region Location Bit Usage Note	Usage Note 5	X	X	X
Packet DMA Clock-Gating Usage Note	Usage Note 6	X	X	X
VCP2 Back-to-Back Debug Read Usage Note	Usage Note 7	X	X	X
Disable KICK Registers Usage Note	Usage Note 8	X		
DDR3 ZQ Calibration Usage Note	Usage Note 9	X	X	X
I^2C Bus Hang After Master Reset Usage Note	Usage Note 10	X	X	X
MPU Read Permissions for Queue Manager Sub-system Usage Note	Usage Note 11	X	X	X
$\overline{POR}$ and $\overline{RESETFULL}$ Sequence Usage Note	Usage Note 12	X	X	X
PLL ENSAT Bit Usage Note	Usage Note 13	X		
AIF2 GSM Compressed Mode and Multi-mode Usage Note	Usage Note 14	X	X	
Queue Proxy Access Usage Note	Usage Note 15	X	X	X
TAC E-AGCH Diversity Mode Usage Note	Usage Note 16	X	X	X
Minimizing Main PLL Jitter Usage Note	Usage Note 17	X	X	X
SRIO and PA_SS PKTDMA RX Descriptor Buffer Size Usage Note	Usage Note 18	X		
AIF2 LTE 3Mhz and 1.4Mhz Support Usage Note	Usage Note 19	X	X	X
PCIe Bar5 Windows Size Configuration in Boot ROM Usage Note	Usage Note 20	X	X	
Sticky Bits in PCIe MMRs Usage Note	Usage Note 21	X	X	X
The Clock Input To NETCP Usage Note	Usage Note 22	X	X	X
Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note	Usage Note 23	X	X	X
PLL Boot Configuration Settings and DEVSPEED Register Usage Note	Usage Note 24	X	X	
CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note	Usage Note 25	X	X	X
IDMA1 Performance Limitation Usage Note	Usage Note 26	X	X	
Revised PLL Programming Sequence Usage Note	Usage Note 27	X	X	X
Core Wake Up on $\overline{RESET}$ Usage Note	Usage Note 28			X
BSDL Testing Support Usage Note	Usage Note 29	X	X	X
AIF2 CPRI FastC&M Restrictions and Usage Note	Usage Note 30	X	X	X
Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note	Usage Note 31	X	X	X

Table 3 Silicon Revision Updates (Part 3 of 3)

Silicon Update Advisory	See	Applies To Silicon Revision		
		1.0	1.0A	2.0
DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note	Usage Note 32	X	X	X
Incorrect Output from TAC for Certain Channels When Configuring TAC_Gn_DATA_FETCH and TAC_Gn_HEAD_FETCH Registers with Values Having a Difference of 8 or 16 Usage Note	Usage Note 33	X	X	X
End of Table 3				

Advisory 1 SRIO Deadlock Issue

Revision(s) Affected: 1.0

Details: At least two devices exchanging a continuous stream for some period of time of SRIO transactions-with-responses may encounter a deadlock. SRIO transactions with responses include NREAD, NWRITE_R, DOORBELL, and Type 11 messages. SRIO transactions without responses are not impacted. SRIO transactions without responses include NWRITE, SWRITE, and Type 9 packets.

Deadlock results because both devices have requests being processed in the receive part (MAU and RXU) that must provide responses. The inbound buffers in the logical layer are getting filled with incoming requests and responses. The responses in the inbound buffers are blocked by the requests that are being processed. The outbound buffers in the logic layer also become filled with requests and responses making it impossible for the MAU and RXU to obtain the buffer credit that is needed to provide the response and get to the next request or response in the inbound buffer.

Workaround 1: Use mostly transactions that do not require responses. If responses are infrequent then transactions will continue to progress until the response can be returned. Use Type 9 packets instead of Type 11 messages. Use NWRITE (push) rather than NREAD (pull) and in preference to NWRITE_R. An occasional NWRITE_R to confirm that data has been successfully written or DOORBELL to alert the other device is acceptable.

Workaround 2: Limit Type 11 messages and DIO blocks to one (or possibly two) SRIO segments. This limits the number of buffers (segments) that can be outstanding. Impacts of this method are as below:

- For Type 11 messages this increases overhead with respect to the Queue Manager and the management of the buffer descriptors.
- For DIO (NREAD, NWRITE_R) this increases overhead with respect to programming the LSUs.

Step 1 to Minimize Risk of Issue: The 2× mode of operation has a higher probability of deadlock occurrence than the 1× mode because the 2× mode has more physical layer outbound buffers than the 1× mode, which can increase the egress traffic, which, in turn, can fill the ingress buffers of the link partner with low priority traffic. So, changing from 2× mode to 1× mode will reduce the probability of occurrence of this deadlock. Additional reduction of the egress packet rate can be done by using priority 0 only for outbound requests, which has the least number of outbound buffers, therefore reducing the traffic. Further reduction of the egress rate could be managed with outbound watermark settings, which can reduce the number of required outbound buffers for a particular priority.

Step 2 to Minimize Risk of Issue: For DIO transactions, use only a single LSU between two devices, which limits the max outstanding non-posted packets to 16. The LSU will naturally progress. Outbound buffer usage will depend on the responses. Impacts of this method are as below:

- This allows hardware to throttle the non-posted requests to 16 at any given time.
- However, the LSUs have shadow registers providing a means of programming a transaction while another transaction is in progress. The maximum for a single LSU is 9, each supporting 1MB transactions.
- Some head of line blocking (HOL) characteristics may arise due to larger blocks (presumably for a lower priority transaction) having to complete before the next transaction can proceed.

- Increased overhead to manage sharing of a single LSU among the cores especially if block sizes are reduced to avoid the possible HOL characteristics.

Step 3 to Minimize Risk of Issue: Use a single priority for all transactions, and allow for auto priority promotion to make it easier for responses to get through the transmit path. Impact is that this may cause some HOL characteristics, especially for Type 11 messages.

Step 4 to Minimize Risk of Issue: Limit LSU/TXU channels to one per destination. Because, at the same priority, this will guarantee round robin, which will effectively interleave outgoing requests and spread out requests to an individual destination.

Advisory 2 — *Queue Manager and Infrastructure Packet DMA Stall Issue*

Revision(s) Affected: 1.0.

Details: If the Queue Manager accesses an external linking RAM entry at the same time that the Packet DMA inside the Queue Manager Subsystem performs a memory access, a deadlock can occur. This scenario is possible only with both external linking RAM usage and Infrastructure Packet DMA data movement running simultaneously. This issue does not apply to any of the other Packet DMAs in the system (AIF2, FFTC, SRIO, PA_SS).

Workaround: Eliminate one of the features to eliminate the possibility of deadlock. Either do not use an external linking RAM, or do not use the infrastructure Packet DMA.

Advisory 3 — *SPI Boot Issue*

Revision(s) Affected: 1.0.

Details: Booting the device via SPI is not functional. The boot code is too aggressive in sending/receiving data on the SPI in a polling mode. This results in data being lost by the boot code during the poll. The SPI, itself, is still functional and can be used for normal data transfer; this issue affects only the ability to boot via SPI.

Workaround: None.

Advisory 4 — *Delayed SRIO PLL Lock Results In Incomplete SRIO Configuration Issue*

Revision(s) Affected: 1.0.

Details: The Boot ROM waits for up to 7 microseconds for the SRIO PLL to lock. After this, it times out and starts writing to the SRIO configuration registers. In some cases, the SRIO is not ready to accept the register writes and they are discarded.

As a result of this issue, all SRIO registers with offsets at 0xB000 or above, may not be programmed and may default to their reset values. The following registers are affected:

```
RIO_DEV_ID              0xb000
RIO_DEV_INFO            0xb004
RIO_PE_FEAT             0xb010
RIO_SW_PORT             0xb014
RIO_SRC_OP              0xb018
RIO_DEST_OP             0xb01c
RIO_LCL_CFG_BAR         0xb05c
RIO_RIO_BASE_ID         0xb060
RIO_SP_MB_HEAD          0xb100
RIO_SP_GEN_CTL          0xb13c
RIO_SP0_ERR_STAT        0xb158
RIO_SP0_CTL             0xb15c
RIO_SP1_ERR_STAT        0xb178
RIO_SP1_CTL             0xb17c
RIO_SP3_CTL             0xb1bc
RIO_PLM_SP0_PATH_CTL    0x1b0b0
RIO_IP_PRESCAL          0x1bd30
RIO_TLM_SP0_CONTROL     0x1b380
```

Workaround: Booting in 1× mode is supported using the default settings in the registers above. Successful boot can be achieved by booting in 1× mode, and then modifications can be made to the peripheral using application software or SRIO maintenance packets to adjust some of these registers. The following registers are recommended to be modified after link initialization has occurred.

- By default the SRIO peripheral is not set up to be in promiscuous mode, so it will accept only packets with DESTID = 0xFFFF. By initially sending a maintenance packet (DESTID = 0xFFFF) to adjust the RIO_TLM_SP0_CONTROL register, setting the TGT_ID_DIS and MTC_TGT_ID_DIS bits, the promiscuous mode can be enabled.
- The peripheral by default is not allowed to issue packets or respond to packets other than maintenance packets. This can be modified by sending a maintenance packet and writing the Output_Port_Enable and Input_Port_Enable of the RIO_SP0_CTL register.
- Finally, the port width can be changed to the desired 4× or 2× configuration by writing the appropriate value in the PATH_MODE field of the RIO_PLM_SP0_PATH_CTL register.

Also, it is possible to boot in the default 1× mode, change the SRIO mode to 2× or 4× mode (via software or maintenance packets) and then boot again in 2× mode if desired.

In addition, the RIO_DEV_ID and RIO_DEV_INFO register will not reflect the correct identifiers for this chip. These registers are supposed to be programmed by the bootloader using the values found in the device level JTAG ID register. The values can be read from the JTAG ID register using NREADs if desired. Bits 27:12(PartID) of the JTAG ID register should be used as the DEVICEIDENTITY field of the RIO_DEV_ID register, and bits 31:28 (Variant) should be used as the DEVICE_REV of the RIO_DEV_INFO register.

Advisory 5 — *Potential E-Fuse Autoload Errors During Device Initialization Issue*

Revision(s) Affected: 1.0.

Details: An issue has been uncovered involving the e-fuse autoload sequence. The e-fuse autoload sequence may provide the incorrect e-fuse information to the device at nominal core voltage. The exact e-fuse fields that are incorrect will determine the end impact to the system, ranging from no impact to severe impact.

Note—E-fuses are used in multiple areas within the device. They are used for memory repair, device ID, EMAC ID, etc.

Workaround: Guarantee that the CVDD Supply is 0.8 V during the e-fuse autoload sequence.

Therefore, the on-board CVDD supply that powers the CVDD rail on TCI6616 should default to 0.8 V instead of the recommended default. The e-fuse autoload occurs immediately after $\overline{\text{POR}}$ or $\overline{\text{RESETFULL}}$ is deasserted. E-fuse autoload does not happen on Hard Resets, Soft Resets or Local Resets (i.e. triggered by $\overline{\text{RESET}}$ pin, PLLCTL register, LPSC MMRs, or Watchdog timers). Every time a $\overline{\text{POR}}$ or $\overline{\text{RESETFULL}}$ is asserted, the CVDD supply must default to 0.8 V before $\overline{\text{POR}}$ or $\overline{\text{RESETFULL}}$ is deasserted. Once the e-fuse autoload sequence completes, the SmartReflex VCNTL interface on TCI6616 will begin outputting the correct sequence to the power controller and that will adjust the CVDD supply to the correct voltage before the device boot occurs.

If desired, the user can verify that the e-fuse autoload completed successfully by checking the value of the E-fusefarm Error Register (FUSERR) at address 0x023100e0 after boot. The register is shown below.

Figure 2 FUSERR Register

31	5	4	0
Reserved		ERR	
R-0		R-0	

Legend: R = Read only; -*n* = value after reset

Table 4 FUSERR Register Description

Bits	Name	Description
31-5	Reserved	Reserved
4-0	ERR	Efuse Error Code • 00000b – Efuse Autoload Pass • 00011b – Efuse Autoload Fail All other values are reserved.

Advisory 6 — *SRIO Potential Message Header Corruption Issue*

Revision(s) Affected: 1.0.

Details: This issue can occur if both SRIO Type 9 and Type 11 messaging are used in the same device. When the RXU submodule inside the SRIO module times out while waiting for the rest of the data of a multi-segment message, an incorrect message type may be used to construct the protocol-specific (PS) data in the returned message header. The PS data is filled according to the last message type received instead of the message type that generated the timeout. Because Type 9 and Type 11 have different structures, if the wrong type is used, incorrect fields are used to fill the PS data in the RX packet.

Workaround 1: The DSP can detect the error condition using certain bits in the received PS data. This requires the knowledge of the SRIO TT setting used in the system. It also requires always setting the least-significant bit of the StreamID to 0, i.e. using only even StreamIDs. The following algorithm can then be used.

```
if (Pkt_type in Navigator descriptor == 31) (Type 11 RX packet was received)
{

    if (known value of TT used in system == 0)
    {

        if (PS info word 2, bit 9 == '1')
            Header corruption occurred. RX packet actually timed out.
        else
            Header corruption did NOT occur. Process RX packet normally.
    }
    else (known value of TT used in system == 1)
    {
        if (PS info word 2, bit 10 == '1')
            Header corruption occurred. RX packet actually timed out.
        else
            Header corruption did NOT occur. Process RX packet normally.
    }
}
else if (Pkt_type in Navigator descriptor == 30) (Type 9 RX packet was received)
{
    if (PS info word 2, bit 16 == '1')
        Header corruption occurred. RX packet actually timed out.
    else
        Header corruption did NOT occur. Process RX packet normally.
}
```

Workaround 2: Avoid mixing Type 9 and Type 11 messages.

Advisory 7 — *Boot ROM CM Setting For SGMII Issue*

Revision(s) Affected: 1.0.

Details: Boot ROM sets Ethernet Common Mode (CM bit in SGMII_SERDES_CFGTX registers) to 0, but the field should be set to 1 for AC-coupled applications. Using 0 for AC-coupled applications can result in less noise margin on the SerDes lines and lead to higher bit error rates.

Workaround: Software can set the bit to the ideal value of 1 after boot. This requires setting bit 7 to 1 in the SGMII_SERDES_CFGTX0 (address 0x02620348) and SGMII_SERDES_CFGTX1 (address 0x02620350) registers. Read-modify-writes can be used to make sure other bits in the registers are not affected.

Advisory 8 *Main PLL Clock Divider Not Properly Configured Issue*

Revision(s) Affected: 1.0.

Details: The main PLL Output Divider (divide-by-2) is not enabled by default. This means the main PLL output clock is twice as fast as specified in the data manual during boot and may cause the device to be over-clocked, depending on which BOOTMODE is selected.

Workaround: Because of this issue, it will be necessary to change the values of BOOTMODE[12:10] on the board. The bootloader uses BOOTMODE[12:10] to program the PLL during the boot sequence. These configured values are listed in the device data manual under Section 2.5 "Boot Modes and Supported PLL Settings". Table 5 is an update to the data manual table because of this issue. The Documented BOOTMODE[12:10] column shows the intended value. The Errata BOOTMODE[12:10] shows what this needs to be changed to for each particular clock input frequency. Making this change is necessary so that the device is not over-clocked. The *DSP ƒ* columns show the actual frequency with the new BOOTMODE[12:10] settings.

Table 5 C66x CorePac System PLL Configuration

Documented BOOTMODE [12:10]	Errata BOOTMODE [12:10]	Input Clock Freq (MHz)	800 MHz Device			1000 MHz Device			1200 MHz Device		
			PLLD	PLLM	DSP ƒ	PLLD	PLLM	DSP ƒ	PLLD	PLLM	DSP ƒ[1]
0b000	0b011	50	0	15	800	0	19	1000	0	23	1200
0b001	0b100	66.67	24	255	682.7008	4	63	853.376	24	383	1024.051
0b010	0b101	80	4	31	512	0	7	640	4	47	768
0b011	0b101	100	4	31	640	0	7	800	4	47	960
0b100	0b110	156.25	24	127	800	4	31	1000	24	191	1200
0b101	N/A	250	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0b110	N/A	312.5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
0b111	0b101	122.88	4	31	786.432	0	7	983.04	4	47	1179.648

1. *ƒ* represents frequency in MHz.

Example: 50 MHz Main PLL Input Clock provided to a 1000 MHz speed grade device – The data manual says to set BOOTMODE[12:10] to 0b000. Because of the issue this will produce a DSP clock of 2000 MHz. The board should be changed to use 0b011.

Once the boot is complete, it is highly recommended that software reconfigure the PLL to the desired frequency, even if it is already achieved by the initial settings. To minimize the overall output jitter, the PLLs should be operated as close as possible to the maximum operating frequency. To maximize the VCO frequency within the PLL, the PLL should be clocked to 2× the intended frequency and the PLL Output Divider should be set to /2. The main PLL Output Divider should be set to divide-by-2 by the software by writing 0b0001 to bits [22:19] of the SECCTL register (address

0x02310108) in the PLL controller. A read-modify-write can be used to make sure other bits in the register are not affected. This register is documented in the *TMS320TCI6616 Communications Infrastructure Keystone SoC* data manual (literature number SPRS624).

Note—PA PLL settings are not affected by this issue. Use the original table in the data manual.

Note—Main PLL clock inputs greater that 156.25 MHz cannot be supported because of this issue.

Note—Some of the configurations above no longer give the max frequency for the device as it was intended to. This will not create an issue for any of the boot modes.

Advisory 9 — *HyperLink Temporary Blocking Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: A temporary blocking condition can occur on HyperLink on a Keystone I local (slave) device while returning high read activity initiated by a remote device (master.) If the read accesses are sufficiently high from the remote devices, the responses from the local device can keep the response path continuously busy. As a result, the local device that is returning the read data cannot issue any commands to the remote device since the read return data is highest priority. This is called arbitration head of line blocking (HOLB). Usually to initiate such high read activity, it is expected that the remote device is using master peripheral like EDMA issuing the reads, where the read burst size is 64 byte or higher, there needs to be multiple outstanding read requests and the read accesses are initiated to high speed memory like L2 or MSMC on the local device. Typically read accesses initiated to external memory (DDR3) will not see this issue, as the read responses are slower relative to on chip memory, additionally if the read accesses are initiated by CPU, it is not expected to cause HOLB, as the CPU waits for read response data prior to issuing subsequent read commands, which limits both the read activity and number of outstanding transactions.

Workaround 1: Use a push messaging model instead of pull, if possible.

Workaround 2: If a pull model is required, use CPU for reading instead of EDMA for local (high speed/low latency) memories, so the schedule breaks when the return path is continuously busy and HOLB is avoided.

Advisory 10 — *SRIO Reset Isolation Not Functional Issue*

Revision(s) Affected: 1.0, 1.0A

Details: The SRIO module has the capability via reset isolation to continue forwarding SRIO packets intended for other devices while the rest of the local device containing the SRIO module is in reset. However, the SRIO reset isolation feature is not functional on silicon revision 1.0. For this version of the silicon, the software should configure SRIO to be reset every time a device reset is asserted to the device.

For revision 1.0A, the SRIO reset isolation feature is available to a limited extent. If the SRIO reset isolation feature is activated when any SRIO traffic is present, part of the logic in the SRIO module may not properly enter the reset isolation state. This may cause potential instability in the system when the SRIO block comes out of reset isolation.

Workaround: A workaround for the issue is to perform a controlled reset for the device where all SRIO traffic going to and from the device (including forwarded traffic) is stopped before issuing the device reset. This will allow the SRIO module to enter reset isolation mode correctly. This workaround only applies to revision 1.0A.

Advisory 11 — *SRIO Type 9 / Type 11 RX Message Uses Wrong Context Issue*

Revision(s) Affected: 1.0.

Details: This issue can occur when a Type 9 or Type 11 SRIO multi-segment message has been partially received by the device, meaning that at least the first segment has been received, but the last segment has not yet arrived. If a second message of the opposite type is received during this time with the same header identification information as the first, the second message will incorrectly use the SRIO receive context assigned to the first message. This happens because the hardware logic does not check the message type before selecting the context based on the message identification information.

Workaround: In order for this issue to occur, the bits making up the message identification information must match between the two packets arriving at the same time. The issue is avoided if the information for packets arriving at the same time is different. One way to accomplish this would be to make sure that the least-significant six bits of the COS field for any Type 9 messages do not match the 6-bit mailbox field for any Type 11 messages that have the same src_id and dest_id.

Advisory 12 — *Total Number of SRIO RXU Contexts Decreases Due to Timeout or Teardown Issue*

Revision(s) Affected: 1.0.

Details: The total number of SRIO RXU contexts that can be used to store received messages may decrease when an RXU timeout or teardown occurs. Eventually, if timeouts and teardowns keep occurring, there may be no contexts available and no messages can be received.

Workaround: The RXU teardown will not cause the issue to occur if the RXU is reset (by disabling and then enabling the submodule) after the teardown is performed.

To Minimize Risk of Issue: The possibility of an RXU timeout can be minimized by making sure the device is not oversubscribed.

Advisory 13 — *DDR3 Automatic Leveling Issue*

Revision(s) Affected: 1.0., 1.0A, 2.0

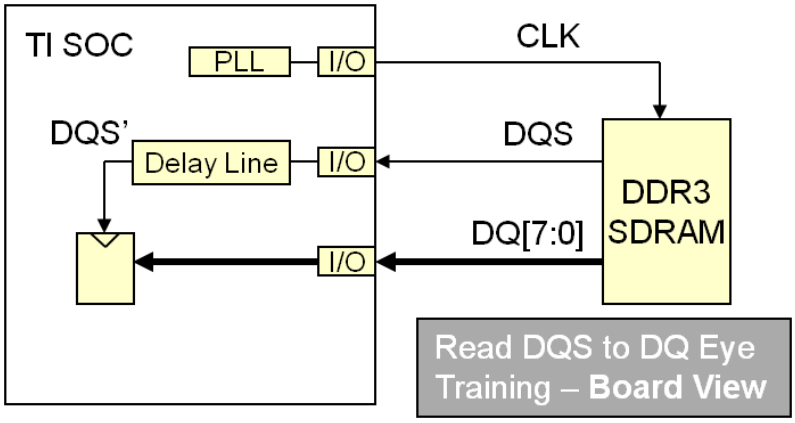
Details: The DDR3 PHY integrated into the DDR3 Memory Controller of the device has a problem with one of the read-write leveling hardware features. When this issue occurs, the DDR3 memory cannot be written to and/or read from correctly.

These features are described in Read-Write Leveling section of the *KeyStone Architecture DDR3 Memory Controller* (SPRUGV8). The three leveling types are: Write Leveling, Read DQS Gate Training, and Read Data Eye Training.

The leveling feature with the problem is the Read Data Eye Training. Depending on the jitter seen on the DQ and DQS signals, this leveling feature will fail to converge on a good data eye, resulting in corrupted DSP to DDR3 SDRAM reads.

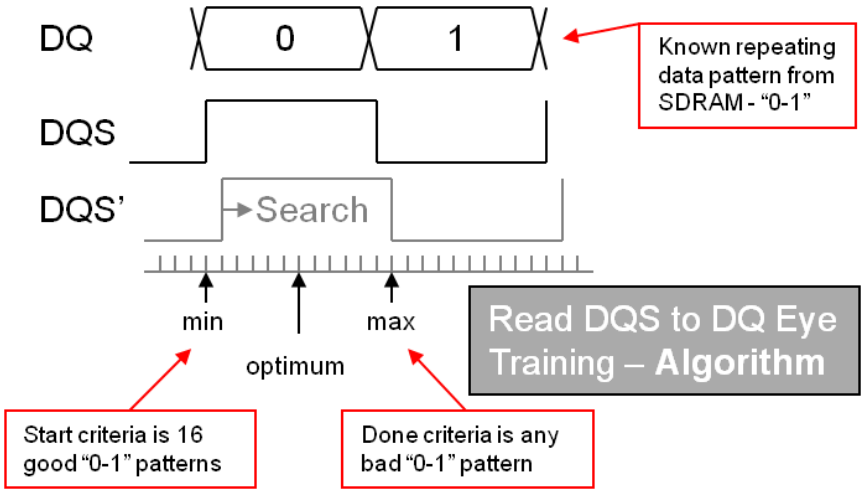
The intended purpose of the Read Data Eye Training feature is to align the DQS sampling transitions in the middle of the DQ data eye within individual byte lanes of the DDR3 memory interface. As seen in Figure 3, the DQS signal can be delayed relative to the DQ signal within the DDR3 PHY.

Figure 3 Read DQS to DQ Eye Training - Board View



As shown in Figure 4, the Read Data Eye Training algorithm should correctly detect the extents of the DQ data eye. The SDRAM is put into a DQS-to-DQ eye training mode and read back an alternating 0, 1 pattern.

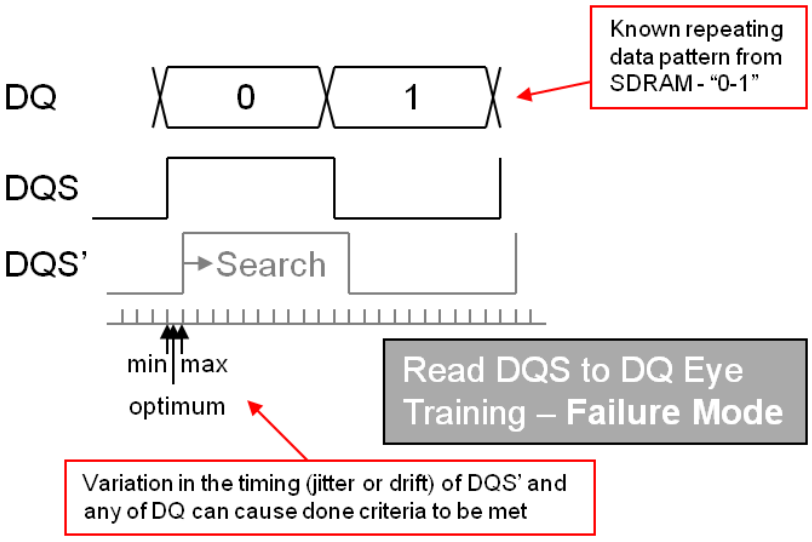
Figure 4 Read DQS to DQ Eye Training - Algorithm



The DDR3 PHY uses this pattern to detect good or bad data being sampled by the DQS transitions. The DDR3 PHY searches by delaying DQS relative to DQ until these constraints are found.

However, if too much jitter is present on the DQ or DQS signals, the search can converge to a false edge as shown in Figure 5.

Figure 5 Read DQS to DQ Eye Training - Failure Mode



This failure results in the data sampling location being placed in a very noisy transition region to the left of the DQ data eye. At this point corrupted data will then be read back to the DSP when a DDR3 read command is issued.

Workaround 1: This workaround is available only on revision 1.0. Minimizing jitter on the DDR3 byte lane signals (DQS, DQS#, DQn, DMn, CBn) is the only way to avoid this issue. Ensuring as much margin as possible from a signal integrity perspective is a necessity for designs not to manifest this issue.

1. All layout guidelines outlined in the DDR3 Design Guidelines (SPRABI1) must be strictly followed.
2. Layout keep-out regions suggested in DDR3 Design Guidelines must be strictly enforced.
3. All high-frequency noise sources must be kept as far as possible from the DDR3 interface.
4. All power filtering suggestions must be followed.
5. All DDR3 nets must be simulated pre-layout, using both TI IBIS models as well as those models provided by SDRAM vendors to ensure that the basic layout meets and exceeds signal integrity requirements for DDR3 design functionality.

Workaround 2: This workaround is available only on revision 1.0A and 2.0 of the device. The DDR3 PHY in the device has the ability to complete auto-leveling of the write leveling values and the read DQS gate training values separate from the read data eye training. Then a fixed value is used as the read data eye sample point. This solution is functional on standard DDR3 fly-by layouts and is referred to as Partial Automatic Leveling. It has been validated for robust operation at DDR3-1333 when connected to either a UDIMM or with a discrete SDRAM implementation.

This mode is enabled by setting bit 9 (0-indexed) of the DDR3_CONFIG_REG_23 at address 0x02620460. The lower 8 bits (bits 7:0) may be written with a read data eye sample value or left at their default value of 0x34. The default value is recommended for use. After programming DDR3_CONFIG_REG_23, proceed to enable auto-leveling. Please refer to the *DDR3 Initialization Application Report* (literature number SPRABL2) for details of the sequence of steps.

Workaround 3: This workaround is available only on revision 1.0A and 2.0 of the device. The user has the ability to completely disable the automatic leveling features of the DDR3 controller and rely exclusively on a set of ratio-forced register values. These values control the DQ and DQS delay between all byte lanes for gate leveling, write leveling, and read data eye training. The specific registers and values to set in the registers are described in the *KeyStone Architecture DDR3 Memory Controller User Guide* (literature number SPRUGV8). The values programmed are dependent on the specific board characteristics.

Because these delay values cannot be tailored to specific byte lanes, only layouts with very small signal skews between DRAMs can use this workaround. Rather that following the routing guidelines which allow different lengths for each byte lane, all signals must be closely matched. Specifically:

1. Route lengths for address, command, control and clock between the DSP and DRAMs must be very similar. We understand that the fly-by topology prevents these lengths from matching.
2. All DQ, DQM and DQS signals must be skew-matched to within ± 10.00 mils across all byte lanes. This length should be the average of the fly-by lengths measured from the DSP to the DRAMs.

These additional routing skew constraints attempt to minimize the delay variance as much as possible, which enable s the single set of values to adequately control the PHY timing. The time variation from the first to the last DRAM reduces the available sampling window. This restriction means that use of this work-around reduces the DDR3 operating speed and it can only be used with topologies with a small number of devices.

It is estimated that the following rates as shown in Table 6 can be achieved based on the routing skew on the board when using ratio-forced register settings:

Table 6 Routing Skew vs. Maximum Data Rate

Maximum Skew	Maximum Data rate
20 mils	1333 MT/s
1.0 inches	1066 MT/s
2.5 inches	800 MT/s

Workaround 4: This workaround is available only on revision 1.0A and 2.0 of the device. The read data eye sample point can now be optimized by incremental read eye leveling. To do this, leave the read data eye training enabled (leave bit 9 in DDR3_CONFIG_REG_23 = 0), trigger automatic leveling and then follow up with at least 64 read data eye incremental leveling events. The incremental leveling events converge the read data eye sample point to a robust sample location.

The time between incremental leveling events is set by the incremental leveling prescalar and incremental data eye training interval fields in the RDWR_LVL_CTRL register at 0x210000DC. The initialization routine can enable the incremental read eye leveling, pause long enough to allow the 64 events to occur and then it can disable the incremental read eye leveling, if desired. When implementing this workaround, the DDR3 interface must not be used until after these 64 incremental read eye leveling events are completed.

Advisory 14 *Potential L2 Cache Corruption During Block Coherence Operations Issue*

Revision(s) Affected: 1.0. and 1.0A.

Details: A potential L2 cache corruption issue during block coherence operations has been identified. Under a specific set of circumstances, L1D or L2 block coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back-to-back in the same L2 set:

1. L1D write miss
2. Writeback or invalidate or writeback-with-invalidate due to block coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block coherence operations listed below:

- L1D writeback
- L1D invalidate
- L1D writeback with invalidate
- L2 writeback
- L2 invalidate
- L2 writeback with invalidate

Workaround: The workaround requires that the memory system be idle during the block coherence operations. Hence programs must wait for block coherence operations to complete before continuing. This applies to L1D and L2 memory block coherence operations. To issue a block coherence operation follow the sequence below.

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register.
4. Wait for completion by one of the following methods:
   a. Issue an MFENCE instruction (preferred).
   b. Poll the WC register until the word count field reads as 0.
5. Perform 16 NOPs
6. Restore interrupts.

For further information about the cache control registers (BAR and WC) see the *TMS320C66x DSP CorePac User Guide* (SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the *C66x DSP and Instruction Set Reference Guide* (SPRUGH7).

Advisory 15 *Multiple PLLs May Not Lock After Power-on Reset Issue*

Revision(s) Affected: 1.0.

Details: After power-on reset release, the boot mode for the device is selected based on the status of the BOOTMODE pins. The main PLL stays in bypass mode for no-boot and I^2C boot. For all other boot modes, a PLL initialization sequence executes inside the boot ROM to configure the main PLL in PLL mode.

In order to ensure proper PLL startup, the PLL power_down pin needs to be toggled. This is accomplished by toggling the PLLPWRDN bit in the PLLCTL register. This needs to be done before the main PLL initialization sequence. The PLL initialization sequence in the boot ROM does not toggle the PLLPWRDN bit. So it may cause the main PLL and/or DDR3 PLL and/or PA PLL not to be locked.

The common symptoms of this issue would be: not being able to connect in CCS, not being able to read or write DDR configuration registers, and/or the SYSCLKOUT pin not showing SYSCLK.

Workaround: The effect of the incomplete PLL initialization sequence in the boot ROM can be avoided using the following steps:

1. Perform a read/modify write to bit-1 (PLLPWRDN) of the PLL control register (PLLCTL). The address of the PLL control register is 0x0231_0100.
2. Stay in a loop such that the bit is set for 5 µs (minimum) and then clear the bit.

Steps 1 and 2 must complete before the main PLL initialization sequence executes from the boot ROM. They must also be executed before the DDR3 PLL and the PA PLL are configured. These two steps control all the above mentioned three PLLs (Main, DDR3 and PA). Below are the reserved pins on the device which will indicate the status of the main PLL lock, DDR3 PLL lock and PA PLL lock.

- RSV20 - COREPLLLOCK
- RSV21 - DDR3PLLLOCK
- RSV22 - PAPLLLOCK

If those pins are high then respective PLL is locked and if those pins are low then the respective PLL is not locked. Above pins can be checked to identify whether or not the device encountered the error condition.

The workaround could be implemented in one of several ways for different chip states:

- **For CCS emulation mode with a no-boot configuration:** The workaround is inside a GEL file provided by TI.
- **For no-boot and I^2C boot modes:** The workaround should be added before the main PLL initialization sequence inside the application code. Please see the I^2C boot file provided by TI.
- **For other boot modes:** The workaround is to force the chip to start an I^2C boot with a boot configuration table. The boot configuration table is used to poke a small program (the fix in steps 1 and 2) to execute before the main PLL initialization code in the device boot ROM runs. An example for this implementation is provided in above mentioned I^2C boot file. In this scenario, the I^2C primary boot implements the workaround and then configures the appropriate peripheral for a secondary boot.

Advisory 16 — *Queue Manager External Linking RAM Location Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: When the Queue Manager is setup to use an external linking RAM and the external linking RAM is placed in the same memory endpoint as the packet descriptors or data buffers, then the Queue Manager may encounter a stall from which it cannot recover. The different memory endpoints are L2RAM, MSMC SRAM, and DDR3.

Workaround: The Queue Manager's external linking RAM must be placed in a different memory endpoint from the packet descriptors or data buffers. As an example, if the external linking RAM is placed in MSMC SRAM, then the packet descriptors and data buffers must be placed in L2RAM and/or DDR3. Note that the packet descriptors and data buffers do not have to be placed in the same memory endpoints.

Advisory 17 — *AIF2 AIF_TX_MONO_MODE Not Compatible With Other Navigator Modules Issue*

Revision(s) Affected: 1.0, 1.0A

Details: When the AIF2 is used in Packet DMA mode, there is an optional setting that can be enabled for the AIF2 Packet DMA called AIF_TX_MONO_MODE. This special mode enables an internal optimization in the AIF2 that requires the use of a specific format of monolithic packet that has a 16-byte header immediately followed by data. This special mode is described in more detail in the *KeyStone Architecture Antenna Interface 2 (AIF2)* user guide (literature number SPRUGV7), *KeyStone Architecture Multicore Navigator* user guide (literature number SPRUGR9), and *Connecting AIF2 to FFTC Guide for KeyStone Devices* application report (literature number SPRABF3). One additional requirement is that the TX packets sent to the AIF2 must be sent with a descriptor size field (found in the lowest 4 bits of the value written when pushing a descriptor onto a queue) that is set to 64 bytes.

It is possible for other Navigator-compliant modules (such as FFTC, SRIO, and the Packet Accelerator) to send output packets directly to the AIF2 without any DSP intervention. However, when the monolithic packet format described above is used, the Packet DMA in the module sending the data uses a descriptor size field of 16 bytes. This means that the special AIF_TX_MONO_MODE of the AIF2 Packet DMA cannot be used for AIF2 TX queues that are receiving packets directly from any of the Navigator-compliant modules.

Workaround: Make sure the AIF_TX_MONO_MODE setting (bit 24 in the TX Channel N Global Configuration Register B in the AIF2 Packet DMA, see the Navigator user guide for more details) is set to *Normal Monolithic Mode* (value = 0) for each of the AIF2 TX queues that are expected to receive packets directly from any of the Navigator-compliant modules.

Advisory 18 DDR3 Excessive Refresh Issue

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: DDR3 JEDEC standard specifies that at any given time, a maximum of 16 refresh commands can be issued within a 2 × tREFI interval (2 × 7.8 = 15.6 µs). Failing to meet this requirement could result in a high current draw and the possibility of DDR3 device failure. The DDR3 controller will violate the above requirement if the following actions occur:

1. The DDR3 memory is put in to self-refresh mode by setting the LP_MODE field in the Power Management Control register to 0x2.
2. One or more read/write commands are sent by the DDR3 controller while the DDR3 memory is in self-refresh such that the memory exits self-refresh to execute commands.
3. After command execution is complete and there are no more commands to execute, the DDR3 controller is then idle for a certain number of DDR clock cycles before putting the external memory in self-refresh mode. The number of idle DDR clock cycles is defined by SR_TIM value field in the Power Management Control register.
4. Because the DDR3 controller issues one refresh command on self-refresh entry and another refresh command on self-refresh exit, if this sequence repeats more than eight times within a 2 × tREFI interval, the DDR3 controller will issue more than 16 refresh commands in a 2 × tREFI interval and violate the JEDEC requirement.

Note if SR_TIM value is greater than or equal to 0x9, the DDR3 controller does not violate the JEDEC requirement. This is because the DDR3 controller will wait for at least 4096 clock cycles of idle time before putting DDR3 memory into self-refresh mode. Therefore, the only possible way the above scenario could occur is by setting the LP_MODE field to 0x2 to put the DDR3 memory in self-refresh mode with the SR_TIM value field less than 0x9, and then sending periodic read/write commands.

Workaround: When using LP_MODE=0x2 to enter self-refresh mode, SR_TIM needs to be programmed greater than or equal to 0x9. For further information about the Power Management Control register see the *KeyStone Architecture DDR3 Memory Controller User Guide* (literature number SPRUGV8).

Advisory 19 — *SRIO Register Aliasing Issue*

Revision(s) Affected: 1.0, 1.0A

Details: Due to an error in a portion of the address decoding logic in the SRIO module, writes to SRIO MAC registers with an address offset in the range of 1B400h to1B9FCh will cause the same data to also be written to the SRIO RXU mapping register with an address offset in the range of 00400h to 009FCh and whose address has the same lowest 12 bits. This issue can potentially make the RXU mapping registers have random values that might permit spurious messages to enter a mailbox or not permit valid messages to enter a mailbox.

Workaround: The MAC registers affected should be initialized before initializing the RXU registers, so that after all the initialization is done, the RXU registers still have the correct value. These affected MAC registers should be written only during initialization. If they are written again later, they will corrupt the corresponding RXU registers.

There are some SRIO MAC registers that may need to be written after initialization for normal operation. They are listed in the two following tables, along with their corresponding RXU mapping register. To allow these MAC registers to be written after initialization, the corresponding RXU mapping entries cannot be used by the application. This means that the affected RXU mapping registers cannot be used for normal operation and should be avoided by the application software. The Type 9 RXU mapping registers affected are RXU_TYPE9_MAP{45, 47, 49, 50, 52}. The Type 11 RXU mapping registers affected are RXU_MAP{01, 12, 22}.

The RXU mapping registers listed in Table 7 should be disabled during initialization by writing a value of **11b** to the **tt** field in the RXU_MAPxx_H or RXU_TYPE9_MAPxx_1 register for that RXU mapping entry. The base address of the SRIO block must be added to the register offsets given in the table to obtain the full address of each register.

Table 7 Affected SRIO MAC Registers That May Need to Be Written After Initialization

MAC Register Offset	MAC Register Name	RXU Register Offset	RXU Register Name
1B490h	TLM_SP2_STATUS	00490h	RXU_MAP12_L
1B510h	TLM_SP3_STATUS	00510h	RXU_MAP22_QID
1B924h	EM_PW_EN	00924h	RXU_TYPE9_MAP45_2
1B934h	EM_DEV_PW_EN	00934h	RXU_TYPE9_MAP47_0
1B93Ch	EM_MECS_STAT	0093Ch	RXU_TYPE9_MAP47_2
1B94Ch	EM_MECS_REQ	0094Ch	RXU_TYPE9_MAP49_0
1B960h	EM_RST_PORT_STAT	00960h	RXU_TYPE9_MAP50_2
1B970h	EM_RST_PW_EN	00970h	RXU_TYPE9_MAP52_0

The two MAC registers in Table 8 may also need to be written after initialization, but because they end up writing data to the corresponding RXU_MAPxx_H or RXU_TYPE9_MAPxx_1 register directly, they require special handling. The two RXU mapping registers listed should still be disabled during initialization like the other registers above.

Table 8 Affected SRIO MAC Registers That May Need to Be Written After Initialization and Require Special Handling

MAC Register Offset	MAC Register Name	RXU Register Offset	RXU Register Name
1B410h	TLM_SP1_STATUS	00410h	RXU_MAP01_H
1B950h	EM_MECS_PORT_STAT	00950h	RXU_TYPE9_MAP49_1

However, in order to make sure they stay disabled, the two reserved bits in the MAC register that match the bit indices of the **tt** field in the corresponding RXU mapping register must be set to a value of **11b** whenever the MAC register is written. For example, bits [14:13] of the TLM_SP1_STATUS register must be written with a value of 11b because those bits match the tt field location in the RXU_MAP01_H register. Also, bits [14:13] of the EM_MECS_PORT_STAT register must be written with a value of 11b because those bits match the tt field location in the RXU_TYPE9_MAP49_1 register. The base address of the SRIO block must be added to the register offsets given in the table to obtain the full address of each register.

Advisory 20 — *AIF2 CPRI LTE Ingress Antenna Carrier Packing Issue*

Revision(s) Affected: 1.0, 1.0A

Details: In order to fully support the CPRI protocol for LTE, the AIF2 module should allow flexible antenna carrier sample packing for the multiple antenna carrier use-case. For example, if the user sets the dual bit map (DBM) X value to 16 and transfers two antenna carriers, the user can choose any packing pattern for the CPRI basic frame. Several examples are shown as follows where AxC0 and AxC1 are antenna carriers 0 and 1, respectively.

- {Control, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1}
- {Control, AxC0, AxC0, AxC1, AxC1, AxC0, AxC0, AxC1, AxC1, AxC0, AxC0, AxC1, AxC1, AxC0, AxC0, AxC1, AxC1}
- {Control, AxC0, AxC0, AxC0, AxC0, AxC1, AxC1, AxC1, AxC1, AxC0, AxC0, AxC0, AxC0, AxC1, AxC1, AxC1, AxC1}
- {Control, AxC0, AxC0, AxC0, AxC0, AxC0, AxC0, AxC0, AxC0, AxC1, AxC1, AxC1, AxC1, AxC1, AxC1, AxC1, AxC1}
- {Control, AxC0, AxC0, AxC0, AxC1, AxC1, AxC1, AxC1, AxC1, AxC0, AxC0, AxC0, AxC0, AxC0, AxC1, AxC1, AxC1}

However, the current AIF2 protocol decoder supports only one type of packing pattern for CPRI incoming (ingress) frames. It is the default packing order and allows only complete interleaving, where each sample from one antenna carrier is followed by a sample from the next antenna carrier (like AIF1 supported) and does not allow a DBM X value greater than the number of antenna carriers. The following is an example of this pattern for two antenna carriers (DBM X value is two).

- {Control, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1, AxC0, AxC1}

Workaround: The AIF2 ingress (output) packet should be configured as a super packet, which contains all the samples of an OFDM symbol for all the antenna carriers of the link. Because the antenna packing pattern is regular, the EDMA can be used to de-interleave the data without CPU intervention. Based on system bandwidth and link rate, the EDMA parameters can be set statically.

Upon reception of the super packet, a system event is generated to trigger the EDMA to deinterleave the data. The user can use an accumulator queue as the AIF2 output queue to trigger EDMA. Alternatively, an AT event can be used to trigger EDMA. Because the AT event is not tied with the availability of the super packet in system memory, enough headroom needs to be used to guarantee the availability of the super packet. When the EDMA de-interleaving is done, the EDMA can be used to clear the system event.

Here is an example for a system that uses a 20-MHz LTE bandwidth with 8× link rate. Figure 6 shows the antenna carrier packing used, where four antenna carriers are interleaved in clumps of eight samples each.

Figure 6 Antenna Carrier Packing

	Big Endian			
Address	0	4	8	C
0x00	AxC0 s0	AxC0 s1	AxC0 s2	AxC0 s3
0x10	AxC0 s4	AxC0 s5	AxC0 s6	AxC0 s7
0x20	AxC1 s0	AxC1 s1	AxC1 s2	AxC1 s3
0x30	AxC1 s4	AxC1 s5	AxC1 s6	AxC1 s7
0x40	AxC2 s0	AxC2 s1	AxC2 s2	AxC2 s3
0x50	AxC2 s4	AxC2 s5	AxC2 s6	AxC2 s7
0x60	AxC3 s0	AxC3 s1	AxC3 s2	AxC3 s3
0x70	AxC3 s4	AxC3 s5	AxC3 s6	AxC3 s7
0x80	AxC0 s8	AxC0 s9	AxC0 s10	AxC0 s11
0x90	AxC0 s12	AxC0 s13	AxC0 s14	AxC0 s15
0xA0	AxC1 s8	AxC1 s9	AxC1 s10	AxC1 s11
0xB0	AxC1 s12	AxC1 s13	AxC1 s14	AxC1 s15
...	...	...	...	...

In this case, a set of PaRAM entries are linked together where each PaRAM entry is set up to de-interleave all the antenna carriers in a single OFDM symbol time. At least seven PaRAM entries are needed to be able to differentiate long cyclic prefix (CP) and short CP in the LTE normal CP case. Figure 7 shows examples of long CP and short CP PaRAM entries.

Figure 7 Long CP and Short CP PaRAM Entries

PaRAM for long CP

OPT=(ABSYNC , ITCCHEN)	
SRC	
BCNT=276	ACNT=32
DST	
DSTBIDX =32	SRCBIDX =128
BCNTRLD=276	LINK
DSTCIDX=8832	SRCCIDX=32
Rsvd	CCNT=4

PaRAM for short CP

OPT=(ABSYNC , ITCCHEN)	
SRC	
BCNT=274	ACNT=32
DST	
DSTBIDX=32	SRCBIDX=128
BCNTRLD=274	LINK
DSTCIDX=8768	SRCCIDX=32
Rsvd	CCNT=4

Depending on the buffer management, more PaRAM entries may be needed to be statically configured to avoid CPU intervention. In the case of a ping-pong buffer, 14 PaRAM entries are needed. The PaRAMs are linked to each other based on symbol number. If the antenna carriers are set up differently, up to 14 × num_antenna_carriers PaRAM entries could be required for a ping-pong buffer. De-interleaved buffers can be pre-linked with descriptors. When the de-interleaving is done, the EDMA can be used to push the descriptors to an FFTC TX queue for further processing.

Here are some additional notes on the workaround.

- Regular packets can still be used for egress (input) packets when using super packets on ingress (output).
- Special notes for TDD
  - If all antenna carriers on the same link have the same TDD configuration, then the AIF2 TDD mode can be used to output only the super packet as needed.
  - If antenna carriers on the same link have different TDD configuration, then AIF2 TDD mode cannot be used. The EDMA needs to be set up to de-interleave only the useful part.
- If the antenna carriers on the same link have different offsets, then the current workaround cannot be used.
- Additional internal memory is needed because of the EDMA de-interleaving.
  - Internal memory is preferred because the de-interleaved data needs to be sent to FFTC for processing, which would have lower throughput if the input packet is in DDR.
  - A symbol-based ping-pong buffer may be enough to serve as the de-interleaved buffer. For example, for one sector of a 20-MHz 4 RX antenna system, 71k bytes of memory is needed.

Advisory 21 *SRIO Packet Forwarding Issue*

Revision(s) Affected: 1.0, 1.0A

Details: When a high volume of SRIO packets with varying priorities are arriving at a device and are being forwarded on to another device, there is a possibility that the local SRIO module will lockup and stop processing incoming packets. The logic in the SRIO module places all packets to be forwarded in the same internal queue, allowing the possibility that a high volume of lower priority packets may block a higher priority packet and cause all packet processing to stop.

Workaround: Pace the packet forwarded traffic so that congestion is avoided and make all packet forwarded traffic the same priority. This priority should be higher than the other incoming traffic destined for the local device.

Advisory 22 — *SRIO Messaging in Highly Oversubscribed System Issue*

Revision(s) Affected: 1.0, 1.0A

Details: When the SRIO message-passing mode is used in a highly oversubscribed system, it is possible for the number of available TX contexts to decrease over time and eventually cause the module to stop transmitting TX packets. An oversubscribed system is one where the number of RXU contexts required in any of the devices tends to exceed the maximum number of RXU contexts in that device. There are 16 RXU contexts per device.

There are two scenarios that can cause the failure. In the first, a retry that happens at nearly the same time as a DONE response can trigger an error in the internal logic that causes a TX context to become unusable. In the second, an error/timeout condition can cause a TX context to be incorrectly cleared twice, making it unusable. In either case, all TX contexts may eventually become unavailable and cause the SRIO module to stop transmitting TX packets.

To reduce the possibility of a system being oversubscribed, there should be at least one RXU context in a receiving device for every TXU in the other devices that may send a message to the receiving device. If any of the TXUs are using Type 11 messages, they should be allocated 1.25 RXU contexts in the receiving device due to possible delays in the message responses that may cause slightly more RXU contexts to be used in the receiving device.

Workaround: None.

Advisory 23 — *SRIO Direct I/O NREAD Data Corruption Issue*

Revision(s) Affected: 1.0, 1.0A

Details: When SRIO direct I/O traffic in a system causes error conditions like timeouts or error responses to be generated, it is possible that an error in the SRIO module's internal logic will cause NREAD commands to return their read data to an incorrect address, thus potentially causing data corruption in the system.The issue only occurs when multiple LSUs are in use.

Workaround 1: Limit the size of all NREAD transactions to 256 bytes so they fit in a single SRIO message. This will avoid the internal error in the SRIO module that causes the issue.

Workaround 2: Use only one LSU within the system. This will prevent the corruption of LSU data used to return the NREAD payload to the correct address.

Advisory 24 — *TAC P-CCPCH QPSK Symbol Data Mode with STTD Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: When processing P-CCPCH channels on the TAC (Transmit Accelerator) using QPSK symbol data format mode with STTD enabled, the P-CCPCH channel output is incorrect. The QPSK+DTX symbol data format mode is not affected by this issue.

Workaround: To work around this issue, the QPSK+DTX symbol data format mode can be used for P-CCPCH all the time, or at least when STTD is enabled. As the P-CCPCH uses a spreading factor of 256, ten additional DTX 32-bit all-zero DTX words need to be interleaved by the application software to use the QPSK+DTX symbol data format.

Advisory 25 — *AIF2 GSM T1, T2, T3 Miscalculation Issue*

Revision(s) Affected: 1.0, 1.0A

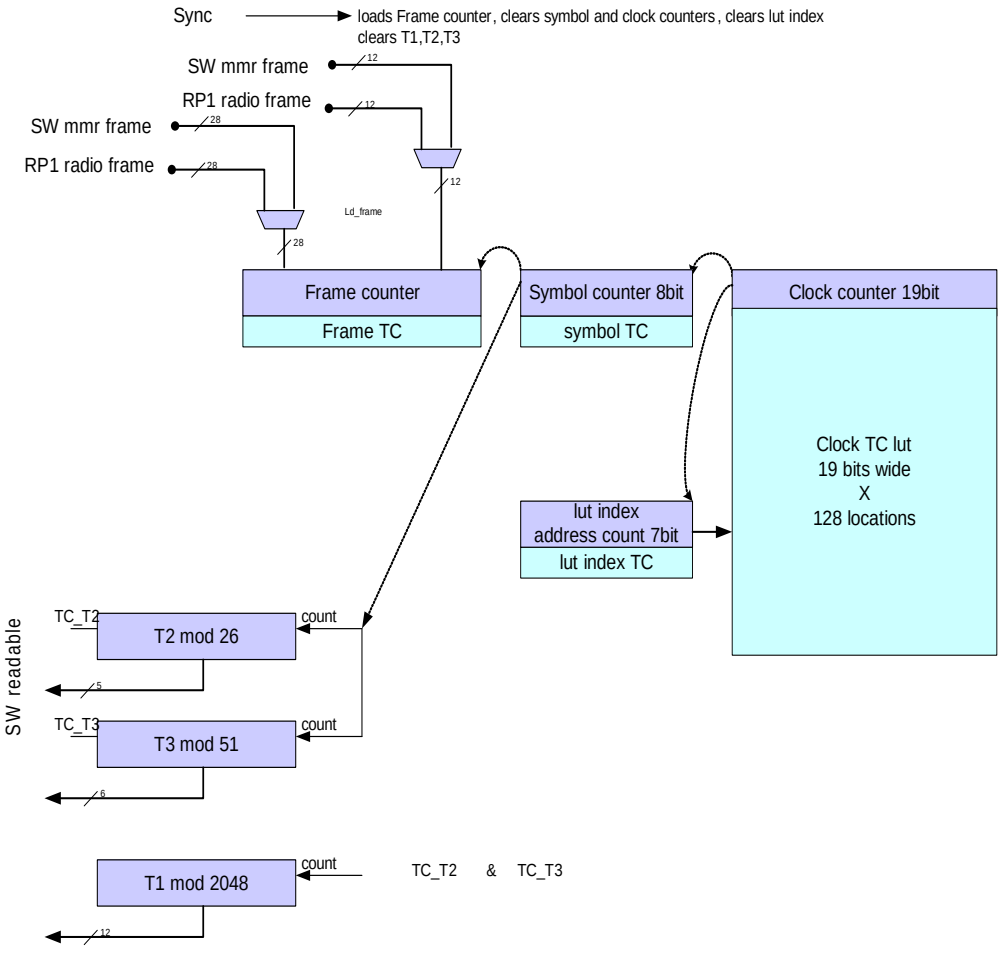
Details: The GSM T1, T2, and T3 counters in the AIF2 timer are designed to increment by one every time the radio timer frame count increments by 1. However, this is not correct operation because the GSM uses 104 as a symbol number terminal count to make it align with the 60-ms Phy frame boundary, even though the original GSM frame has only13 symbols inside. As a result, the T counters have a much longer period between steps.

The GSM has a group of three timers that are incremented on a GSM frame-by-frame basis. They will be incremented every time the symbol (GSM slot) counter wraps around. These are the T1, T2, and T3 timers, and they are fixed as shown:

- T1 = modulo 2048
- T2 = modulo 26
- T3 = modulo 51

Figure 8 shows how T1, T2, and T3 counters operate with the radio timer.

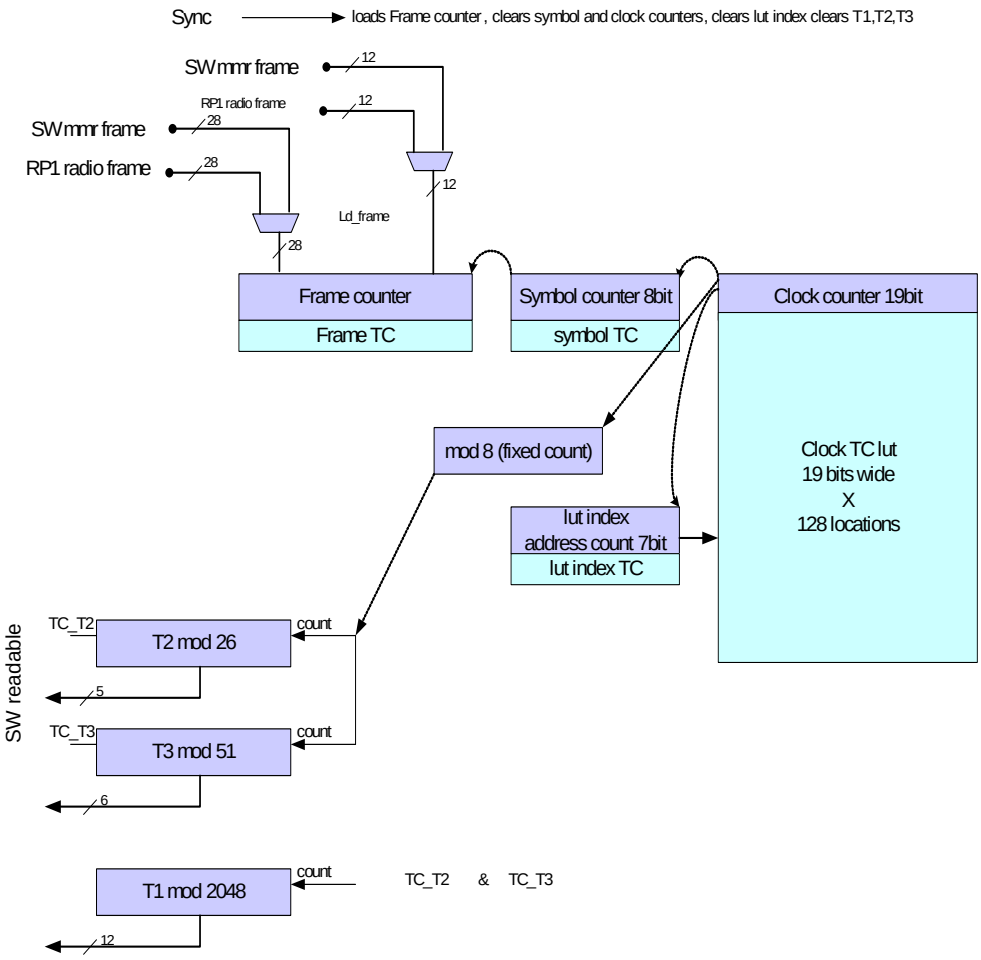
Figure 8 Radio Timer with GSM Time



However, one architecture problem was found while running the GSM system level test. The radio timer frame counter is programmed to count 60 ms, which is 13 GSM frame lengths. This configuration had to be done, because 60 ms is the first n × 10 ms frame boundary which aligns to the OBSAI/CPRI PHY frame boundary (10 ms). In this case, the radio timer should have 104 as a symbol terminal count because there are a total of 104 GSM time slots within 60 ms. In this case, the radio timer should have 104 as a symbol terminal count because there are 104 GSM time slots within 60 ms. With a 104 symbol terminal count, there are not enough calculations to show the correct values on the T1, T2, and T3 counters.

To fix this problem, it is necessary to add a mod 8 counter that increases its value every radio symbol time. When the mod 8 count wraps around, the T1, T2, and T3 counters are incremented by 1. As a safety feature, the mod 8 counter resets to 0 every time the radio frame count is incremented by 1. The T1, T2, and T3 counters reset only when the radio timer is fully re-synchronized. Figure 9 shows how the current GSM T counter design should be changed.

Figure 9 GSM T Counter Design Change



Workaround: An additional AT event can be assigned to generate an event every GSM slot time. The ISR may check mod 8 count per visit and the T2 and T3 count array can be updated manually when the mod 8 count wraps around. TC_T1 count can be calculated by a bitwise AND operation between TC_T2 and TC_T3.

Advisory 26 *AIF2 Multiple DBMR Operation Issue*

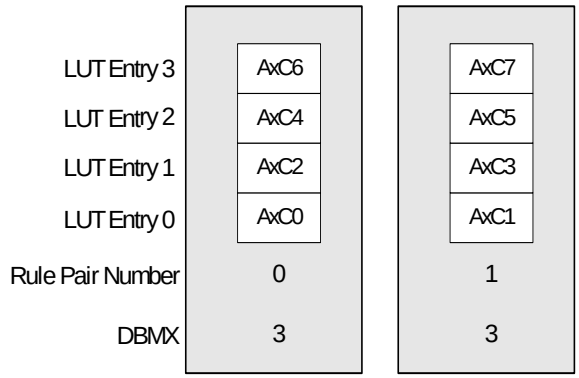
Revision(s) Affected: 1.0, 1.0A

Details: The antenna streams for a link can be separated into multiple rules (modulo and dual bitmap rule (DBMR)). Each rule uses multiple entries (AxC or packet channels) in its lookup table. The lookup table index was reset to 0 after each visit for all but the last rule when users use multiple numbers of rules together within one link.

For OBSAI (Open Base Station Architecture Initiative), the protocol encoder supports two kinds of transmission rules—modulo and DBRM—to interleave multiple channel data into one link. There are 64 modulo rules and 64 DBMR rules. When using both rules together they share the same index numbers. If the user sets only one modulo rule and DBMR for a specific link, it does not cause any lookup table confusion and perfectly supports multiple numbers of entries (channels) with the correct interleaving. However, if the user assigns multiple numbers of rules with multiple entries (channels), the operation does not work correctly. The lookup table index always resets to 0 after visiting the rules except for the last rule pair.

Figure 10 shows the expected framing order for a modulo and DBMR pair and the actual framing order using the current hardware.

Figure 10 OBSAI Message Frame Ordering for Two Rule Pairs



Expected framing order: **AxC0**, AxC1, **AxC2**, AxC3, **AxC4**, AxC5, **AxC6**, AxC7

Framing order of current hardware **AxC0**, AxC1, **AxC0**, AxC3, **AxC0**, AxC5, **AxC0**, AxC7

This is a very simple example of a two rule pair case. In the case of rule 0, only AxC0, which is in entry 0, is inserted into the frame and AxC2, AxC4, AxC6 data is ignored. The entry index works perfectly only with the last rule.

Workaround: None.

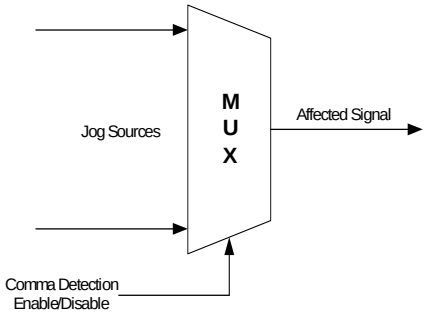
Advisory 27 AIF2 SerDes Comma Alignment Failure Issue

Revision(s) Affected: 1.0, 1.0A

Details: Data corruption encountered on AIF2 lane0 to lane5 (any one or multiple lanes at a time) causes a loss of frame synchronization. The AIF2 SerDes incorporates a comma (K28) alignment mechanism to identify symbol boundaries in 8B10B-encoded data. Comma detection can be disabled after alignment has been achieved. Due to a circuit design issue, disabling the comma detection can corrupt the data alignment. If AIF2 RM (Receive Media Access Control) sd_auto_align_en is enabled, RM disables comma detection after frame sync is achieved. Due to this issue, the data alignment is corrupted and AIF2, once again, goes out of frame sync.

SerDes alignment is achieved by *jogging* one bit at a time until the required alignment is achieved. As an alternative to the comma alignment mechanism, jogs can be requested directly. As shown in Figure 11, the jog request source is selected by a mux, which is controlled by whether comma detection is enabled or disabled.

Figure 11 Affected Signal



The jog request (mux output), which is shown as Affected Signal in Figure 11, is fed to a synchronizer located elsewhere in the SerDes. This signal can glitch while disabling comma detection. Depending on the PVT and line frequency, this glitch can be misinterpreted as a spurious jog request, which can corrupt the alignment. PVT dependency is such that not all devices will exhibit the issue.

All the six lanes are affected with this issue. AIF2 has two SerDes modules inside. One module has lane0 to lane3 and the other has lane4 and lane5. The glitch is dependent on layout differences between these two macros so it is possible that only some of the six lanes will show this issue and some will not.

No other peripherals that are using similar SerDes are affected by this. SRIO does not disable the comma detection. HyperLink does not use the SerDes comma alignment.

All alignment is handled internal to HyperLink module. NetCP and PCIe have a different SerDes module than the one that is used in the AIF2 and it is not affected by this issue.

Workaround: A software workaround is described below.

Keep the RM SD auto_align_en disabled and keep the SD ALIGN enabled. This can be done by setting the sd_auto_align_en bit field of the RM_LK_CFG0 register to 0 for each link and by setting the ALIGN field of the SD_RX_R1_CFG register to 01b for each link.

Table 9 and Table 10, show the RM Link Config and SD Rx Config register and bit-field snapshots from the *AIF2 User Guide*:

Table 9 RM Link Config Register Snapshot

Bits	Field Name	Type	Description
12	sd_auto_align_en	READ_WRITE	Enables the RM to automatically disable SerDes symbol alignment when the receiver state machine reaches state ST3 0 = Disable auto alignment 1 = Enable auto alignment
RM_LK_CFG0[0]			**Address [0x50000]**
RM_LK_CFG0[1]			**Address [0x50800]**
RM_LK_CFG0[2]			**Address [0x51000]**
RM_LK_CFG0[3]			**Address [0x51800]**
RM_LK_CFG0[4]			**Address [0x52000]**
RM_LK_CFG0[5]			**Address [0x52800]**

Table 10 RM Link Config Register Snapshot

Bits	Field Name	Type	Description
5-4	ALIGN	READ_WRITE	The receiver frame synchronizer must control the byte alignment feature of each receiver so that a *jog* can be initiated based on detection of non-alignment. The ALIGN bits must be set to 2b00 for normal operation. 00 = Alignment Disabled 01 = Comma alignment enabled 10 = Alignment jog (The symbol alignment will be adjusted by one bit position when this mode is selected)
SD_RX_R1_CFG[0]			**Address [0x8004]**
SD_RX_R1_CFG[1]			**Address [0x8804]**
SD_RX_R1_CFG[2]			**Address [0x9004]**
SD_RX_R1_CFG[3]			**Address [0x9804]**
SD_RX_R1_CFG[4]			**Address [0xA004]**
SD_RX_R1_CFG[4]			**Address [0xA804]**

The reason for turning the comma detection off after getting a frame sync is that a bit error can cause a faulty comma detect and an incorrect bit alignment. Typically, the practice is to disable the comma detect during normal operation. But with the above workaround, it is recommended to keep comma detection enabled; hence, a bit error may cause the faulty comma detect. It may cause a virtual reset of the AIF2 interface and cause a loss of network connection due to a large number of line code violations before the next possible comma character.

The probability that a bit error could cause a bit alignment may be in the time frame of years. To help this problem, it might be best to set the number of consecutive code violation blocks (in RM_LK_CFG2 register) that reset the state machine to a very high number (> 400 for OBSAI & Max threshold for CPRI). This will most likely keep the state machine in sync in the case that a bit error causes an improper alignment. However, a faulty alignment will cause some data to be corrupted.

CSL Workaround Release Version:

CSL 1.0.0.24 and PDK 1.0.0.13 will have the Workaround added in it.

Advisory 28 SRIO Control Symbols Are Sent More Often Than Required Issue

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: Control symbols are SRIO physical layer message elements used to manage link maintenance, packet delimiting, packet acknowledgment, error reporting, and error recovery. Control symbols are used to manage the flow of transactions in the SRIO physical interconnect. The SRIO input status control symbols communicate the status of the physical link and packets in flight between the two SRIO link partners.

The bandwidth of the SRIO link is reduced because status control symbols are sent more often than required. Worst case impact is a 2.73 percent reduction in bandwidth for a 1× port operating at 1.25 Gbaud. This impact is reduced to 0.1 percent for a 4× port operating at 5 Gbaud. More details about this impact on various lane speed and port configurations can be found in Table 11.

Table 11 Impact on Various Lane Speeds

	Percentage of Bandwidth Reduction		
Lane Speed (Gbaud)	**1x Port**	**2x Port**	**4x Port**
1.25	2.73	1.37	0.68
2.5	1.17	0.59	0.29
3.125	0.86	0.43	0.21
5.0	0.39	0.20	0.10

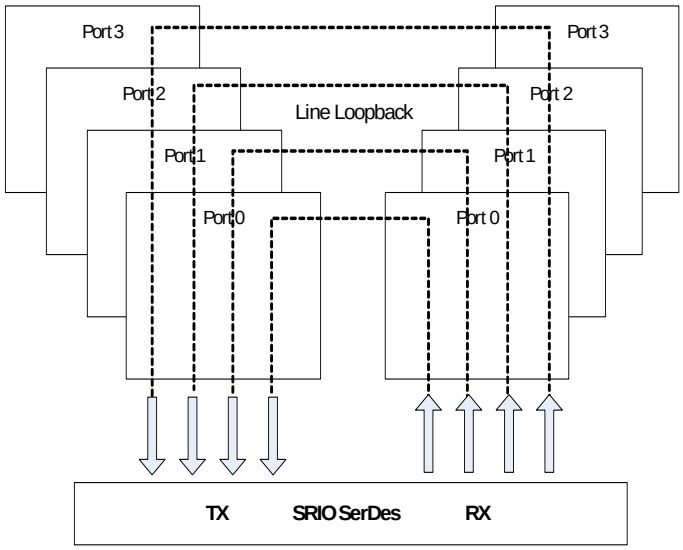
Workaround: None.

Advisory 29 *Corruption of Control Characters In SRIO Line Loopback Mode Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: The SRIO physical layer is configured in line-loopback mode on a per-port basis, by setting the LLB_EN bit in PLM_SP(n)_IMP_SPEC_CTL register. In line-loopback mode, the data from the SerDes receiver is looped back to the SerDes transmitter. Figure 12 shows the line loopback from SerDes RX to SerDes TX:

Figure 12 SRIO SerDes in Loopback Mode



The port does not provide any clock compensation when line loopback is enabled. The transmit clock must be externally synchronized with the receive clock. There is only a small FIFO that can compensate for PLL jitter or wander. Hence, correct operation of line loopback requires that the link partners use the same reference clock for the SRIO physical layer, in order to avoid overruns or underruns due to clock frequency mismatch between the link partners. As a result, line loopback mode is generally restricted to validation and qualification of board signal integrity in a lab environment.

When line loopback is enabled on one or more SRIO ports, any valid 10b code group that decodes to an illegal control character as defined by the RapidIO Specification (Revision 2.1) and whose most significant bit is 0, will be corrupted on transmission. This issue can be summarized as follows:

- 10b code group -> Legal control character -> No problem
- 10b code group -> Illegal control character and most significant bit is 0 -> Corruption

Workaround: Instead of using PRBS sequences, users can qualify boards by using RapidIO-compliant data on the link and monitoring either per-lane error counters or port-level error counters. RapidIO-compliant data is less stressful than PRBS sequences, as the RapidIO-complaint 10b data has shorter 0s and 1s run lengths than PRBS sequences. Hence, RapidIO-compliant data represent a more accurate stimulus for this test. This should be acceptable for users whose RapidIO links are of short reach, which can be either 20 cm + 1 connector or 30 cm without a connector.

Advisory 30 — *SerDes Transit Signals Pass ESD-CDM up to ±150 V Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: All data manual specifications associated with the SerDes high-speed functional pins are guaranteed to a maximum component Electrostatic Discharge - Charged Device Model (ESD-CDM) pulse threshold of 150 V.

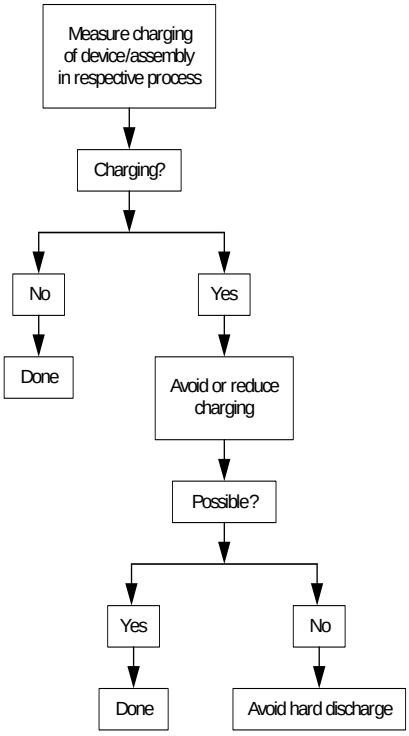
Due to the sensitive nature of the SerDes high-speed pins, exceeding component ESD CDM stress pulses of 150 V on the functional pins listed below may cause permanent changes to the output swing and the de-emphasis behavior associated with these pins.

The following is the list of pins that fall into this category. See the device-specific data manual to see if these pins are present:

- Serial HyperLink Transmit Data Pins
  - MCMTXN0
  - MCMTXP0
  - MCMTXN1
  - MCMTXP1
  - MCMTXN2
  - MCMTXP2
  - MCMTXN3
  - MCMTXP3
- PCI Express Transmit Data Pins
  - PCIETXN0
  - PCIETXP0
  - PCIETXN1
  - PCIETXP1
- Serial RapidIO Transmit Data Pins
  - RIOTXN0
  - RIOTXP0
  - RIOTXN1
  - RIOTXP1
  - RIOTXN2
  - RIOTXP2
  - RIOTXN3
  - RIOTXP3
- Ethernet MAC SGMII Transmit Data Pins
  - SGMII0TXN
  - SGMII0TXP
  - SGMII1TXN
  - SGMII1TXP

Workaround: While there is no strict workaround for this issue, there are several ways to analyze the risk with regards to CDM. Figure 13 shows the basic flow for analyzing this risk:

Figure 13 Basic Flow for Analyzing CDM Risk



Using generally accepted good practices during the assembly process can help to minimize the likelihood of a hard discharge. These practices include:

- The use of an ionizer near the PCB before, during, and after placement of parts
- The use of grounded, conductive/dissipative suction cups when using pick-and-place machines
- The use of dissipative materials for downholder pins and/or plastic covers as well as two-stage pogo-pins while performing in-circuit-test.

Advisory 31 — *AIF2 CPRI 8x UL Peak Bandwidth Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: A single CPRI 8x link in RSA mode with all antennas aligned exposes a peak bandwidth limitation when more than 22 AxCs are enabled; this results in data corruption for channels higher than 22. Similarly, multiple links that support more than a total of aligned 44 AxCs result in the same peak bandwidth limitation and data corruption.

Details: A single AIF2 CPRI 8x link can transfer a maximum of 32 WCDMA AxCs for a 15-bit sample and 30 AxCs for a 16-bit sample. When using fewer than 22 AxCs, no problem was detected. However, the problem appeared when more than 22 aligned AxC channels were enabled. Correct data was received for the first 22 AxCs (channel 0 ~ 21), but corrupted values were inserted for AxC channels 23 ~ 31. This problem occurs only on CPRI, WCDMA (DIO) UL RSA mode. OBSAI and CPRI DL mode is fine even with 64 AxCs (two 8x links). The cause of this problem is in the PD (Protocol Decoder) module. When the peak bandwidth exceeds the AIF2 PD staging buffer, buffer corruption occurs.

The peak bandwidth is reduced by either reducing channel count or by offsetting the time for some of the AxC channels so that no more than 22 AxC channels are time-aligned.

Workaround 1: AIF2 PD has the pd_cpri_id_lut[0][0:127] register with the cpri_8wd_offset field. It allows for a 0-to-7 word (or WCMDA chip) offset which is programmed on an AxC-by-AxC basis and only applicable for CPRI. This offset is used in the staging buffer of PD to align word data into Quad Words.

The normal use of this offset is fine control of AxC Offset. In RSA mode of operation, bits [1:0] are the fine offset control and bit [2] must match bit [0] of the AxC offset field. In cases where no AxC offsets are used, these fields are normally programmed to 0.

In the workaround use case, the input CPRI PHY data is not necessarily shifted. Instead, we are effectively introducing an alignment error of 1 chip (4Bytes – 2 Samples) by setting cpri_8wd_offset to 1 for the higher number channels. The data is presented to RAC with a shift of two samples. Both path search and finger de-spread operations work from the same shifted antenna data; the AxC shift results only in one chip additional processing latency (260.4 ns). If the RAC is used for Ue positioning (i.e the absolute delay is estimated, as opposed to the relative delay between the Ue and P-CCPCH, the application must be modified to compensate for the shift, less the Ue would appear 78 meters farther away for the base station.

It is important that all AxCs for a specific Diversity sector be given the same AIF2 delay.

Workaround 2: If there are 40 or fewer total AxC channels and we can control the allocation of channels to two links, we can allocate a maximum of 20 AxC channels to each of two links without needing to use workaround 1.

Advisory 32 — *L2 Cache Corruption During Block and Global Coherence Operations Issue*

Revision(s) Affected: 2.0

Details: Under a specific set of circumstances, L1D or L2 block and global coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back-to-back in the same L2 set:

1. L1D write miss
2. Invalidate or writeback-with-invalidate due to block and global coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block and global coherence operations EXCEPT:

- L1D block writeback
- L1D global writeback
- L2 block writeback
- L2 global writeback

See also: Advisory 14 — Potential L2 Cache Corruption During Block Coherence Operations Issue

Generic Workarounds: The workarounds below are generic and may have performance impacts. Customers are requested to understand the application and see which one suits them better.

Workaround 1: This workaround requires that the memory system be idle during the block and global coherence operations. Therefore, programs must wait for block and global coherence operations to complete before continuing. This applies to L1D and L2 memory block and global coherence operations.

To issue a block coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register.
4. Wait for completion by one of the following methods
   a. Issue an MFENCE instruction (preferred)
   b. Poll the WC register until the word count field reads as 0
5. Perform 16 NOPs.
6. Restore interrupts.

To issue a global coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write 1 to the corresponding global coherence register (L1DINV, L1DWBINV,L2DINV, and L2DWBINV).
3. Wait for completion by one of the following methods
   a. Issue an MFENCE instruction (preferred)

b. Poll the corresponding global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV) until the bit [0] field reads as 0

4. Perform 16 NOPs.
5. Restore interrupts.

Workaround 2: This workaround is also generic, but will allow CPU traffic to go on in parallel with cache coherence operations. To issue a block coherence operation, follow the sequence below:

1. Issue a MFENCE command.
2. Freeze L1D cache.
3. Start L1D WBINV.
4. Restart CPU traffic.

   (CPU operations happen in parallel with WBINV and do not need to wait for cache coherency operation to complete)
5. Poll the WC register until the word count field reads as 0.
6. WBINV completes when word count field reads 0.
7. Issue an MFENCE command.
8. Unfreeze L1D cache.

For more information about the cache control registers (BAR, WC, L1DINV, L1DWBINV, L2DINV, and L2DWBINV) see the *TMS320C66x DSP CorePac User Guide* (literature number SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the *C66x DSP and Instruction Set Reference Guide* (literature number SPRUGH7).

Advisory 33 *Data Corruption on C66x CorePac Initiated PCIe MMRs Read in Big Endian Mode Issue*

Revision(s) Affected: 1.0, 1.0A

Details: When the device operates in big endian mode, the read return data from the PCIe memory-mapped register (MMR) may be corrupted when the C66x CorePac reads the PCIe MMR value directly. Any remote read from the PCIe MMR is not impacted.

Workaround: Use EDMA to read from the PCIe MMR when the device operates in big endian mode. EDMA read access should be limited to 32 bits each time.

Advisory 34 — *AIF2 Line Code Violation Error from B8 SerDes to B4 SerDes Lanes Cross Connection Issue*

Revision(s) Affected: 1.0, 1.0A

Summary: Line Code Violations (LCVs) or 8b10b decode errors can occur between different OBSAI RP3 links or CPRI links. This problem may occur when the B8 SerDes macro links are directly connected to B4 SerDes links of other DSPs in the chain and when the Master Sync (MSYNC) is set to 0. This problem is also dependent on the power supply voltage variation and at very low temperature of the device.

Details: AIF2 has 6 external links. The first four links are connected to B8 SerDes macro and the other two links are connected to B4 macro. In some applications, data can be transmitted across, up to, 6 transmit links. In this case, data is not physically synchronous, but aligned using special 8b10b symbols in the data stream and need to be able to treat all lanes as one logical clock domain to minimize inter-lane skew at differential outputs. This design also simplifies the SoC design. It requires that txbclk outputs from all transmit lanes are logically phase aligned.

Each SerDes Tx has an independent divider generating TX byte clock and use MSYNC input to indicate which link is a master (MSYNC = 1) or slave (MSYNC = 0). Slave links align their TX byte clock divider to their lower numbered neighbor and an alignment pulse is passed from master link to slave link. To achieve timing closure, reference clock skew between macros was required to be within 30ps, but the problem is that the static phase offset of SerDes PLLs is not properly accounted for, so 30ps budget is not enough, which results in tbsync not having enough hold margin. Sometimes, depending on extreme low temperatures with supply voltage variances, this may cause serious trouble like losing symbol alignment which causes LCV errors.

Workaround 1: Set SD_TX_R1_CFG register MSYNC field to 1 on slave macro (B4) Tx lanes until the B4 and B8 macro PLLs both locked and set it back to 0 after PLL lock. This will be the solution for preventing alignment changing.

Workaround 2: Select B8 macro as a source of AIF2 Tx byte clock. This can be done by setting 0 for SD_CLK_SEL_CFG register. This reduces setup margin on SerDes internal bus.

Advisory 35 — *SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue*

Revision(s) Affected: 1.0, 1.0A

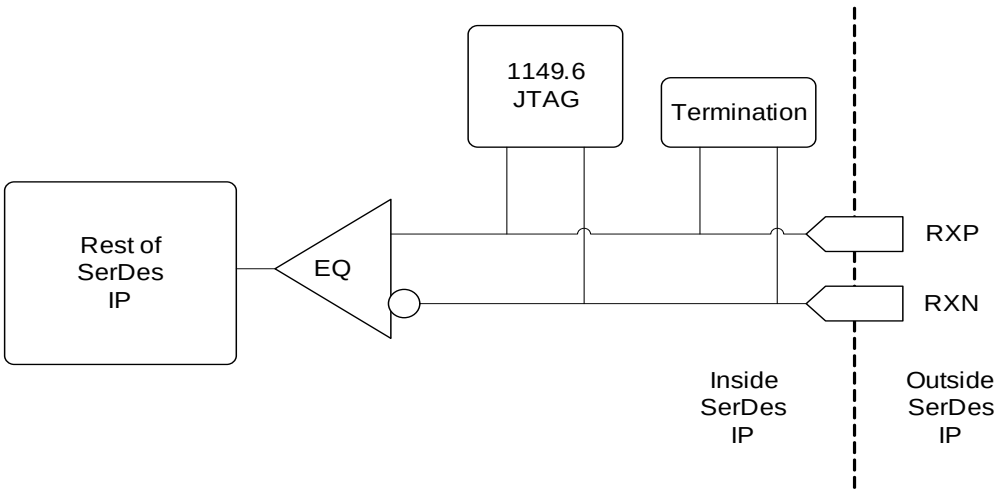
Details: In a production or laboratory test environment the IEEE 1149.1 and IEEE 1149.6 scan-chain cells are used to verify the proper connectivity of the TCI6618 SoC to other board components (other TCI661x, or third-party IC, or connectors) through the use of the JTAG BSDL scan-chain commands.

The IEEE 1149.1 scan-chain cells are designed for use with DC-coupled signals and rely on signal levels; these cells will detect normal VIH or VIL thresholds. Most I/O on TCI661x devices fall under this category. The IEEE 1149.6 scan-chain cells are design for use with AC-coupled signals and rely on signal edge detection; these cells will detect falling edge transitions or rising edge transitions since DC offsets cannot be maintained through an AC-coupled signal path.

The SerDes receivers as shown in Figure 14 for AIF2, SRIO, PCIe, SGMII and Hyperlink are all designed to be AC coupled to their associated transmitters. For these AC-coupled SerDes receivers the IEEE 1149.6 scan-chain provides edge-sensitive sensing.

A problem has been identified in some cases where the AC-JTAG (IEEE 1149.6, edge-sensitive) input cells on receivers do not reliably respond to input edge transitions from an associated transmitter. Because of this failure, the 1149.6 scan-chain users can receive faulty signal levels transitions or no transitions at all.

Figure 14 Simplified SerDes receiver block diagram depicting relative placement of major receiver components



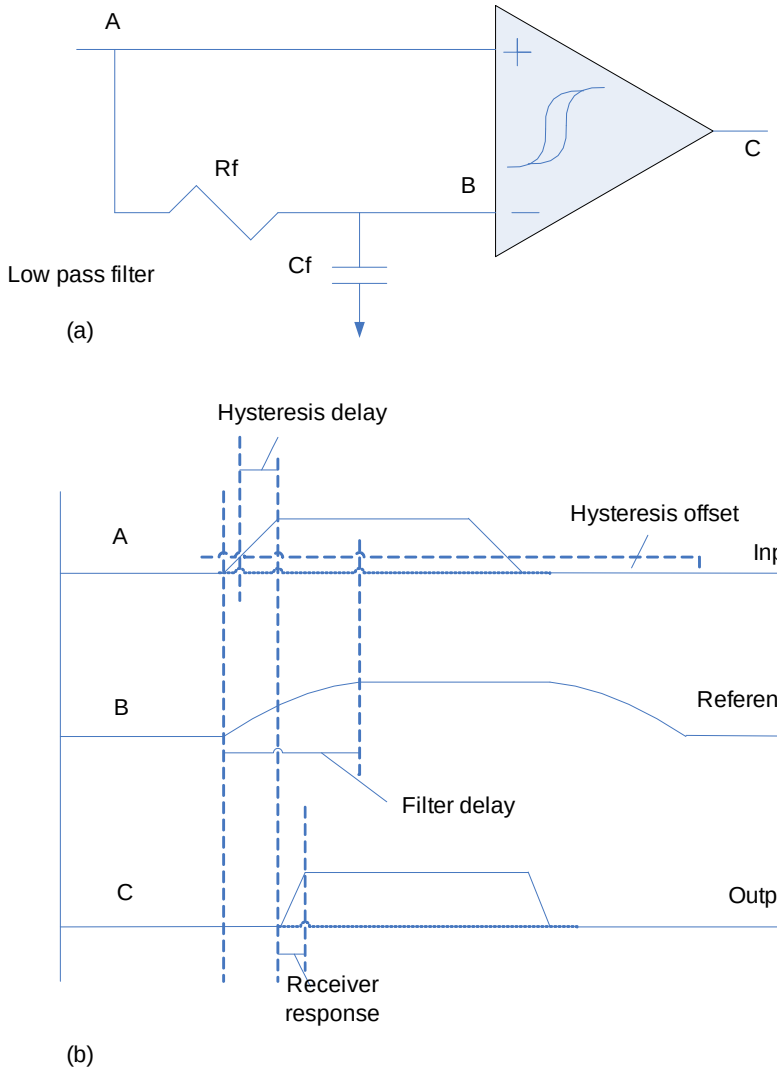
IEEE 1149.1, level-sensitive, input cells show no issue and normal operation of the SerDes receivers is also unaffected by this problem.

The reason for the issue is due to the original AC-JTAG input cell performance being overly sensitive to rise/fall times of the signals interfacing to it and input transition frequency.

The AC-JTAG input cell is based on a self-referenced, hysteretic comparator. See Figure 15. If the SerDes receiver pins are idle (no transitions are applied), the voltage of the pins will settle to the programmed, receiver common-mode voltage (0.80V nominally). From this idle state a single edge transition applied to the AC-JTAG input cell can push both terminals of the AC-JTAG comparator inputs past the VDDT (1.0V) supply voltage level.

When the comparator inputs are saturated the comparator gain is minimized and under certain PVT conditions (strong silicon, 125° C junction temperature, minimum VDDT), the comparator can fail to switch to the proper state, or not respond at all.

Figure 15 Self-referenced Comparator and Response (See Note Below)



Note—To find the transitions in a signal, a self-referencing, hysteretic comparator (a) receives a signal, A, and a delayed version of the signal, B (b). The comparator responds to an ac-coupled signal with a reconstruction of the original (dc) signal, C.

Workaround: Only one potential workaround has been identified so far. The failure mode of the AC-JTAG comparator is most pronounced by single edge transitions at low frequency as this creates the highest likelihood of the comparator inputs being saturated and switching incorrectly. More frequent transitions can result in proper switching of the comparator output.

The basic procedure is to initiate multiple edge transitions and then read the state of the RX input cell in question. However, due to the nature of the failure, the exact number would have to be determined experimentally for a given device.

Additionally, keeping the temperature of the device lower in the operating range and the VDDT voltage higher in the operating range can help avoid the issue.

Advisory 36 — *System Reset Operation Disconnects SoC from CCS Issue*

Revision(s) Affected: 2.0

Details: The CCS connection to targets will fail after a system reset is issued via CCS. The CCS connection to targets will also fail after $\overline{\text{RESET}}$ reset of the device. A system reset, issued from CCS or by the $\overline{\text{RESET}}$ pin, can cause power reset to all C66x CorePacs and can cause the hardware states of debug logic (including hardware breakpoints) to get cleared. The result is that any existing CCS connection to those targets will get corrupted, terminating further access to the target.

Workaround 1: A new configuration option called Domain Power Loss Mode is added in the CCS target configuration for enabling the debug software to detect and handle the power loss event automatically.

To enable this option, in the CCS target configuration window, click on the sub-path of ICEPICK_D for each individual C66x CorePacs. Then click on the property option **Domain Power Loss Mode** and select **Auto**.

The support for this new option will be released in the emupack update v5.0.586.0 or newer, patched to CCS5.1 GA.

Workaround 2: Before issuing a system reset, disconnect CCS from all DSP targets, issue the system reset, then reconnect CCS to the targets to continue debug operations.

Advisory 37 — *Power Domains Hang During Power-Up at the Same Time as a RESET (Hard Reset) Received Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: Certain power domains, like those mentioned below, have multiple RAMs and the controllers associated with them are daisy-chained. This issue occurs whenever the software is powering up one of the power domains and at the same time a RESET (hard reset) is received.

Details: Power domains affected: Power Domain 7 (MSMC RAM), Power Domain 8 (RAC_A, RAC_B, and TAC) and Power Domains 13-16 (C66x Core 0-3, L1/L2 RAMs).

The Global PSC state machine associated with the power domain in transition hangs; as a result, the device does not come out of reset as expected. The RESETSTAT pin status will be stuck low indicating that the device is in reset. The only option to exit from this hang condition is to apply a RESETFULL or POR.

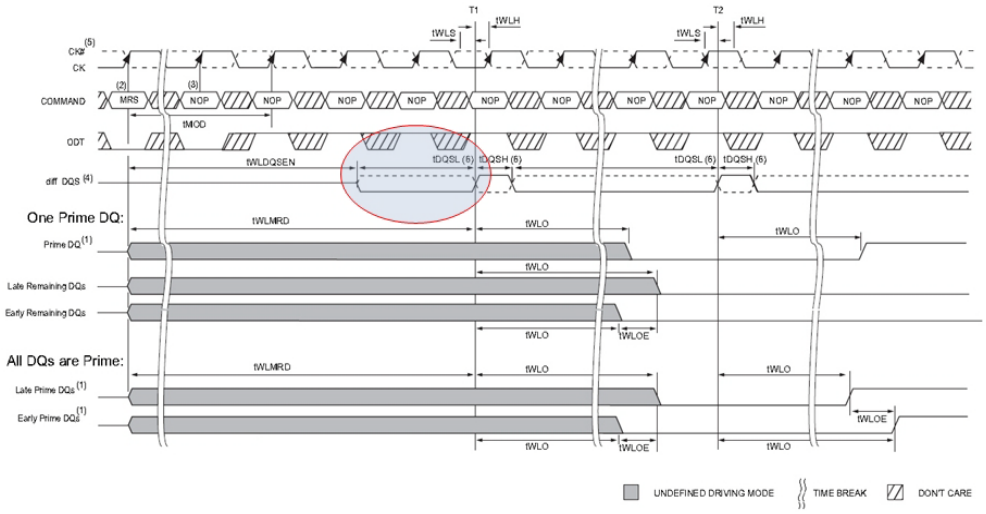
Workaround: Whenever the external host controller applies a RESET (hard reset) to the device, the host is normally expected to wait for the RESETSTAT status pin to toggle from low to high (this indicates that the device is out of reset). If the RESETSTAT pin does not toggle and is stuck at low, the external host controller can infer that this issue has occurred. To come out of this issue, the external host has to apply a RESETFULL or POR to the device. Make sure that the boot configuration pins are re-latched during RESETFULL or POR.

Advisory 38 — *DDR3 Incremental Write Leveling Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: As shown in Figure 16, the DDR3 JEDEC standard requires that the DQS strobe be sent with a preamble (DQS=0) for at least tDQSL during write leveling, prior to the first rising edge of the DQS strobe (DQS=1). It was observed in simulations that the first DQS strobe is sent immediately following the high impedance state of the DQS line and without a preamble. This Z->1 transition on the DQS line may allow incremental write leveling to fail. Note that this issue impacts only the incremental write leveling feature of the DDR3 Memory controller. No issue is expected with automatic write leveling.

Figure 16 Timing Details of Writing Leveling Sequence



NOTES: 1. DRAM has the option to drive leveling feedback on a prime DQ or all DQs. If feedback is driven only on one DQ, the remaining DQs must be driven low, as shown in above Figure, and maintained at this state through out the leveling procedure.
2. MRS: Load MR1 to enter write leveling mode.
3. NOP: NOP or Deselect.
4. diff_DQS is the differential data strobe (DQS, DQS#). Timing reference points are the zero crossings. DQS is shown with solid line, DQS# is shown with dotted line.
5. CK, CK# : CK is shown with solid dark line, where as CK# is drawn with dotted line.
6. DQS, DQS# needs to fulfill minimum pulse width requirements tDQSH(min) and tDQSL(min) as defined for regular Writes; the max pulse width is system dependent.

Workaround: At this time, incremental write leveling is not supported on this device. It is recommended that the incremental write leveling intervals in RDWR_LVL_CTRL and RDWR_LVL_RMP_CTRL be programmed to 0 to disable this feature.

Advisory 39 — *Single MFENCE Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The MFENCE instruction is used to stall the instruction fetch pipeline until the completion of all CPU-triggered memory transactions.

Under very particular circumstances, MFENCE may allow the transaction after the MFENCE to proceed before the preceding STORE completes.

For example,

1. STORE_A
2. MFENCE
3. TRANSACTION_B

The MFENCE implementation stalls the CPU until the memory system asserts that there are no transactions "in flight," i.e. it is idle. This prevents the CPU from proceeding to TRANSACTION_B before STORE_A completes. A small window exists where the memory system prematurely asserts that it is idle when STORE_A moves from L1D to L2 when it is otherwise idle. This can cause incorrect program behavior if TRANSACTION_B must occur strictly after STORE_A. For example, suppose STORE_A writes to DDR3, and TRANSACTION_B triggers an EDMA which reads the location written by STORE_A. MFENCE should guarantee that STORE_A commits before the EDMA executes, so that the EDMA sees the updated value. Due to the issue in this advisory, TRANSACTION_B could trigger the EDMA before STORE_A commits, so that the EDMA sees stale data.

Workaround: Replace a single MFENCE with two MFENCEs back to back. This remedies the issue by resuming the stall in the case where the memory system prematurely indicated that it was idle when STORE_A passed from L1D to L2.

1. STORE_A
2. MFENCE
3. MFENCE
4. TRANSACTION_B

Note on coherence operations:

For the following advisory , double MFENCE can also be used as a workaround in addition to the workarounds already listed:

- Advisory 14—Potential L2 Cache Corruption During Block Coherence Operations Issue
- Advisory 32—L2 Cache Corruption During Block and Global Coherence Operations Issue

Note that the second advisory listed above does not cover L1D and L2 block and global writebacks. If L1D and L2 block and global writebacks are followed by an MFENCE and a transaction that depends on the completion of the writebacks, the double MFENCE workaround should be used.

Please also note that the current release of the MCSDK software package does not include this workaround. It will be included in a future release.

Note—Special Considerations for Trace. When trace generation is expected through software that includes the use of MFENCE, there are additional requirements for the workaround. Trace generation for MFENCE requires that every occurrence of the MFENCE instruction be followed with a NOP and a MARK instruction. The workaround is described in this white paper: http://processors.wiki.ti.com/images/c/c5/TracingMfenceWhitePaper.pdf

Advisory 40 — *Read Exception and Data Corruption Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: Under specific circumstances, a pre-fetch for a cacheable data access (program pre-fetches are not affected) that bypasses L2 can result in a read exception and/or data corruption.

Details: A pre-fetch for a cacheable data access can result in a read exception and/or data corruption when all of the following conditions are satisfied:

1. The address being accessed lies in the range 0x0C000000 – 0xFFFFFFFF and does not lie within the CorePac's global alias 0x1n000000 – 0x1nFFFFFF, where n equals the CorePac ID number as indicated by either the DSP core number register (DNUM) or the MMID field in the L2 configuration register (L2CFG).
2. The MAR register for the given address space enables caching (MAR.PC = 1) and pre-fetch (MAR.PFX = 1).
3. L1D cache is enabled (L1DCFG.L1MODE is non-zero) and not frozen (L1DCC.OPER = 0).
4. The address does not get cached by L2 cache. This can occur for the following reasons:
   a. The address lies in the range 0x0C000000 – 0x0FFFFFFF
   b. The address lies above this range and L2 cache is frozen (L2CFG.L2CC = 1) or disabled (L2CFG.L2MODE = 0)

Before going into a detailed explanation of the root cause, it is worthwhile to understand the different types of XMC pre-fetch hits that can occur in a system.

1. Data Hit – An access matches an allocated entry with successful read data present in buffer. The access is serviced via the pre-fetch buffer.
2. Data Hit Wait – An access matches an allocated entry with outstanding pre-fetch reads. The access is serviced via the pre-fetch buffer when the read returns.
3. Address Hit – An access matches an allocated entry with neither successful read data nor an outstanding pre-fetch. This access is forwarded to the MSMC, but allocation for this stream will continue if applicable.
4. Miss – An access does not match an allocated entry. The access is forwarded to MSMC. If the access hits in the candidate buffer, it will be allocated as a stream.

Of the different types of pre-fetch hits listed above, this failure mode specifically requires an "Address Hit" where the allocated slot contains no data. This can happen due to a number of reasons:

1. If pre-fetches collide in the XMC pipeline, the earlier (in time) pre-fetch will be discarded.
2. When MSMC's data pre-fetch holding buffers are full, MSMC will discard the oldest pre-fetch to eliminate pre-fetch head-of-line blocking and reduce bandwidth expansion from pre-fetch.
3. The pre-fetch buffer is write-invalidate; any write that matches on an active stream invalidates any present pre-fetch data for that address.
4. Pre-fetch returned with unsuccessful read status.

The root cause of the issue is in the way the data pre-fetcher inside the XMC behaves when accessed by L1D directly (not caching in L2) as opposed to when accessed through the L2 controller (allocating in L2 cache):

The data pre-fetcher operates on 128 byte lines (the same as L2 cache line size). However, the L1D has a 64 byte line size (not matching L2 or the pre-fetch buffer).

The hardware operates differently for pre-fetches generated for L1D and L2. The L2 always consumes the entire pre-fetch line at once whereas the L1D can only consume half of the pre-fetch line.

When an L1D access (say, to address A) hits a pre-fetch stream, the pre-fetcher may 'look back' at that stream by generating a re-fetch of the other 64 bytes of the line.

In the failing case, the hardware generates a re-fetch for an L1D access to re-fill half of the 128 byte pre-fetch line, and subsequently re-allocates that pre-fetch line to a new stream due to a subsequent data read (say, to address X). The hardware must sort the resulting pre-fetches by marking older results as stale.

The fault lies in the 'sorting' hardware which can lead to the first pre-fetch not being marked stale under certain boundary conditions (see pathological sequence below). The subsequent fetch access when the pre-fetch access wasn't marked stale will access the stale data that has been cleared (but not yet marked as stale.) This results in the read exception and/or incorrect data being fetched. Thus, the issue occurs only when the re-fetch feature is triggered and only L1D accesses use the re-fetch feature.

Since the L2 always consumes entire pre-fetch lines, it is not susceptible to the fault.

The above pre-fetch behavior leading up to the failure can be illustrated with the help of the following sequence:

Pathological sequence:

1. Initial state of the data pre-fetch buffer: Must have allocated all 8 entries [0-7] as detected streams

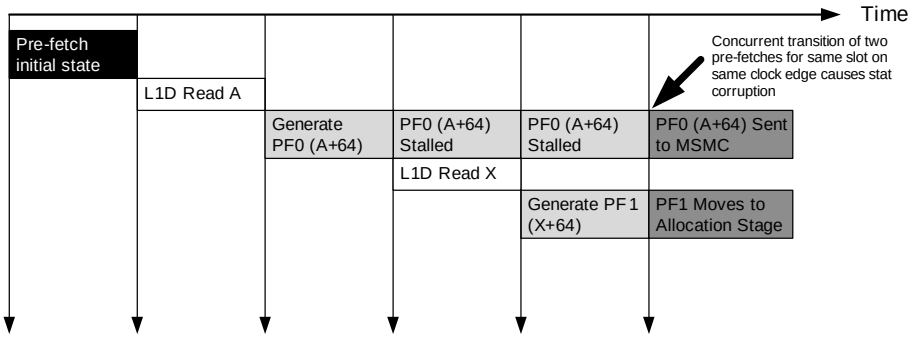
Figure 17 Initial state of pre-fetch buffer

Pre-fetch Buffer Initial State		
Entry	Address	Entry Status
[0]	A,A+64	Stream Valid, No Data Present
[1]	B,B+64	Stream Valid
[2]	C,C+64	Stream Valid
[3]	D,D+64	Stream Valid
[4]	E,E+64	Stream Valid
[5]	F,F+64	Stream Valid
[6]	G,G+64	Stream Valid
[7]	H,H+64	Stream Valid

2. L1D pre-fetchable data read to address A hits pre-fetch entry [0] for early half of 128 byte pre-fetch line
   a. Must be Address Hit only, both 64 byte halves of entry [0] must not contain pre-fetch data.
   b. Entry [0] must be the 'oldest' entry, next entry to reallocate.

3. Data pre-fetcher re-fetch behavior is triggered which generates pre-fetch (PF0) to A+64.
4. PF0 stalls in XMC pipeline until step 7
5. L1D pre-fetchable data read to address X hits in candidate or pre-fetch buffer, triggering allocation of entry [0] to stream starting at X+64
6. A pre-fetch for address X+64 (PF1) is generated for entry [0] early half and followed in a subsequent cycle by X+128 (PF2).
   a. Either PF1 or PF2 can map to the same buffer slot as PF0 (this example maps PF1)
7. PF0 transitions to the XMC output (to MSMC) on the exact same clock edge that PF1 is transitioning to the buffer allocation pipeline stage

Figure 18 Sequence diagram



8. Instead of PF0 being marked stale and thrown away when returned, both PF0 and PF1 are marked valid and outstanding.
9. This leads to two valid outstanding pre-fetches for the same data buffer slot which is an invalid state and can result in unexpected behavior for a subsequent pre-fetchable data access to address X+64.

Depending on whether the subsequent pre-fetchable data access to X+64 arrives before or after the second pre-fetch returns, the result can be a read exception and/or data corruption. The table below describes the different possible outcomes depending on the relative timing between following events:

- PF0 Read Return and Status
- PF1 Read Return and Status and
- Pre-fetchable data access to X+64
  - Column 2 represents the result when the demand access arrives before the second pre-fetch return
  - Column 4 represents the result when the demand access arrives after the second pre-fetch return

Table 12 Possible outcomes depending on relative timing of subsequent pre-fetchable data access to X+64, PF0 return and PF1 return

1. First Pre-Fetch Return		2. X+64 CPU Access	3.Second Pre-Fetch Return		4. X+64 CPU Access
PF0	With Data	Hit, Data Corruption	PF1	With Data	Hit, Correct
PF0	With Data	Hit, Data Corruption	PF1	Cancelled	Hit, Exception
PF0	Cancelled	Miss, Correct	PF1	With Data	Miss, Correct
PF0	Cancelled	Miss, Correct	PF1	Cancelled	Miss, Correct
PF1	With Data	Hit, Correct	PF0	With Data	Hit, Data Corruption
PF1	With Data	Hit, Correct	PF0	Cancelled	Hit, Exception
PF1	Cancelled	Miss, Correct	PF0	With Data	Miss, Correct
PF1	Cancelled	Miss, Correct	PF0	Cancelled	Miss, Correct
Color Key:		Always Correct Operation	Sometimes Correct Operation		Never Correct Operation

Workaround: Software must disable the PFX bits in the MARs for address ranges 0x0C000000 – 0x0FFFFFFF corresponding to cacheable data (MARs can be written to only in supervisor mode. The PFX bit for MARs 12-15 which define attributes for 0x0C000000 – 0x0FFFFFFF is set to 1 by default). This will disable pre-fetching for accesses to those addresses, while still allowing those accesses to be cached in L1D.

If pre-fetching for MSMC SRAM and other memory spaces is desired, it can still be done provided they are remapped to a space other than 0x0C00 0000 – 0x0FFF FFFF within the MPAX registers (the remapped MSMC will act as shared level 3 memory and will be cacheable in L1D and L2).

The L2 cache must remain on and set to a cache size greater than zero, and must not be frozen when accessing pre-fetchable data, otherwise XMC will apply the previously described L1D-specific behavior for the data prefetcher and subject the system to the same issue.

Advisory 41 — *Incorrect Output from TAC for HS-PDSCH and P-CCPCH Channels in Certain Spreader Allocation Sequences Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: In certain sequences of allocation of spreaders of TAC, TAC produces incorrect output for HS-PDSCH and P-CCPCH channels.

Details:

For HS-PDSCH:

3GPP release 5 introduced HSDPA for high-speed downlink data traffic. HSDPA transmission (at physical layer) involves several data channels (HS-PDSCH) and several control channels (HS-SCCH) for each cell. The data for HSDPA is transmitted in time units (TTI) of 2 ms called sub-frames. A sub-frame consists of 7680 chips in WCDMA time terminology. In each sub-frame, HS-PDSCH channels are organized into groups where each group of channels transmits data for a certain user. The number and grouping of HS-PDSCH channels serving a particular user can change from sub-frame to sub-frame. The physical layer standards require all HS-PDSCH channels serving a user in a sub-frame to have the same gain and diversity weights (if diversity is enabled).

TAC supports spreaders of type HS-PDSCH and HS-SCCH to generate data for respective channels. To support grouping of HS-PDSCH channels, TAC supports 'dummy' spreaders which are responsible for obtaining gain and diversity weights for each group of HS-PDSCH channels.

The use of TAC in a BTS solution is facilitated by accompanying software called TAC functional library (TAC FL). TAC FL has APIs that application software calls at run time to set up different channels. TAC FL in turn has responsibility of allocation and configuration of TAC resources such as spreaders for generating the antenna output samples for the channels.

The current TAC issue causes incorrect antenna output to be generated for a HS-PDSCH channel in two cases.

Case 1:

In the first case, the problem happens only when using 64QAM modulation in the last sub-frame of a channel frame (10ms interval in the channel time domain). Here, when either the latest non-idle spreader preceding (spreader Id sequence-wise) the HS-PDSCH spreader or the first non-idle spreader following (spreader Id sequence-wise) the HS-PDSCH spreader has a different channel timing relative to the HS-PDSCH spreader, half of the last sub-frame of the HS-PDSCH's channel frame may have incorrect output.

Case 2:

The second case affects HS-PDSCH for all modulations and when diversity or beamforming is not enabled for HS-PDSCH. In this case, if first non-idle spreader following (spreader Id sequence-wise) has different channel timing relative to the HS-PDSCH spreader, then the output of HS-PDSCH spreader may be incorrect.

We refer to the HS-PDSCH spreader with incorrect output as the affected spreader. We refer to the spreader with the different channel timing as the affecting spreader.

For P-CCPCH:

P-CCPCH is one of the common channels defined by 3GPP R99 specifications. STTD diversity may be enabled for P-CCPCH which causes transmission on a secondary diversity stream in addition to the primary stream. The diversity stream will use the same input data as primary stream, but the input data is STTD encoded before spreading operation as defined by the specifications.

The current TAC issue causes incorrect output on the diversity stream when the STTD diversity is enabled on P-CCPCH spreader and the first non-idle spreader following it (spreader Id sequence-wise) has different channel timing relative to the P-CCPCH channel. The incorrect output occurs in two blocks of 256 chips every 10ms.

Again, we refer to the P-CCPCH spreader with incorrect output as the affected spreader and the spreader with the different channel timing as the affecting spreader.

Note that for both HS-PDSCH and P-CCPCH cases, there may be some idle spreaders between the affecting and the affected spreaders, and the problem may still happen.

Workaround: The TAC issue causes antenna output to be bad for HS-PDSCH and P-CCPCH channels in certain spreader allocation sequences as described above. One workaround is to ensure for HS-PDSCH spreaders that the spreaders immediately preceding and immediately following them have the same timing, while for P-CCPCH spreaders that the spreaders immediately following them have the same timing. Since TAC FL allocates spreaders upon API calls from application software, the spreader allocation logic in TAC FL is being changed to ensure such spreader allocation sequences that cause the problem do not occur. The change in the spreader allocation sequences will have impact in channel capacity supported by TAC/TAC FL solution. The channel capacity impact is being analyzed.

To work around the issue, it is recommended that application software in BTS solution integrate with the updated version of TAC FL.

Advisory 42 — *Incorrect Output from TAC for a Fetching Spreader on a Collision between Input Header's Fetch Write to the Spreader's SPM and Software's Write to SPM Issue*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: TAC produces incorrect output in an access slot or a frame or a sub-frame for a fetching spreader when there is a collision on a TAC clock cycle between TAC fetch process' write of access slot/frame/sub-frame header fields to the spreader's SPM and software's write to any spreader's SPM word.

Details:

TAC spreader overview:

TAC supports WCDMA downlink operations conforming to 3GPP physical layer specifications. TAC has several internal hardware components the most important of which are the spreaders. Spreaders are responsible for spreading and scrambling operations for different channels while also interacting with other TAC internal hardware components to perform supplementary operations such as gain application or diversity weight application for the channels. Each spreader may be independently configured to process a specific WCDMA physical layer channel type.

The configuration of a spreader is through its control memories namely spreader parameter memory (SPM) and spreader request memory (SRM). SPM supports configuration including channel type and parameters specific to the channel type. For example, a spreader may be configured to be of DPCH channel type by setting the channel type field of the spreader's SPM to DPCH. Further, parameters of DPCH channel type such as channel time offset, modulation scheme and input frame buffer pointers can be configured in the SPM. SRM is used to configure when the spreader has to change state (such as become active from idle) and also whether the spreader must produce output for one output stream or two output streams.

Spreaders may be fetching or non-fetching depending on whether the channel type needs to fetch and process input data or not. P-SCH, S-SCH, P-CPICH and S-CPICH channel spreaders are non-fetching while other channel spreaders are fetching. In case of fetching spreaders, the input data is prepared by an external source (software) in memory on an access slot/frame/sub-frame basis and processed by the spreaders as such. Access slot based channels include AICH and E-AICH channels, frame based channels include P-CCPCH, S-CCPCH, PICH, MICH, DPCH and F-DPCH while sub-frame based channels include HSPA channels (HS-SCCH, HS-PDSCH, E-HICH/RGCH, E-AGCH). The input data for each access slot/frame/sub-frame consists of a header and a payload. Part of the fetch process for a spreader consists of writing some fields from access slot/frame/sub-frame header to the SPM. Some header fields are written to SPM because 3GPP specifications require such fields to be changeable on an access slot/frame/sub-frame basis, and the values of these fields in header affect the spreader data path behavior for that access slot/frame/sub-frame. The following are fields of header that are written to the SPM by the fetch process:

Table 13 Header Fields Written to SPM by the Fetch Process

Channel Type	Fetched Header Fields Written to SPM
DPCH	Slot mask, DTX, compressed mode type, nextBlockPtr
MICH/PICH/P-CCPCH	Channel gain, DTX
S-CCPCH[1]	DTX, nextBlockPtr
F-DPCH	Slot mask, DTX
AICH/E-AICH	DTX
HS-PDSCH	Message handler RLS Id, Channelization code, Modulation, DTX, diversity mode, nextBlockPtr
HS-SCCH	Diversity mode, DTX
E-HICH/RGCH	DTX
E-AGCH	DTX

1. Even though for S-CCPCH, Channel gain field is written by header fetch process to SPM in every frame, Channel gain field is not affected by the current TAC issue.

Note that SPM has other fields written by software at channel setup time that do not change on an access slot/frame/sub-frame basis.

The current issue relates to writes to SPM by the fetch process and writes to SPM by software.

TAC issue:
When the TAC clock cycle in which the TAC fetch process writes the fields of an access slot/frame/sub-frame header to a spreader's SPM coincides with the TAC clock cycle in which software writes to SPM of any spreader, then the write by the header fetch process will not happen.

Note that a fetching spreader fetches header periodically on an access slot/frame/sub-frame basis as long as the spreader's data path processing (spreading/scrambling operations) is active. However, writes to SPM by software to configure a spreader are random (determined by base station system scenarios).

Issue consequences:
If the current TAC issue results in a missing write to SPM by the header fetch process in an access slot/frame/sub-frame, this may cause incorrect output from the spreader for that access slot/frame/sub-frame. Specifically, in case of a missing write to SPM by the header fetch process, the spreader data path will use the same values for above header fields in the access slot/frame/sub-frame (N) as were used two access slots/frames/sub-frames before (N-2). The TAC issue does not have any consequence if the values of above header fields are not expected to change across access slots/frames/sub-frames (specifically, if the system populates header values for access slot/frame/sub-frame N that has missing header fetch write same as access slot/frame/sub-frame N-2).

Writes to SPM by software happen when a new channel needs to be set up in the system or an existing channel needs to be reconfigured with new parameters. While the writes to SPM by software can be at random times, the likelihood of missing write to SPM by header fetch process is more in a HSPA scenario compared to a non-HSPA scenario.

This is because in case of a HSPA scenario, header fetches are more frequent (sub-frame basis) than in case of frame based channels. Also, the likelihood is more in a system where there are frequent HSPA reconfigurations (more writes to SPM by software) than in a system where that is not the case.

Workaround: Analysis for the TAC issue was done, but no software workaround has been identified.

Advisory 43 — *False DDR3 Write ECC Error Reported Under Certain Conditions*

Revision(s) Affected: 1.0, 2.0

Details

An L1D or L2 block writeback or writeback invalidate operation to ECC protected DDR3 space will flag a DDR3 write ECC error, even though neither the data nor the ECC values stored in the SDRAM will be corrupted.

The write ECC error interrupt can be enabled by setting the WR_ECC_ERR_SYS bit in the Interrupt Enable Set Register (IRQENABLE_ SET_SYS) of the DDR3 controller.

Under normal conditions, a write access performed within the ECC protected address range to a 64-bit aligned address with a byte count that is 64-bit quanta is not expected to flag a write ECC error interrupt. The C66x cache controller always operates on whole cache lines, which are 128 bytes for the L2 cache and 64 bytes for L1D cache. However, a block writeback or writeback invalidate always generates a bounding single byte write (with its byte enables disabled) to the last address in that block.

This single byte write violates both the alignment and quanta conditions causing the DDR3 controller to flag a write ECC error in the Interrupt Raw Status Register (IRQSTATUS_RAW_SYS) register. Since this bounding write is sent with its byte enables disabled, it does not actually reach the DDR memory and does not corrupt the stored data or ECC values. The write ECC error is thus a spurious error since no data or ECC value is actually corrupted.

It should be noted that the DDR3 controller, in response to this sub-quanta write (the bounding single byte write), will report an error on an internal status line to the CPU that executed the writeback. This error status flags the MDMA error interrupt to the CPU and is interpreted as an MDMA data error (the STAT field in the CPU's MDMA Bus Error Register will be set to 0x4).

Note—1: Since no bounding writes are generated with global writeback or global writeback invalidate operations, this issue is limited only to block writebacks to ECC protected region.

Note—2: If the MDMA error flagged by a block coherence operation is followed by a true MDMA error flagged by a master executing a direct sub-quanta write, only the first MDMA error will be captured. Software must clear an MDMA error as possible in order for future errors to be captured.

Workaround 1

In order to differentiate a false write ECC error from a true error generated by alignment/quanta violations, the system should keep track of block writeback/writeback invalidate operations to the ECC protected memory space. If a write ECC error is confirmed for that operation, it can be safely ignored.

There is only a single DDR3 error interrupt that must be processed by one of the C66x cores. Therefore, some special mechanism will be required for the system to keep track of which core performed the block writeback that caused the error. This mechanism may involve checking for the data error reported in the MDMA Bus Error Register.

Note—3: A system must satisfy the alignment/quanta conditions so a true write ECC error is not expected and such errors should be isolated and removed as part of system software evaluation.

Note—4: The system software must clear the write ECC error and MDMA error before they can be re-triggered by any successive error conditions. Note that a race condition can exist if a subsequent ECC error (real or false) or MDMA error interrupt is triggered before the previous interrupt is cleared.

Workaround 2 A global coherence operation can be performed instead of a block operation. It should be noted that a global operation can possibly operate on more cache lines than the block operation, causing a larger than necessary cycle overhead and negatively impact memory system performance.

NOTES The C66x CorePac will receive an MDMA error in response to the DDR3 ECC error. Other masters may also see the DDR3 ECC error when transaction they have sent result in the error. The responses of the C66x CorePac, ARM CorePac, and other masters to the non-zero values returned on rstatus and sstatus due to DDR3 ECC errors are summarized in Table 14.

Table 14 Master Behavior in Response to sstatus/rstatus Flagged Due to DDR3 ECC Errors

Master	Bus Error Returned	Error Status Captured	Alarm Notification
C66x CorePac	status, rstatus	Error status captured in the STAT field in the C66x CorePac's MDMA Bus Error Register will be set to 0x4.	The C66x CorePac's MDMA error interrupt is asserted.
PCIe	status, rstatus	Interrupts are not generated and error status is not logged.	PCIe returns completion abort to requestor only for rstatus error. No completion abort is returned for a write error (sstatus).
EDMA TC	status, rstatus	BUSERR and ERRDET registers capture the error information. The transfer of data from source to destination happens irrespective of error.	Error interrupt is generated if enabled within EDMA.
Multicore Navigator Infrastructure PktDMA	status, rstatus	Error status is not logged.	Error interrupt is not reported
TSIP	status, rstatus	Errors will be stored in the channels' interrupt queue along with the error codes.	TSIP asserts an error event (TSIPx_ERRINTn) when an error is queued for channel 'n'. CorePac[n] will receive TSIPx_ERRINTn.
SRIO	status, rstatus	Error responses set a bit in the AMU_INT_ICSR register based on the CPRIVID of the transaction. The RIO_AMU_ERR_CAPT0, RIO_AMU_ERR_CAPT1 registers will contain the address of the nonposted transaction that failed along with the CPRIVID and CMSTID.	Each bit can be routed by software configuration through the Interrupt Condition Routing Register (ICRR) to a specific ARM or C66x CorePac for error handling. See the device data manual for interrupt mapping.
HyperLink	status, rstatus	When the serial link is active HyperLink will pass the rstatus. rstatus is will be that of the remote slave read.	HyperLink Error interrupt (HyperLink_INT or VUSR_INT) are generated when error is received and are provided to CorePacs as secondary interrupts.
End of Table 14			

Usage Note 1 *AIF2 Protocol Encoder Registers Usage Note*

Revision(s) Affected: 1.0, 1.0A

Details: It is not possible to read the AIF2 Protocol Encoder PE Channel LUT 0 ~ 7 memory mapped registers (MMRs) while AIF2 is processing antenna data. Reading these registers from software or CCS for debug while the AIF2 is processing antenna data will make it appear like the data is moving around from location to location.

Workaround: Disable the AIF2 processing before attempting to read the PE Channel LUT 0 ~ 7 registers in run time.

Usage Note 2 *TAC DL TPC Timing Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: In softer-handover, downlink power updates may be implemented one slot early in a specific use-case, but by the next slot the system corrects itself.

While a UE is in softer-handover, the update for downlink TPC, TFCI, and pilot power offsets and min/max accumulated gain limits for the radio links may be issued up to one slot early. This applies only to the radio links with TX Offsets greater than four chips away from the first radio link in the set. The wireless terminal may measure power a little high/low for one slot and report it to the NodeB.

The issue does not affect F-DPCH-only UE operation, because the P0x values are always 0. Reconfiguration from DPCH radio links to F-DPCH radio links and vice-versa as well as DPCH-only reconfigurations can potentially show this problem within part of the last slot before the reconfiguration boundary.

Workaround: None.

Usage Note 3 *PCIe Device ID Field Reset Value Usage Note*

Revision(s) Affected: 1.0

Details: The PCIe vendor ID / device ID field reset value is incorrect.

Workaround: The vendor ID / device ID PCIe register needs to be set by software with the proper values. The vendor ID should be 0x104C. The device ID should be 0xB004. In addition, the register should be set to the correct value before enabling the link training sequence.

Usage Note 4 *DDR3 Turn-Around Time For DQ & DQS Usage Note*

Revision(s) Affected: 1.0

Details: There could be a potential issue with turn-around time for the DQ and DQS lines when performing a read followed by a write to DDR3.

The JEDEC specification allows for a minimum of one clock cycle for the turn-around time when performing a read command followed by a write command. The DDR3 controller is implemented with exactly one clock cycle of turn-around time when CAS latency minus Write Latency is an even value. It has two clock cycles of turn-around time when CAS latency minus Write Latency is an odd value. The tDQSCK value specified in the JEDEC specification must also be subtracted from this cycle time to determine the final routing limit.

The data signals from the DSP take time to reach the memory chips. In addition, data signals from the memory chips take time to reach the DSP. This round trip delay must complete in the turn-around time mentioned above. If the round trip delay is too long, output contention will occur, which will corrupt data. It may also impact device reliability.

The following equation provides an approximation of the maximum data route length:

data_route_delay < (TA – tDQSCK) / 2 - margin

Where,

- tDQSCK = DQS to CK skew limit from DRAM data sheet – stated for each standard speed grade in the JEDEC standard (tDQSCK = 255 ps for DDR3-1333)
- TA = 1 clock cycle when CAS latency minus write latency is an even value
- TA = 2 clock cycle when CAS latency minus write latency is an odd value

Workaround 1: Limit board delays to be smaller than the maximum length per the equation above.

Table 15 shows the data routing limitations for both the Even and Odd latency deltas. The first column for each lists the maximum delay in picoseconds. The second column for each lists the maximum routing length in inches assuming a signal propagation rate of 180 ps/in.

Table 15 DDR3 Data Routing Limitations

	CL-WL Even		CL-WL Odd	
DDR3-1333	623 ps	3.46 in	1373 ps	7.63 in

Because this is preliminary guidance and some small margin should be subtracted from these delays to account for additional terms such as multi-rank delay skew and write leveling error, it is recommended that the maximum routing lengths be reduced by 10%.

Workaround 2: Run at a lower frequency.

Usage Note 5 *Packet DMA Does Not Update RX PS Region Location Bit Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The Packet DMA inside each of the Navigator-compliant modules fails to update the Protocol-Specific Region Location bit (bit 22 of Packet Descriptor Word 0) to a 1 when it is writing an RX host-mode packet to memory with protocol-specific (PS) words located in the start of the data buffer instead of the descriptor. This means that the software cannot use this bit to determine if any PS information present is located in the RX packet descriptor or at the beginning of the data buffer. The same problem will occur if the packet is sent directly to another Navigator-compliant module, as that module will not be able to determine the PS info location. This issue affects only host-type packets.

Workaround 1: Use monolithic-type packets only, thus eliminating the issue.

Workaround 2: Always place PS info in the descriptor instead of in the data buffer so that the PS location bit is always 0 and the issue does not apply.

Workaround 3: The software is responsible for configuring the Packet DMA RX flow tables, which include the PS location each flow will use. Thus, for packets sent to the DSP (not to another module directly), the software can be designed to keep track of the PS info location so it does not have to rely on the bit in the RX packet descriptor. This can be accomplished in many different ways. The following are a couple of examples:

- The software can always use the same PS location setting. This eliminates the need to find out the location from the RX descriptor.
- The software can place some form of identifier in one of the user-defined tag fields in the TX descriptor and configure the Packet DMA to pass that information through to the RX descriptor. The software can then use the identifier along with previously stored information to determine the PS info location.

Usage Note 6 *Packet DMA Clock-Gating Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: Clock-gating a module with a Navigator interface (Packet DMA) while it is writing RX packets to memory can cause undefined behavior.

Workaround: Disable/teardown all of the Packet DMA's channels before clock-gating the module in the PSC.

Usage Note 7 *VCP2 Back-to-Back Debug Read Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: In a debug scenario, back-to-back config bus reads from VCP2 where the first read is to an invalid VCP2 address and the second read is to a valid register are not supported and the second read will not return valid data. Emulation (CCS memory window) accesses do not use back-to-back reads and are not affected.

Workaround: Either guarantee that invalid VCP2 config bus accesses do not occur or do not perform back-to-back VCP2 debug (config bus) reads from DSP software. A single cycle between reads is sufficient for proper operation.

Usage Note 8 *Disable KICK Registers Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The Bootcfg module contains a kicker mechanism to prevent any spurious writes from changing any of the Bootcfg register values. When the kicker mechanism is locked (which it is initially after power on reset), none of the Bootcfg MMRs are writable (they are only readable). This mechanism requires two register writes (one to KICK0 and one to KICK1) with exact data values before the kicker lock mechanism is un-locked. Once released, then all the Bootcfg MMRs having *write* permissions are writable (the read-only MMRs are still read only). The device has only one set of kicker registers to lock/unlock the Bootcfg registers. This creates potential race conditions in the multi-core environment when the different DSP cores try to access Bootcfg registers at the same time.

Workaround: Unlock the kicker mechanism and do not re-lock it. To unlock it, write 0x83e70b13 to KICK0 and then write 0x95a4f1e0 to KICK1. Do not write any other different values afterward to these registers because that will lock the kicker mechanism and block any writes to Bootcfg registers.

Usage Note 9 *DDR3 ZQ Calibration Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: Incorrect impedance calibration will occur if DDR3 devices on the board share ZQ resistors. This is the resistor connected to the ZQ pin on a DDR3 device.

Workaround: Use independent ZQ resistors for all DDR3 devices on the board. The reg_zq_dualcalen field in the EMIF's SDRAM Output Impedance Calibration Config register must also be set to 1.

Usage Note 10 *I²C Bus Hang After Master Reset Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: It is generally known that the I²C bus can hang if an I²C master is removed from the bus in the middle of a data read. This can occur because the I²C protocol does not mandate a minimum clock rate. Therefore, if a master is reset in the middle of a read while a slave is driving the data line low, the slave will continue driving the data line low while it waits for the next clock edge. This prevents bus masters from initiating transfers. If this condition is detected, the following three steps will clear the bus hang condition:

1. An I²C master must generate up to 9 clock cycles.
2. After each clock cycle, the data pin must be observed to determine whether it has gone high while the clock is high.
3. As soon as the data pin is observed high, the master can initiate a start condition.

Usage Note 11 *MPU Read Permissions for Queue Manager Sub-system Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The Memory Protection Unit (MPU) has the ability to restrict the write and read permissions for bus masters like the DSP cores and System EDMAs when they attempt to access various portions of the address space on the device. One of the peripherals that may be access-controlled is the Queue Manager Sub-system (QMSS). For proper device operation, all of the read permissions for the VBUSM slave port of the QMSS must be enabled. If any of the read permissions for the VBUSM slave port of the QMSS are disabled, invalid read data and EDMA malfunction may occur. This usage note does not impact the VBUSP slave port on the QMSS or the QMSS write permissions, which may be enabled or disabled.

Usage Note 12 *POR and RESETFULL Sequence Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: For boot configuration pins to be latched correctly, during the power sequencing and reset control for chip initialization, RESETFULL must be held low for a period after the rising edge of POR, but may be held low for longer periods if necessary. The configuration bits shared with the GPIO pins will be latched on the rising edge of RESETFULL and must meet the setup and hold times. Timing requirements are specified in the device-specific data manual, *TMS320TCI6616 Communications Infrastructure Keystone SoC* data manual (literature number SPRS624).

Usage Note 13 *PLL ENSAT Bit Usage Note*

Revision(s) Affected: 1.0.

Details: For optimal PLL operation, the *ENSAT* bit in the PLL control registers for the Main PLL, DDR3 PLL, and PA PLL should be set to 1. The PLL initialization sequence in the boot ROM sets this bit to 0 and could lead to non-optimal PLL operation. Software can set the bit to the optimal value of 1 after boot. This requires setting the ENSAT bit (bit 6) to a value of 1 in the MAINPLLCTL1 register (address 0x0262032C), DDR3PLLCTL1 register (address 0x02620334), and PAPLLCTL1 register (address 0x0262033C). Read-modify-writes can be used to make sure other bits in the registers are not affected.

Usage Note 14 *AIF2 GSM Compressed Mode and Multimode Usage Note*

Revision(s) Affected: 1.0, 1.0A

Details: The AIF2 was intended to handle both compressed and non-compressed GSM formats. The concept of compression is simply that some portion of the end of an AIF2 Time Slot is not populated with antenna data. The implications on transmit are that the AIF2 Protocol Encoder (PE) should not fail the link if the DMA data does not fully fill the GSM Time Slot. The PE should transition to empty messages (OBSAI) or zero antenna data (CPRI) in this event. PE should commence with the next packet at the beginning of the next GSM Time Slot.

The implications on reception are nearly identical. The AIF2 Protocol Decoder should be tolerant of streaming data which does not quite fill the GSM Time Slot and will expect Time Stamp to re-start at the beginning of the next GSM Time Slot.

The AIF2 does support this functionality for reception side, but transmission side (PE) cannot support the functionality which was originally planned, so GSM users can not use the various sizes of GSM compressed data for downlink with the current hardware.

Workaround: The workaround for this issue involves dynamically modifying a series of AIF2 memory mapped registers (MMRs). The application software maintains two tables for the values written to these MMRs. One table contains the active configuration to enable the transmission of the antenna data and the other table contains all zeros to disable the transmission. The AIF2_AT system events can be used to trigger the EDMA to transfer the values from the tables to the AIF2 MMRs at the proper times. The active configuration table should be transferred just before the start of the GSM compressed mode time slot and the table with all zeros should be transferred just before the end of the compressed mode time slot. The details of this workaround will be presented in a separate application report.

Usage Note 15 *Queue Proxy Access Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: When there are multiple DSP cores potentially accessing the Queue Manager, the Queue N register A, B,C, and D should be accessed in the same burst. However, the C66x CorePac cannot generate bursts larger than 8 bytes. The Queue Proxy is designed to allow C66x CorePac to push/pop descriptors using multiple transactions. However, when the C66x CorePac uses the Queue Proxy region for push and pop, the Queue Proxy may mix the transactions from non-CorePac system masters. This may lead to an error transaction, which causes a system deadlock.

Workaround: The C66x CorePac should not use the Queue Proxy region to push/pop descriptors. The C66x CorePac should use VBUSM region (base address starts from 0x34000000) to push descriptors. When Queue N register C is needed for a push, the C66x CorePac should issue a DoubleWord write to generate an 8-byte burst write to Queue N register C and Queue N register D. When device is little endian mode, the Queue N register C and Queue N register D value need to be swapped. The C66x CorePac should use the VBUSP region (base address starts from 0x02A00000) to pop Queue N register D only. When packet size, byte count, and queue size information are needed for a certain queue, C66x CorePac should use the queue peek region to get the information.

Usage Note 16 *TAC E-AGCH Diversity Mode Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: The TAC (Transmit Accelerator) E-AGCH diversity mode cannot be changed from TTI to TTI for different UEs. The E-AGCH channel is transmitted for different users from TTI to TTI (2 ms or 10 ms). Because the diversity mode of this channel should match the diversity mode of the UE's associated downlink dedicated channel (DPCH or F-DPCH), the E-AGCH channel diversity mode should be changeable each TTI. With TAC, the diversity mode is statically configured for the life of the channel. It is specified in the spreader parameter memory (SPM) for the associated spreader.

Workaround: To change the diversity mode on a TTI boundary would require that the current spreader be stopped, and a new spreader be started with the changed diversity mode. The TAC FL needs to be more closely involved during the life of the channel to carry out these changes, and it requires the diversity mode to be known to the TAC FL at least 1.5 slots before the TTI boundary for the change to be made.

An alternative workaround is to configure two E-AGCH channels, one in diversity mode and one in non-diversity mode and the application will decide which channel to use each TTI.

Both methods consume extra TAC resources, which may affect deployments because 1 extra spreader and 1 extra cycle (and 1 extra DL TPC message handler if separate channels are used) will be required.

Usage Note 17 *Minimizing Main PLL Jitter Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0.

Details: Once the boot is complete, it is highly recommended that software reconfigure the Main PLL to the desired frequency, even if it is already achieved by the initial settings. To minimize the overall output jitter, the PLLs should be operated as close as possible to the maximum operating frequency. To maximize the VCO frequency within the PLL, the PLL should be clocked to 2× the intended frequency and the PLL Output Divider should be set to /2. The main PLL Output Divider should be set to divide-by-2 by the software by writing 0b0001 to bits [22:19] of the SECCTL register (address 0x02310108) in the PLL controller. A read-modify-write can be used to make sure other bits in the register are not affected. This register is documented in the *TMS320TCI6616 Communications Infrastructure Keystone SoC* data manual (literature number SPRS624).

Note—It is only after programming the SECCTL register to enable the divide-by-2 that the following equation can used to program the PLL as specified in the data manual.

CLK = CLKIN × (PLLM+1) ÷ (2 × (PLLD+1))

Usage Note 18 *SRIO and PA_SS PKTDMA RX Descriptor Buffer Size Usage Note*

Revision(s) Affected: 1.0.

Details: In the silicon revision(s) listed, the PKTDMA in the SRIO and PA_SS blocks requires that the RX descriptor buffer size be at least one byte more than the output payload data sent in the buffer. If the RX buffer size is equal to the output payload data size, the PKTDMA will use up an extra RX descriptor which it will link to the RX descriptor, even though there will be no output data placed in the extra descriptor's buffer.

Workaround: One workaround is to size the RX descriptor buffers so they are at least one byte larger than the maximum expected output payload data size. An alternative workaround is to allocate enough extra RX descriptors with associated buffers to make up for the extra ones that will be required when the RX descriptor buffer size is equal to the output payload data size.

Usage Note 19 *AIF2 LTE 3 MHz and 1.4 MHz Support Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: AIF2 requires the packet length to be divisible by 4 due to the OBSAI message restriction but LTE 3 MHz and 1.4 MHz symbol length do not satisfy the requirement due to the cyclic prefix (CP) length in normal cyclic prefix case as shown in Table 16.

Table 16 Cyclic Prefix Length for Each LTE Bandwidth

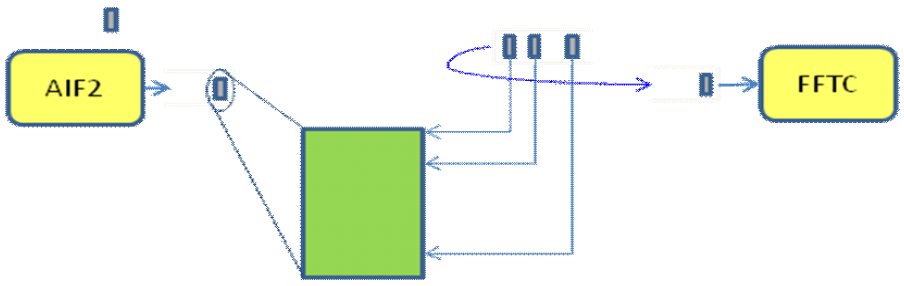
Channel Bandwidth	FFT Size	Sampling Rate	Long CP Length	Short CP Length	Extended CP Length	Total Number of Samples per Subframe
20	2048	30.72	160	144	512	30720
15	1536	23.04	120	108	384	23040
10	1024	15.36	80	72	256	15360
5	512	7.68	40	36	128	7680
3	256	3.84	20	**18**	64	3840
1.4	128	1.92	**10**	**9**	32	1920

Workaround AIF2 is only a data streaming engine and doesn't care the content nor the real length of a LTE symbol. AIF2 can be set to have only one jumbo packet per slot for both ingress and egress. As AIF2 is a real-time data streaming engine, it writes the data out as it receives and only reads data when it needs to transmit. Therefore, to minimize impact on system latency:

- On ingress, the symbol processing is not triggered upon receiving the jumbo packet, instead it is triggered based on AT event. As AT event doesn't have association with the availability in memory, application needs to program AT event with enough offset to the actual symbol timing to guarantee the data availability in memory.
- On egress, no need to produce the entire jumbo packet before pushing the packet to AIF2, symbols can be produced on symbol basis with enough margin before AIF2 needs it. The margin is related to the pre-fetch buffer size set in AIF2.

Figure 19 shows detailed description on Ingress:

Figure 19 Symbol Processing on Ingress

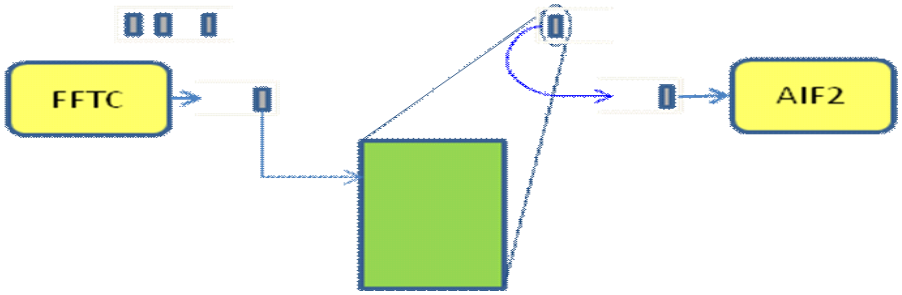


Preconfigure a set of descriptors that point to the correct offset in jumbo packet for each symbol, store the descriptors in the FFTC Tx FDQ or an array. AT event trigger the EDMA to pop and push the descriptors from FFTC Tx FDQ to FFTC Tx Q. EDMA needs to pop the descriptor from the Tx FDQ and push to the FFTC Tx Q. If the AIF2

output packet PS words are needed, then it will need to be copied by EDMA before pushing the descriptor to the FFTC Tx Q. If using AT event to trigger EDMA, after pushing the descriptor, EDMA needs to clear the CP_INTC to prepare the subsequent AT event trigger. As there is no real need for the AIF2 output queue itself, AIF2 Rx Q can be set to be the same queue as the Rx FD, Therefore no need to recycle the AIF2 output descriptor.

Detailed description on Egress is shown in Figure 20.

Figure 20 Symbol Processing on Egress



Assuming FFTC generates the packet, FFTC Rx FDQ needs to be prepared such that the descriptors point to the right position of the AIF2 jumbo packet for each symbol. Once every jumbo packet period, the jumbo packet descriptor can be pushed to AIF2 Tx queue. This can potentially be done by CPU as it is likely that it could be aligned with LTE DL symbol processing. As there is no real need for the FFT output descriptor itself, FFTC Rx Q can be set to be the same queue as the Rx FDQ, Therefore no need to recycle the FFTC output descriptor.

Usage Note 20 *PCIe BAR5 Window Size Configuration in Boot ROM Usage Note*

Revision(s) Affected: 1.0

Details: In silicon revision 1.0, the boot ROM code does not configure PCIe BAR5 Mask Register. The BAR5 address space (window size) is disabled after boot when operating PCIe as an end point (EP).

Workaround: If BAR5 is required when operating PCIe as EP, the user application can configure the BAR5 Mask Register on the EP side prior to the PCIe enumeration process. The user application should set DBI_CS2 bit (bit 5 in Command Status Register) first, then configure the BAR5 Mask Register and clear the DBI_CS5 bit after configuration. Please note that the user should not attempt modification of the BAR Mask Registers from serial link side (from RC or host). Please refer to the *PCIe for KeyStone Devices User's Guide* (literature number SPRUGS6) for details.

Usage Note 21 *Sticky Bits in PCIe MMRs Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The sticky bits in PCIe memory-mapped registers (MMRs) are those bits that are neither initialized nor modified by a hot reset, as defined in the PCI Express base specification. The values of the sticky bits are unchanged after a hot reset, which is important in error handling to ensure that error-related control and status information is not lost due to a hot reset.

In the TCI6616 device, the sticky bits in PCIe MMRs are not changed after a soft reset; only the non-sticky bits are initialized. During a hard reset, both the sticky and non-sticky bits get reset. For more information, see the PCI Express base specification.

Usage Note 22 *The Clock Input To NETCP Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: The clock input to PASS is programmable. There is a multiplexer which selects between SYSCLK1 or output of PASS PLL as the input to Network Coprocessor (NETCP). This multiplexer is controlled by bit 13 (zero indexed) in PASSPLLCTL1 register. The default value of this bit is 0 which selects SYSCLK1 as the input to PASS, however this is not the recommended mode of operation.

Workaround 1: In order to set PASS PLL output as the input to NETCP, bit 13 should be set to 1. This can be done as part of the PASS PLL initialization sequence. Read-modify-write can be used to make sure other bits in the register are not affected. Note that in case of Ethernet boot, the Boot ROM code is correctly setting this bit to 1. But in all other boot modes, this bit will have a default value of 0 and software must change this to 1.

Usage Note 23 *Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: During power-up and power-down cycles of the DSP, it is possible that some current may flow between the VDDR and VDDT rails. The VDDR rail tracks the VDDT rail as VDDT is ramping up to 1.0V. When VDDR gets approximately to the 400mV mark, the VDDR rail stops tracking the VDDT rail. Leakage observed here comes from the SerDes module that contains 1V transistors. It has been verified that this leakage is expected and has no impact on the reliability of the device.

Usage Note 24 *PLL Boot Configuration Settings and DEVSPEED Register Usage Note*

Revision(s) Affected: 1.0, 1.0A

Details: In silicon revision 1.0 and 1.0A, the CorePac main PLL is configured as 800MHz, 1000MHz, or 1200MHz in the BootROM code based on the DEVSPEED register reading. For the device to support a higher frequency, such as 1250MHz, the main PLL will not be configured correctly by the boot mode configurations because the DEVSPEED register reading is limited to 1200MHz.

Workaround: Reprogram the main PLL configuration settings appropriately to achieve the target device frequency at boot time of the user application. The application should follow the PLL programming procedure that is documented in the device-specific data manual.

Usage Note 25 *CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: A multi-rank configuration presents certain limitations when using lower speed bins like DDR3-800 that require CWL = 5 to be programmed.

It is recommended that below guidelines be followed for multi-rank configuration when using CWL=5:

- Set bit 24 (indexed to 0) in DDR3_CONFIG_REG_12 at 0x02620434 to 1. This will constrain the DDR PHY to use only rank0 delays for reads/writes to both ranks. The reset value is 0 which allows each rank to use its own delay.
- Using only rank0 delays means that the fly-by/round-trip delay across ranks must be balanced for a given byte lane.

Since rank0 delays are used by the PHY for both ranks, only the single rank equations for fly-by and round-trip delays from *DDR3 Design Requirements for KeyStone Devices* (literature number SPRABI1) need to be satisfied. The multi-rank equations will not be valid.

Usage Note 26 *IDMA1 Performance Limitation Usage Note*

Revision(s) Affected: 1.0, 1.0A

Details: In silicon revision 1.0, 1.0A, the IDMA1 transaction performance is lower than theoretical number due to the insufficient buffering in EMC. This issue has been fixed in silicon revision 2.0 and the performance meets the theoretical number. The following table shows the realistic IDMA1 transaction performance.

Table 17 IDMA1 Transaction Performance

IDMA1 Transaction	Silicon Revision 1.0, 1.0A (Bytes/Cycle)	Silicon Revision 2.0 (Bytes/Cycle)	Theoretical Performance (Bytes/Cycle)
L1D to L2	4.54	7.69	8
L2 to L1D	3.41	7.67	8
L2 to L2	3.75	7.76	8
L1D Fill	2.66	7.84	8
L2 Fill	7.49	15.44	16

Usage Note 27 *Revised PLL Programming Sequence Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: It has been observed that on a few devices, the CorePacs lock up after reprogramming of the Core PLL. It has been identified that the incorrect PLL programming sequence was causing the CorePacs to lock up.

Workaround: TI has revised the PLL programming sequence for Main PLL, DDR PLL and PASS PLL to eliminate the possibility of this lock-up issue. The revised sequence enables the by-pass mode in the PLL Controller via a MUX (by clearing PLLEN and PLENSRC bits) when the Main PLL is being re-programmed. Also, the PLLM, PLLD and BWADJ fields are programmed prior to assertion of PLLRST signal for all three PLLs in the revised sequence.

Please use the revised Main PLL, DDR PLL and PASS PLL programming sequence as described in *KeyStone Architecture Phase-Locked Loop (PLL) User Guide* (literature number SPRUGV2).

Usage Note 28 *Core Wake Up on RESET Usage Note*

Revision(s) Affected: 2.0

Details: Execution may start only on some C66x CorePacs if CCS is connected to the device and reset is applied via the RESET pin on the device. In order to make sure that all the CorePacs wake up after reset via RESET pin, device needs to be completely disconnected from the CCS before applying reset via RESET pin.

Some of the C66x CorePacs do not wake up on RESET reset when the device is connected via CCS. When the device is connected via CCS the device stays in the emulation debug state. If the RESET reset is applied while the device is in the emulation debug state it causes some of the C66x CorePacs to go into an unknown state and they don't start execution.

Resets using POR and RESETFULL does not exhibit this behavior.

This does not affect the normal usage of the device when CCS/emulator is not connected to the device since the device is not in emulation debug state when reset is applied using RESET pin. This behavior can only happen in the lab environment where CCS/emulator is connected to the device.

Workaround: Below is the sequence which must be followed to completely disconnect the device from CCS before applying RESET.

1. "Free Run" all the C66x CorePacs
2. Disconnect all the C66x CorePacs from CCS
3. Apply RESET

Steps 1 and 2 insure that all the debug states are cleared in the device. This will allow the C66x CorePacs to wake up correctly on reset via RESET. Bypassing either step 1 or 2 will result in C66x CorePacs that do not begin execution after reset.

Usage Note 29 *BSDL Testing Support Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: It has been observed that IEEE1149.6 testing on the device may not function properly. During AC boundary scan testing, errors (either stuck at or shorts) may be reported by the BSDL test software and BSDL controller.

Proper use of the SerDes AC boundary scan cells requires that the SoC (processor) be powered with clocks supplied in the recommended sequencing as outlined in the data manual for the device in use, and the device is out of reset prior to running the boundary scan tests.

Additionally, it has been identified that the default SerDes RX termination value, "RXTERM", is set to either 000 or 111 and must be re-programmed to 001 (0.7VDDT (or 0.8VDDT) common point in the memory mapped registers (MMR)) for each SerDes prior to use.

This correction has been found to be valid for all SerDes except PCIe where the register termination control had been hard coded and is not configurable except through bit 5 (pcs_fix_term) of the PCS Configuration 0 Register [PCS_CFG0]. Configuring bit 5 high forces the termination value to be set to common point to vsst; setting this MMR bit to a "0" sets the termination value to common point floating. In either case the boundary scan cell may not function properly.

Workaround: The only known workaround involves correctly powering and providing clocking for the SoC/DSP (processor) first. Each SerDes RX termination (RXTERM) register will then be required to be re-programmed from its default value of "000" with a value of "001" prior to BSDL testing. Depending on the SerDes peripheral, there may be more than one RXTERM register requiring programming.

There are two types of BSDL tools currently available:

1. Those capable of running a BSDL controller and TI emulation/debugger software over the same hardware platform
2. Those types of tools that are designed to run emulation/debugger software independently from boundary scan software

The following are the known workaround steps for both scenarios described above:

Single hardware/software tool

1. The SoC/DSP must be correctly powered with clocks applied
2. Using the emulation/debugger, enable all the respective SerDes peripherals
3. Using the emulation/debugger, change the RXTERM value from "000" to "001"
4. Disable the emulation\debugger software and connect using your BSDL software
5. Run your respective 1149.6 boundary scan tests

Separate hardware/software tool

1. The SoC/DSP must be correctly powered with clocks applied
2. Using the emulation/debugger, enable all the respective SerDes peripherals
3. Using the emulation/debugger, change the RXTERM value from "000" to "001"
4. Disconnect the emulation/debugger hardware and software

5. Connect the boundary scan hardware and software
6. Run your respective 1149.6 boundary scan tests

Please see the indicated peripheral user's guide for details on enabling a given peripheral.

Table 18 Register Information for the Peripheral

Peripheral	Register	Bit	Address
AIF (SPRUGV7)	SD_RX_R2_CFG[0]	4:2	0x01F08008
	SD_RX_R2_CFG[1]	4:2	0x01F08808
	SD_RX_R2_CFG[2]	4:2	0x01F09008
	SD_RX_R2_CFG[3]	4:2	0x01F09808
	SD_RX_R2_CFG[4]	4:2	0x01F0A008
	SD_RX_R2_CFG[5]	4:2	0x01F0A808
HyperLink (SPRUGW8)	HYPERLINK_SERDES_CFGRX0	9:7	0x026203B8
	HYPERLINK_SERDES_CFGRX1	9:7	0x026203C0
	HYPERLINK_SERDES_CFGRX2	9:7	0x026203C8
	HYPERLINK_SERDES_CFGRX3	9:7	0x026203D0
PCIe (SPRUGS6)	PCS_CFG0 (RXTERM value not configurable to 001)	5	0x21800380
SGMII (SPRUGV9)	SGMII_SERDES_CFGRX0	9:7	0x02620344
	SGMII_SERDES_CFGRX1	9:7	0x0262034C
SRIO (SPRUGW1)	SRIO_SERDES_CFGRX0	9:7	0x02620364
	SRIO_SERDES_CFGRX1	9:7	0x0262036C
	SRIO_SERDES_CFGRX2	9:7	0x02620374
	SRIO_SERDES_CFGRX3	9:7	0x0262037C

Usage Note 30 *AIF2 CPRI FastC&M Restrictions and Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: This usage note describes some hardware (HW) limitation and restriction of the AIF2 Fast Ethernet CPRI control word channel operation. This includes one HW limitation about Fast Ethernet SSD (Start of Stream Delimiter) handling. This usage note also explains the HW restrictions of 4B5B encoded data nibble level swapping, Hyperframe boundary delimitation and Ethernet packet CRC32 usage.

1.CPRI Fast Ethernet Preamble Issue

Details: IEEE 802.3 Fast Ethernet spec Start-of-Stream Delimiter chapter states: A Start-of-Stream Delimiter (SSD) is used to delineate the boundary of a data transmission sequence and to authenticate carrier events. On transmission, the first 8 bits of the MAC preamble are replaced by the SSD, a replacement that is reversed on reception.

TI AIF2 Protocol encoder does not overwrite the first 8 bits of the MAC preamble. Instead, it appends 8bits SSD at the end of preamble and this is a violation of 802.3 spec.

Workaround: The best solution for this problem is to turn off the Ethernet header append/strip feature from the AIF2 PE/PD. Instead, use software (SW) to append/strip the 6 bytes pre-amble and one byte SOF. In this case, SSD will be attached by 4B5B encoder and 4B5B encoding option should be enabled regardless of Ethernet header append/strip feature status. With this approach, the CRC generation/checking feature in AIF2 must also be disabled since there is no way to prevent the CRC calculation over the Ethernet header. SW would need to implement this CRC in addition to the Ethernet header. This should be applied to both Egress and Ingress configuration.

2.CPRI Fast Ethernet 4B5B encoded data nibble level swap

Details: CPRI spec does not clearly describe how 4B5B encoded data is to be transferred with regards to the ordering of which bit or nibble data block from Ethernet MAC. AIF2 HW is natively supports big endian data order and it supports bit level swap within byte before 4B5B encoding or after 4B5B encoding (from Rev 2.0), but it doesn't support 5-bit nibble level swap for 4B5B encoded data (AIF2 transfers MSB 5bit first and LSB 5bit last) which have been found to be required by some Remote Radio Head products.

Workaround 1: Use NULL delimiter instead of 4B5B encoding. This is only allowed when RRH can accept NULL delimiter as an alternative.

Workaround 2: Use an FPGA between AIF2 and RRH where FPGA performs nibble level swap for 4B5B encoded data.

Workaround 3: Modify RRH HW to accept current 4B5B encoded data from AIF2 (MSB 5bits first and LSB 5bits last).

3.CPRI Fast Ethernet Hyper-frame boundary delimitation restriction

Details: AIF2 has a special CPRI control word and packet encoding option "Hyper-frame boundary delimitation" and packet boundaries are inferred to be on Hyper-frame boundaries. This feature does not work with DMA layer of AIF2; resulting operation can cause the Protocol Encoder to misalign the data into the CPRI hyper-frame. The user may see one or two quad word amount of unexpected data shift if there is any minor delay on DMA layer.

Workaround: Use “NULL” or “4B5B” for packet encoding option instead of Hyper-frame boundary option.

4. CPRI Fast Ethernet CRC generation feature restriction

Details: AIF2 supports CRC generation by HW for CPRI Fast Ethernet and all CRC8, CRC16 and CRC32 is supported.

For CRC16 and CRC32 the CPRI Fast Ethernet Generation requires the Ethernet packet size to be multiple of 4 bytes. If the packet size is not a multiple of 2 bytes for CRC16 or 4 bytes for CRC32, the HW will generate an invalid CRC value.

Workaround: Use CRC8 option, if the packet size is not a multiple of 2 bytes for CRC16 and 4 bytes for CRC32.

Usage Note 31 *Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Details: Users are required to program their board CVDD supply initial value to 1.1V on the device. The initial CVDD voltage at power-on will be 1.1V nominal and it must transition to VID set value, immediately after being presented on the VCNTL pins. This is required to maintain full power functionality and reliability targets guaranteed by TI.

SmartReflex voltage scheme as defined by the device specific data manual and *Hardware Design Guide for KeyStone I Devices* (literature number SPRABI2) is absolutely required.

Usage Note 32 *DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: For commands with a higher class of service, larger than expected latency may be observed due to the DDR3 memory controller failing to properly elevate the priority of execution of the command.

Details: The DDR3 memory controller's 'Class of service' (COS) feature allows the user to prioritize commands that are scheduled inside the controller's command FIFO based on bus priority of a master or its master ID. A latency counter is programmed for a specific class of service. The counter value decides how long the command may wait inside the FIFO before it is moved to head of the FIFO. The commands assigned to a class of service with a lower counter value are considered to have a higher class of service. Please see the *KeyStone Architecture DDR3 Memory Controller User Guide* (literature number SPRUGV8) for details.

The following are examples of how read and write transactions behave when the class of service feature is enabled. The class of service for reads and writes are handled independently i.e. if the counter for a read with a higher class of service expires, all writes irrespective of their class of service will not be blocked. All other reads with a lower class of service, however, will be blocked. Similarly, if the counter for a write with a higher class of service expires, all reads irrespective of their class of service will not be blocked. All other writes with a lower class of service, however, will be blocked. This can lead to a situation where the controller fails to elevate the priority of execution of a command that has been assigned to a higher class of service.

An example of such a situation is when the COS counter for a read command with a higher class of service expires and there is a continuous stream of write traffic to an open bank in the memory with lower class of service than the read command.

Note—The class of service counter or class of service latency tracks how long a particular command that is mapped to the specific COS (in this example, the read command) should wait in the FIFO before it is prioritized for execution.

Note—The PR_OLD_COUNT tracks how long the oldest command (regardless of the assigned COS) should stay in the command FIFO before it is prioritized for execution.

In the above example, the COS counter associated with that read command expires. However, since the controller will always prioritize writes to an open bank, the read command will become the oldest in the FIFO. From this point, the PR_OLD_COUNT starts tracking how long it has been in the FIFO. It is only when PR_OLD_COUNT expires that the read command will be moved to the head of the FIFO and executed. Thus, commands with a higher class of service may see a higher than expected latency of execution which can be a problem if a master is latency sensitive.

Note—The situation that exposes this issue (like the one mentioned above) is expected to be a corner case. In a real world system that has random traffic generated by multiple masters, the probability of this issue is very slim.

Note—This does not affect systems where the class of service feature is not used.

Workaround: The PR_OLD_COUNT field in the VBUSM configuration register decides the number of DDR3 clock cycles after which the controller raises the priority of the oldest command in the FIFO. By default, this is 0xFF which translates to 0xFF x 16 x 2 DDR3 clock cycles (refer to the *KeyStone Architecture DDR3 Memory Controller User Guide* (literature number SPRUGV8)). This field should be reduced until the higher than expected latency for the latency sensitive master is reduced to an acceptable level. The time spent by the oldest command in the FIFO depends on the traffic pattern and the aggregate traffic hitting the DDR interface. Thus, there is no optimal value that can be recommended for all systems. The user is expected to determine the optimal PR_OLD_COUNT.

Potential effect of reducing PR_OLD_COUNT:

The controller attempts to manage the traffic to/from DDR as efficiently as possible by rescheduling commands in the FIFO as per its arbitration logic. The user should note that while reducing PR_OLD_COUNT too low (0x0 to 0x20) will reduce the time spent by a command inside the FIFO, this may impact the scheduling and negatively affect throughput.

Usage Note 33 *Incorrect Output from TAC for Certain Channels When Configuring TAC_Gn_DATA_FETCH and TAC_Gn_HEAD_FETCH Registers with Values Having a Difference of 8 or 16 Usage Note*

Revision(s) Affected: 1.0, 1.0A, 2.0

Summary: When software configures TAC_Gn_DATA_FETCH and TAC_Gn_HEAD_FETCH registers with values having a difference of 8 or 16, TAC produces incorrect output for DPCH with spreading factor 4 and for S-CCPCH with slot formats 15, 16 and 17.

Details: Transmit Accelerator (TAC) on Keystone devices has several internal hardware components, the most important of which are the spreaders. The spreaders can be configured to carry out chip-rate processing for various channel types defined by WCDMA specifications. The spreaders inside TAC are grouped into internal blocks called Spreader Group Co-Processors (SGCPs) where each SGCP contains several spreaders.

While there are certain types of WCDMA channels for which the antenna output data is generated completely based on channel setup-time configuration, the antenna output data for most types of channels is dependent on channel setup-time configuration as well as input data that is periodically supplied to TAC. Channel types for which periodic input data is required include R99 channels PICH, PCCPCH, SCCPCH, AICH, E-AICH, MICH, F-DPCH and DPCH, and HSPA channels. The input data is supplied periodically on an access slot basis (AICH/E-AICH channels), frame basis (all R99 channels except AICH/E-AICH) or sub-frame basis (HSPA channels). The input data consists of a header and a payload.

A spreader in TAC that is configured for a channel type requiring periodic input data is called a fetching spreader. Before the start of a given access slot, frame or sub-frame (depending on the channel type), a fetching spreader needs to fetch input data header and input data payload. For data payload, TAC fetches the payload in chunks of 64 bits, so the first 64-bit chunk of payload is fetched before the access slot, frame or sub-frame boundary while the subsequent 64-bit chunks of payload are fetched periodically thereafter throughout the access slot, frame or sub-frame duration. TAC provides configurability for software to determine how many 4-chip iterations in advance of an access slot, frame or sub-frame boundary TAC issues a fetch request for the header and a fetch request for the first 64-bit payload chunk. This configurability is available on an SGCP basis through registers TAC_Gn_DATA_FETCH and TAC_Gn_HEAD_FETCH registers, where TAC_G0_DATA_FETCH and TAC_G0_HEAD_FETCH are on SGCP #0, TAC_G1_DATA_FETCH and TAC_G1_HEAD_FETCH are on SGCP #1 and so on. For example, if TAC_G0_HEAD_FETCH is 2 and TAC_G0_DATA_FETCH is 1, then all fetching spreaders on SGCP #0 issue a fetch request for header 2 iterations (8 chips) before access slot, frame or sub-frame boundary while they issue a fetch request for the first 64-bit payload chunk 1 iteration (4 chips) before the boundary. This configurability is available so that a system designer takes into account system data flow constraints and gives enough time for the TAC fetches to complete before the actual data processing starts for a spreader.

The current silicon issue results in data payload fetch problems for DPCH channels with spreading factor 4 and S-CCPCH channels with slot formats 15, 16 and 17 when TAC_Gn_HEAD_FETCH and TAC_Gn_DATA_FETCH are configured with values differing by 8 or 16. This will result in incorrect antenna output for these channels. For example, if a spreader on SGCP #0 is configured as DPCH with spreading factor 4 or as S-CCPCH with slot format 15, 16 or 17, then it will produce bad output under the following configurations (not exhaustive listing):

Table 19 TAC Registers Configuration

TAC_G0_DATA_FETCH	TAC_G0_HEAD_FETCH
1	9
1	17
11	19
11	27

Workaround: Software can work around this silicon issue by configuring values for TAC_Gn_HEAD_FETCH and TAC_Gn_DATA_FETCH that differ by values other than 8 or 16. For example, corresponding to the values of TAC_G0_DATA_FETCH given in the above table, the following table shows values of TAC_G0_HEAD_FETCH that work around the issue:

Table 20 TAC Registers Configuration

TAC_G0_DATA_FETCH	Bad TAC_G0_HEAD_FETCH	Good TAC_G0_HEAD_FETCH
1	9	8
1	9	10
1	9	16
1	17	18
11	17	18
11	19	20
11	27	26
11	27	28

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions of TI 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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