TMS320TCI6612
Communications Infrastructure KeyStone SOC
Silicon Revisions 1.0, 1.1, 1.3

Silicon Errata

Literature Number: SPRZ369D
December 2012–Revised June 2015

Contents

List of Figures

List of Tables

TMS320TCI6612 Communications Infrastructure KeyStone SOC Errata Silicon Revisions 1.0, 1.1, 1.3

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320TCI6614 fixed-/floating-point digital signal processor. See the device-specific data manual, *TMS320TCI6612 Communications Infrastructure Keystone SoC* data manual (SPRS784) for more information.

2 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., TMS320TCI6614CMS). 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.

3 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 CMS package is shown in Figure 1. Figure 1 also shows an example of TCI6614 package symbolization.

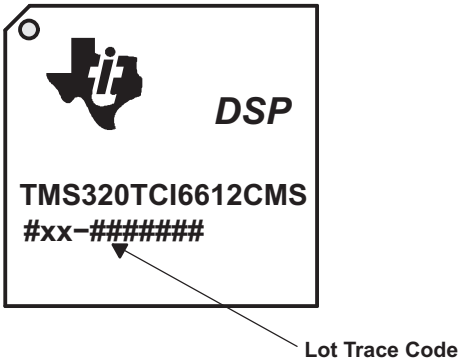


Figure 1. Lot Trace Code Example for TMS320TCI6612 (CMS Package)

Silicon revision correlates to the lot trace code marked on the package. This code is of the format #xx-#######. Note that there may be an additional leading character (not shown in this example) and xx may actually be two or three characters. 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 TCI6614 devices.

Table 1. Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision
11	1.1	ARM L2 cache issue fixed
13	1.3	Silicon revision 1.3

The TCI6614 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 TCI6614.

The JTAG ID register (JTAGID) is a read-only register that identifies to the customer the JTAG/Device ID. The value in the VARIANT field of the JTAG ID Register changes based on the revision of the silicon being used.

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 *TMS320TCI6612 Communications Infrastructure Keystone SoC* data manual (SPRS784) for more information.

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* (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 VARIANT field for each silicon revision of the TCI6614 device.

Table 2. Silicon Revision Variables

Silicon Revision	CPU CSR Register	C66x CorePac MM_REVID Register	TCI6614 JTAGID Register
1.0, 1.1	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 2.0 MM_REVID[REVISION] = 0001h	JTAGID = 0x0B96202F
1.3	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 2.0 MM_REVID[REVISION] = 0001h	JTAGID = 0x1B96202F

More details on the JTAG ID and CorePac Revision ID Registers can be found in the *TMS320TCI6612 Communications Infrastructure Keystone SoC* data manual (SPRS784) for more information.

4 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

Silicon Update Advisory	See	Applies To Silicon Revision		
		1.0	1.1	1.3
HyperLink Temporary Blocking Issue	Advisory 1	X	X	X
BCP DIO Reading From DDR Memory Issue	Advisory 3	X	X	X
DDR3 Excessive Refresh Issue	Advisory 4	X	X	X
TAC P-CCPCH QPSK Symbol Data Mode with STTD Issue	Advisory 5	X	X	X
SRIO Status Control Symbols are Sent More Often Than Required Issue	Advisory 6	X	X	X
Corruption of Control Characters in SRIO Line Loopback Mode Issue	Advisory 7	X	X	X
SerDes Transit Signals Pass ESD-CDM up to ±150V Issue	Advisory 8	X	X	X
AIF2 CPRI 8x UL Peak BW Issue	Advisory 9	X	X	X
AIF2 SERDES Lane Aggregation Issue	Advisory 10	X	X	X
ARM L2 Cache Content Corruption Issue	Advisory 11	X		
L2 Cache Corruption During Block and Global Coherence Operations Issue	Advisory 12	X	X	X
System Reset Operation Disconnects the SoC from CCS Issue	Advisory 13	X	X	X
Power Domains Hang when Powered Up Simultaneously with $\overline{\text{RESET}}$ (Hard Reset) Issue	Advisory 14	X	X	X
SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue	Advisory 16	X	X	X
DDR3 Automatic Leveling Issue	Advisory 17	X	X	X
DDR3 Incremental Write Leveling Issue	Advisory 18	X	X	X
ARM Memory Corruption Issue	Advisory 19	X	X	
NAND Boot Failure Issue	Advisory 20	X	X	
Tracer_DDR_2 Master ID Filtering Limitation Issue	Advisory 21	X	X	X
Single MFENCE Issue	Advisory 22	X	X	X
Read Exception and Data Corruption Issue	Advisory 23	X	X	X
EMIF16 (NOR) Boot Error From Chip Selects Region CE2 and CE3 Issue	Advisory 24	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 25	X	X	X
BCP Soft Slicer Lock-Up Issue	Advisory 26	X	X	X
False DDR3 Write ECC Error Reported Under Certain Conditions	Advisory 27	X	X	X
Descriptors Placed in PCIe Memory Space can Cause Problems	Advisory 28	X	X	X
TAC DL TPC Timing Usage Note	Usage Note 1	X	X	X
Packet DMA Clock-Gating for AIF2 and Packet Accelerator Subsystem Usage Note	Usage Note 2	X	X	X
VCP2 Back-to-Back Debug Read Usage Note	Usage Note 3	X	X	X
DDR3 ZQ Calibration Usage Note	Usage Note 4	X	X	X
I^2C Bus Hang After Master Reset Usage Note	Usage Note 5	X	X	X

Table 3. Silicon Revision Updates (continued)

Silicon Update Advisory	See	Applies To Silicon Revision		
		1.0	1.1	1.3
MPU Read Permissions for Queue Manager Subsystem Usage Note	Usage Note 6	X	X	X
Queue Proxy Access Usage Note	Usage Note 7	X	X	X
TAC E-AGCH Diversity Mode Usage Note	Usage Note 8	X	X	X
Minimizing Main PLL Jitter Usage Note	Usage Note 9	X	X	X
MSMC and Async EMIF Accesses from ARM Core	Usage Note 10	X	X	X
OTP Memory Controller Does Not Operate at Full Speed	Usage Note 11	X	X	X
$\overline{\text{POR}}$ and $\overline{\text{RESETFULL}}$ Sequence Usage Note	Usage Note 12	X	X	X
AIF2 LTE 3Mhz and 1.4Mhz Support Usage Note	Usage Note 13	X	X	X
Packet DMA Does Not Update RX PS Region Location Bit Usage Note	Usage Note 14	X	X	X
The Clock Input to NETCP Usage Note	Usage Note 15	X	X	X
Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note	Usage Note 16	X	X	X
CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note	Usage Note 17	X	X	X
USIMIO Pin IPU Usage Note	Usage Note 18	X	X	
Revised PLL Programming Sequence Usage Note	Usage Note 19	X	X	X
Core Wake Up on $\overline{\text{RESET}}$ Usage Note	Usage Note 20	X	X	X
BSDL Testing Support Usage Note	Usage Note 21	X	X	X
Ethernet Boot Size Limitation Usage Note	Usage Note 22	X	X	
Secure Boot Size Limitation Usage Note	Usage Note 23	X	X	
AIF2 CPRI FastC&M Restrictions and Usage Note	Usage Note 24	X	X	X
Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note	Usage Note 25	X	X	X
Performance Degradation for Asynchronous Accesses Caused by An Unused Feature Enabled in EMIF 16 Usage Note	Usage Note 26	X	X	X
DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note	Usage Note 27	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 28	X	X	X

Silicon Updates

Silicon Updates (continued)

Advisory 1 *HyperLink Temporary Blocking Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: A HyperLink temporary blocking condition can exist when a local device is reading from a remote device so heavily that the responses from the remote device keep the response path continuously busy. Because responses have a higher priority than outgoing commands, the remote device is temporarily unable to send commands to the local device.

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

Workaround 2: If a pull model is needed, schedule breaks so the return path is not continuously busy.

Advisory 3 *BCP DIO Reading From DDR Memory Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: The BCP Direct I/O (DIO) submodule and the DIO interface in the Rate De-matcher (RD) submodule only support start addresses that are multiples of 64 bytes when reading data from DDR memory. The addresses only need to be a multiple of 16 bytes when reading from L2RAM or MSMC SRAM.

Workaround: The DIO interface used in the RD submodule is used for uplink HARQ data for LTE, HSUPA, and WiMAX. If DDR memory is used, the application software should always allocate the HARQ buffers so they start on 64-byte boundaries.

The stand-alone DIO submodule is used only for WCDMA Rel-99 downlink processing, where it performs the transport channel multiplexing. If DDR memory is used, the normal flow of writing and reading the data via DIO cannot be used since the Interleaver (INT) submodule will place the radio frames on 16-byte boundaries instead of 64-byte boundaries. Instead, the DIO can still write the data, but the transport channel multiplexing can be performed by using the BCP PacketDMA to read the data for the physical channel processing instead of the DIO. A set of TX descriptors can be setup so each one points to the current radio frame of each of the transport channels. These descriptors can then be linked together using the “Next Descriptor Pointer” field in each descriptor. When the head descriptor is pushed onto one of the BCP TX queues, the BCP PacketDMA will read the data for each of the linked descriptors, thus performing the transport channel multiplexing just like the DIO would have done.

Advisory 4 *DDR3 Excessive Refresh Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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* (SPRUGV8).

Advisory 5 *TAC P-CCPCH QPSK Symbol Data Mode with STTD Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 of 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 6 *SRIO Control Symbols Are Sent More Often Than Required Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 4.

Table 4. Impact on Various Lane Speeds

Lane Speed (Gbaud)	Percentage of Bandwidth Reduction		
	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 7 *Corruption of Control Characters In SRIO Line Loopback Mode Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 2 shows the line loopback from SerDes RX to SerDes TX:

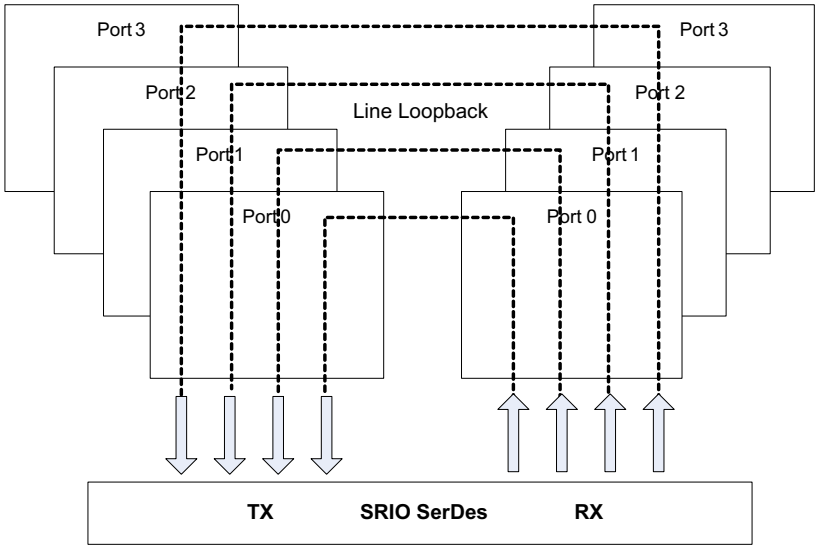


Figure 2. SRIO SerDes in Loopback Mode

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 8 *SerDes Transit Signals Pass ESD-CDM up to ±150 V Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 3 shows the basic flow for analyzing this risk:

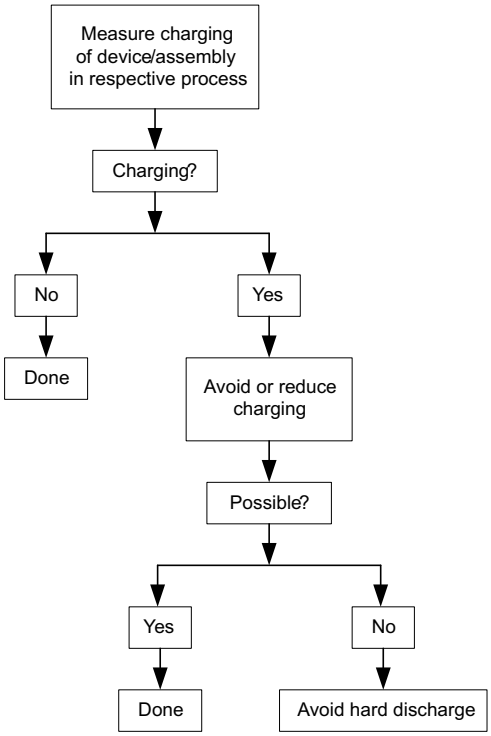


Figure 3. Basic Flow for Analyzing CDM Risk

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 9 *AIF2 CPRI 8x UL Peak BW Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: A single CPRI 8x link in RSA mode with all antennas aligned exposes a peak bandwidth limitation when more than 22 AxCs are enabled resulting in data corruption for channel numbers higher than 22. Similarly, multiple links supporting more than a total of aligned 44 AxCs result in the same peak bandwidth limitation and data corruption.

A single AIF2 CPRI 8x link can transfer a maximum of 32 WCDMA AxCs for 15 bit samples and 30 AxCs for 16 bit samples. When we use less than 22 AxCs, no problem is detected. However, the problem arises when more than 22 aligned AxC channels are enabled. We receive correct data for the first 22 AxCs (channels 0 ~ 21), but corrupted values are inserted for AxC channels 23 ~ 31. This problem only occurs on CPRI, WCDMA (DIO) UL RSA mode. OBSAI and CPRI DL modes do not exhibit this behavior, even with 64 AxCs (two 8x links).

The cause of this problem is in the PD (Protocol Decoder) module. Once the peak bandwidth exceeds the limitations of the AIF2 PD staging buffer, buffer corruption occurs. The peak bandwidth is reduced by either reducing channel count or by modifying the timing offset of some of the AxC channels such that no more than 22 AxC channels are time aligned.

Workaround 1: The AIF2 PD contains 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 the PD to align word data into Quad Words.

The normal use of this offset is to fine control the 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 zero.

In the work around 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 the cpri_8wd_offset to one for the higher number channels. The data is presented to the RAC with a shift of two samples. Both path search and finger de-spread operations work from the same shifted antenna data, and the AxC shift results in only one chip of 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 AxC for a given diversity sector be given the same AIF2 delay.

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

Advisory 10 *AIF2 SERDES Lane Aggregation Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: Line Code Violations (LCVs) or 8b10b decode errors can occur between different Nyquist 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 zero. This problem is also dependent on the power supply voltage variation and very low temperature behavior of the device.

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, the data is not physically synchronous but rather aligned using special 8b10b symbols in the data stream. AIF2 needs 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 and 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. As a result, the 30ps budget is not sufficient, which results in tbsync not having enough hold margin. In a situation with extreme low temperatures with supply voltage variances, this may cause serious issues such as losing symbol alignment which will also result in LCV errors.

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

Workaround 2: Select the B8 macro as a source of the AIF2 Tx byte clock. This can be done by setting the SD_CLK_SEL_CFG register to 0. The result is a reduced setup margin on the SerDes internal bus.

Advisory 11 *ARM L2 Cache Content Corruption Issue*

Revision(s) Affected: 1.0

Details: The ARM L2 cache does not show the correct content from cacheable DDR.

The ARM L2 cache line size is 64B. A 64B cache line fetch from DDR does not show the correct values from DDR memory. Some part of the 64B cache line is repeated with the same values from another part of the cache line.

Code executed from cacheable DDR will show abnormal behavior and will result in exceptions. Data accessed from cacheable DDR will be corrupted.

Workaround 1: Disable L2 cache. SC-MCDSK version 2.00.00.01 disables L2 cache (note that this can also result in performance degradation). All subsequent SC-MCDSK releases will be cache disabled until PG1.1 is available.

Advisory 12 *L2 Cache Corruption During Block and Global Coherence Operations Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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

Generic Workarounds: The workarounds below are very 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. Hence 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* (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 13 *System Reset Operation Disconnects the SoC from CCS Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 resetting the device using the $\overline{\text{RESET}}$ pin.

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** on the right side of the window and select **Auto**.

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 any DSP targets, and then re-connect CCS to the targets for debug purposes after the system reset.

Advisory 14 *Power Domains Hang When Powered Up Simultaneously with RESET (Hard Reset) Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: Power domains affected:

- Power Domain 5 (BCP)
- Power Domain 7 (MSMC RAM)
- Power Domain 8 (RAC_A, RAC_B, and TAC)
- Power Domains 13 and 14 (C66x Core 0 and 1, L1/L2 RAMs)

Certain power domains, such as the ones mentioned above, have multiple RAMs. The controllers associated with these RAMs are daisy chained. This issue occurs whenever the software is powering up one of these power domains and at the same time a RESET (hard reset) is received.

The Global PSC state machine associated with the power domain in transition hangs and as a result the chip 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 transition indicates that the device is out of reset). If the RESETSTAT pin doesn't toggle and is stuck at low, then 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 16 *SerDes AC-JTAG (1149.6) Receiver Sensitivity Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 TCI6614 SoC to other board components (other TCI6614, 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 TCI6614 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 4 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 Rincewind and Draco 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.

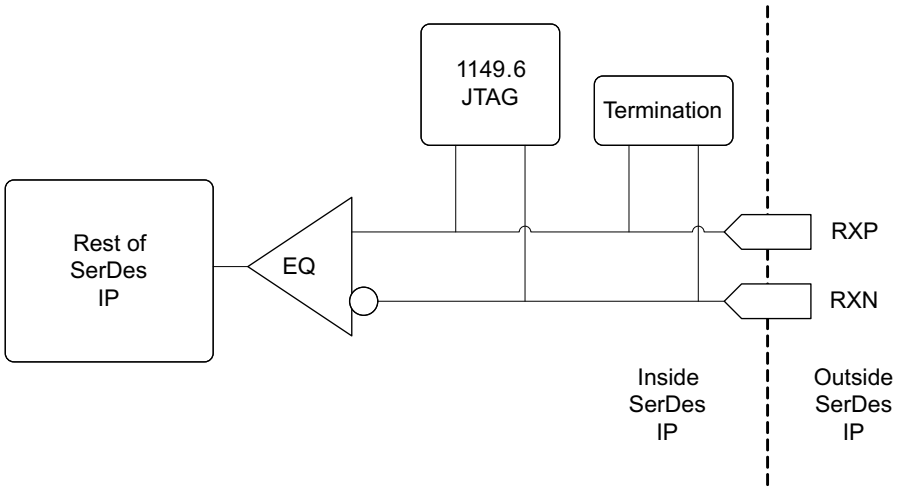


Figure 4. Simplified SerDes Receiver Block Diagram Depicting Relative Placement of Major Receiver Components

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 5. 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, 125C junction temperature, minimum VDDT), the comparator can fail to switch to the proper state, or not respond at all.

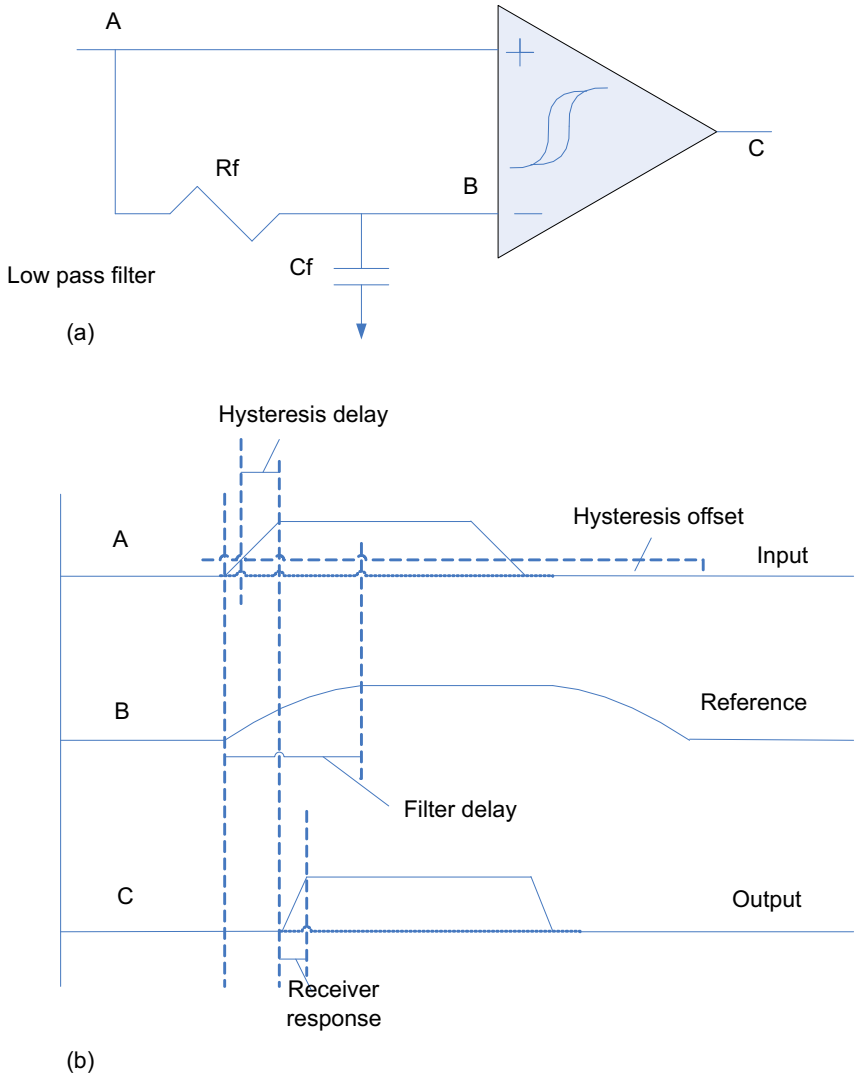


Figure 5. Self-referenced Comparator and Response (See Note Below)

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:

One potential workaround has been identified. 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 17 *DDR3 Automatic Leveling Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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.

The read-write leveling hardware features are described in Read-Write Leveling section of the *KeyStone Architecture DDR3 Memory Controller User Guide* (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 6, the DQS signal can be delayed relative to the DQ signal within the DDR3 PHY.

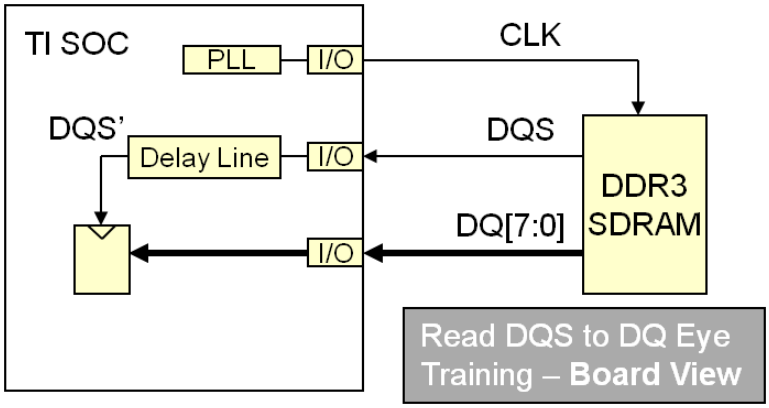


Figure 6. Read DQS to DQ Eye Training - Board View

As shown in Figure 7, the Read Data Eye Training algorithm should correctly detect the extents of the DQ data eye. The SDRAM are put into a DQS to DQ eye training mode and read back an alternating 0, 1 pattern.

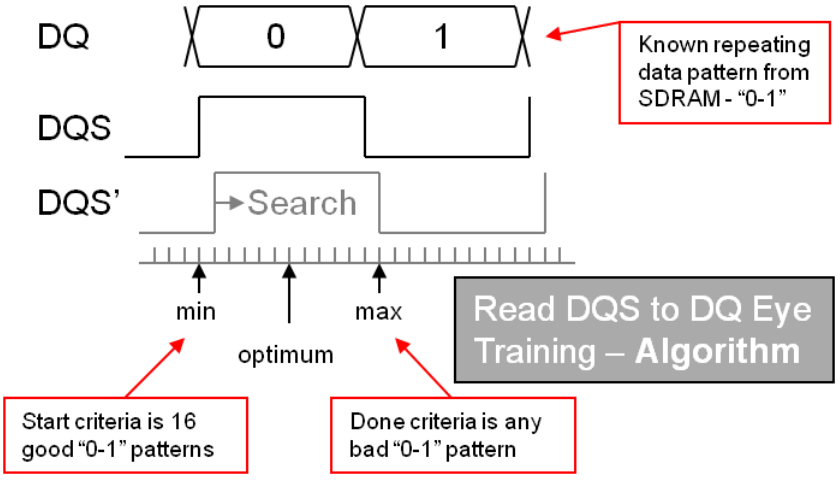


Figure 7. Read DQS to DQ Eye Training - Algorithm

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 8.

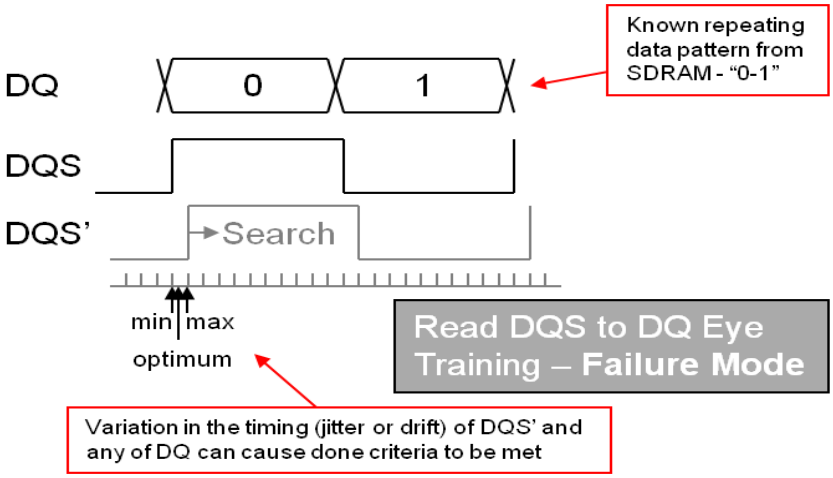


Figure 8. Read DQS to DQ Eye Training - Failure Mode

Workaround 1: 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 following registers (one register each for the four data byte lanes and the ECC byte lane): DDR3_CONFIG_REG_52, DDR3_CONFIG_REG_53, DDR3_CONFIG_REG_54, DDR3_CONFIG_REG_55, DDR3_CONFIG_REG_56, DDR3_CONFIG_REG_57, DDR3_CONFIG_REG_58, DDR3_CONFIG_REG_59 and DDR3_CONFIG_REG_60 at addresses 0x026204D4, 0x026204D8, 0x026204DC, 0x026204E0, 0x026204E4, 0x026204E8, 0x026204EC, 0x026204F0 and 0x026204F4 respectively. 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. After programming above registers, proceed to enable auto-leveling. Please refer to the *DDR3 Initialization Application Report* (SPRABL2) for details of the sequence of steps.

Workaround 2: 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* (SPRUGV8). The values programmed are dependent on the specific board characteristics.

Workaround 3: 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_52, DDR3_CONFIG_REG_53, DDR3_CONFIG_REG_54, DDR3_CONFIG_REG_55, DDR3_CONFIG_REG_56, DDR3_CONFIG_REG_57, DDR3_CONFIG_REG_58, DDR3_CONFIG_REG_59 and DDR3_CONFIG_REG_60= 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.

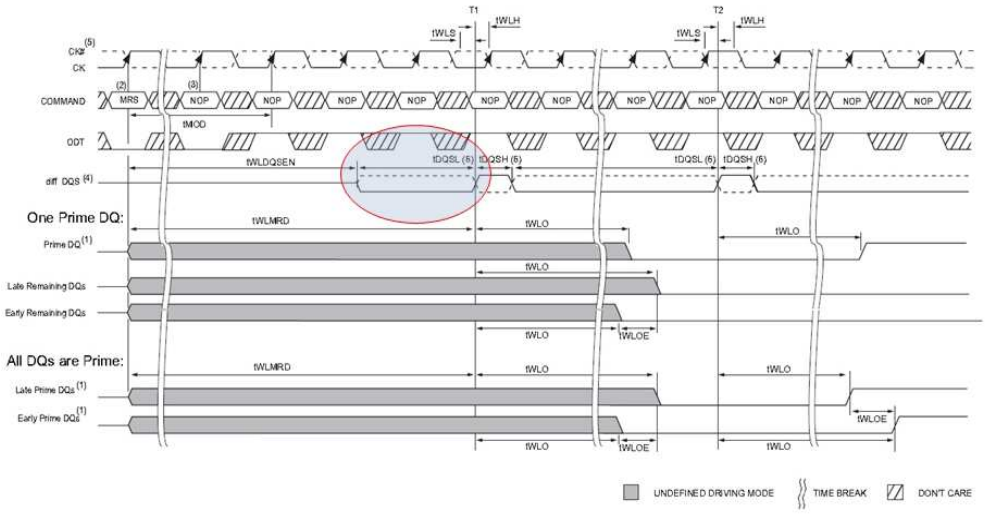
The time between incremental leveling events is set by the incremental leveling

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 18 *DDR3 Incremental Write Leveling Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: As shown in Figure 9, 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.



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.

Figure 9. Timing Details of Writing Leveling Sequence

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

Advisory 19 *ARM Memory Corruption Issue*

Revision(s) Affected: 1.0, 1.1

Details: ARM may evict a line from its cache and then issue a write to a location in the line that was evicted. In some instances it is possible that the “second write” will actually end up being written to the DDR first. When the evicted line is eventually written into DDR it will overwrite the updated value corresponding to the “second write.”

Workaround: Forcing strict ordering of all commands in the EMIF will eliminate this issue. By default re-ordering is enabled. To disable re-ordering, set PR_OLD_COUNT field in LAT_CONFIG register (0x2100_0054) in the DDR memory controller to 0. Setting PR_OLD_COUNT to 0 forces the DDR memory controller to order all memory commands in arrival order. See SPRUGV8C for details on the memory controller and its registers.

Advisory 20 *NAND Boot Failure Issue*

Revision(s) Affected: 1.0, 1.1

Details: Booting from a NAND flash device will fail when the device is run at full speed. The root cause of the problem is that insufficient time is allowed for the NAND device to complete the initial reset command sent to it by the ARM. After sending the reset command, the ARM executes a wait loop. This wait loop allows a maximum number of iterations before halting and declaring that the NAND is not responding correctly. The wait loop does not allow enough time for the NAND to complete its initial reset sequence.

Workaround 1: Force the ARM to run at a lower speed. NAND boot is supported only when the ARM is the boot master. At startup, the PLL will be programmed for the ARM to run at the max speed rating for the specific device. The max speed rating for the device is read from the efuse. The speeds for device are 1 GHz and 1.2 GHz. In some cases, the ARM can be tricked to run slower by indicating false input clock information in the boot mode pins read by the device at power up. For example, if device is rated to run at 1.2 GHz and the actual input clock is 122.88 MHz, the ARM can be made to run slower by setting the boot mode pins to indicate that the input clock speed is 312.50 MHz. This will result in the PLL being programmed for the ARM to run at approximately 472 MHz instead of 1.2 GHz (122.88 / 312.5 * 1200). This will slow down the wait loop execution time enough to allow the NAND to complete its reset sequence, resulting in a successful boot. This workaround has been verified to work with two versions of Micron NAND devices whose reset sequences both take approximately 300 micro-seconds. This workaround is valid for some combinations of the device's speed rating, actual input clock, and indicated clock. A pass/fail table of the various combinations is shown in Table 5. After NAND boot is complete, the downloaded code may then re-program the PLL for full speed device operations, i.e. the slow execution is only needed for the boot sequence. This workaround should not be used for any boot mode except NAND.

NOTE: Forcing the ARM to run slower may have undesired side effects for other boot modes.

Table 5. Pass/Fail Results of Various Speed Combinations

Input Clock (MHz)	Boot Mode Clock Info (MHz)	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot
50	50	1000	1000	Fail	1200	1200	Fail
50	66.67	1000	749.96	Fail	1200	899.955	Fail
50	80	1000	625	Fail	1200	750	Fail
50	100	1000	500	Pass	1200	600	Fail
50	156.25	1000	320	Pass	1200	384	Pass
50	250	1000	200	Pass	1200	240	Pass
50	312.5	1000	160	Pass	1200	192	Pass
50	122.88	1000	406.90	Pass	1200	488.28	Pass
66.67	50	1000	1333.4	Invalid	1200	1600.08	Invalid
66.67	66.67	1000	1000	Fail	1200	1200	Fail
66.67	80	1000	833.375	Fail	1200	1000.05	Fail
66.67	100	1000	666.7	Fail	1200	800.04	Fail
66.67	156.25	1000	426.69	Pass	1200	512.03	Pass
66.67	250	1000	266.68	Pass	1200	320.02	Pass
66.67	312.5	1000	213.34	Pass	1200	256.01	Pass
66.67	122.88	1000	542.56	Pass	1200	651.07	Fail
80	50	1000	1600	Invalid	1200	1920	Invalid

Table 5. Pass/Fail Results of Various Speed Combinations (continued)

Input Clock (MHz)	Boot Mode Clock Info (MHz)	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot
80	66.67	1000	1199.94	Invalid	1200	1439.93	Invalid
80	80	1000	1000	Fail	1200	1200	Fail
80	100	1000	800	Fail	1200	960	Fail
80	156.25	1000	512	Pass	1200	614.4	Fail
80	250	1000	320	Pass	1200	384	Pass
80	312.5	1000	256	Pass	1200	307.2	Pass
80	122.88	1000	651.04	Fail	1200	781.25	Fail
100	50	1000	2000	Invalid	1200	2400	Invalid
100	66.67	1000	1499.925	Invalid	1200	1799.91	Invalid
100	80	1000	1250	Invalid	1200	1500	Invalid
100	100	1000	1000	Fail	1200	1200	Fail
100	156.25	1000	640	Fail	1200	768	Fail
100	250	1000	400	Pass	1200	480	Pass
100	312.5	1000	320	Pass	1200	384	Pass
100	122.88	1000	813.80	Fail	1200	976.56	Fail
156.25	50	1000	3125	Invalid	1200	3750	Invalid
156.25	66.67	1000	2343.63	Invalid	1200	2812.36	Invalid
156.25	80	1000	1953.125	Invalid	1200	2343.75	Invalid
156.25	100	1000	1562.5	Invalid	1200	1875	Invalid
156.25	156.25	1000	1000	Fail	1200	1200	Fail
156.25	250	1000	625	Fail	1200	750	Fail
156.25	312.5	1000	500	Pass	1200	600	Fail
156.25	122.88	1000	1271.57	Invalid	1200	1525.88	Invalid
250	50	1000	5000	Invalid	1200	6000	Invalid
250	66.67	1000	3479.81	Invalid	1200	4499.78	Invalid
250	80	1000	3125	Invalid	1200	3750	Invalid
250	100	1000	2500	Invalid	1200	3000	Invalid
250	156.25	1000	1600	Invalid	1200	1920	Invalid
250	250	1000	1000	Fail	1200	1200	Fail
250	312.5	1000	800	Fail	1200	960	Fail
250	122.88	1000	2034.51	Invalid	1200	2441.41	Invalid
312.5	50	1000	6250	Invalid	1200	7500	Invalid
312.5	66.67	1000	4687.27	Invalid	1200	5624.72	Invalid
312.5	80	1000	3906.25	Invalid	1200	4687.5	Invalid
312.5	100	1000	3125	Invalid	1200	3750	Invalid
312.5	156.25	1000	2000	Invalid	1200	2400	Invalid
312.5	250	1000	1250	Invalid	1200	1500	Invalid
312.5	312.5	1000	1000	Fail	1200	1200	Fail
312.5	122.88	1000	2543.13	Invalid	1200	3051.76	Invalid
122.88	50	1000	2457.6	Invalid	1200	2949.12	Invalid
122.88	66.67	1000	1843.11	Invalid	1200	2211.73	Invalid
122.88	80	1000	1536	Invalid	1200	1843.2	Invalid
122.88	100	1000	1228.8	Invalid	1200	1474.56	Invalid
122.88	156.25	1000	786.43	Fail	1200	943.72	Fail
122.88	250	1000	491.52	Pass	1200	589.83	Fail
122.88	312.5	1000	393.21	Pass	1200	471.86	Pass

Table 5. Pass/Fail Results of Various Speed Combinations (continued)

Input Clock (MHz)	Boot Mode Clock Info (MHz)	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot	ARM Rating (MHz)	ARM Frequency (MHz)	Power Up Boot
122.88	122.88	1000	1000	Fail	1200	1200	Fail

Workaround 2: Select UART-primary, NAND-secondary boot mode. When the ARM is the boot master, the boot mode pins always specify primary and secondary boot modes. The primary boot mode will be attempted first and, if not successful, the secondary mode will be tried. When UART is the primary boot mode, the ARM will poll the UART for boot data. If no boot data appears before the time-out, it will then attempt the secondary boot mode. When NAND is the secondary boot mode, the ARM will then attempt to boot from NAND. In this case, the initial NAND boot attempt still fails. The ARM then re-tries UART boot. When UART boot fails the second time, it will again try NAND boot. The second NAND boot attempt will succeed since the NAND devices finish their second reset sequence in a much shorter amount of time (5 micro-seconds vs. 300 micro-seconds). The wait loop is long enough to let the second reset command complete. This will work for all device speeds. Because of the long UART boot time-out periods, this workaround adds approximately 75 seconds to the boot time of the device.

Workaround 3: Because NAND devices complete the second and subsequent reset sequences faster when compared to the initial reset sequence (5 micro-seconds vs. 300 micro-seconds) NAND boot can be made to work by simply taking the device back into reset and then releasing it from reset. This extra reset sequence must be done without cycling power to the NAND device. Resetting the device after an initial NAND boot attempt works since the wait loop is long enough for success on a second boot attempt.

Advisory 21 *Tracer_DDR_2 Master ID Filtering Limitation Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: In silicon revisions 1.0 and 1.1, Tracer_DDR_2 is not able to trace the transactions from ARM CorePac to DDR.

In silicon revision 1.3, Tracer_DDR_2 is able to trace the transactions from ARM CorePac to DDR. But the tracer could not differentiate the transactions of master ID (n) and master ID (n+128). It will only have the limitation when both BCP (with master IDs 96~98) and ARM CorePac (with master IDs 224~226) are trying to access DDR.

Workaround: In silicon revision 1.3, the user could avoid the limitation in Tracer_DDR_2 by not tracing the access from BCP and ARM CorePac to DDR at the same time.

Advisory 22 *Single MFENCE Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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 12- L2 Cache Corruption During Block and Global Coherence Operations Issue

Note that the 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 23 *Read Exception and Data Corruption Issue*

Revision(s) Affected: 1.0, 1.1, 1.3

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

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

Figure 10. Initial state of pre-fetch buffer

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

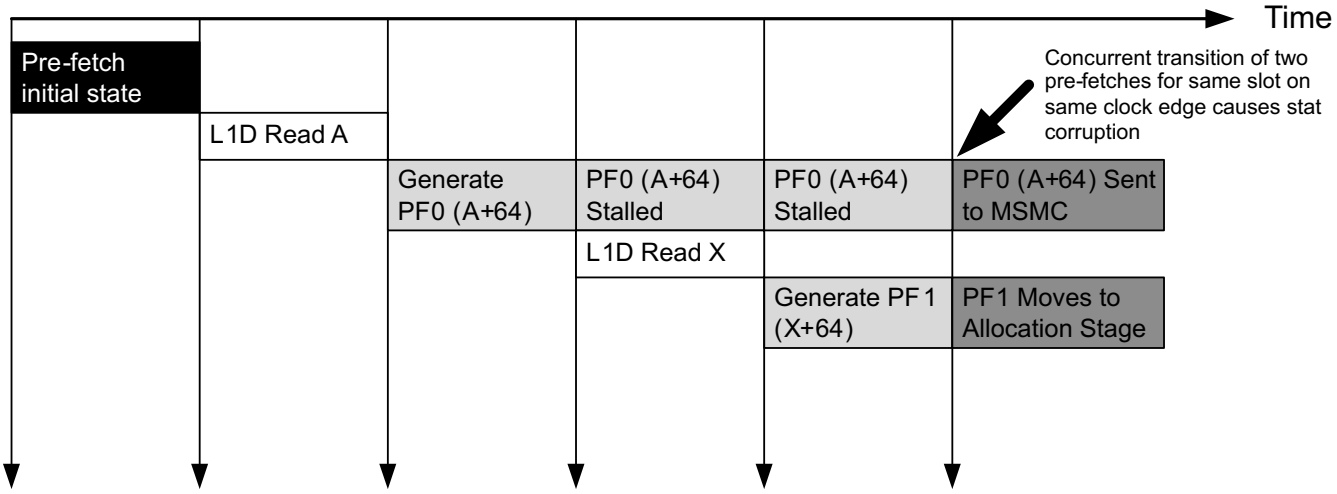


Figure 11. Sequence diagram

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 6. 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 24 *EMIF16 (NOR) Boot Error From Chip Selects Region CE2 and CE3 Issue*

Revision(s) Affected: 1.0, 1.1

Summary: An error exception will be generated when performing EMIF16 boot from NOR with half of EMIF chip select region CE2, addresses 0x7A00_0000 – 0x7BFF_FFFF, and all of EMIF chip select region CE3, addresses 0x7C00_0000 – 0x7FFF_FFFF. This is applicable for non-secure devices only.

Details: EMIF16 (NOR) boot is performed from NOR flash on non-secure devices. Access to the NOR flash is handled by the EMIF16 hardware block. A MPU (memory protection unit) controls read/write/execute privileges for memory areas handled by the EMIF16 block. Due to a problem in the initialization of the MPU some of the memory areas handled by the EMIF do not have the execute privilege required for EMIF16 (NOR) boot. Any memory area without execute privilege will generate an error exception if the processor attempts to execute from that area.

The EMIF block contains 4 chip selects regions, CE0 through CE3. Execute privilege is granted for CE0, addresses 0x7000_0000 – 0x73FF_FFFF, CE1, addresses 0x7400_0000 – 0x77FF_FFFF, and the first half of CE2, addresses 0x7800_0000 – 0x79FF_FFFF. Execute privilege is not granted for the second half of CE2, addresses 0x7A00_0000 – 0x7BFF_FFFF, and all of EMIF CE3, addresses 0x7C00_0000 – 0x7FFF_FFFF. The means EMIF16 (NOR) boot cannot be performed when using the second half of CE2 and all of CE3. When booting from NOR on a secure device the boot image is copied from the NOR and then authenticated/decrypted and executed from RAM. The MPU initialization gives read access to the full NOR, this means that on a secure device NOR boot is available from all chip selects.

Workaround1: When EMIF16 boot from NOR is desired, connect the NOR flash to the device using only CE0, CE1, or CE2. If CE2 is used, the boot image size must be limited to 2 MB.

Workaround2: The address space covered by the second half of CE2 can be “recovered” if code being executed from the first half of the chip select modifies the MPU settings to allow execute privilege prior to executing from the second half of the chip select region.

Advisory 25 *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, 1.1, 1.3

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 7. 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

(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.

Table 7. Header Fields Written to SPM by the Fetch Process (continued)

Channel Type	Fetched Header Fields Written to SPM
E-AGCH	DTX

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 26 *BCP Soft Slicer Lock-Up Issue*

Revision(s) Affected 1.0, 1.1, 1.3

Summary The Bitrate Coprocessor (BCP) Soft-Slicer (SSL) sub-module can enter a dead-locked state and cause the BCP Uplink chain to lock up on some devices. The issue is seen in WCDMA Uplink processing.

Details The Start-of-Packet (SOP) signal in the WCDMA SSL, which is responsible for controlling data processing in the SSL sub-module, is not initialized correctly. This can cause the SSL to enter a deadlocked state on some devices. Incoming packets are blocked from entering the SSL sub-module and BCP will not make progress in processing the packets. The data logger does not show the output packet count incrementing for the SSL sub-module and there is no Data Log Entry in the SSL Data Logger RAM.

The behavior is random and is dependent upon the power-on/reset value of the Start-of-Packet signal.

As a result, the SSL sub-module is not usable on the affected devices.

Workaround The issue can be resolved by using a software workaround to initialize the BCP SSL internal memory. TI provides a pre-compiled library which the user needs to link with the application code. The library contains a C routine which should be called right after the BCP is enabled. A header file is also provided along with the library for forward function prototype declaration.

The BCP SSL (Soft-slicer) works correctly after executing the workaround function, until the BCP is power cycled.

NOTE: The software workaround is effective for 1.3 silicon only.

The software workaround patch files can be downloaded from the following location:

http://software-dl.ti.com/keystone/Keystone-1_BCP/TI_keystone_bcp_ssl_patch/index_FDS.html

Advisory 27 *False DDR3 Write ECC Error Reported Under Certain Conditions*

Revision(s) Affected 1.0, 1.1, 1.3

Problem Summary: 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.

Details The write ECC error interrupt can be enabled by setting the WR_ECC_ERR_SYS bit in the Interrupt Enable Set Register (IRQSTATUS_ 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 will have to 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. It should be noted 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.

FAQ: Q: 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 transactions they have sent result in the error. How do these masters respond to a DDR3 ECC error?

A: 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 8.

Table 8. Master Behavior in Response to sstatus/rstatus Flagged Due to DDR3 ECC Errors

Master	Bus Error Returned	Error Status Captured	Alarm Notification
C66x CorePac	sstatus, 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.
ARM CorePac	sstatus, rstatus	Details on the exception are captured in the CP15 registers: Data Fault Status Register (DFSR), Instruction Fault Status Register (IFSR) or Auxiliary Data Fault Status Register (ADFSR). The address that generated the abort can be seen by reading Data Fault Address Register (DFAR) for synchronous aborts.	The ARM CorePac will trigger an external abort exception. They are disabled by default and should be enabled via the CPSR.A bit (Current Program Status Register).
PCIe	sstatus, 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	sstatus, 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	sstatus, rstatus	Error status is not logged.	Error interrupt is not reported
TSIP	sstatus, 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	sstatus, rstatus	Error responses set a bit in the AMU_INT_ICSR register based on the CPRIVID of the transactions. The RIO_AMU_ERR_CAPT0, RIO_AMU_ERR_CAPT1 registers will contain the address of the non-posted transactions 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	sstatus, 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.
10GE	sstatus, rstatus are not used	NA	NA

NOTE: Check your device data manual to see which masters are applicable.

Advisory 28 *Descriptors Placed in PCIe Memory Space can Cause Problems*

Revision(s) Affected 1.0, 1.1, 1.3

Problem Summary: Packet DMA can generate write transactions with partial byte enables when trying to access descriptors. This can cause problems if the descriptors are stored in PCIe memory space since PCIe cannot handle partial byte enables.

Details Per the JEDEC specification for DDR3, the PHY is expected to transmit the Data Strobe, DQS, and Data signals, DQn, such that the Data Strobe transitions near the center of the Data signal bit period or 'eye'. The DDR3 PHY contains values in DLLs that control the launch timing for the DQS and DQn signals for each byte lane. To meet this timing in the PHY, the offset between the Data Strobe and the Data signals associated with that strobe should be 1⁄4 of a DDR3 clock period (which is half the data eye width). Since this offset is controlled by a DLL that has 256 taps per clock period, the offset between the DQS and DQn signals should always be 64 (or 0x40). This offset is fixed by the DATA_REG_PHY_DQ_OFFSET field in the DDR3_CONFIG_REG_1 register. Unfortunately, the default value in this register bit-field is 32 (0x20) which programs an offset equal to 1⁄8 of a DDR3 clock period.

Workaround As long as host-mode descriptors are used and these descriptors are located in a memory space that can properly handle partial byte enables (such as L2 SRAM, DDR3 or MSMC), the issue will not affect Packet DMA accesses to PCIe memory space. As mentioned earlier, data buffers can be stored in PCIe memory space without any problems.

Usage Note 1 *TAC DL TPC Timing Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 2 *Packet DMA Clock-Gating for AIF2 and Packet Accelerator Subsystem Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: Clock-gating the AIF2 or Packet Accelerator Subsystem 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 3 *VCP2 Back-to-Back Debug Read Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: In a debug scenario, back-to-back configuration 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 configuration bus accesses do not occur or do not perform back-to-back VCP2 debug (configuration bus) reads from DSP software. A single cycle between reads is sufficient for proper operation.

Usage Note 4 *DDR3 ZQ Calibration Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 5 *I²C Bus Hang After Master Reset Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 6 *MPU Read Permissions for Queue Manager Subsystem Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 Subsystem (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 7 *Queue Proxy Access Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: When there are multiple C66x/ARM 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 or ARM CorePac cannot generate bursts larger than 8 bytes. The Queue Proxy is designed to allow C66x or ARM CorePac to push/pop descriptors using multiple transactions. However, when the C66x or ARM 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 or ARM CorePac should not use the Queue Proxy region to push/pop descriptors. The C66x or ARM 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 or ARM 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 or ARM 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 or ARM CorePac should use the queue peek region to get the information.

Usage Note 8 *TAC E-AGCH Diversity Mode Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 9 *Minimizing Main PLL Jitter Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 *TMS320TCI6612 Communications Infrastructure Keystone SoC* data manual (SPRS784) for more information.

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 10 *MSMC and Async EMIF Accesses from ARM Core Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: The ARM Core accesses MSMC SRAM through the SMS (System MSMC SRAM slave) port. The MSMC SMS port does not support cache line wrap accesses. For this reason all cache line accesses from the ARM core to the MSMC memory (whether they be data or instruction cache access) will not go through.

The Async EMIF module supports only cache line wrap accesses at 16B or 32B cache line sizes. The ARM Core data and instruction cache line size is 64B. As the Async EMIF slave port does not cache line wrap for a cache line size of 64B, all cache line wrap accesses of 64B size will fail and result in unknown behavior.

The ARM Core must be configured to have all of MSMC and all of Aysnc EMIF as non-cacheable.

Usage Note 11 *OTP Memory Controller Does Not Operate at Full Speed Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: The OTP Memory controller is programmable in secure device only and operates on the CPU/6 clock. The OTP memory controller does not operate as expected at full speed. Therefore, we must ensure that the OTP memory is programmed only when the main PLL is 750MHz or lower. This restriction applies only during programming of the OTP memory. Outside of this time, the main PLL can operate normally.

Workaround: In secure device, program the OTP memory when the device main PLL is 750MHz or lower (such as clock frequency in bypass mode). If the main PLL is bypassed, the device will run using a bypass clock. The OTP memory can be programmed with any supported bypass clock frequency.

Usage Note 12 *$\overline{\text{POR}}$ and $\overline{\text{RESETFULL}}$ Sequence Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: For boot configuration pins to be latched correctly, during the power sequencing and reset control for chip initialization, $\overline{\text{RESETFULL}}$ must be held low for a period after the rising edge of $\overline{\text{POR}}$ 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 $\overline{\text{RESETFULL}}$ and must meet the setup and hold times. Timing requirements are specified in the device-specific data manual, *TMS320TCI6612 Communications Infrastructure Keystone SoC* data manual (SPRS784) for more information.

Usage Note 13 *AIF2 LTE 3MHz and 1.4MHz Support Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: AIF2 requires the packet length to be divisible by 4 due to the OBSAI message restriction but LTE 3MHz and 1.4MHz symbol length do not satisfy the requirement due to the cyclic prefix (CP) length in normal cyclic prefix case as shown in Table 9.

Table 9. 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 12 shows detailed description on Ingress:

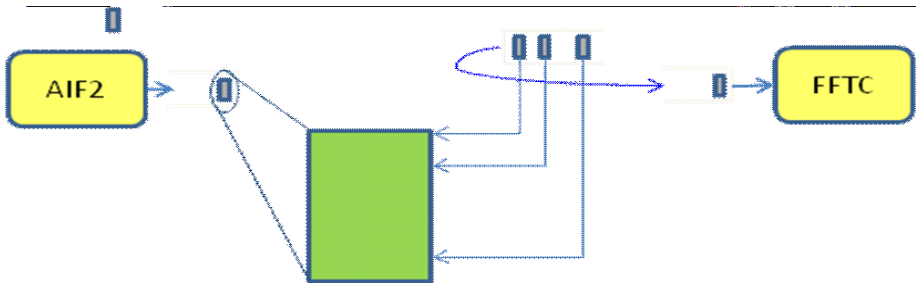


Figure 12. Symbol Processing on Ingress

Pre-configure 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 13.

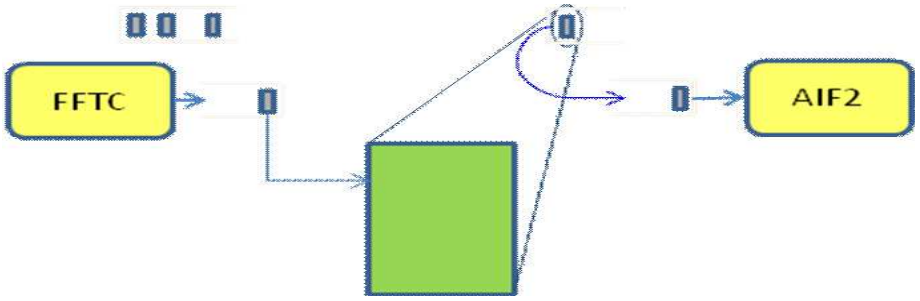


Figure 13. Symbol Processing on Egress

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

Revision(s) Affected: 1.0, 1.1, 1.3

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 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 15 *The Clock Input to NETCP Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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

Workaround: In order to set the 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 during Ethernet boot, the Boot ROM code correctly sets this bit to 1. However, in all other boot modes this bit has a default value of 0 and the software must change the value to 1.

Usage Note 16 *Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: During the 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 1.0 V. When VDDR gets approximately to the 400 mV mark, the VDDR rail stops tracking the VDDT rail. Leakage observed here comes from the SerDes module that contains 1 V transistors. It has been verified that this leakage is expected and has no impact on the reliability of the device.

Usage Note 17 *CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 the following guidelines be observed 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.
- Because 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* (SPRABI1) need to be satisfied. The multi-rank equations will not be valid.

Usage Note 18 *USIMIO Pin IPU Usage Note*

Revision(s) Affected: 1.0, 1.1

Details: The USIMIO pin was implemented as IPD (internal pulldown) pin in silicon revisions 1.0 and 1.1. This pin is fixed to be IPU (internal pullup) pin in silicon revision 1.3.

Usage Note 19 *Revised PLL Programming Sequence Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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* (SPRUGV2).

Usage Note 20 *Core Wake Up on RESET Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 21 *BSDL Testing Support Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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 10. 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 22 *Ethernet Boot Size Limitation Usage Note*

Revision(s) Affected: 1.0, 1.1

Details: Booting the device by Ethernet will fail when the boot image is larger than 256 Kbytes. The root cause of the problem is the fixed size 256 Kbyte buffer allocated for storing the image when it is received via Ethernet. The ROM boot loader will discard any bytes over the 256 Kbyte size limit. There is no indicator sent to the host that a truncated image has been stored. When the last bytes of the image have been received the ROM boot loader will begin to execute it even if the end of the image has been discarded due to it being longer than the buffer size.

Workaround 1: Keep the size of the boot image below 256 Kbytes.

Workaround 2: Create an intermediate boot loader that is smaller than 256 Kbytes. It can be successfully loaded by the ROM boot loader via Ethernet. When the intermediate loader is executed it can then load an image of any size.

Usage Note 23 *Secure Boot Size Limitation Usage Note*

Revision(s) Affected: 1.0, 1.1

Details: Booting a secure device will fail when the boot image is larger than 128 Kbytes. The root cause of the problem is a conversion error when converting an image size in bytes to the number of 32-bit words. The conversion results in the number of words being incorrectly calculated when the image byte size is larger than 128 Kbytes. This causes authentication to fail since the wrong number of words is validated. When this error happens no indication is given to the host that authentication failed, it simply appears that secure boot failed.

Workaround 1: Keep the size of the secure boot image below 128 Kbytes.

Workaround 2: Create an intermediate boot loader that is smaller than 128 Kbytes. It can be successfully loaded by the ROM boot loader on a secure device. When the intermediate loader is executed it can then load an image of any size.

Usage Note 24 *AIF2 CPRI FastC&M Restrictions and Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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

Revision(s) Affected: 1.0, 1.1, 1.3

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* (SPRABI2) is absolutely required.

Usage Note 26 *Performance Degradation for Asynchronous Access Caused by an Unused Feature Enabled in EMIF16 Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

Details: Although it supports only asynchronous mode operation on the device, the EMIF16 module has a legacy 'synchronous mode' feature that is enabled by default. While this synchronous mode is enabled, EMIF16 issues periodic refresh commands that take precedence over asynchronous accesses commands and stall the execution of the latter until the refresh command is executed. This stall results in reduced throughput of asynchronous accesses when EMIF16 tries to read or write to the asynchronous memory. The stall will manifest itself as a long delay between asynchronous accesses.

Workaround: Programming bit 31 at the 32-bit address 0x20C00008 to 1 will disable the synchronous mode feature.

```
*(Uint32*) 0x20C00008 |= 0x80000000; //Disable synchronous mode feature
```

When the synchronous mode is disabled, EMIF16 will not issue any refresh commands. This will no longer result in stall cycles between asynchronous accesses and thus the performance will be improved.

This bit affects only the refreshes issued by EMIF16. It does not affect the rest of the device.

Usage Note 27 *DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note*

Revision(s) Affected: 1.0, 1.1, 1.3

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. See the *KeyStone Architecture DDR3 Memory Controller User Guide* (SPRUGV8) for details.

The following are examples of how read and write transactions behave when the Class of Service feature is enabled. The COS 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* (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 28 *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, 1.1, 1.3

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 11. TAC Registers Configuration

TAC_G0_DATA_FETCH	TAC_G0_HEAD_FETCH
1	9
1	17

Table 11. TAC Registers Configuration (continued)

TAC_G0_DATA_FETCH	TAC_G0_HEAD_FETCH
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 12. 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

Revision History

Changes from December 17, 2013 to May 30, 2015 (from C Revision (December 2013) to D Revision) **Page**

- Added Advisory 27, ***False DDR3 Write ECC Error Reported Under Certain Conditions*** 45
- Added Advisory 28, ***Descriptors Placed in PCIe Memory Space can Cause Problems*** 47

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