TMS320C6652/54/55/57
Multicore Fixed and Floating-Point Digital Signal Processor
Silicon Revision 1.0

Silicon Errata

Literature Number: SPRZ381C
April 2012–Revised May 2016

Contents

1 Introduction 4
2 Device and Development Support Tool Nomenclature 4
3 Package Symbolization and Revision Identification 5
4 Silicon Updates 6
Revision History 53

List of Figures

1 Lot Trace Code Example for TMS320C6654 (CZH Package) 5
2 Read DQS to DQ Eye Training - Board View 10
3 Read DQS to DQ Eye Training - Algorithm 11
4 Read DQS to DQ Eye Training - Failure Mode 11
5 Timing Details of Writing Leveling Sequence 13
6 SRIO SerDes in Loopback Mode 15
7 Basic Flow for Analyzing CDM Risk 17
8 Initial state of pre-fetch buffer 25
9 Sequence diagram 26
10 DQ to DQS offset with default incorrect value (0x20) for DATA_REG_PHY_DQ_OFFSET in DDR3_CONFIG_REG_1 32
11 DQ to DQS offset with correctly programmed value (0x40) for DATA_REG_PHY_DQ_OFFSET in DDR3_CONFIG_REG_1 32

List of Tables

1 Lot Trace Codes 5
2 Silicon Revision Variables 5
3 Silicon Revision Updates 6
4 Impact on Various Lane Speeds 14
5 Possible Outcomes Depending on Relative Timing of Subsequent Pre-Fetchable Data Access to X+64, PF0 Return and PF1 Return 26
6 Master Behavior in Response to sstatus/rstatus Flagged Due to DDR3 ECC Errors 29
7 Register Information for the Peripheral 48

TMS320C6652/54/55/57 Multicore Fixed and Floating-Point Digital Signal Processor Silicon Revision 1.0

1 Introduction

This document describes the silicon updates to the functional specifications for the *TMS320C6652/54/55/57* fixed- and floating-point digital signal processor. See the device-specific data manual, TMS320C6655/57 Multicore Fixed and Floating-Point Digital Signal Processor, TMS320C6652/54 Multicore Fixed and Floating-Point Digital Signal Processor 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., TMS320C6652/54/55/57CMS). 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 CZH package is shown in Figure 1. Figure 1 also shows an example of the C6654CZH package symbolization.

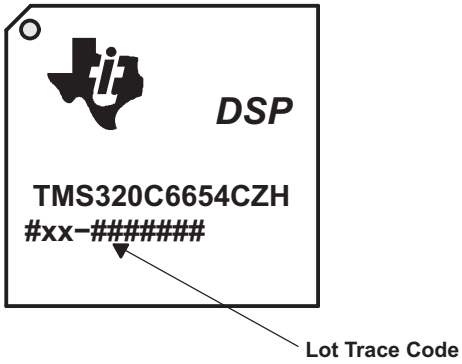


Figure 1. Lot Trace Code Example for TMS320C6654 (CZH Package)

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 C6652/54/55/57 devices.

Table 1. Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision

The C6652/54/55/57 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 C6652/54/55/57.

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 TMS320C6655/57 Multicore Fixed and Floating-Point Digital Signal Processor, TMS320C6652/54 Multicore Fixed and Floating-Point Digital Signal Processor 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.

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 C6654/55/57 device.

Table 2. Silicon Revision Variables

Silicon Revision	CPU CSR Register	C66x CorePac MM_REVID Register	C6654/55/57 JTAGID Register
1.0	CSR[CPU_ID] = 15h CSR[REVISION_ID] = 00h	Rev. 2.0 MM_REVID[REVISION] = 0001h	JTAGID = 0x0B97A02F

More details on the JTAG ID and CorePac Revision ID Registers can be found in the TMS320C6655/57 Multicore Fixed and Floating-Point Digital Signal Processor, TMS320C6652/54 Multicore Fixed and Floating-Point Digital Signal Processor 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
HyperLink Temporary Blocking Issue(1)	Advisory 1	X
DDR3 Excessive Refresh Issue	Advisory 2	X
DDR3 Automatic Leveling Issue	Advisory 3	X
DDR3 Incremental Write Leveling Issue	Advisory 4	X
SRIO Status Control Symbols are Sent More Often Than Required Issue(1)	Advisory 6	X
Corruption of Control Characters in SRIO Line Loopback Mode Issue(1)	Advisory 7	X
SerDes Transit Signals Pass ESD-CDM up to ±150V Issue	Advisory 8	X
L2 Cache Corruption During Block and Global Coherence Operations Issue	Advisory 9	X
System Reset Operation Disconnects the SoC from CCS Issue	Advisory 10	X
Power Domains Hang when Powered Up Simultaneously with $\overline{\text{RESET}}$ (Hard Reset) Issue	Advisory 11	X
Boundary Scan for ECC Bits in DDR3 Not Reflecting Pin State Issue	Advisory 12	X
Single MFENCE Issue	Advisory 13	X
Read Exception and Data Corruption Issue	Advisory 14	X
False DDR3 Write ECC Error Reported Under Certain Conditions	Advisory 20	X
Descriptors Placed in PCIe Memory Space can Cause Problems	Advisory 28	X
Improper DDR3 PHY DQ to DQS	Advisory 29	X
Packet DMA Does Not Update RX PS Region Location Bit Usage Note	Usage Note 1	X
Packet DMA Clock-Gating Usage Note	Usage Note 2	X
VCP2 Back-to-Back Debug Read Usage Note(1)	Usage Note 3	X
DDR3 ZQ Calibration Usage Note	Usage Note 4	X
I^2C Bus Hang After Master Reset Usage Note	Usage Note 5	X
$\overline{\text{POR}}$ and $\overline{\text{RESETFULL}}$ Sequence Usage Note	Usage Note 6	X
MPU Read Permissions for Queue Manager Subsystem Usage Note	Usage Note 7	X
Queue Proxy Access Usage Note	Usage Note 8	X
Minimizing Main PLL Jitter Usage Note	Usage Note 9	X
Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note	Usage Note 10	X
CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note	Usage Note 11	X
Changing EMAC Priority Usage Note	Usage Note 12	X
Revised PLL Programming Sequence Usage Note	Usage Note 13	X
Core Wake Up on $\overline{\text{RESET}}$ Usage Note	Usage Note 14	X
BSDL Testing Support Usage Note	Usage Note 15	X
Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note	Usage Note 16	X
Performance Degradation for Asynchronous Accesses Caused by An Unused Feature Enabled in EMIF 16 Module Usage Note	Usage Note 17	X
DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note	Usage Note 18	X

(1) This item does not apply to the C6654 device. See the device-specific data manual for information about available peripheral modules.

Silicon Updates

Title	Page
Advisory 1 — *HyperLink Temporary Blocking Issue*	8
Advisory 2 — *DDR3 Excessive Refresh Issue*	9
Advisory 3 — *DDR3 Automatic Leveling Issue*	10
Advisory 4 — *DDR3 Incremental Write Leveling Issue*	13
Advisory 6 — *SRIO Control Symbols Are Sent More Often Than Required Issue*	14
Advisory 7 — *Corruption of Control Characters In SRIO Line Loopback Mode Issue*	15
Advisory 8 — *SerDes Transit Signals Pass ESD-CDM up to ±150 V Issue*	16
Advisory 9 — *L2 Cache Corruption During Block and Global Coherence Operations Issue*	18
Advisory 10 — *System Reset Operation Disconnects the SoC from CCS Issue*	20
Advisory 11 — *Power Domains Hang When Powered Up Simultaneously with $\overline{\text{RESET}}$ (Hard Reset) Issue*	21
Advisory 12 — *Boundary Scan for ECC Bits in DDR3 Not Reflecting Pin State Issue*	22
Advisory 13 — *Single MFENCE Issue*	23
Advisory 14 — *Read Exception and Data Corruption Issue*	24
Advisory 20 — *False DDR3 Write ECC Error Reported Under Certain Conditions*	28
Advisory 28 — *Descriptors Placed in PCIe Memory Space can Cause Problems*	30
Advisory 29 — *Improper DDR3 PHY DQ to DQS*	31
Usage Note 1 — *Packet DMA Does Not Update RX PS Region Location Bit Usage Note*	33
Usage Note 2 — *Packet DMA Clock-Gating Usage Note*	34
Usage Note 3 — *VCP2 Back-to-Back Debug Read Usage Note*	35
Usage Note 4 — *DDR3 ZQ Calibration Usage Note*	36
Usage Note 5 — *I^2C Bus Hang After Master Reset Usage Note*	37
Usage Note 6 — *$\overline{\text{POR}}$ and $\overline{\text{RESETFULL}}$ Sequence Usage Note*	38
Usage Note 7 — *MPU Read Permissions for Queue Manager Subsystem Usage Note*	39
Usage Note 8 — *Queue Proxy Access Usage Note*	40
Usage Note 9 — *Minimizing Main PLL Jitter Usage Note*	41
Usage Note 10 — *Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note*	42
Usage Note 11 — *CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note*	43
Usage Note 12 — *Changing EMAC Priority Usage Note*	44
Usage Note 13 — *Revised PLL Programming Sequence Usage Note*	45
Usage Note 14 — *Core Wake Up on $\overline{\text{RESET}}$ Usage Note*	46
Usage Note 15 — *BSDL Testing Support Usage Note*	47
Usage Note 16 — *Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note*	49
Usage Note 17 — *Performance Degradation for Asynchronous Accesses Caused by An Unused Feature Enabled in EMIF 16 Usage Note*	50
Usage Note 18 — *DDR3 Class of Service Feature Can Cause Higher Than Expected Latency Usage Note*	51

Advisory 1 *HyperLink Temporary Blocking Issue*

Revision(s) Affected: 1.0

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, if possible.

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

Advisory 2 *DDR3 Excessive Refresh Issue*

Revision(s) Affected: 1.0

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

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

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

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

Advisory 3 *DDR3 Automatic Leveling Issue*

Revision(s) Affected: 1.0

Details:

The DDR3 PHY integrated into the DDR3 Memory Controller of the device has a problem with one of the read-write leveling hardware features.

The read-write leveling hardware features are described in Read-Write Leveling section of the KeyStone Architecture DDR3 Memory Controller User Guide. 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 2, the DQS signal can be delayed relative to the DQ signal within the DDR3 PHY.

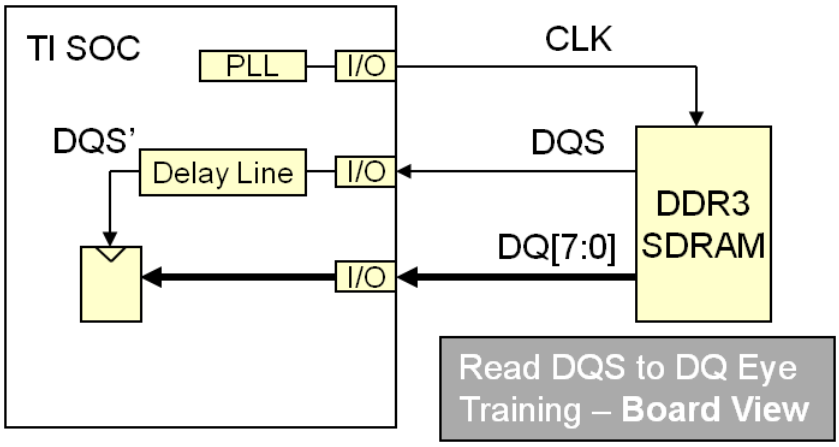


Figure 2. Read DQS to DQ Eye Training - Board View

As shown in Figure 3, 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.

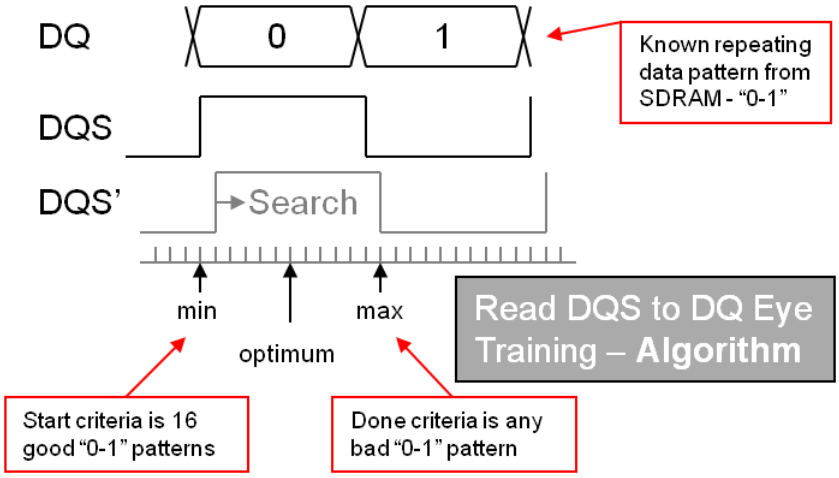


Figure 3. Read DQS to DQ Eye Training - Algorithm

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 4.

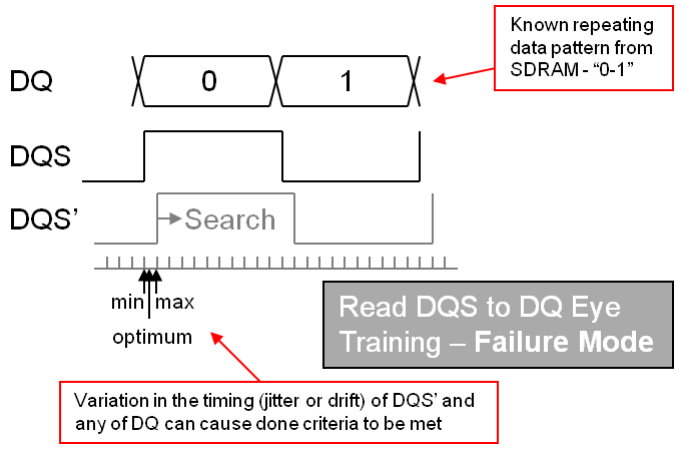


Figure 4. Read DQS to DQ Eye Training - Failure Mode

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 and DDR3_CONFIG_REG_60 at addresses 0x026204D4, 0x026204D8, 0x026204DC, 0x026204E0 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 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. The values programmed are dependent on the specific board characteristics. Limited support is available for this solution. The customer will need to assume responsibility for validating all required timing margins are met.

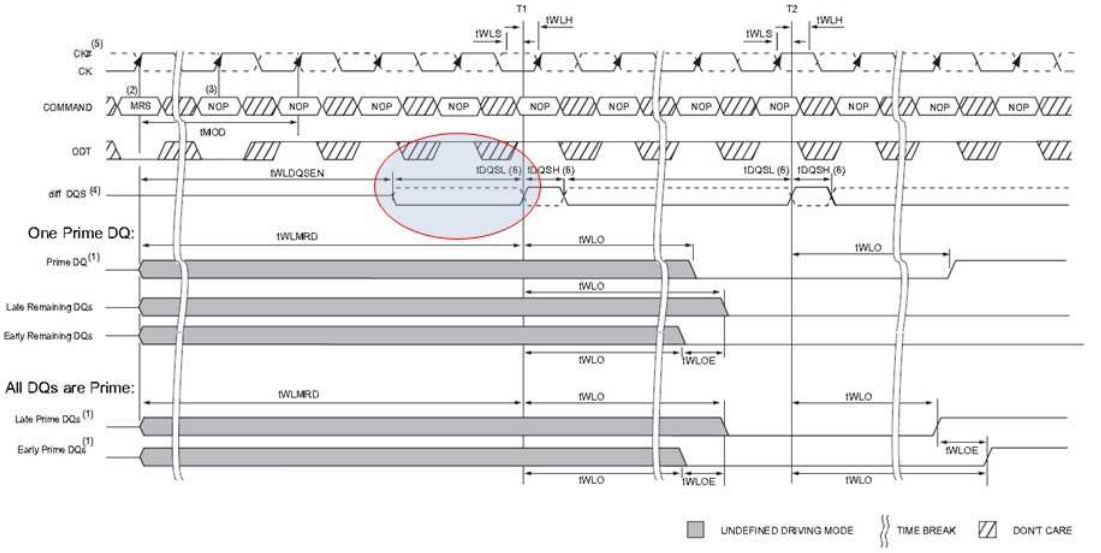
Workaround 3: The read data eye sample point can now be optimized by incremental read eye leveling. To do this, leave the read data eye training enabled (leave bit 9 in DDR3_CONFIG_REG_52, DDR3_CONFIG_REG_53, DDR3_CONFIG_REG_54, DDR3_CONFIG_REG_55 and DDR3_CONFIG_REG_60= 0), trigger automatic leveling 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. This solution is functional on standard DDR3 fly-by layouts and is referred to as Full Automatic Leveling.

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 4 *DDR3 Incremental Write Leveling Issue*

Revision(s) Affected: 1.0

Details: As shown in Figure 5, 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 5. 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 6 *SRIO Control Symbols Are Sent More Often Than Required Issue*

Revision(s) Affected: 1.0

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

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

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

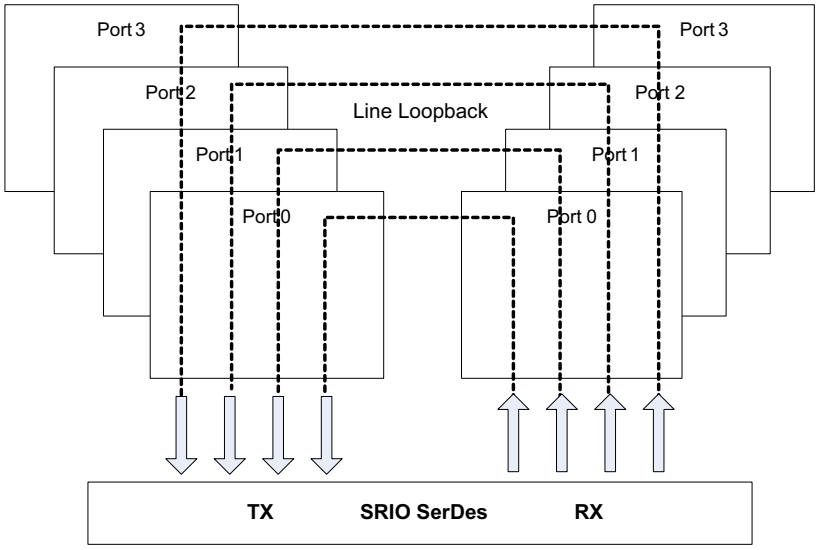


Figure 6. SRIO SerDes in Loopback Mode

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

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 7 shows the basic flow for analyzing this risk:

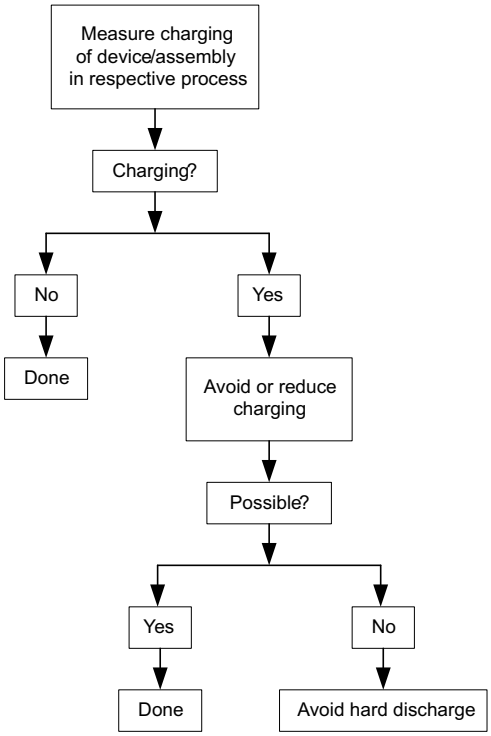


Figure 7. Basic Flow for Analyzing CDM Risk

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

Revision(s) Affected: 1.0

Details:

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

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

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

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

Workaround:

The 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:

7. Disable interrupts
8. Write 1 to the corresponding Global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV)
9. 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
10. Perform 16 NOPs
11. Restore interrupts

For further information about the cache control registers (BAR, WC, L1DINV, L1DWBINV, L2DINV, and L2DWBINV) see the TMS320C66x DSP CorePac User Guide. 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 .

Advisory 10 *System Reset Operation Disconnects the SoC from CCS Issue*

Revision(s) Affected: 1.0

Details: The CCS connection to targets will fail after a system reset is issued via CCS. The CCS connection to targets will also fail after 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 11 *Power Domains Hang When Powered Up Simultaneously with RESET (Hard Reset) Issue*

Revision(s) Affected: 1.0

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

Details: Power Domain affected:

- Power Domain 13 (CorePac 0, L1/L2 RAMs) for C6654;
- Power Domain 7 (MSMC RAM) and Power Domain 13 (CorePac 0, L1/L2 RAMs) for C6655;
- Power Domain 7 (MSMC RAM) and Power Domain 13-14 (CorePac 0-1, L1/L2 RAMs) for C6657.

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 indicates that the device is out of reset). If the RESETSTAT pin does not toggle and is stuck at low, the external host controller can infer that this issue has occurred. To recover, 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 12 *Boundary Scan for ECC Bits in DDR3 Not Reflecting Pin State Issue*

Revision(s) Affected: 1.0

Details: The check bits in the DDR EMIF (pins DDRCB[3:0]) do not accurately report the status of the IO pin when engaged in boundary scan. These pins will report indeterminate data when scanned. Boundary scan tests should ignore the reported state of these pins when examining the scan output data. Board connectivity tests should not rely on boundary scan for testing these pins.

Advisory 13 *Single MFENCE Issue*

Revision(s) Affected: 1.0

Details:

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

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

For example,

1. STORE_A
2. MFENCE
3. TRANSACTION_B

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

Workaround:

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

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

Note on coherence operations:

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

- Advisory 9 - 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 14 *Read Exception and Data Corruption Issue*

Revision(s) Affected: 1.0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Pathological sequence:

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

Allocation Pointer →

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 8. 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

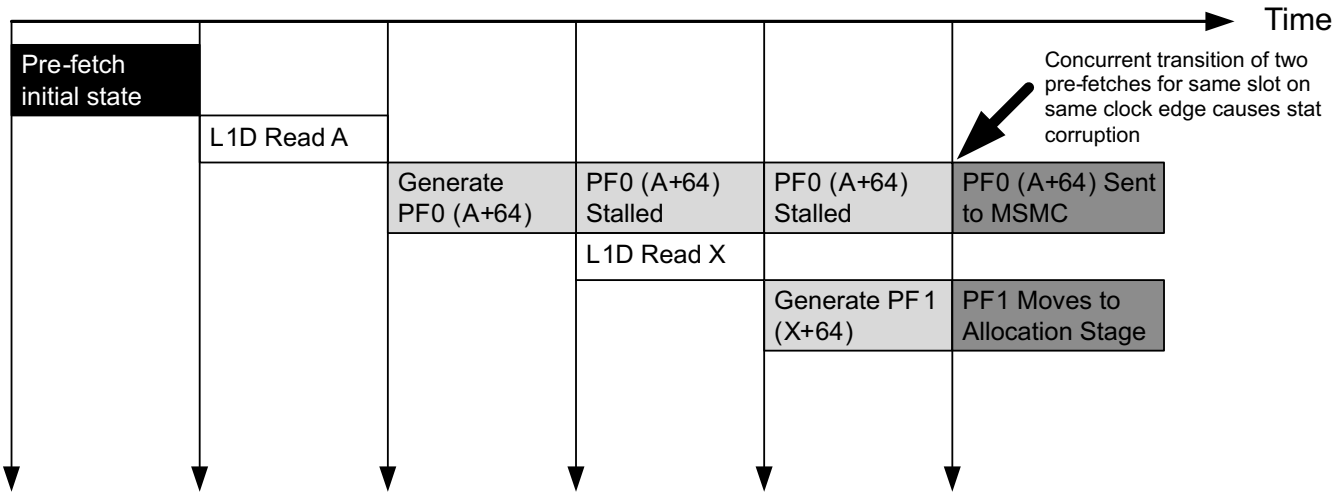


Figure 9. Sequence diagram

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 5. 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 20 *False DDR3 Write ECC Error Reported Under Certain Conditions*

Revision(s) Affected 1.0 for C665x family

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 soon 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 6.

Table 6. 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 for C665x family

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.

Advisory 29 *Improper DDR3 PHY DQ to DQS*

Revision(s) Affected 1.0 for C665x family

Problem Summary: On C665x devices, the default value of the DATA_REG_PHY_DQ_OFFSET field in the DDR3_CONFIG_REG_1 register causes DQ and DQS to be offset incorrectly during writes.

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 The DDR3 initialization software will need to properly set this bit field to 0x40. The GEL file snippets below show addition in bold of the proper assignment to DDR3_CONFIG_REG_1 near the beginning of the configuration sequence prior to the PHY reset. Figure 10 and Figure 11 represent the DQ-DQS offset resulting from incorrect (0x20) and correct (0x40) DATA_REG_PHY_DQ_OFFSET values.

```
#define DDR3_CONFIG_REG_1 (*(unsigned int*)(0x02620408))
...

DATA0_GTLVL_INIT_RATIO = 0xB0;
DATA1_GTLVL_INIT_RATIO = 0xB0;
DATA2_GTLVL_INIT_RATIO = 0xBD;
DATA3_GTLVL_INIT_RATIO = 0xC3;
.
.
DATA8_GTLVL_INIT_RATIO = 0xA4;

//Correct DQS-DQ write timing offset
DDR3_CONFIG_REG_1 = 0x01000000;

//Do a PHY reset. Toggle DDR_PHY_CTRL_1 bit 15 0->1->0
DDR_DDRPHYC &= ~(0x00008000);
DDR_DDRPHYC |= (0x00008000);
DDR_DDRPHYC &= ~(0x00008000);
```

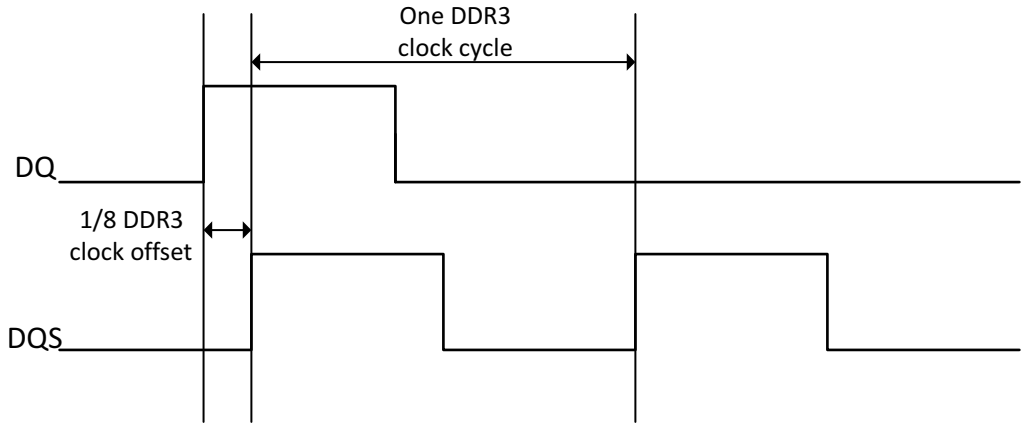


Figure 10. DQ to DQS offset with default incorrect value (0x20) for DATA_REG_PHY_DQ_OFFSET in DDR3_CONFIG_REG_1

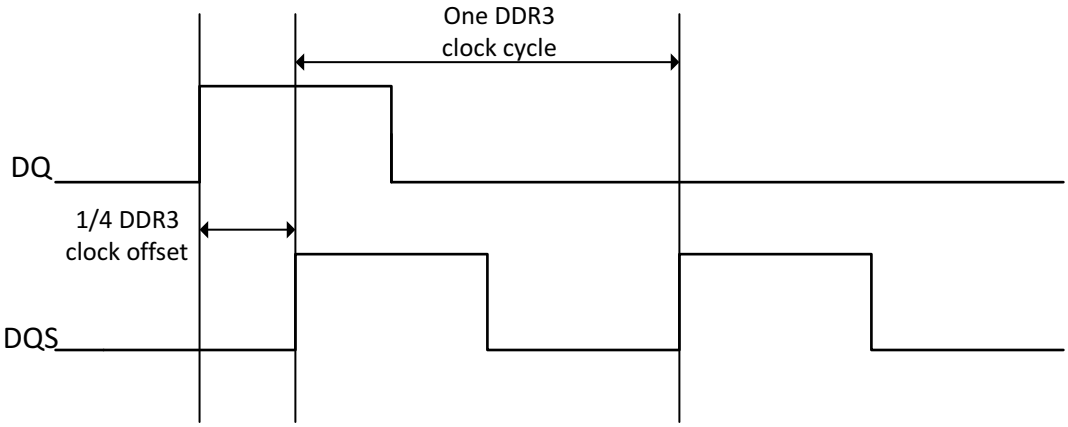


Figure 11. DQ to DQS offset with correctly programmed value (0x40) for DATA_REG_PHY_DQ_OFFSET in DDR3_CONFIG_REG_1

Usage Note 1 *Packet DMA Does Not Update RX PS Region Location Bit Usage Note*

Revision(s) Affected: 1.0

Details: The Packet DMA inside each of the Navigator-compliant modules fails to update the Protocol-Specific Region Location bit (bit 22 of Packet Descriptor Word 0) to a 1 when it is writing an RX host-mode packet to memory with protocol-specific (PS) words located in the start of the data buffer instead of the descriptor. This means that the software cannot use this bit to determine if any PS information 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 2 *Packet DMA Clock-Gating Usage Note*

Revision(s) Affected: 1.0

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

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

Usage Note 3 *VCP2 Back-to-Back Debug Read Usage Note*

Revision(s) Affected: 1.0

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

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

Usage Note 4 *DDR3 ZQ Calibration Usage Note*

Revision(s) Affected: 1.0

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

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

Usage Note 5 *I²C Bus Hang After Master Reset Usage Note*

Revision(s) Affected: 1.0

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

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

Usage Note 6 *$\overline{\text{POR}}$ and $\overline{\text{RESETFULL}}$ Sequence Usage Note*

Revision(s) Affected: 1.0

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, TMS320C6655/57 Multicore Fixed and Floating-Point Digital Signal Processor, TMS320C6652/54 Multicore Fixed and Floating-Point Digital Signal Processor.

Usage Note 7 *MPU Read Permissions for Queue Manager Subsystem Usage Note*

Revision(s) Affected: 1.0

Details: The Memory Protection Unit (MPU) has the ability to restrict the write and read permissions for bus masters like the DSP cores and System EDMAs when they attempt to access various portions of the address space on the device. One of the peripherals that may be access-controlled is the Queue Manager 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 8 *Queue Proxy Access Usage Note*

Revision(s) Affected: 1.0

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

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

Usage Note 9 *Minimizing Main PLL Jitter Usage Note*

Revision(s) Affected: 1.0

Details: Once the boot is complete, it is highly recommended that software reconfigure the Main PLL to the desired frequency, even if it is already achieved by the initial settings. To minimize the overall output jitter, the PLLs should be operated as close as possible to the maximum operating frequency. To maximize the VCO frequency within the PLL, the PLL should be clocked to 2× the intended frequency and the PLL Output Divider should be set to /2. The main PLL Output Divider should be set to divide-by-2 by the software by writing 0b0001 to bits [22:19] of the SECCTL register (address 0x02310108) in the PLL controller. A read-modify-write can be used to make sure other bits in the register are not affected. This register is documented in the TMS320C6655/57 Multicore Fixed and Floating-Point Digital Signal Processor, TMS320C6652/54 Multicore Fixed and Floating-Point Digital Signal Processor 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 *Current Flow Between VDDT and VDDR Rails During Power Sequencing of DSP Usage Note*

Revision(s) Affected: 1.0

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

Usage Note 11 *CAS Write Latency (CWL) at Low Speed Bins in DDR3 Multi-rank Configuration Usage Note*

Revision(s) Affected: 1.0

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

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

- Set bit 24 (indexed to 0) in DDR3_CONFIG_REG_12 at 0x02620434 to 1. This will constrain the DDR PHY to use only rank0 delays for reads/writes to both ranks. The reset value is 0 which allows each rank to use its own delay.
- Using only rank0 delays means that the fly-by/round-trip delay across ranks must be balanced for a given byte lane.
- Since rank0 delays are used by the PHY for both ranks, only the single rank equations for fly-by and round-trip delays from DDR3 Design Requirements for KeyStone Devices need to be satisfied. The multi-rank equations will not be valid.

Usage Note 12 *Changing EMAC Priority Usage Note*

Revision(s) Affected: 1.0

Details: The EMAC priority is set in the EMAC / UPP Priority Allocation (EMAC_UPP_PRI_ALLOC) Register. This value should not be changed while the EMAC is reading or writing data within the SoC. This value can only safely be changed before the EMAC is active or after it is shut down. It is recommended to set the priority desired at system startup time and not change it. Should changing the priority be necessary after EMAC is active, the EMAC should be shut down, then the priority changed, then EMAC re-started.

Usage Note 13 *Revised PLL Programming Sequence Usage Note*

Revision(s) Affected: 1.0

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

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

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

Usage Note 14 *Core Wake Up on RESET Usage Note*

Revision(s) Affected: 1.0

Details:

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

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

Resets using POR and RESETFULL does not exhibit this behavior.

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

Workaround:

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

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

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

Usage Note 15 *BSDL Testing Support Usage Note*

Revision(s) Affected: 1.0

Details:

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

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

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

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

Workaround:

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

There are two types of BSDL tools currently available:

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

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

Single hardware/software tool

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

Separate hardware/software tool

1. The SoC/DSP must be correctly powered with clocks applied
2. Using the emulation/debugger, enable all the respective SerDes peripherals
3. Using the emulation/debugger, change the RXTERM value from “000” to “001”
4. Disconnect the emulation/debugger hardware and software
5. Connect the boundary scan hardware and software
6. Run your respective 1149.6 boundary scan tests

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

Table 7. Register Information for the Peripheral

Peripheral	Register	Bit	Address
HyperLink[1] (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[1] (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

(1) C6655 and C6657 only

Usage Note 16 *Initial Voltage Level Setting of CVDD Rail Power Supplies Usage Note*

Revision(s) Affected: 1.0

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

SmartReflex voltage scheme as defined by the device specific data manual and Hardware Design Guide for KeyStone I Devices is absolutely required.

Usage Note 17 *Performance Degradation for Asynchronous Accesses Caused by An Unused Feature Enabled in EMIF 16 Usage Note*

Revision(s) Affected: 1.0

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

Revision(s) Affected: 1.0

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

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

The following are examples of how read and write transactions behave when the class of service feature is enabled. The class of service for reads and writes are handled independently, i.e., if the counter for a read with a higher class of service expires, all writes irrespective of their class of service will not be blocked. However, all other reads with a lower class of service 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. However, all other writes with a lower class of service 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 is 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). 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.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from B Revision (July 2015) to C Revision	Page
• Global change: Added C6652 device	4

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