66AK2H06/12

Communications Infrastructure KeyStone SOC

Silicon Revision 1.0, 1.1

Silicon Errata

Literature Number: SPRZ402
June 2013

Release History

Release	Date	Description/Comments
SPRZ402	June 2013	Initial Release

Contents

List of Figures

List of Tables

66AK2H06/12 Communications Infrastructure KeyStone SOC Silicon Revision 1.0, 1.1

Introduction

This document describes the silicon updates to the functional specifications for the 66AK2H06/12 fixed-/floating-point digital signal processor. See the device-specific data manual for more information.

Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all devices and support tools. Each family member has one of two prefixes: X or [blank]. These prefixes represent evolutionary stages of product development from engineering prototypes through fully qualified production devices/tools.

Device development evolutionary flow:

- **X:** Experimental device that is not necessarily representative of the final device's electrical specifications
- **[Blank]:** Fully qualified production device

Support tool development evolutionary flow:

- **X:** Development-support product that has not yet completed Texas Instruments internal qualification testing.
- **[Blank]:** Fully qualified development-support product

Experimental (X) and fully qualified [Blank] devices and development-support tools are shipped with the following disclaimer:

Developmental product is intended for internal evaluation purposes.

Fully qualified and production devices and 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 experimental devices (X) 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.

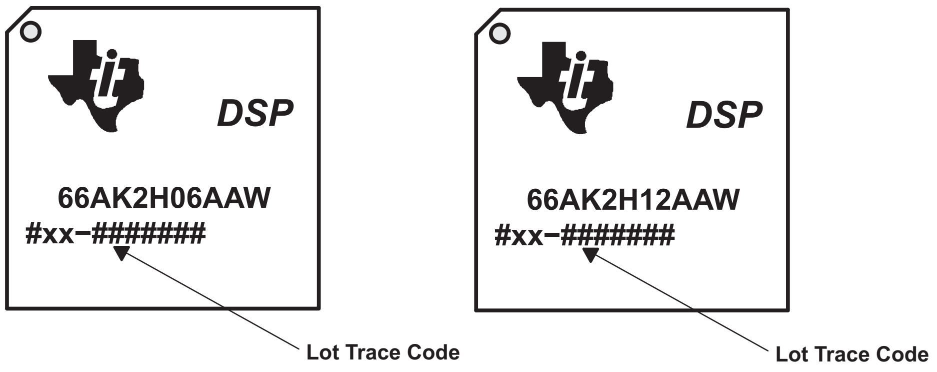
TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, AAW), the temperature range (for example, blank is the default case temperature range), and the device speed range, in Megahertz (for example, blank is 1000 MHz [1 GHz]).

For device part numbers and further ordering information for 66AK2H06/12in the AAW package type, see the TI website www.ti.com or contact your TI sales representative.

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 AAW package is shown in Figure 1. Figure 1 also shows an example of 66AK2H06/12 package symbolization.

Figure 1 Lot Trace Code Example for 66AK2H06/12 (AAW 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 66AK2H06/12 devices.

Table 1 Lot Trace Codes

Lot Trace Code (xx)	Silicon Revision	Comments
10	1.0	Initial silicon revision
11	1.1	Silicon revision 1.1

The 66AK2H06/12 device contains multiple read-only register fields that report revision values. The JTAG ID (JTAGID) and C66x CorePac Revision ID registers allow the customer to read the current device and CPU level revision of the 66AK2H06/12.

The JTAG ID register (JTAGID) is a read-only register that identifies to the customer the JTAG/Device ID.

The C66x CorePac Revision ID register 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 part-specific data manual.

Table 2 shows the contents of the C66x CorePac REVID Register, and the JTAGID register for each silicon revision of the 66AK2H06/12 device.

Table 2 Silicon Revision Variables

Silicon Revision	C66x CorePac REVID Register (address location: 0x0181_2000)	66AK2H06/12 JTAGID Register (address location: 0x0262_0018)
1.0	0x0009_ 0000	0x0b98_102f
1.1	0x0009_0002	0x1b98_102f
End of Table 2		

More details on the JTAG ID and CorePac Revision ID Registers can be found in the device-specific data manual.

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 1.0 Updates

Silicon Update Advisory	See[1]	Applies To Silicon Revision	
		1.0	1.1
Boot ROM Missed EMIF PHY Configuration Registers	KeyStoneII.BTS_errata_advisory.1	X	
Execution in Parallel (XIP) from NOR Flash on ARM Does Not Work	KeyStoneII.BTS_errata_advisory.2	X	X
ARM L2 Latency is Not Set Correctly	KeyStoneII.BTS_errata_advisory.3	X	
SRIO Boot Mode Packet DMA Cleanup Missing	KeyStoneII.BTS_errata_advisory.4	X	
Sync-Ethernet Push Event IO Control Tie-Off Incorrect	KeyStoneII.BTS_errata_advisory.5	X	
ACE Snoop Collision with Internal ARM A15 Snoop May Deadlock Core	KeyStoneII.BTS_errata_advisory.7	X	
EMIF Boot (NOR flash) Mode Not Working When ARM Master Boot	KeyStoneII.BTS_errata_advisory.8	X	
1.4-Ghz ARM is Not Supported	KeyStoneII.BTS_errata_advisory.9	X	
DDR3A Lanes 5 and DDR3B Lane 8 Have PHY to PUB DFI Hold Time Failures	KeyStoneII.BTS_errata_advisory.10	X	
Simultaneous Write to SPM by Fetcher and VBUS Causes Fetch Write to be Lost	KeyStoneII.BTS_errata_advisory.11	X	
ARM WriteBack-NoAllocate Hang	KeyStoneII.BTS_errata_advisory.12	X	X
SPI Boot Size Limitation, C66x Master Boot	KeyStoneII.BTS_errata_usagenote.1	X	
ARM Core Hangs if Timer is Accessed While ARM Core is in Wait-In-Reset State	KeyStoneII.BTS_errata_usagenote.2	X	X
Debug System (DebugSS) Trace Buffer (TBR) EDMA via System Port is not Functional	KeyStoneII.BTS_errata_usagenote.3	X	X
ARM Boot Can Fail When Interrupt Enabled	KeyStoneII.BTS_errata_usagenote.4	X	X
Boot I^{2}C Frequency Incorrect	KeyStoneII.BTS_errata_usagenote.5	X	X
Access to DDR3 Without Configuring PHY Properly Can Cause Hang	KeyStoneII.BTS_errata_usagenote.6	X	X
C66x CorePac and ARM CorePac AVS Rails	KeyStoneII.BTS_errata_usagenote.7	X	X
End of Table 3			

1. Not all KeyStone II errata apply to all KeyStone II parts. Therefore, numbering gaps in the errata list are normal.

KeyStoneII.BTS_errata_advisory.1

Boot ROM Missed EMIF PHY Configuration Registers

Revision(s) Affected	1.0
Details	Boot ROM missed initializing some EMIF PHY Configuration registers.

On PG1.0 Boot ROM, the built- in DDR3 PHY configuration for the DWCPUB PHY does not include values needed for PHY Config. The following values are needed to support configuring the PHY registers:

DDR3A_PLLCR	offset 0x00000018
DDR3A_DSGCR	offset 0x00000040
DDR3A_ZQ0CR1	offset 0x00000184
DDR3A_ZQ1CR1	offset 0x00000194
DDR3A_ZQ2CR1	offset 0x000001A4
DDR3A_ZQ3CR1	offset 0x000001B4

Workaround If the non-default values are required in these registers, a two stage boot must be used.

KeyStoneII.BTS_errata_advisory.2

Execution in Parallel (XIP) from NOR Flash on ARM Does Not Work

Revision(s) Affected 1.0, 1.1

Details ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF.

On these devices, ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF. Consequence of this is that the parallel NOR flash boot from ARM does not work.

Direct execution from a parallel NOR flash does not work as ARM always generates 64-byte cacheline wrap mode accesses to EMIF. This is irrespective of marking this memory region as Device or Strongly Ordered as confirmed by ARM. ASYNC EMIF does not support 64-byte cacheline wrap accesses and therefore generates a bus error on receiving it. This causes an abort to happen in ARM.

Workaround None. Use C66x boot master if NOR flash boot is required, or boot NOR flash over SPI.

KeyStoneII.BTS_errata_advisory.3

ARM L2 Latency is Not Set Correctly

Revision(s) Affected	1.0
Details	ARM L2 Latency is not set correctly in the Boot ROM In PG1.0 devices, the boot ROM did not change the default value of L2 latency in the L2 Control register. The default value is 0 (which means two cycles). The required value is 3 (which means 4 cycles).
Workaround	Either use one of the C66x boot modes or the I^2C or SPI boot mode in ARM master boot mode which does not setup the ARM PLL. Once the boot is done, the PLL can be changed and the latency setup can be programmed accordingly.

KeyStoneII.BTS_errata_advisory.4

SRIO Boot Mode Packet DMA Cleanup Missing

Revision(s) Affected	1.0
Details	The issue is within the Boot ROM. In SRIO Boot Mode, the Packet DMA is not torn down when the SRIO boot is complete. On PG1.0 devices, in SRIO boot mode, the boot ROM did not perform the Packet DMA cleanup (the Packet DMA is not torn down) when the boot is complete. All queues are emptied and the QM is torn down, but the Packet DMA is left up.
Workaround	Before reconfiguration, the existing configuration should be torn down.

KeyStoneII.BTS_errata_advisory.5

Sync-Ethernet Push Event IO Control Tie-Off Incorrect

Revision(s) Affected 1.0

Details The tie-off for IO control for the two Sync-Ethernet Push Events is connected incorrectly in the design.

On PG1.0 devices, the tie-off for IO control for the two Sync-Ethernet Push Event (TSPUSHEVT0 (PPS Push Event from GPS for IEEE1588) and TSPUSHEVT1 (Push Event from BCN for IEEE1588)) IOs is connected incorrectly in the design. Consequence of this is that the TSPUSHEVT0 and TSPUSHEVT1 pads are being forced to outputs when not in reset, and thus **cannot** be selected as inputs. These pads are functionally used as inputs.

Workaround None.

KeyStoneII.BTS_errata_advisory.7

ACE Snoop Collision with Internal ARM A15 Snoop May Deadlock Core

Revision(s) Affected 1.0

Details A combination of Non-cacheable or streaming writes, a DSB instruction, and a snoop can cause a core to deadlock

If a certain combination of snoops, DSB instruction execution, and stores occurs with a particular timing, a Cortex-A15 MPCore CPU can deadlock. The deadlocked CPU will be unable to complete an eviction until earlier writes complete, and those writes will not be able to complete until that eviction completes.

Conditions

1. One core (coreX) is executing store instructions that are not allocating to the L1 cache. These will be a combination of:
   - Writes to Non-cacheable, Strongly-ordered, or Device memory locations.
   - Write-streaming full line cacheable writes.
2. These stores back up in the memory system and allocate 12 entries of the L2 Write Request Queue
3. CoreX begins the eviction of a cache line (A) that is being replaced by a returning cache line fill
4. Another core (coreY) executes a DSB instruction that generates a TLB synchronization request to coreX
5. A memory request occurs (from coreZ, an ACP request, an external snoop, or a translation table walk request) that
   - hits the cache line (A) being evicted from coreX, OR
   - hits another line in the L1 data cache of coreX that is in uniqueClean or uniqueDirty state.

Implications If the above scenario occurs with a particular timing, it is possible that the eviction from coreX cannot complete until another L2 write buffer credit is returned, and no such credit can be returned until the eviction completes. In this situation, the processor deadlocks.

Workaround There is no known workaround.

KeyStoneII.BTS_errata_advisory.8

EMIF Boot (NOR flash) Mode Not Working When ARM Master Boot

Revision(s) Affected	1.0

Details

ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF.

ARM cannot perform direct execution from a parallel NOR flash connected via ASYNC EMIF. Consequence of this is that the parallel NOR flash boot from ARM does not work.

Direct execution from a parallel NOR flash does not work as ARM always generates 64byte cacheline wrap mode accesses to EMIF. This is irrespective of marking this memory region as Device or Strongly Ordered as confirmed by ARM. ASYNC EMIF does not support 64-byte cacheline wrap accesses and therefore generates a bus error on receiving it. This causes an abort to happen in ARM.

Workaround

Use C66x boot master if NOR flash boot is required.

KeyStoneII.BTS_errata_advisory.9

1.4 GHz ARM is Not Supported

Revision(s) Affected	1.0
Details	1.4 GHz ARM is not supported silicon revision listed in the table. The ARM cannot fully function at 1.4 GHz with core power supplies of 900 mV nominal voltage.
Workaround	Run the ARM at a 1.2 GHz.

KeyStoneII.BTS_errata_advisory.10

DDR3A Lanes 5 and DDR3B Lane 8 Have PHY to PUB DFI Hold Time Failures

Revision(s) Affected 1.0

Details

DDR3A Bit Errors in Data Byte Lane 5

It has been observed that operating DDR3A in 64-bit mode results in occasional read errors. The problem is observed across all frequencies of operation.

The read errors occur across byte lane 5 of the DDR3A interface due to hold time violations on the DFI interface between the PHY Utility Block and SDRAM PHY on this specific byte lane. Since the hold violations are only on byte lane 5, this issue impacts only 64-bit DDR3A operation and not 32-bit or 16-bit DDR3A operations.

Table 4 DDR3A

72 bit with ECC	Nonfunctional
64 bit without ECC	Nonfunctional
36 bit with ECC	Functional
32 bit without ECC	Functional
16 bit without ECC	Functional

Workaround The only stable work around available at this time is to operate the DDR3A interface in 32-bit or 16-bit mode. These configurations would ensure the faulty byte lane is never activated.

Details

DDR3B Bit Errors in ECC Byte Lane

It has been observed that error correction is currently nonfunctional on the DDR3B interface. Turning on ECC during DDR3B operation results in occasional read errors as a result of miscalculated ECC values.

Table 5 DDR3B

72 bit with ECC	Nonfunctional
64 bit without ECC	Functional
36 bit with ECC	Nonfunctional
32 bit without ECC	Functional
16 bit without ECC	Functional

Workaround The only stable work around available at this time is to operate DDR3B interface in non ECC mode. This configuration would ensure the faulty byte lane is never activated.

KeyStoneII.BTS_errata_advisory.11

Simultaneous Write to SPM by Fetcher and VBUS Causes Fetch Write to be Lost

Revision(s) Affected 1.0

Details 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's write of access slot/frame/sub-frame header fields to the spreader's SPM and software's write to any spreader's SPM word.

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: spreader parameter memory (SPM) and spreader request memory (SRM).

- SPM supports configuration of 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 changing from 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, HSPDSCH, 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 the 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:

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*	DTX, nextBlockPtr
F-DPCH	Slot mask, DTX
AICH/E-AICH	DTX
HS-PDSCH	Message handler RLS Id, Channelization code, Modulation, DTX, diversity mode, nextBlockPtr
HS-SCCH	Diversity mode, DTX
E-HICH/RGCH	DTX
E-AGCH	DTX

* 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 silicon bug.

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 silicon bug relates to writes to SPM by the fetch process and writes to SPM by software.

TAC silicon bug: 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).

Bug consequences: If the current TAC bug 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 uses 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 silicon bug 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 a missing write to SPM by the header fetch process is greater in an HSPA scenario compared to a non-HSPA scenario. This is because in case of an HSPA scenario, header fetches are more frequent (sub-frame basis) than in the 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 None

KeyStoneII.BTS_errata_advisory.12

ARM WriteBack-NoAllocate Hang

Revision(s) Affected 1.0, 1.1

Details An eviction from L1 data cache might stall indefinitely in the L2 write buffer preventing a snoop from completing.

If a very specific timing occurs with the following behavior, an ARM A15 core lockup is possible.

1. A full line cacheable write from the L1 write streaming logic.
2. A store to a page marked “writeBack NoAllocate” in the page tables.
3. A store to a page marked “writeBack” with any allocation hints when the cache is disabled and the MMU is enabled.

Workaround

- Do not use the write-back no-allocate memory type.
- Do not issue write-back cacheable stores at any time when the cache is disabled SCTLR.C=0) and the MMU is enabled (SCTLR.M=1). Because it is implementation defined whether cacheable stores update the cache when the cache is disabled, it is not expected that any portable code will execute cacheable stores when the cache is disabled.

KeyStoneII.BTS_errata_usagenote.1

SPI Boot Size Limitation, C66x Master Boot

Revision(s) Affected	1.0
Details	The issue is within the Boot ROM. In C66x Master, SPI Boot Mode, the SPI Boot Size is limited by the Boot ROM to a single block. In SPI Boot Mode, the SPI Boot Size is limited by the Boot ROM to a single block. The max block read size is 8K bytes for C66x master boot.
Workaround	A workaround is available that contains the fix in that first block read. The workaround can simply be prepended to customer boot images when placed on the flash. To obtain the workaround contact the TI E2E Forum or your local TI representative.

KeyStoneII.BTS_errata_usagenote.2

ARM Core Hangs if Timer is Accessed While ARM Core is in Wait-In-Reset State

Revision(s) Affected 1.0, 1.1

Details While the ARM CorePac is held in Wait-In-Reset (WIR) state by the DEBUGSS, any access to the timer memory mapped registers by the Code Composer Studio will cause the ARM CorePac to hang.

Access to Timers associated with the ARM CorePac will hang when the ARM cores are held in reset by the DEBUGSS (a $\overline{\text{LRESET}}$ applied to the ARM CorePac). This scenario would happen typically in CCS code development environment when bringing up the CCS with all cores held in wait-in-reset. If there is a memory window that points to the Timer configuration space, the ARM CorePac will be hung.

Note—For a description of Wait-In-Reset, see the TI Embedded Processors Wiki.

Workaround To avoid this issue, use the GEL command, GEL_MapReset() that masks (unmap) the memory map configuration (including the Timer configuration space. Note that you cannot access those Timer registers from the CCS memory window.) To view other memory spaces, use the GEL_MapAddStr () to add them in.

KeyStoneII.BTS_errata_usagenote.3

Debug System (DebugSS) Trace Buffer EDMA via System Port Not Functional

Revision(s) Affected 1.0, 1.1

Details Debug System (DebugSS) Trace Buffer (TBR) EDMA via System Port is not functional.

The TBR in the DebugSS is used to capture the output of CP-Tracers and System Trace Module (STM). The trace data can be accessed by debugger or application from either the configuration port or the system port. The system port is ideal for another master such as EDMA to read the trace data to a larger memory region. This allows more trace data to be captured into on-chip memory beyond the limited size of trace buffer.

The DebugSS TBR clock is integrated incorrectly in the design. Consequence of this is preventing the usage of EDMA to drain the contents of TBR.

Workaround None for the EDMA use case from the system port. The TBR can still be accessed via the configuration port.

KeyStoneII.BTS_errata_usagenote.4

ARM Boot Can Fail When Interrupt Enabled

Revision(s) Affected 1.0, 1.1

Details ARM boot can fail when an interrupt is pending when interrupts are enabled.

From the ARM Boot ROM code, the "Enable interrupts" code below in the ROM is not implementing correctly.

```
********************************************************************************
* Enable interrupts
********************************************************************************

.def _chipEnableInts

_chipEnableInts:

cpsie i
bx  lr
```

If an interrupt is pending when the cpsie instruction executes the interrupt is immediately taken. The interrupt code executes normally, and even returns with the correct mode. However the next instruction, bx lr, does not execute. The code simply continues into the next instruction, which happens to be a data value. This results in an invalid instruction exception.

In a case of the C66x boot master and a $\overline{\text{RESET}}$ is applied, the $\overline{\text{RESET}}$ resets the ARM PLL causing the ARM boot code to execute very slowly. The system PLL was reset isolated and executed very quickly. The C66x boot code was loaded, and this code poked the IPC interrupt to wake up the ARM, but the slow ARM was still setting up translation tables. If there is an interrupt pending, the interrupt is taken when the interrupts were enabled.

This could also happen when the ARM is the boot master in the PCIe and HyperLink boot modes. If the remote end generates an interrupt immediately after link detection it is possible that the ARM code has not yet reached the cpsie instruction. But in both of these modes, the ARM PLL will be enabled and the race is very short.

The consequence of this is that the ARM generates an invalid instruction exception.

Workaround Delay interrupts to the ARM for a few ms. This applies mostly to ARM core 0. Secondary ARM cores are usually woken up without an interrupt, however the same code can execute if the secondary ARM core wakes up and does not see a branch address, and in which case it enables interrupts and idles.

KeyStoneII.BTS_errata_usagenote.5

Boot I^{2}C Frequency Incorrect

Revision(s) Affected 1.0, 1.1

Details The issue is within the Boot ROM. Initial boot I^{2}C frequency incorrect on $\overline{\text{RESET}}$ reset.

For I^{2}C boot, the code to determine the device frequency assumes that the PLL is in bypass. This is true for power on reset, but for $\overline{\text{RESET}}$, with the PLL reset isolated this is incorrect. The code should check to see if the PLL is enabled and if it is, it should return the e-fuse device frequency.

On a normal $\overline{\text{POR}}$ or $\overline{\text{RESETFULL}}$, boot with I^{2}C as the boot master, the boot code will correctly assume the system is running at 312 MHz (max possible) and scale the I^{2}C clock to run at 20 KHz. So the actual frequency will scale based on the actual reference clock. After boot execute a $\overline{\text{RESET}}$, the initial frequency will be much higher, running faster by a multiple equal to the actual effective PLL multiplier value.

For example if the actual reference clock is 50 MHz and the device frequency is e-fused for 1400 MHz, the initial I^{2}C will read using a data clock of 20 * 50 / 312 = 3.2 KHz. After a $\overline{\text{RESET}}$ reset, with the PLL reset isolated, the initial read will be at 20 * 1400 / 312 = 89 KHz. This will work with almost all I^{2}C devices that are compliant to the 100 KHz bus standard specified in the original I^{2}C specification. This represents the worst case for these devices.

Workaround None.

KeyStoneII.BTS_errata_usagenote.6

Access to DDR3 Without Configuring PHY Properly Can Cause Hang

Revision(s) Affected 1.0, 1.1

Details If the DDR3 is not configured properly before the C66x and/or ARM CorePac issuing an access to DRR3, the device could lock up.

If the DDR3 PHY Utility Block (PUB), DDR3 PHY and the EMIF Controller are not configured or improperly configured, any access to the DRR3 memory space is issued by the C66x and/or ARM CorePac, including opening a memory window view from the Code Composer Studio (CCS) that is pointed to the DDR3 memory space, the device could lock up.

Workaround It is recommended that before issuing an access to the DDR3, the device must properly initialize the DDR3 PUB, DDR3 PHY, and EMIF controller.

Refer to the KeyStone II DDR3 Programming Sequence documented in the KeyStone II DDR3 User Guide (SPRUGV8) or the KeyStone II DDR3 Initialization Sequence document (SPRABL2).

KeyStoneII.BTS_errata_usagenote.7

C66x CorePac and ARM CorePac AVS Rails

Revision(s) Affected	1.0, 1.1
Details	Separating the AVS rails for the C66x CorePac and the ARM CorePac exhibits a transient voltage on the ARM CorePac. There are two different AVS rails specified on these devices, one for the C66x CorePacs, CVDD, and one for the ARM CorePac, CVDDT. On PG 1.0 silicon, an issue with transient voltage of the ARM CorePac is observed by TI test engineering if these two rails are supplied separately.
Workaround	On PG 1.0 silicon, TI recommends that both AVS rails must be tied together. On PG 1.1 silicon, these two rails are tied together internally.

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		