TMS320DM6467T
Digital Media System-on-Chip (DMSoC)
Silicon Revision 3.0

Silicon Errata

Literature Number: SPRZ307A
January 2010–Revised July 2010

List of Figures

List of Tables

TMS320DM6467T DMSoC Silicon Revision 3.0

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320DM6467T DMSoC devices (i.e., TMS320DM6467T). For more detailed information on these devices, see the device-specific data manual:

- *TMS320DM6467T Digital Media System-on-Chip* data manual (Literature Number SPRS605)

Throughout this document, unless otherwise specified, TMS320DM646x, DM646x, TMS320DM646xT, and DM646xT, refer to the TMS320DM6467T device. For additional peripheral information, see the latest version of the *TMS320DM646x DMSoC Peripherals Overview* Reference Guide (Literature Number SPRUEQ0).

The advisory numbers in this document are not sequential. Some advisory numbers have been moved to the next revision and others have been removed and documented in the user's guide. When items are moved or deleted, the remaining numbers remain the same and are not resequenced.

1.1 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.,TMX320DM6467TZUT). Texas Instruments recommends two of three 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.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, ZUT), the temperature range (for example, "Blank" is the commercial temperature range), and the device speed range in megahertz (for example, "1" is the default [1000-MHz DSP, 500-MHz ARM9]).

1.2 Revision Identification

Figure 1 provides an example(s) of the TMS320DM6467T device markings. The device revision can be determined by the symbols marked on the top of the package.

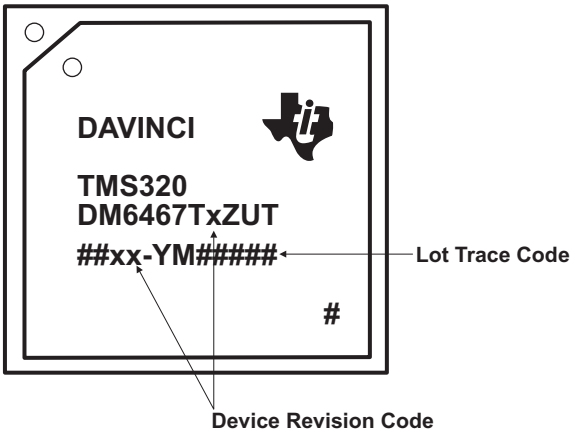

Figure 1. Example, Device Revision Codes for TMS320DM6467T (ZUT Package)

Silicon Revision and date of manufacturing are identified by device-revision code and lot trace code markings on the package. The codes are of the format ##xx-YM##### where "xx" denotes the Die PG Code. If xx is "30" then the silicon is revision 3.0 and the device part number (x) has *no letter* (blank). Table 1 lists the information associated with each silicon revision. "YM" in the lot trace code denotes the manufacturing date in Hex, where "Y" denotes the year and "M" denotes the month. For example, if "YM" is "9B", then the part is manufactured in 2009 November.

Table 1. TMS320DM6467T Device Revision Codes

DEVICE REVISION CODE (xx) and (x)		SILICON REVISION	PART NUMBERS/COMMENTS
DIE PG CODE (xx)	DEVICE PART NUMBER (x)		
30	(blank)	3.0	TM**S**320DM6467TZUT

2 Silicon Revision 3.0 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 3.0 of the TMS320DM6467T device.

2.1 Usage Notes for Silicon Revision 3.0

Usage notes highlight and describe particular situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness. These usage notes will be incorporated into future documentation updates for the device (such as the device-specific data sheet), and the behaviors they describe will not be altered in future silicon revisions.

2.1.1 PCI Cannot Burst More Than 32 Bytes When Used in Master Mode

On DM646x silicon revision 3.0 and earlier, the device PCI can operate as a PCI master and slave. As a slave, the device PCI responds to accesses initiated by an off-chip PCI master. As a master, the device PCI initiates transfers on the PCI bus. Usually, for memory read and write transfers, another DM646x master such as the EDMA is configured to move data to/from the PCI.

As a PCI master, the device PCI is capable of bursting only a maximum of up to 32 bytes. In other words, for memory transfers larger than 32 bytes, the device PCI initiates a transfer, transfers 32 bytes, stops the transfer, and then repeats. As a PCI slave, external PCI masters can burst an infinite amount of data to the DM646x PCI. Note that the PCI may insert wait states or generate a target retry if it cannot meet the latency requirement set forth by the PCI system. For example, a PCI access to DDR2 memory may stall due to other master accesses or because of a scheduled DDR2 memory refresh. In this case, the PCI generates a target retry until the DDR2 memory controller is ready to service the PCI request.

Because of this limitation, the DM646x PCI throughput will be lower in master mode than in slave mode. To avoid low throughput performance, external PCI masters should be used to move data to/from the DM646x PCI whenever possible.

2.1.2 VPIF Resynchronization After Disconnecting External Video Source

The VPIF module does not support resynchronization after disconnecting an external video source and the VP_ERRINT is not designed to flag a loss-of-sync event. In order to capture correctly, the VPIF has to be reset each time a source is reconnected.

One approach to reliably reset the VPIF is to have the DM646x device periodically poll the status register of the ADC which interfaces to the VPIF via the I2C bus. For example, on the DM646x EVM, the DM646x CPU can poll the TVP7002 or TVP5147 Sync Detect Status register via the I2C interface. When an invalid sync status is detected, the DM646x CPU disables the VPIF and continues polling periodically. When a valid sync status is detected, the VPIF is reset, reenabled, and reinitialized to capture.

Furthermore, if the ADC can send a toggle signal during a loss-of-sync, this signal can be connected directly to the GPIO of the DM646x to trigger a GPIO interrupt. Upon receiving the interrupt, the CPU can then reset the VPIF. This avoids the CPU from constantly reading the I2C bus.

2.2 Silicon Revision 3.0 Known Design Exceptions to Functional Specifications

Table 2. Silicon Revision 3.0 Advisory List

Advisory 3.0.2 *Pin Muxing Functions: EMIFA-to-PCI Pin Mux Switching Resets Device*

Revision(s) Affected 3.0 and earlier

Details On the DM646x device, the EMIFA address 22 (EM_A[22]) and the $\overline{\text{PCI\_RST}}$ functions share the same pin (C10). The $\overline{\text{PCI\_RST}}$ function is internally connected to the chip reset. If the EM_A[22] function is low before the EMIFA-to-PCI pin mux switch happens (e.g., after any EMIFA boot mode execution), the device resets.

Workaround(s) Before switching the pin mux from EMIFA to PCI, drive EM_A[22] high by performing a dummy EMIFA read.

Advisory 3.0.3 *DSP SDMA/IDMA: Unexpected Stalling and Potential Deadlock Condition When DSP L2 Memory Ports Used as RAM When L2 Memory Configured as Non-cache (RAM)*

Revision(s) Affected 3.0 and earlier

Details

Note: This advisory is ***not*** applicable if the DSP L2 memory is configured as 100% cache, ***or*** if L2 RAM and HDVICP0/1 RAM/Buffers (through the DSP SDMA port) *are not* accessed by IDMA or SDMA during run-time.

Note: Masters can access HDVICP0/1 RAM/Buffers either through HDVICP EDMA ports (that are connected to SCR1 through BR9 to BR14) or through HDVICP ports that are connected to DSP SDMA port via BR25, BR26, and SCR8 (as shown in the TMS320DM646x System Interconnect Block Diagram in *TMS320DM6467 SoC Architecture and Throughput Overview* Application Report (literature number SPRAAW4)).

The C64x+ Megamodule has a Master Direct Memory Access (MDMA) bus interface and a Slave Direct Memory Access (SDMA) bus interface. The MDMA interface provides DSP access to resources outside the C64x+ Megamodule (i.e., DDR2, EMIFA, HDVICP0/1 EDMA ports, PCI, ARM TCM, and VLYNQ remote memory). The MDMA interface is typically used for CPU/cache accesses to memory beyond the Level 2 (e.g., L2 RAM/Cache, HDVICP0/1 RAM/Buffers through DSP SDMA port) memory level. These accesses include cache line allocates, write-backs, and non-cacheable loads and stores to/from system memories. The SDMA interface allows other master peripherals, including ARM (data port), EDMA transfer controllers, HPI, USB2.0, ATA, EMAC, PCI, and VLYNQ, to access Level 1 Data (L1D), Level 1 Program (L1P), L2 DSP memories, and HDVICP0/1 RAM/Buffers (through DSP SDMA port). The DSP Internal DMA (IDMA) is a C64x+ Megamodule DMA engine used to move data between internal DSP memories (L1,L2) and/or the DSP peripheral configuration bus. The IDMA engine shares resources with the SDMA interface.

The C64x+ Megamodule has an L1D cache and L2 cache both implementing write-back data caches–it holds updated values for external memory as long as possible. It writes these updated values, called "victims", to external memory when it needs to make room for new data *or* when requested to do so by the application, or when a load is performed from a non-cacheable memory for which there is a set match in the cache (i.e., the non-cacheable line would replace a dirty line if cached). The L1D sends its victims to L2. The caching architecture has pipelining, meaning multiple requests could be pending between L1, L2, and MDMA. For more details on the C64x+ Megamodule and its MDMA and SDMA ports, see the *TMS320C64x+ Megamodule* Reference Guide (literature number SPRU871).

Ideally, the MDMA (dashed-dotted line in Figure 2) and SDMA/IDMA paths (dashed lines in Figure 2) operate independently with minimal interference. Normally MDMA accesses may stall for extended periods of time due to expected system level delays (e.g., bandwidth limitations, DDR2 memory refreshes). However, when using L2 as RAM, SDMA and IDMA accesses to L2/L1/HDVICP RAM/Buffers (through DSP SDMA port) may experience unexpected stalling in addition to the normal stalls seen by the MDMA interface. For latency-sensitive traffic, the SDMA stall can result in missing real-time deadlines. In a more severe case, the SDMA stall can produce a deadlock condition in the SoC. An IDMA stall cannot produce a deadlock condition.

Note: SDMA/IDMA accesses to L1P/D ***will not*** experience an unexpected stall if there are no SDMA/IDMA accesses to Level 2 memory ports (L2 RAM and HDVICP0/1 RAM/Buffers through DSP SDMA port). Unexpected SDMA/IDMA stalls to L1 happen *only* when they are pipelined behind Level 2 memory port accesses. Additionally, the deadlock scenario will be avoided if there are no SDMA accesses to the Level 2 memory ports.

Note: Figure 2 is provided for illustrative purposes and is incomplete for simplification. The IDMA/SDMA (dashed-lines) path could also go to L1D/L1P memories, and IDMA

can go to DSP CFG peripherals. MDMA transactions can originate also from L1P or L1D through the L2 controller or directly from the DSP.

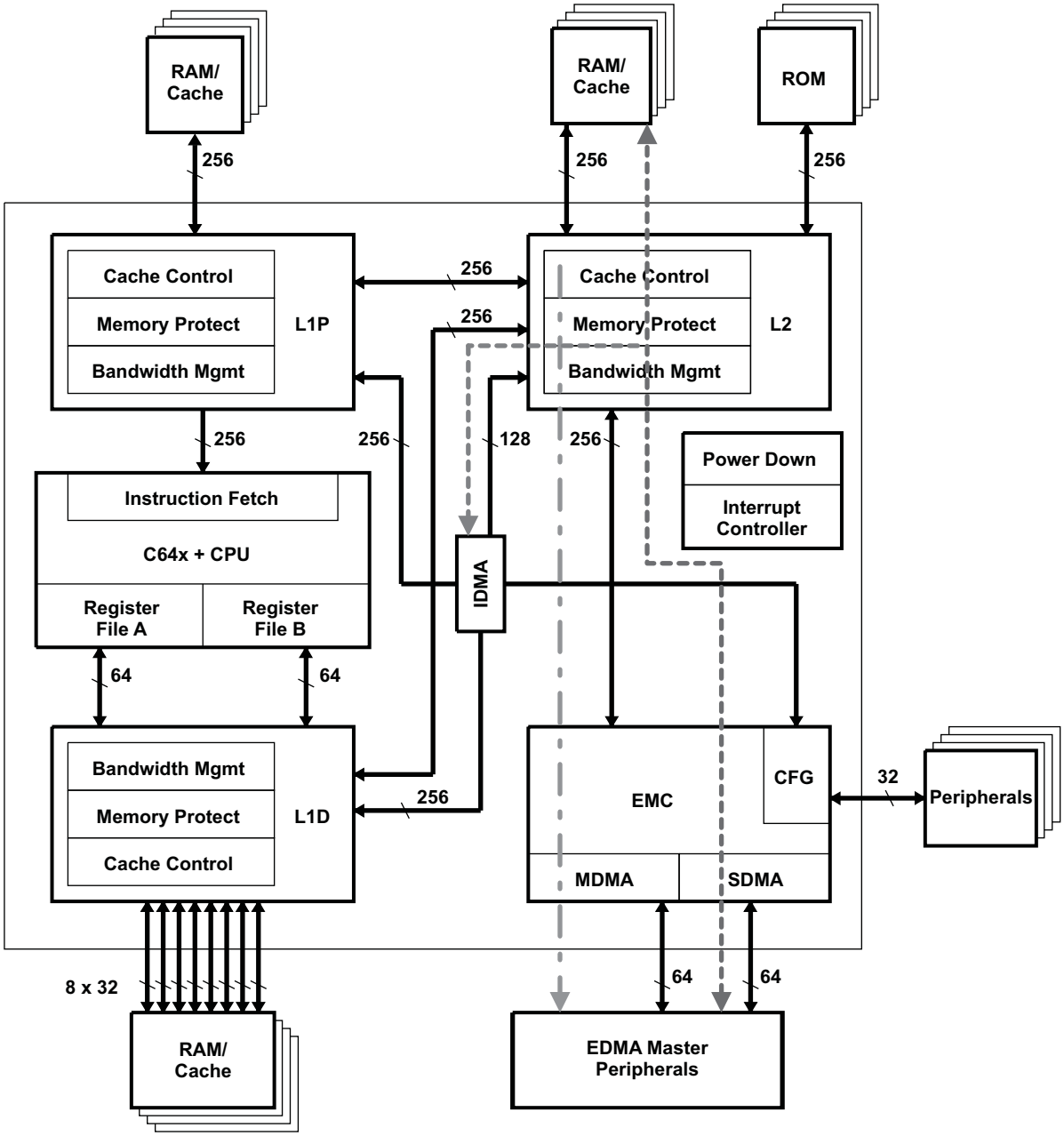

Figure 2. IDMA, SDMA, MDMA Paths

NOTES:

1. The dashed lines in Figure 2 represent SDMA/IDMA paths.
2. The dashed-dotted line in Figure 2 represents the MDMA path.

SDMA/IDMA stalls may occur during the following scenarios. Each of these scenarios describes expected normal DSP functionality, but the SDMA/IDMA access potentially exhibits additional unexpected stalling.

1. Bursts of writes to non-cacheable MDMA space (i.e., DDR2, EMIFA, ARM TCM, HDVICP0/1 EDMA ports, PCI, and VLYNQ remote). The DSP buffers up to 4 non-cacheable writes. When this buffer fills, SDMA/IDMA is blocked until the buffer is no longer full. Therefore, bursts of non-cacheable writes longer than three writes can stall SDMA/IDMA traffic.
2. Various combinations of L1 and L2 cache activity:
   (a) L1D read miss generating victim traffic to L2 (Cache or SRAM) or external memory. The SDMA/IDMA may be stalled while servicing the read miss and the victim. If the read miss also misses L2 cache, the SDMA/IDMA may be stalled until data is fetched from external memory to service the read miss. If the read access is to non-cacheable memory there will still potentially be an L1D victim generated even though the read data will not replace the line in the L1D cache.
   (b) L1D read request missing L2 (going external) while another L1D request is pending. The SDMA/IDMA may be stalled until the external memory access is complete.
   (c) L2 victim traffic to external memory during any pending L1D request. SDMA/IDMA may be stalled until external memory access and the pending L1D request is complete.

The duration of the IDMA/SDMA stalls depends on the quantity/characteristics of the L1/L2 cache and MDMA traffic in the system. In cases 2a., 2b., and 2c., stalling may or may not occur depending on the state of the cache request pipelines and the traffic target locations. These stalling mechanisms may also interact in various ways, causing longer stalls. Therefore, it is difficult to predict if and how long stalling will occur.

IDMA/SDMA stalling and any system impact is most likely in systems with excessive context switching, L1/L2 cache miss/victim traffic, and heavily loaded EMIFA.

Use the following procedure to determine if SDMA/IDMA stalling is the cause of real-time deadline misses for existing applications. Situations where real-time deadlines may be missed include loss of McASP samples and low peripheral throughput.

1. Determine if the transfer missing the real-time deadline is accessing L2, HDVICP0/1 RAM/Buffers through DSP SDMA port or L1D memory. If not, then SDMA/IDMA stalling is ***not*** the source of the real-time deadline miss.
2. Identify all SDMA transfers to/from Level 2 memory ports (L2 RAM or HDVICP0/1 RAM/buffers through DSP SDMA port) [e.g., EDMA transfer to/from L2 from/to an McASP, PCI, *or* HPI block transfer to/from L2, EDMA transfer to/from HDVICP0/1 RAM/Buffers]. If there are no SDMA transfers going into Level 2 memory ports, then SDMA/IDMA stalling is ***not*** the source of the problem.
3. Redirect all SDMA transfers to L2 memory to other memories using one of the following methods:
   (a) Temporarily transfer all the L2 SDMA transfers to L1D SRAM or ARM RAM.
   (b) If not all L2 SDMA transfers can be moved to L1D memory or ARM RAM memory, temporarily direct some of the transfers to DDR2 memory and keep the rest in L1D memory/ARM RAM memory. Still, there should be ***no*** L2 SDMA transfers.
   (c) If 'a.' and 'b.' are not possible, move the transfer with the real-time deadline to EMAC CPPI RAM. If the EMAC CPPI RAM is not big enough, a two-step mechanism can be used to page a small working buffer defined in EMAC CPPI RAM into a bigger buffer in L2 SRAM. The EDMA module can be setup to automate this double buffering scheme without CPU intervention for moving data from EMAC CPPI RAM. Some throughput degradation is expected when the buffers are moved to EMAC CPPI RAM.

   Note: EMAC CPPI RAM memory is only word-addressable. Therefore, EDMA transfers to/from EMAC CPPI RAM ***must*** have SRCBIDX/DSTBIDX = 4.

If real-time deadlines are still missed after implementing any of the options in Step 3, then IDMA/SDMA stalling is likely ***not*** the cause of the problem. If real-time deadline misses are solved using any of the options in Step 3, then IDMA/SDMA stalling ***is*** likely the source of the problem.

Note: The above Step 3 is applicable only to the SDMA accesses to L2 RAM.

Accesses to HDVICP RAM/Buffers via SDMA interface are typically driven by an application-specific requirement and cannot be redirected to some other memory. Additionally, for memory to memory transfers involving HDVICP, the issue would translate more into unexpected longer stalls/delays, and thereby performance degradation, not necessarily missing real-time deadlines. For accesses to HDVICP RAM/Buffers via the SDMA interface, if unexpected performance degradation is seen, use the guidelines under Method 2 in the Workaround(s) section.

Deadlock Scenario

As previously mentioned, a possible deadlock scenario is introduced in the presence of the SDMA stalls just described. This scenario occurs for certain masters connected to main data pipelined SCR1 either directly or indirectly through a bridge. For DM646x devices the masters that fall into this category, as shown in the TMS320DM6467 System Interconnect Block Diagram in the *TMS320DM6467 SoC Architecture and Throughput Overview* Application Report (literature number SPRAAW4), are:

- The ARM data port (ARM-D), connected to SCR1 through Bridge 2
- EDMA transfer controllers (EDMA TCs), connected to SCR1 directly
- PCI, connected to SCR1 through Bridge 3
- The masters HPI and ATA connected to a satellite SCR2, connected to SCR1 through Bridge 6
- The masters EMAC, USB, and VLYNQ connected to a satellite SCR3, connected to SCR1 through Bridge 7

If the following sequence of events occurs, then a deadlock situation might arise.

1. One of the following accesses occur:
   (a) ARM-D issues a write command to the DSP's SDMA followed by a subsequent write command to slave memories DDR2, EMIFA, or HDVICP0/1 EDMA ports.
   (b) An EDMA TC issues a write command to the DSP's SDMA followed by a subsequent write command to slave memories DDR2, EMIFA, HDVICP0/1 EDMA ports, or ARM TCM.
   (c) The PCI issues a write command to the DSP's SDMA followed by a subsequent write command to slave memories DDR2 or ARM TCM.
   (d) Any master connected to Bridge 6 (HPI and ATA) issues a write command to the DSP's SDMA and the same or different master connected to Bridge 6 issues a subsequent write command to slave memories DDR2 or ARM TCM.
   (e) Any master connected to Bridge 7 (EMAC, USB and VLYNQ) issues a write command to the DSP's SDMA and the same or different master connected to Bridge 7 issues a subsequent write command to to slave memories DDR2, EMIFA, HDVICP0/1 R/W EDMA ports, or ARM TCM.

   The following is a list of slave memories corresponding to each of the Bridge 7 masters:
   - Master EMAC has access to slave memory DDR.
   - Master USB has access to slave memories DDR, AEMIF, and ARM TCM.
   - Master VLYNQ has access to slave memories DDR, AEMIF, HDVICP0/1 R/W EDMA ports, and ARM TCM.
2. The DSP's SDMA asserts itself not ready and is unable to accept more write data, and a MDMA cache line writeback is initiated from DSP memory to DDR2 memory or to another slave memory (e.g., EMIFA).

In the above scenario it is possible for data phases from the write command issued to DDR2 (or EMIFA) to be stuck behind the data phases for the write to the DSP's SDMA in the SCR.

Therefore, if the DSP issues victim traffic to the same slave (DDR2 or EMIFA) then data associated with the victim traffic (#2) intended for DDR2 (or EMIFA) will be stuck behind write commands issued for #1. However, due to the MDMA/SDMA blocking issue, the SDMA traffic for #1 will be waiting for the MDMA traffic for #2 to finish, manifesting itself into a deadlock situation.

Note: If the slave memories (listed above in Step 1) are not cached in L2 (via the MAR bit settings) and there are no long distance read/writes to these memories from DSP then, the deadlock scenario will not arise (as there would be no MDMA accesses).

Workaround(s)

Method 1

For applications involving L2 RAM only, entirely eliminate IDMA/SDMA stalling stalling and potential for a deadlock condition using one or both of the following:

1. Configure the entire L2 RAM as 100% cache (e.g., move all data buffers to L1D/P, ARM RAM, EMAC CPPI memory, or external memory).

   Note: Some throughput degradation is expected when the buffers are moved to ARM RAM or EMAC CPPI RAM. Additionally, CPPI memory is only word-addressable. Therefore, EDMA transfers to/from EMAC CPPI RAM ***must*** have SRCBIDX/DSTBIDX = 4.
2. Eliminate all IDMA/SDMA access to L2 RAM during any time IDMA/SDMA stalling would have an impact (e.g., could preload data/code through IDMA/SDMA during system initialization/re-configuration).

Method 2

For applications involving IDMA/SDMA accesses to L2 RAM and/or HDVICP0/1 RAM/Buffers through DSP SDMA port, perform any of the following to reduce the IDMA/SDMA stalling system impact:

1. Improve system tolerance on DMA side (IDMA/SDMA/MDMA):
   (a) Understand and minimize latency-critical SDMA/IDMA accesses to Level 2 memories or L1P/D.
   (b) Directly reduce critical real-time deadlines, if possible, at peripheral and/or I/O level (e.g., increase word size and/or reduce bit rates on serial ports).
   (c) Reduce DSP MDMA latency by:
      (i) Increase priority of DSP access to DDR2/EMIFA such that MDMA latency of MDMA accesses causing stalls is minimized.
      Note: Other masters, such as VPIF, may have real-time deadlines that dictate higher priority than DSP.
      (ii) Lower PR_OLD_COUNT bit field setting in the DDR2 memory controller's Burst Priority Register. Values ranging between 0x10 and 0x20 should give decent performance and minimize latency; lower values may cause excessive SDRAM row thrashing.
      (iii) Do not perform EMIFA accesses using EMIFA WAIT handshaking during DSP run-time. Devices using WAIT potentially insert excessive latency to external memory accesses.
2. Minimize offending scenarios on DSP/Caching side:
   (a) If the DSP performing non-cacheable writes is causing the issue, insert protected non-cacheable reads (as shown in the last list item f. below) every few writes to allow write buffer to drain.
   (b) Avoid caching from slow memories such as, Asynchronous Memory. Instead, page the data via the EDMA from the off-chip Async Memory to L2 SRAM or SDRAM space before accessing the data from the DSP.

   Note: Paging cannot occur while real-time deadlines must be met.

(c) Use explicit cache commands to trigger cache write-backs during appropriate times (e.g., L1D Writeback All and L2 Writeback All).
Do not use these commands when real-time deadlines must be met.

(d) Restructure program data and data flow to minimize the offending cache activity.

(i) Define the read-only data as “const.” The const C keyword tells the compiler that the array will not be written to. By default, such arrays are allocated to the “.const” section as opposed to BSS. With a suitable linker command file, the developer can link the .const section off-chip, while linking .bss on-chip. Because programs initialize .bss at run time, this reduces the program’s initialization time and total memory image.

(ii) Explicitly allocate lookup tables and writeable buffers to their own sections. The #pragma DATA_SECTION(label, “section”) directive tells the compiler to place a particular variable in the specified COFF section. The developer can explicitly control the layout of the program with this directive and an appropriate linker command file.

(iii) Avoid directly accessing data in slow memories (e.g., flash); copy at initialization time to faster memories.

(e) Modify troublesome code.

(i) Rewrite using DMAs to minimize data cache writebacks. If the code accesses a large quantity of data externally, consider using DMAs to bring in the data, using double buffering and related techniques. This will minimize cache write-back traffic and the likelihood of IDMA/SDMA stalling.

(ii) Re-block the loops. In some cases, restructuring loops can increase reuse in the cache and reduce the total traffic to external memory.

(iii) Throttle the loops. If restructuring the code is impractical, then it is reasonable to slow it down. This reduces the likelihood that consecutive SDMA/IDMA blocks “stack up” in the cache request pipelines resulting in a long stall.

(f) Protect non-cacheable reads from generating an SDMA stall by freezing the L1D cache during the non-cacheable read access(es). The following example code contains a function that protects non-cacheable reads, avoids blocking during the reads, and, therefore, avoids the deadlock state.

```
;; ========================================================================= ;;
;;  Long Distance Load Word                                                  ;;
;;                                                                           ;;
;;      int long_dist_load_word(volatile int *addr)                          ;;
;;                                                                           ;;
;;  This function reads a single word from a remote location with the L1D    ;;
;;  cache frozen.  This prevents L1D from sending victims in response to     ;;
;;  these reads, thus preventing the L1D victim lock from engaging for the   ;;
;;  corresponding L1D set.                                                   ;;
;;                                                                           ;;
;;  The code below does the following:                                       ;;
;;                                                                           ;;
;;      1.  Disable interrupts                                               ;;
;;      2.  Freeze L1D                                                       ;;
;;      3.  Load the requested word                                          ;;
;;      4.  Unfreeze L1D                                                     ;;
;;      5.  Restore interrupts                                               ;;
;;                                                                           ;;
;;  Interrupts are disabled while the cache is frozen to prevent affecting   ;;
;;  the performance of interrupt handlers.  Disabling interrupts during      ;;
;;  the long distance load does not greatly impact interrupt latency,        ;;
;;  because the CPU already cannot service interrupts when it's stalled by   ;;
;;  the cache.  This function adds a small amount of overhead (~20 cycles)   ;;
;;  to that operation.                                                       ;;
;;                                                                           ;;
;; ========================================================================= ;;

        .asg    0x01840044,     L1DCC           ; L1D Cache Control
        .global _long_dist_load_word
        .text
        .asmfunc
; int long_dist_load_word(volatile int *addr)
_long_dist_load_word:
        MVKL    L1DCC,          B4
        MVKH    L1DCC,          B4
||      DINT                                    ; Disable interrupts
||      MVK     1,              B5
        STW     B5,             *B4             ; \_ Freeze cache
        LDW     *B4,            B5              ; /
        NOP     4
        SHR     B5,     16,     B5              ; POPER -> OPER
||      LDW     *A4,            A4              ; read value remotely
        NOP     4
        STW     B5,             *B4             ; \_ Restore cache
        RET     B3
||      LDW     *B4,            B5              ; /
        NOP     4
        RINT                                    ; Restore interrupts
        .endasmfunc

;; ========================================================================= ;;
;;  End of file: ldld.asm                                                    ;;
;; ========================================================================= ;;
```

Perform the following to eliminate the potential for a deadlock condition:

1. Force the ARM-D to perform writes to either the DSP SDMA (L1, L2, and HDVICP RAM/Buffers through SDMA port) or slave memories (DDR2, EMIFA, or HDVICP0/1 EDMA ports), but not to both.
2. Force each EDMA TC to perform writes to either DSP SDMA (L1, L2, and HDVICP RAM/Buffers through SDMA port) or slave memories (DDR2, EMIFA, HDVICP0/1 EDMA ports, or ARM TCM), but not to both.
3. For PCI, HPI, ATA, EMAC, USB, and VLYNQ, do one of the following:
   (a) Force the completion of pending write commands to either DSP SDMA (L1, L2, and HDVICP RAM/Buffers through SDMA port) or slave memories before initiating writes to a different destination. Pending write commands from a particular master are forced to complete when the same master initiates a read from the same destination memory. **Note**: In the case of DDR2 and EMIFA, a read command only forces the completion of write commands within a 2KB-aligned window.
   (b) Force each master (PCI) or all masters in a group ([HPI and ATA] one group, [EMAC, USB, and VLYNQ] another group) to perform writes to either DSP SDMA memory space or slave memories, but not to both (For list of slave memories corresponding to each master, see the ***Deadlock Scenario*** section of this advisory).

   Note: In the case of group of masters, for example [HPI and ATA] as a group (connected to Bridge 6), both of these masters must only perform writes to either DSP SDMA or slave memories, but not to both. For example, if HPI writes to DSP SDMA memory and ATA writes to DDR2 memory, the potential for the deadlock condition is still present. The same condition applies for other group of masters [EMAC, USB, and VLYNQ] connected to Bridge 7.

Note: If the slave memories (listed in the steps above) are not cached in L2 (via the MAR bit settings) and there are no long-distance read/writes to these memories from DSP, then the deadlock scenario will not arise (as there would be no MDMA accesses) and no need to perform the workaround.

Advisory 3.0.4 *Glitch on I2C Bus During Device Power-Up Sequence*

Revision(s) Affected: 3.0 and earlier

Details: The suggessted power-up sequence was 3.3 V→1.8 V→1.2 V. However, there is a small glitch on I2C bus during this power-up sequence. If the board design is already done with this power-up sequence (3.3 V→1.8 V→1.2 V), there is no need to change. This glitch should not be an issue because it is not a valid I2C bus transition (i.e., the glitch does not match either the START or STOP condition).

Workaround: Reverse the power-up sequence, that is, 1.2 V→1.8 V→3.3 V.

Advisory 3.0.16 *HDVICP – H.263 Encode: Quantized AC Coefficient Clipping*

Revision(s) Affected: 3.0 and earlier

Details: This low impact issue is rare and occurs only when ***all*** the following conditions are met:

- Encode
- H.263

In H.263 or MPEG4 (short_video_header = 1) encoding, the quantized AC coefficient should be clipped within a range of [-127:127]. But, the HD-VICP CALC QiQ function only supports the range of [- (2^N-1):2^N-1] (N = 8 to 15). Therefore, the minimum clipping range will be [-255:255].

This saturation can be mitigated by clipping the QP to a minimum value of 8 without perceivable quality loss.

Workaround: Control the QPmin for the encoder.

The DCT transform coefficient value will be in the range of -1024 to 1023, so the minimum value QP can be 8 to limit the quantized AC coefficient within the range of [-127:127].

Advisory 3.0.19 *DMA Access to L2 SRAM May Stall When the C64x+ CPU Command Priority is Lower Than or Equal to the DMA Command Priority*

Revision(s) Affected: 3.0 and earlier

Details:

Note: DMA refers to all non-CPU requests. This includes Internal Direct Memory Access (IDMA) requests and all other system DMA master requests via the Slave Direct Memory Access (SDMA) port.

The C64x+ Megamodule uses a bandwidth management (BWM) system to arbitrate between the DMA and CPU requests issued to L2 RAM. For more information on the BWM feature, see the *TMS320C64x+ DSP Megamodule* Reference Guide (Literature Number: SPRU871). The BWM arbitration grants L2 bandwidth based on programmable priorities and contention-cycle-counters. The contention-cycle-counters count the number of cycles for which the associated L2 requests are blocked by higher-priority requests. When the contention-cycle-counter reaches a programmed threshold (MAXWAIT), the associated L2 request is granted a slice of L2 bandwidth. This prevents indefinite blocking of lower-priority requests when faced with the continuous presence of higher-priority requests.

Ideally, the BWM arbitration will grant equal L2 bandwidth between equal priority DMA and CPU requests. Instead, when requests arrive at the BWM such that the CPU priority is *lower than or equal to* the DMA priority, the bandwidth is always granted in favor of the CPU over the DMA. In the case of successive CPU requests, it is possible for the CPU to block all DMA requests until the CPU traffic subsides. Figure 3 shows a high-level diagram of the arbitration scheme used for L2 RAM requests.

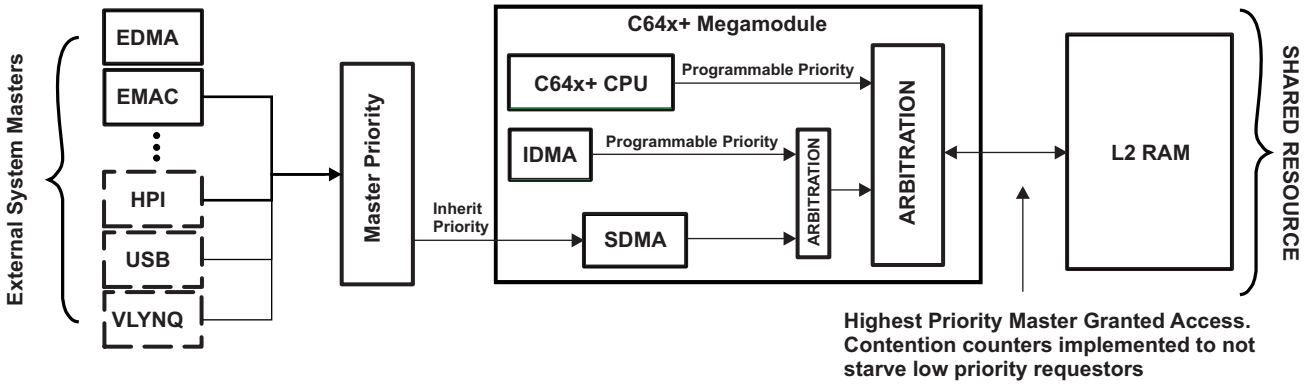

Figure 3. Priority Arbitration Scheme for L2 RAM

When the SDMA has finished sending all of its commands to the L2 controller, the C64x+ Megamodule drops the effective transfer priority to seven. The L2 Controller uses this effective priority to arbitrate the SDMA command with the CPU.

This happens regardles of what the actual SDMA priority is. This means that if the CPU Priority is equal to seven, then it can also trigger the issue highlighted by this advisory.

Workaround: Configure the DMA and CPU requests to different priority levels such that the CPU priority level is higher than the DMA priority level. Priority seven should not be used for the CPU. There is no penalty for setting the IDMA and SDMA priorities equal to each other.

The following table highlights which priority combinations are affected and what combinations are valid:

Table 3. Allowable CPU and SDMA Priorities

CPU PRIORITY	SDMA PRIORITIES ALLOWED
0	Not allowed; Affected by Advisory Issue
1	0
2	0-1
3	0-2
4	0-3
5	0-4
6	0-5
7	Not allowed; Affected by Advisory Issue

Pseudo code provided below highlights how the various requestor priorities can be configured:

CPU request priority is programmed within the CPUARBU register:

```
/** Pseudo code only **/

        Uint32 *CPUARBU;

        CPUARBU = ( Uint32 * ) ( 0x01841000 );

        /* Set priority different from IDMA/SDMA */
        *CPUARBU = [CPU_PRIORITY];
```

IDMA request priority is programmed within the IDMA1_COUNT register

```
/** Pseudo code only **/

        Uint32 *IDMA1_SRC, *IDMA1_DST;
        Uint32 *IDMA1_CNT;

        IDMA1_SRC = ( Uint32 * ) ( 0x01820108 );
        IDMA1_DST = ( Uint32 * ) ( 0x0182010C );
        IDMA1_CNT = ( Uint32 * ) ( 0x01820110 );

        *IDMA1_SRC = sourceAddress;
        *IDMA1_DST = destinationAddress;

        /* Set IDMA priority different from CPU */
        *IDMA_CNT = ( [IDMA_PRI] << [IDMA_PRI_SHIFT] ) | buffsize ;
```

SDMA request priority is inherited from the MSTPRIn registers

```
/** Pseudo code only **/

        Uint32 *MSTPRI1, *MSTPRI2;

        MSTPRI1 = ( Uint32 * ) ( 0x01C40040 );
        MSTPRI2 = ( Uint32 * ) ( 0x01C40044 );

        /* Set SDMA master priorities different from CPU */
        *MSTPRI1 = [MAST_PRI] << [MAST_SHIFT];
        *MSTPRI2 = [MAST_PRI] << [MAST_SHIFT];
```

Advisory 3.0.20 *HDVICP—H.264 Encode/Decode: Intra 16 x 16 Plane MB Values ≥ 512 are Incorrect*

Revision(s) Affected: 3.0 and earlier

Details:

During the calculation of the intra prediction block, pixel values can reach 512 or higher. However, the computation engine has precision only enough to hold values equal to 511 or less.

This issue can occur only when ***all*** the following conditions are met:

- H.264 Encode or Decode
- Intra_16x16_Plane mode MB
- Luma blocks *only*; it does not affect Chroma blocks
- If the prediction block generated using the Intra_16x16_Plane mode has a pixel value of 512 or higher

The Artifacts by this advisory are:

- In Encode, an incorrect (but legal) bitstream will be generated, and a mismatch will happen between the Encoder and Decoder resulting in signficant quality degradation.
- In Decode, the reconstructed macroblock will be incorrect.
- If the following MBs are intra referring to this MB, the error is propagated spatially within the picture.
- If the following pictures refer to this MB, the error is propagated temporally to the following pictures.

This advisory is content dependent. Typically, generating Intra Prediction Values ≥ 512 is rare but can occur; therefore, the suggested workarounds below should be used. A potential exception could be for a closed-system decoder where the user can ensure that the encoder does not use Intra_16x16_Plane mode.

Workaround(s):

Encode

Turn off the Intra_16x16_plane mode evaluation by HDVICP IPE. Typically, the quality degradation for removing this mode is relatively small.

Decode

During a MB pipeline slot, a CALC is done, a workaround by CPU, and CALC again with corrected intra prediction macroblock (see Figure 4).

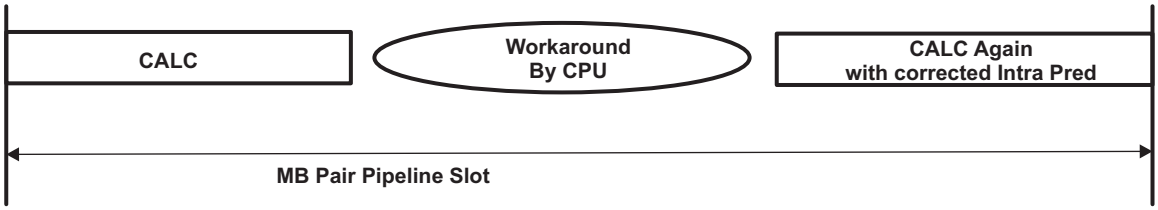

Figure 4. MB Pair Pipeline Slot Example

If the MB is of intra 16x16 plane mode, perform the following steps:

1. Let CALC run.
2. Wait for CALC to complete
3. After CALC completes, switch BFSW for iprdbuf : CALC → DMA
4. Check and correct Intra Prediction (for more details, see the *Check and Correct Intra Prediction* section below)
5. Switch BFSW for iprdbuf : DMA → CALC
6. Switch BFSW for calcmbuf : CALC → DMA
7. Set up CALC commands with intra mode for IQ/IT and inter mode for prediction (for more details, see the *Set up CALC Commands with Intra mode for IQ/IT and Inter Mode for Prediction* section below
8. If MBAFF, retrigger DMA for upper row data
9. Switch BFSW for calcmbuf : DMA → CALC
10. Switch BFSW for reconbuf : LPF → DMA
11. Read left reconstructed column
12. Switch BFSW for reconbuf : DMA → LPF
13. Switch BFSW for calcsbuf : CALC → DMA
14. Set back left reconstructed column
15. Switch BFSW for calcsbuf : DMA → CALC
16. Re-run CALC
17. Wait for CALC to complete

Check and Correct Intra Prediction

Read Pred[0,0], Pred[15,0], Pred[0,15], Pred[15,15], Pred[7,7] and Pred[7,8] (or Pred[8,7]) from iprdbuf

/* This workaround uses the fact that 1D slope is constant on a Plane. */

```
/* This workaround uses the fact that 1D slope is constant on a Plane. */

// right top pixel is wrong
if (Pred[15,0] == 0x00 && Pred[0,15] < Pred[7,8] (or Pred[8,7])) {
    Pred[15,0] = 0xFF;
}
// left bottom pixel is wrong
else if (Pred[0,15] == 0x00 && Pred[15,0] < Pred[7,8] (or Pred[8,7])) {
    Pred[0,15] = 0xFF;
}
// right bottom pixels are wrong
else if (Pred[15,15] == 0x00 && Pred[0,0] < Pred[7,7]) {
    Pred[15,15] = 0xFF
    if (Pred[11,15] == 0x00) Pred[11,15] = 0xFF;
    if (Pred[12,14] == 0x00) Pred[12,14] = 0xFF;
    if (Pred[12,15] == 0x00) Pred[12,15] = 0xFF;
    if (Pred[13,13] == 0x00) Pred[13,13] = 0xFF;
    if (Pred[13,14] == 0x00) Pred[13,14] = 0xFF;
    if (Pred[13,15] == 0x00) Pred[13,15] = 0xFF;
    if (Pred[14,12] == 0x00) Pred[14,12] = 0xFF;
    if (Pred[14,13] == 0x00) Pred[14,13] = 0xFF;
    if (Pred[14,14] == 0x00) Pred[14,14] = 0xFF;
    if (Pred[14,15] == 0x00) Pred[14,15] = 0xFF;
    if (Pred[15,11] == 0x00) Pred[15,11] = 0xFF;
    if (Pred[15,12] == 0x00) Pred[15,12] = 0xFF;
    if (Pred[15,13] == 0x00) Pred[15,13] = 0xFF;
    if (Pred[15,14] == 0x00) Pred[15,14] = 0xFF;
}
```

Set up CALC Commands With Intra Mode for IQ/IT and Inter mode for Prediction

To do the 2nd run, set CALC commands per Table 4 and Table 5. MB mode and Q/IQ information will need to be modified.

Table 4. Format of MB-mode for Spatial Intra Prediction

MB-MODE	
BIT NO.	**DESCRIPTION**
31	TQ Bypass-mode **[0]** :Disable, [1] :Enable.
30:28	Reserved.
27	Chroma DC-Transform (2x2) [0] :Disable, **[1]** : Enable.
26	Luma DC-Transform (4x4) [0] :Disable, **[1]** :Enable.
25:24	Scaling-timing [00] : Do not sue Scaling -function. [01] : Scaling per Luma/Chroma. **[10]** : Scaling per Color-Block. [01] : Scaling per Block.
23:20	Reserved.
19:18	Transform size of Block [5] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
17:16	Transform size of Block [4] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
15:14	Transform size of Block [3] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
13:12	Transform size of Block [2] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
11:10	Transform size of Block [11] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
9:8	Transform size of Block [0] [00] :8x8 , [01] :8x4 , [10] :4x8 , **[11]** :4x4
7	Reserved.
6:4	Chroma Block-mode [000] :Reserved , **[001]** :Intra 8x8 , [010] :Reserved , [011] :Intra w/o Pred. , [100] :Inter , [101] : Reserved , [110] :Reserved , [111] :Skip.
3	Reserved.
2:0	Luma Block-mode [000] :Intra 16x16 , [001] :Intra 8x8 , [010] :Intra 4x4 , [011] :Intra w/o Pred. , **[100]** :Inter , [101] : Reserved , [110] :Reserved , [111] :Skip.

For Table 5, set Inter shift-scale value = Intra shift-scale value.

Table 5. Q/IQ Information

PARAMETER FOR Q/IQ		
ADDRESS OFFSET	**BIT NO.**	**DESCRIPTION**
0x08	63:24	Reserved.
	23:16	**Shift-scale for IQ Intra-DC-Y**
	15:8	**Shift-scale for IQ Inter-DC-Y**
	7:0	Shift-scale for IQ AC-Y
0x10	63:56	Reserved.
	55:48	**Shift-scale for IQ Intra-DC-Cr**
	47:40	**Shift-scale for IQ Inter-DC-Cr**
	39:32	Shift-scale for IQ AC-Cr
	31:24	Reserved.
	23:16	**Shift-scale for IQ Intra-DC-Cb**
	15:8	**Shift-scale for IQ Inter-DC-Cb**
	7:0	Shift-scale for IQ AC-Cb
0x18	63:24	Reserved.
	23:16	**Shift-scale for Q Intra-DC-Y**
	15:8	**Shift-scale for Q Inter-DC-Y**
	7:0	Shift-scale for Q AC-Y
0x20	63:56	Reserved.
	55:48	**Shift-scale for Q Intra-DC-Cr**
	47:40	**Shift-scale for Q Inter-DC-Cr**
	39:32	Shift-scale for Q AC-Cr
	31:24	Reserved.
	23:16	**Shift-scale for Q Intra-DC-Cb**
	15:8	**Shift-scale for Q Inter-DC-Cb**
	7:0	Shift-scale for Q AC-Cb
0x28	63:32	Round coefficient [1]
	31:0	Round coefficient [0]

This advisory has been fixed in TI Decoder S/W Version 01.10.01 or later.

Appendix A Revision History

This silicon errata revision history highlights the technical changes made to the SPRZ307 revision to make it an SPRZ307A revision.

Scope: Applicable updates relating to the TMS320DM6467 devices have been incorporated.

Table 6. DM6467 Revision History

SEE	ADDITIONS/MODIFICATIONS/DELETIONS
Section 1.2	Revision Identification: • Updated/ Changed figure • Updated/ Changed second paragraph

IMPORTANT NOTICE

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

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

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

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

Reproduction of 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

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