OMAP-L137
C6000 DSP+ARM Processor
Silicon Revisions 3.0, 2.1, 2.0, 1.1, and 1.0

Silicon Errata

Literature Number: SPRZ291I
October 2008–Revised June 2014

Contents

OMAP-L137
Silicon Revisions 3.0, 2.1, 2.0, 1.1, and 1.0

1 Introduction

This document describes the known exceptions to the functional specifications for the *C6000 DSP+ARM Processor* devices (for example, OMAP-L137). For more detailed information on these devices, see the device-specific data manual: *OMAP-L137 C6000 DSP+ARM Processor* data manual (literature number SPRS563).

For additional peripheral information, see the latest version of the *OMAP-L137 C6000 DSP+ARM Processor* Technical Reference Manual (Literature Number: SPRUH92).

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

This document also contains Usage Notes. 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 notes will be incorporated into future documentation updates for the device (such as the device-specific data manual), and the behaviors they describe will not be altered in future device revisions.

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 OMAP processors and support tools. Each commercial OMAP platform member has one of three prefixes: X, P, or null (no prefix) (for example, OMAPL137CZKB3). 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 (TMDX) through fully qualified production devices/tools (TMDS).

Device development evolutionary flow:

X	Experimental device that is not necessarily representative of the final device's electrical specifications
P	Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification
NULL	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

X and P 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.

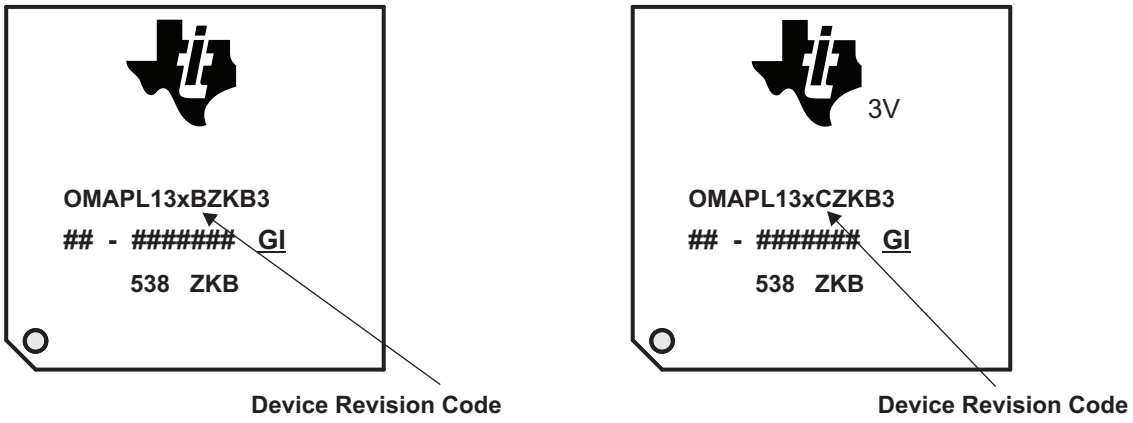
Predictions show that prototype devices (X or P) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are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

1.2 Package Symbolization and Revision Identification

Example, Device Revision Codes for OMAP-L137 (ZKB Package) shows an example of the OMAP-L137 processor package symbolization. The device revision code can be identified by the markings on the top of the package; for example, the "C" between the device number and the package identifier indicates the device revision code (that is, Silicon Revision 2.1) A "D" between the device number and the package identifier would indicate the device revision code for Silicon Revision 3.0.

Table 1 lists the device revision codes for the OMAP-L137 processor, as well as shows the relationship between the device revision codes and silicon revisions.

For more detailed information on device nomenclature breakdowns, see the "*Device and Documentation Support*" section of the device-specific data manual. For orderable addendum information, see the "*Packaging Information*" section of the data manual.



Example, Device Revision Codes for OMAP-L137 (ZKB Package)

Table 1. *OMAP-L137* **Device Revision Codes**

DEVICE REVISION CODE (xx)	SILICON REVISION	COMMENTS
D	3.0	All Silicon Revision 3.0 parts support a minimum operating DVDD (I/O) voltage of 3.0V; therefore, no additional marking is necessary.
C	2.1	Silicon Revision 2.1 parts that have a minimum operating DVDD (I/O) voltage of 3.0V can be identified by the 3V marking next to the TI logo on the package. See Example, Device Revision Codes for OMAP-L137 (ZKB Package) for details.
B	2.0	Silicon Revision 2.0 parts that have a minimum operating DVDD (I/O) voltage of 3.0V can be identified by the 3V marking next to the TI logo on the package. See Example, Device Revision Codes for OMAP-L137 (ZKB Package) for details.
A	1.1	-
(blank)	1.0	-

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 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 McASP: Inactive Slot Usage Note

On all the silicon revisions, McASP underflow (underruns) can occurs if any of the McASP serialzer is configured as a transmit serializer with any of the time slot-n in XTDM register field is set to inactive and EDMA is used to do the transfers. This is independent of whether the EDMA/CPU transfers are through peripheral configuration bus or the DMA port.

If the EDMA is used to service the McASP with any of the time slot-n in XTDM register field is set to inactive, EDMA may not be triggered upon enabling serializer and hence data transfer will be stalled or if data transfer has begun with some random value and goes to underrun errors, that breaks the data transfer operation.

To ensure proper McASP (transmit) data transfer either through config or EDMA port, ensure the time slot-n in XTDM register field is set to active. For example, if a serializer is configured for transmit operation with a 5-slot TDM frame in which it is only required to transmit data in slots 0 to 2, but make sure all five slots (0 to 4) should be configured as active. The EDMA configuration and user application should account for the transfer of irrelevant data to the McASP for slots 3 and 4.

2.1.2 USB0: Generic RNDIS Usage Note

On all silicon revisions, when using Generic RNDIS mode, the user should ensure that the DMA configuration has completed prior to the host starting a transfer. This condition is sometimes violated when performing back-to-back data transfers (not transactions). If a new transfer is scheduled by a host while the device is working on the previous transfer and the data transfer size for the new transfer is different than the previous transfer data size, then there exists a contention between the two transfer sizes creating undesired behavior resulting with a DMA lock up. A case in point where this violation could happen is demonstrated by the example below.

A user configures the DMA in Generic RNDIS mode expecting a data size of 512 bytes or less from a host. The host sends 512 bytes or less of data to the device. While the device is in the process of working on the received data to figure out the size of the next data transfer, the host starts a new data transfer addressing the same endpoint. Since the endpoint FIFO is empty, the device accepts the data and the DMA starts to transfer the received data from the receive FIFO to memory. At the same time, the application on the device side finishes and figures out the next transfer data size (using the data received from previous transfer) and reconfigures the Generic DMA Size register for the second transfer. If the second transfer size is different from the first transfer size, the contention happens at this point. The host has already started the second transfer prior to the device re-configuring the DMA parameters. The application on the device side, updates the DMA size register content for the second transfer while the DMA is in the middle of the second transfer using the DMA size register content of the first transfer. This effectively results with altering the DMA size register content while the DMA is in the middle of a transfer. Changing DMA parameters while in the middle of a transfer is not allowed and when done it will create undesirable outcomes.

Workaround: This is not a bug and for this reason, there exists no workaround. This is a caution for the user to be aware of this issue and hence to ensure that this scenario is avoided. If there exists an idle time in between the two back-to-back transfers, this issue will not exist. When expecting a back-to-back transfer where RNDIS mode can not be used, the user needs to use TRANSPARENT mode. When using TRANSPARENT mode, the application will be receiving more interrupts, that is, interrupts will be generated on each USB packets as opposed to receiving a single interrupt on the completion of a transfer.

2.1.3 USB0: Isochronous Interrupt Loading Usage Note

On all silicon revisions, when the USB Controller Endpoint is enabled to handle Isochronous type of transfer, the controller supports a single configuration for interrupt generation, which is for interrupts to be generated for every ISO packet received or sent, that is, transfer size is equal to packet size. The option of generating interrupt on multiple ISO packets received or sent is not available. Since ISO transfer can be scheduled to happen on every micro-frame or frame, the number of interrupts generated could overwhelm the system. This is not a problem as long as there is enough CPU power available to handle all interrupts. However, some applications may be running low on available CPU time and may desire to service/process multiple ISO packets at a time. The option for handling ISO interrupts in a batch is not available. The user should ensure that enough CPU power is available to handle all ISO interrupts in order to avoid missing interrupts resulting with missing ISO packets.

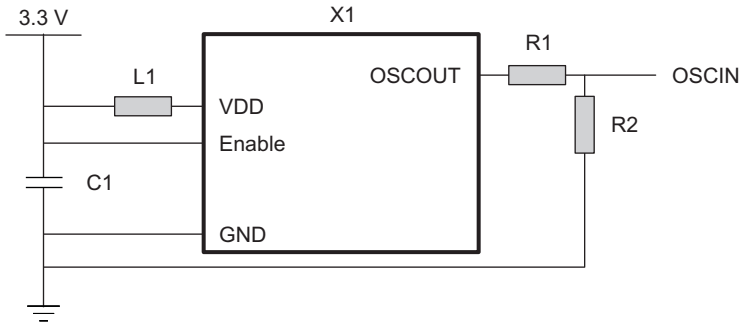
2.1.4 Clock Input Buffers Updated for Noise Immunity

For silicon revision 2.1 and later, the I/O buffers for all McASP clock inputs have been improved to provide more robust noise immunity than in previous silicon revisions. Timing specifications are not affected as a result. In particular, the specification for maximum input transition time remains the same; the improvements simply provide more margin to the existing specification.

2.1.5 System-Level ESD Immunity Usage Note

On all silicon revisions, certain design elements make this device susceptible to radiated noise during an ESD strike, as described in the standard IEC 61000-4-2. Exposure to the electrical noise caused by the ESD can cause soft device failures due to noise coupling on the system clock (OSCIN). ESD events within the IEC spec range do not cause permanent device damage and full functionality is recoverable with a device reset. The sensitivity to this noise issue is primarily due to the 1.2V oscillator/clock input implemented on this device. The low voltage range, coupled with slow rise and fall times, provides a lower noise margin than other TI devices with higher voltage internal oscillators (for example, 1.8V or 3.3V oscillators).

If ESD robustness is a concern, it is strongly recommended to avoid using the internal oscillator as a clock source. An external 3.3V clock source with a resistor voltage divider as in Figure 1 can be used to externally generate the required 1.2V input clock. By using an external clock input with fast rise/fall times (less than 5 ns), the noise margin improves significantly, increasing ESD noise resistance.



Legend: L1 = ferrite bead; C1 = 0.1 uF; R1 = 165 ohm / 5%; R2 = 100 ohm / 5%

Figure 1. External 3.3V Clock Source

In addition to using an external clock source, several other board and software recommendations specific to this device can improve system-level ESD immunity:

- The OSCIN and OSCVSS (and OSCOUT, if used) should be routed as short as possible to reduce their ability to pick up EMI noise.
- Route the OSCIN signal on inner board layers where it is shield by power and ground planes.
- The processor should be provided as much power supply decoupling as is practical and placed as close to the processor as possible.
- Implement the PLL filtering circuits shown in the device datasheet.

These recommendations are in addition to standard methods for increasing system ESD immunity, such as using shielding enclosures, proper grounding and PCB stackup, and ESD protection circuitry.

2.2 Silicon Revision 3.0 Known Design Exceptions to Functional Specifications

The advisories are not enumerated in sequential order and hence some numbers may not appear in the document

Table 2. Silicon Revision 3.0 Advisory List

Advisory 3.0.4 *DMA Access to L2 RAM Can Stall When DMA and C674x CPU Command Priorities are Equal*

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 C674x Megamodule uses a bandwidth management (BWM) system to arbitrate between DMA and CPU requests issued to L2 RAM. See TMS320C674x DSP Megamodule Reference Guide, Literature Number - SPRUFK5 for more information on the BWM. 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 low 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 equal priority DMA and CPU requests arrive at the BWM, bandwidth is always granted in favor of the CPU over DMA. In the case of successive CPU requests, it is possible for the CPU to block all DMA requests until CPU traffic subsides. Additionally, some command logic in the BWM uses priority level 7, which can also result in SDMA stalls when the CPU is also programmed to priority level 7. Figure 2 shows a high level diagram of the arbitration scheme used for L2 RAM requests.

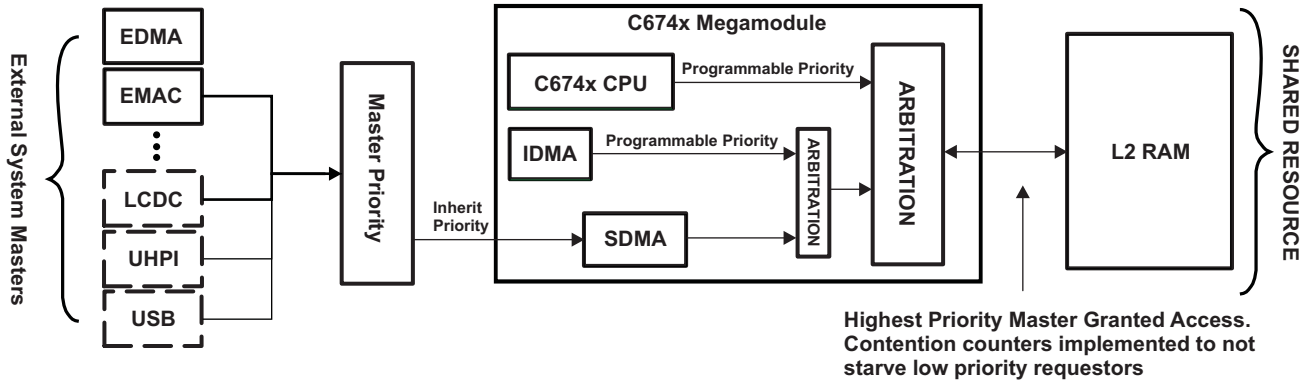


Figure 2. Priority Arbitration Scheme for L2 RAM

Workaround(s)

Configure DMA and CPU requests to different priority levels. The CPU should not be set to priority level 7. There is no penalty for setting the IDMA and SDMA priorities equal to each other.

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 * ) ( 0x01C14114 );
        MSTPRI2 = ( Uint32 * ) ( 0x01C14118 );

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

Advisory 3.0.6 *USB0: Extraneous RESET Interrupt*

Revision(s) Affected 3.0 and earlier

Details When the USB controller is operating as a device and an attached host resets the device after the completion of the “Device Attached” state by driving both differential data lines low, the USB controller operating as a device could receive multiple RESET interrupts for the single RESET signaling invoked by the host. The multiple interrupt generation only happens for the duration of the RESET signaling on the bus. RESET Interrupt is not generated before or after the completion of RESET.

Workaround(s) Software must service every USB RESET interrupt received. Software should not proceed on performing any other task, like initialization, until RESET duration has come to completion. The POWER[RESET] bit field will be cleared by the USB Controller when RESET signaling on the bus is removed by the Host. The USB Controller clearing the POWER[RESET] bit field should be used by software as an indication for the completion of RESET signaling.

Advisory 3.0.7 *LCDC Underflow During Initialization*

Revision(s) Affected 3.0 and earlier

Details

During LCDC initialization, there is the potential for a FIFO underflow condition to occur. A FIFO underflow condition occurs when the input FIFO is completely empty and the LCDC raster controller logic that drives data to the output pins attempts to fetch data from the FIFO. When a FIFO underflow condition occurs, incorrect data will be driven out on the LCDC data pins.

An underflow condition will occur if the EMIFB issues a refresh command to the SDRAM memory during LCDC start up/initialization. The error condition will be captured in the LCD Status register (LCD_STAT) in the FIFO underflow status (FUF) bit field. If the FUF_EN bit is enabled in Raster Control Register (RASTER_CTRL), the LCDC will send an interrupt to the CPU.

The FIFO underflow described above is not expected to be a common occurrence because of the unlikely alignment of events required to produce the underflow condition.

Workaround(s)

The EMIFB hardware automatically schedules refresh commands to the SDRAM memory. Therefore, it is not possible for the user/application code to schedule EMIF refresh commands to prevent them from being initiated during LCDC start up. This means that it is not possible to prevent an EMIFB refresh occurrence during the start up of the LCDC.

Software should poll the FUF bit field in the LCD_STAT register to check if an error condition has occurred or service the interrupt if FUF_EN is enabled when FUF occurs. If the FUF bit field has been set to 1, this will indicate an underflow condition has occurred and then the software should execute a reset of the LCDC via the LPSC.

Advisory 3.0.13 *EMIFA: Asynchronous Memory Timeout Error Persistence*

Revision(s) Affected

3.0 and earlier

Details

In Extended Wait mode, during a read access to an asynchronous memory, if the WAIT input does not go inactive within maximum extended wait cycles programmed in the Async Wait Cycle Config register, the EMIFA will report a time-out error. The data returned for this access will be all zeros. If this access is followed by a read to the EMIFA's memory-mapped register (MMR) space, the EMIFA will still report a time-out error but with the correct data for the MMR read. The EMIFA will hold the time-out error until another asynchronous access without a time-out error or an SDRAM access is performed.

This issue is only applicable if all of the following are true:

- The EMIFA is used for asynchronous memory accesses in Extended Wait mode.
- There is a potential for a time-out error to occur, that is, the asynchronous memory will not de-assert the WAIT input.
- If asynchronous memory read with time-out error is followed by an MMR read.

Workaround(s)

If a time-out occurs, perform any of the following:

- A dummy read to another asynchronous memory chip select that is not configured to be in Extended Wait mode.
- A dummy read to the same asynchronous memory chip select after disabling the Extended Wait mode on that chip select.
- A dummy read to SDRAM

Advisory 3.0.14 *A Single CHIPINTn Interrupt Event Will Register Multiple Times in the DSP Event Combiner Module (ECM)*

Revision(s) Affected 3.0 and earlier

Details The C674x DSP megamodule supports twelve maskable hardware interrupt signals (CPUINT4 through CPUINT15). Single system interrupts may be mapped directly to a CPUINTn hardware interrupt, or multiple system interrupts may be combined by the ECM into a single signal before mapping to a CPUINTn interrupt. See SPRUFK5 - TMS320C674x DSP Megamodule Reference Guide for more information on how DSP interrupts are handled.

The ECM expects all incoming interrupts to be pulse interrupts, however the [SYSCFG_CHIPSIG_]CHIPINTn interrupts are level interrupts. This mismatch in interrupt types will cause a single CHIPINTn interrupt event to register multiple times in the ECM.

Workaround(s) The CPUINTn hardware interrupts can support both pulse and level interrupts so CHIPINTn interrupts should be mapped directly to CPUINTn hardware interrupts. Furthermore, if the ECM is used for other system interrupts, the CHIPINTn interrupts should be masked out in the EVTMASKn registers.

Advisory 3.0.15 *Potential USB2.0 Soft Reset Timing Violation*

Revision(s) Affected

3.0 and earlier

Details

When a soft reset is invoked by setting the RESET bit of the USB CTRLR register (CTRLR[RESET] = 1), the internal reset timing requirements may be violated. Although this timing violation has not been observed in practice, the potential for a timing violation exists.

USB resets initiated by system-reset and power-on-reset are immune from the timing violation.

There is no plan to fix this issue in future silicon revisions because:

1. No functional problems have been observed to date
2. A software workaround has been developed to avoid the problem

Workaround(s)

The reset timing violation can be avoided by providing the modified soft reset activation sequence outlined below:

1. Enable the USB controller module clock
2. Perform a soft USB reset
3. Wait for the USB soft reset bit to clear
4. Disable the USB controller module clock
5. Configure the USB PHY parameters
6. Enable the PHY
7. Enable the USB controller module clock

Advisory 3.0.17 *ARM Interrupt Controller Vector Size Register (VSR) Initialization*

Revision(s) Affected 3.0 and earlier

Details The VSR register in the ARM Interrupt Controller (AINTC) is not correctly initialized after reset. If this register is not explicitly configured, the AINTC will only allocate 1 byte per interrupt (instead of 4).

Workaround(s) The desired value (even if it is the default value) should be written to the VSR prior to using the interrupt controller.

Advisory 3.0.18 *A Single CHIPINTn Interrupt Event Can Register Multiple Times in the AINTC*

Revision(s) Affected 3.0 and earlier

Details

Interrupts destined for the ARM CPU are managed by the ARM Interrupt Controller (AINTC). The AINTC detects, combines, and routes system interrupts to the two native ARM interrupt signals FIQ and IRQ. See the device Technical Reference Manual SPRUH92 for additional information about the AINTC.

The AINTC module expects all incoming interrupts to be pulse interrupts, however the [SYSCFG_CHIPSIG_]CHIPINTn interrupts are level interrupts. This mismatch in interrupt types will cause a single CHIPINTn interrupt event to register as multiple interrupt pulses in the AINTC. However, the AINTC does not have the capacity to count the number of interrupt pulses received per system interrupt – it only maintains interrupt flags. A system interrupt is flagged as active until its status is cleared by the user through the AINTC, regardless of the number of interrupts detected.

If the status flag for AINTC CHIPINTn is cleared while the CHIPINTn interrupt is still active, the AINTC will continue to detect CHIPINTn interrupts and its status flag will be set again. This additional setting of the AINTC CHIPINTn status flag is false.

Workaround(s)

Method 1

Do not execute the intended interrupt service routine code if the associated CHIPSIGn status flag is not set in the SYSCFG_CHIPSIG register. A cleared CHIPSIGn status flag indicates that the device is responding to a false interrupt. This method is easy to implement, but does not eliminate false interrupts.

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

    void CHIPINT0_ISR(void) {
        /* Exit immediately if CHIPSIG0 is not set */
        if( (SYSCFG->CHIPSIG & 0x1) == 0 ) {
            return;
        }

        /* Intended service routine code */
        SYSCFG->CHIPSIG_CLR = 0x1;
        printf("Hello World!\n");
    }
```

Method 2

Do not clear the AINTC CHIPINTn status flag until the CHIPSIGn status has been cleared. This method will eliminate false interrupts, but requires changes to the AINTC interrupt dispatch code. Changing the dispatch code may introduce undesired behavior in the application.

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

    /* Sequence that is susceptible to false CHIPINTn interrupts */
    void AINTC_ISR_DISPATCH_1(void) {
        Get_Interrupt_Information();

        /* CHIPINTn interrupts continue to be generated after */
        /* AINTC CHIPINTn flag is cleared.                    */
        Clear_AINTC_Interrupt_Flag();

        /* CHIPINTn interrupts are only stopped after ISR clears */
        /* the status flag.                                      */
        Branch_To_ISR();
    }
```

```
/* Sequence that is not susceptible to false CHIPINTn interrupts */
   void AINTC_ISR_DISPATCH_2(void) {
       Get_Interrupt_Information();

       /* ISR will clear CHIPSIGn flag and discontinue CHIPINTn */
       /* interrupts to AINTC.                                  */
       Branch_To_ISR();

       /* Ok to clear AINTC CHIPINTn flag now.                  */
       Clear_AINTC_Interrupt_Flag();
   }
```

Advisory 3.0.19 *Incorrect Masking of the C674x CSR:SAT Bit*

Revision(s) Affected

3.0 and earlier

Details

The C674x CPU supports a Saturation feature for key arithmetic operations. If an operation results in saturation, the SAT (saturation) bit in the control status register (CSR) is set. In normal operation, one or more functional units can simultaneously perform arithmetic operations that can result in saturation. In the case of simultaneous arithmetic operations, the SAT bit is set if at least one functional unit's operation results in saturation. The saturation status register (SSR) provides saturation flags for each functional unit, making it possible for the program to distinguish between saturations caused by different instructions in the same execute packet. Also, there is no direct connection to the SAT bit in the control status register (CSR); writes to the SAT bit have no effect on SSR and writes to SSR have no effect on the SAT bit.

In the case where a 2 cycle .M unit instruction is in the delay slot of a 4 cycle instruction of the same .M unit, and if both instructions are expected to generate results in the same cycle, the CSR:SAT bit will be incorrectly masked. Ideally, the CSR:SAT bit should be set if any one of the two .M unit instruction causes a saturation. Instead, the arithmetic saturation result of the 2 cycle .M unit instruction will overwrite the CSR:SAT bit.

All of the following must take place in order for an application to be affected by this advisory:

1. A 2 cycle .M unit instruction and a 4 cycle .M unit instruction are issued simultaneously
2. Both instructions are processed on the same side
3. The 2 cycle instruction is in the delay slot of the 4 cycle instruction so that the results of both instructions are generated in the same cycle
4. The saturation result of the 4 cycle .M unit instruction is different from the saturation result of the 2 cycle .M unit instruction
5. The application checks for the saturation flag and uses the saturation result of the 4 cycle instruction

Workaround(s)

Perform one of the following:

- For the location of code where saturation results are monitored, do not mix datatypes so that 2 cycle and 4 cycle .M unit instructions are not issued together.
- Do not mix floating point .M unit instruction with fixed point 2 cycle .M unit instructions.

Advisory 3.0.21 *SDMA Activity Can Corrupt L1D When L2 Is Configured as Mixed/C ache/SRAM*

Revision(s) Affected 3.0 and earlier

Details

Note: SDMA refers to all non-CPU requests to the EMC SDMA (Slave Direct Memory Access) port (see Figure 3). SDMA requests are defined as external system bus master requests handled via this port.

The C674x Megamodule uses a two-way set associative cache for L1D. This means that every physical memory location in the system has two possible set/way locations in the cache where it can reside. See TMS320C674x DSP Megamodule Reference Guide (Literature Number SPRUFK5) for more information on the L1D cache architecture and related terminology. Updated (dirty) values in L1D cache are not written back to external memory until cache activity evicts a cache-line (victim write-back) or a write-back is requested by software.

An L1D cache-line corruption event occurs when all of the conditions in the following steps are met (see Figure 3):

1. L1D cache Lines 1, 2, and 3 have the following characteristics:
   - Line 1 is associated with L2 SRAM (Line A in Figure 3), was previously read by CPU, and is clean. (CPU has not updated the data.)
   - Line 2 is associated with L2 SRAM (Line B in Figure 3), was previously read by CPU, and is clean. (CPU has not updated the data.)
   - Line 3 was previously read by the CPU and may be either clean or dirty.
2. SDMA receives updated data for L2 SRAM Lines A and B, which correspond to L1D cache Lines 1 and 2.
3. A snoop write operation is initiated by the L2 to overwrite the L1D cache Lines 1 and 2 with updated L2 SRAM Lines A and B. Before the snoop write operation finishes, the CPU performs two reads within the same clock cycle:
   - Line E in L2 cache is read as a cache hit. Line E is destined to replace Line 2 in L1D Cache, which also has a snoop write pending for the updated Line B content.
   - Line D in L2 SRAM is read. Line D will replace Line 3 in L1D cache.
4. When the snoop write operation completes, Line 2 in L1D cache now contains the updated L2 SRAM Line B data instead of the L2 cache Line E data.

The correct behavior would have been to kill the pending snoop write initiated to update L1D cache Line 2 with the updated L2 SRAM Line B data in Step 3. The L1D cache should have evicted Line B and replaced it with Line E data. Instead, the snoop write operation continues and does not complete until after the L1D cache Line 2 has already been replaced with L2 cache Line E data. The snoop write instruction overwrites the L1D cache Line 2 (containing L2 cache Line E data) with the updated L2 SRAM Line B data.

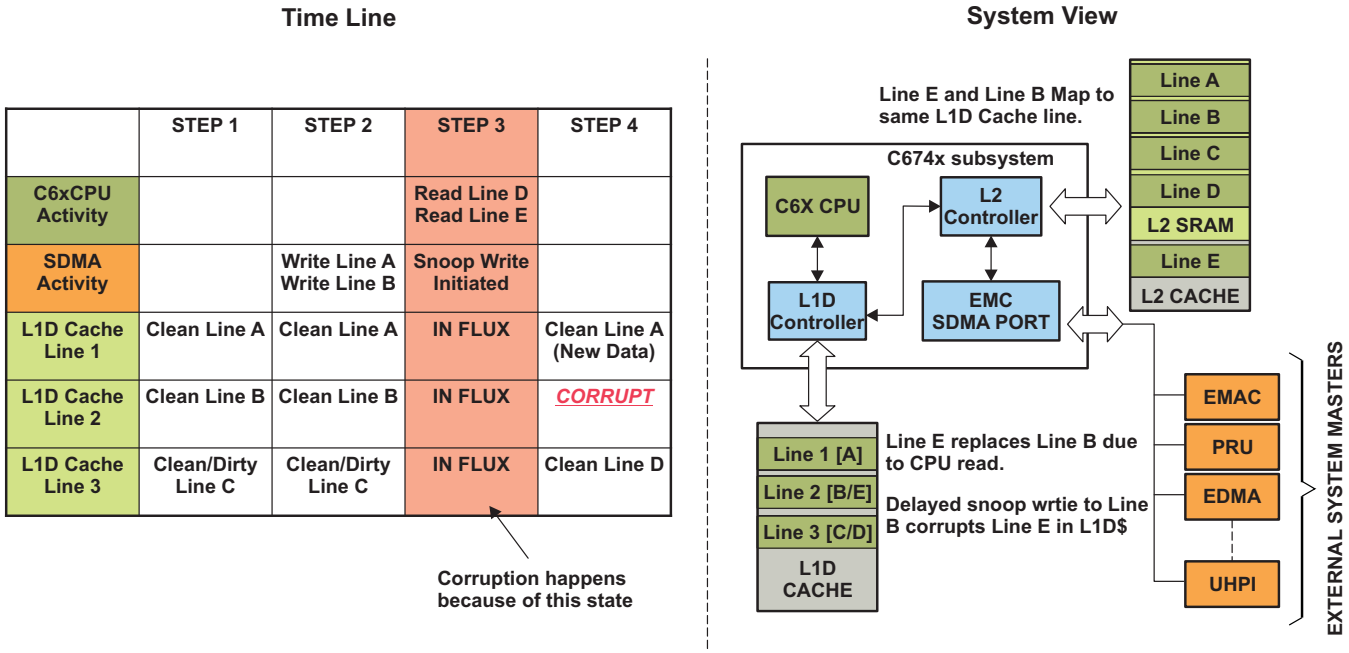


Figure 3. Example of L1D Cache Corruption

Workaround(s)

Method 1: Do not perform two CPU read operations in the same clock cycle. For C code, use compiler flag **(--c64p_dma_l1d_workaround)** available in the C6000 Compiler (CodeGen) Tools version 7.0.2 and later. For assembly code, the --c64p_dma_l1d_workaround flag will only issue a warning.

Method 2: In cases where buffer access will not be shared between CPU and SDMA, unintended CPU/SDMA cache-line sharing can be avoided by aligning CPU and SDMA buffers to 64-byte boundaries. Aligning buffers to 64-byte boundaries will result in wasted space, however it ensures that the CPU and SDMA buffers will not have partial segments which overlap into the same L1D cache line.

```
/** Pseudo code only **/
Uint8 *SDMA_BUFF, *CPU_BUFF;

/* 64-byte aligned allocation Option 1 */
SDMA_BUFF = malloc( (Int32) (( SDMA_BUFF_SIZE + 63)/64) * 64 );
CPU_BUFF = malloc( (Int32) ((CPU_BUFF_SIZE + 63)/64) * 64 );

SDMA_BUFF = (Uint8 *) ( (Int32) SDMA_BUFF & ~63 );
CPU_BUFF = (Uint8 *) ( (Int32) CPU_BUFF & ~63 );

/* 64-byte aligned allocation Option 2 with BIOS Call */
SDMA_BUFF = MEM_alloc( IRAM, SDMA_BUFF_SIZE, 64 );
CPU_BUFF = MEM_alloc( IRAM, CPU_BUFF_SIZE, 64 );
```

Method 3 Manage access to a 64-byte boundary aligned buffer that is shared between CPU and SDMA by implementing a semaphore and forcing cache writeback operations if there are CPU writes. With this method, the semaphore ensures that there is clear ownership of the buffer between CPU and SDMA, and the CPU manages cache coherence by using explicit cache writeback operations.

```
/** Pseudo code only **/
/* Example with EDMA as the external master */
EDMA_ISR() {
/* EDMA releases ownership of buffer */
SEM_post(SyncSemaphore);
return;
}

main() {
       while(COND) {
    /* CPU waits for ownership of buffer */
    SEM_pend(SyncSemaphore);

    /**********************/
    /*** CPU Processing ***/
    /**********************/
    /* Cache writeback for shared block */
    /* Buffer must be 64-byte aligned */
    BCACHE_wbInv( blockPtr, blockSize, WAIT );

    /* Initiate EDMA */
    EDMA_Event_Generate();
}
}
```

Method 4 Do not allow SDMA to access L2 RAM. SDMA can use buffers in L1D RAM or Shared RAM instead of L2 RAM.

Method 5 Configure the entire L2 RAM as cache. Critical peripheral data can be accessed in L1D RAM or Shared RAM instead of L2 RAM.

Method 6 Configure the entire L2 RAM as normal SRAM (no cache).

Method 7 Configure the entire L1D RAM as normal SRAM (no cache).

Advisory 3.0.22 *USB 2.0 On-The-Go (OTG) Session Request Protocol (SRP) Is Not Supported*

Revision(s) Affected 3.0 and earlier

Details The USB 2.0 On-The-Go (OTG) Session Request Protocol (SRP) allows a USB-peripheral to request the USB-host to enable Vbus and start a session. On this device, the SRP protocol is not supported.

The OTG Host Negotiation Protocol (HNP), which allows USB-devices to swap roles between host and peripheral, is supported.

Workaround(s) None

Advisory 3.0.24 *USB0 PLL Mean Frequency Can Drift Across Large Temperature Swings*

Revision(s) Affected 3.0 and earlier

Details Under conditions in which the device is subjected to large variations in operating temperatures, the USB0 PLL temperature compensation circuitry does not have enough margin to guarantee compensation for PLL drift across all temperature ranges.

As a result, the mean frequency generated by the USB0 (USB 2.0 OTG) PHY PLL will begin to drift (relative to the expected 480 Mbps) when the temperature of the device is subjected to large swing from the original temperature in which the USB0 PHY was most recently calibrated (initialized).

Once the onset of the PLL drift occurs, the mean frequency will continue to drift outside the expected frequency eventually resulting in failure of USB packet reception and/or transmission. This break in transmission will continue until the USB0 PHY is recalibrated during a USB0 PHY Reset.

If the device is not exposed to large variations in temperature relative to the temperature at which the USB0 PHY was most recently initialized, the temperature compensation circuitry is expected to provide the proper compensation to prevent the mean PLL frequency from losing lock and beginning to drift.

More specifically, this advisory is most applicable in applications where the device is expected to operate outside the commercial temperature space (0˚C-90˚C). TI has identified a point-to-point device temperature range of 0˚C-65˚C in which there is very high confidence in which the compensation circuitry will properly compensate for all variations in temperature provided that the USB0 PHY was most recently initialized (calibrated) within this same temperature range.

Operating outside the 0˚C-65˚C temperature range increases the susceptibility of the device to experience PLL drift, but does not mean that the application will always experience a failure in USB transmission.

Root Cause The Voltage Controlled Oscillator (VCO) Compensation circuitry local to the USB0 PHY was not designed with a large enough range to compensate for all variations in temperature across the specified operating range of the device.

How to Most Easily Reproduce the Issue: Reproduction of this issue can most easily be accomplished by the following steps:

1. Allowing the unit to soak in an ambient temperature of -35˚C until the device temperature reaches approximately the same temperature.
2. Power up the device and provide the necessarily software programming in order to invoke the USB Signal Quality Test Pattern.
3. Using a USB 2.0 Certified Test Platform, execute the USB signal quality test procedure across the following temperature set points. -35˚C, 0˚C, +35˚C, +70˚C. Record the measured mean frequency by the compliance software.

NOTE: The set points can be varied to obtain finer temperature resolution of when the PLL begins to drift a per platform basis. The above temperature profile is provided for reference.

Workaround(s)

When a break in transmission is detected, USB0 traffic can be recovered by a software reset of the USB0 PHY. A PHY reset implies recalibration of the PHY PLL at the reset temperature. The system has not been observed to reliably recover on its own. A PHY reset also implies re-enumeration of all devices. There is no way to recalibrate the USB0 PHY without a re-enumeration.

In order to invoke the recovery mechanism (that is a USB0 PHY reset) one needs to determine when the issue is present. One such approach is to look for an absence of USB0 Core interrupts over a specified time window. This window should be optimized for the expected USB traffic based upon the application.

As an additional safeguard, an application can also intentionally schedule pre-determined USB PHY resets at specific temperature points if operation over a broad range is expected.

Here is an example of one way to power cycle the USB0 PHY via the Chip Configuration 2 Register in the System Configuration (SYSCFG) Module:

```
#define CFGCHIP2  *((volatile unsigned int *) 0x01C14184)
    #define USBPHY_PHYPDWN 0x00000200

Void phy_reset(void) {
        CFGCHIP2 |= USBPHY_PHYPDWN;        /* Power down the USB PHY */

        mdelay(1);                         /* Wait 500ms */

        CFGCHIP2 &= ~USBPHY_PHYPDWN;       /* Power up the USB PHY */
}
```

Advisory 3.0.25 *USB0: CPU gets Stale Receive Data from the Data Buffer located in External Memory*

Revision(s) Affected 3.0 and earlier

Details When CPPI DMA completes a receive data transaction it posts a write to the Rx data buffer located in external memory, posts a write to update the descriptor located in external memory, and raises an interrupt to CPU. When the system load is high, the posted writes to DDR may not be complete before the CPU receives the interrupt. In this case, the CPU would fetch stale receive data from the Rx data buffer located in external memory.

Workaround(s) Initialize the datalength descriptor field to zero. CPPI DMA updates this field after the completion of an RX DMA operation with the actual number of bytes received. In the ISR (actually in a deferred call context), poll this field until it becomes a non-zero value to ensure data buffer has been updated with actual data. The descriptor buffer write is posted after the data buffer write, so waiting for the descriptor field to be updated ensures the data buffer has been updated. Since this workaround involves deferred procedure calls (whose schedule can be delayed depending on OS load), the latency sensitive application (like ISO Audio) might be affected by delay in notification to the application.

Advisory 3.0.26 *USB0: Early DMA Completion in DMA Receive Mode and More Than One Endpoint is Transferring Data*

Revision(s) Affected 3.0 and earlier

Details The erroneous short packet status can be detected on current endpoint and XDMA closes the Rx transfer in current endpoint. When more than one endpoint have been processed, if one of the endpoints has a short packet, then the short packet status is broadcasting to all endpoints.

This results in premature completion of a Rx descriptor in generic RNDIS CPPI DMA mode.

Workaround(s) The workaround involves monitoring transfer data size before and after transferring and reconfiguring data transfer size by software if the before and after size is different. Software must keep tracking every endpoint data transferring size. When DMA completion interrupt is received, software checks size difference. If the size is not equal, software requests the remaining data.

Advisory 3.0.27 *USB0: DMA Hung up in Frequent Teardowns*

Revision(s) Affected 3.0 and earlier

Details Teardown receive DMA is not working perfectly. This happens when a teardown is initiated by software during the endpoint is still active. Frequent teardown results in XDMA hung up situation.

Workaround(s) Software should make sure that DMA does not get to an unknown state during teardown by disabling the DMAEN bit in the RXCSR register. After this the teardown procedure can be initiated. Software should also add 250 ms delay during teardown.

3 Silicon Revision 2.1 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 2.1 of the *OMAP-L137*.

3.1 Usage Notes for Silicon Revision 2.1

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.

Silicon revision 2.1 applicable usage notes have been found on a later silicon revision. For more details, see Section 2.1 *Usage Notes for Silicon Revision 3.0.*

3.2 Silicon Revision 2.1 Known Design Exceptions to Functional Specifications

Silicon revision 2.1 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2 *Silicon Revision 3.0 Known Design Exceptions to Functional Specifications*

Table 3. Silicon Revision 2.1 Advisory List

Title	Page
Advisory 2.1.23 —Digital I/O Buffers Age Prematurely	29

Advisory 2.1.23 *Digital I/O Buffers Age Prematurely*

Revision(s) Affected 2.1 and earlier

Details

The 3.3 V digital I/O buffers on the device exhibit accelerated aging under use conditions with heavy switching activity. As a result, the recommended Power-On Hours (POH) listed in the device-specific data manual may not apply in all cases. For examples of representative lifetimes, see Examples of Representative Lifetimes (Power-On Hours) .

In a typical use case, the EMIFB clock pin (EMB_CLK) may be impacted due to the high switching rate and may eventually lead to SDRAM-related failures.

NOTE: The information in Examples of Representative Lifetimes (Power-On Hours) and Examples of Representative Lifetimes (Power-On Hours) is provided solely for user convenience and ***does not*** extend or modify the warranty provided under any terms and conditions, including TI's Standard Terms and Conditions of Sale for Semiconductor Products.

Examples of Representative Lifetimes (Power-On Hours)

NOMINAL DVDD VOLTAGE (V)	I/O SWITCHING FREQUENCY (MHz)	DPPM[1] ESTIMATE	FAIL RATE (%)	LIFETIME ESTIMATES (YEARS) HOURS (HRS) PER DAY OF USE					PERIPHERAL
				2 HRS	3 HRS	7 HRS	12 HRS	24 HRS	
3.30	133	1000	0.1	8.9	5.9	2.5	1.5	0.7	EMIFB
		2500	0.25	>15	11.7	5.0	2.9	1.5	
		5000	0.5	> 15	> 15	7.3	4.3	2.1	
3.30	16.94	1000	0.1	> 15	> 15	> 15	11.6	5.8	McASP
		2500	0.25	> 15	> 15	> 15	> 15	11.5	
		5000	0.5	> 15	> 15	> 15	> 15	> 15	

(1) Defective parts per million.

Accelerated aging of buffers depends on I/O switching frequency and I/O voltage. Prolonged high frequency switching and operating at higher voltages causes buffer performance to degrade more rapidly.

Workaround(s) The following methods should be used to prolong the lifetime of the device.

METHOD 1

The 3.3 V output signals should only toggle when required for system functionality. The options are to tristate output buffers when not in use, or to run at the minimum frequency required for the specific application.

METHOD 2

Reduce the I/O voltage, DVDD.

Note: Do not reduce I/O voltage on devices already deployed or showing premature aging effects. To see the effects of premature aging on output signals, See the IO Buffer Premature Aging Assessment Wiki page.

Examples of Representative Lifetimes (Power-On Hours) depicts the subsequent performance improvements by reducing the I/O voltage.

Examples of Representative Lifetimes (Power-On Hours)

NOMINAL DVDD VOLTAGE (V)	I/O SWITCHING FREQUENCY (MHz)	DPPM[1] ESTIMATE	FAIL RATE (%)	LIFETIME ESTIMATES (YEARS) HOURS (HRS) PER DAY OF USE					PERIPHERAL
				2 HRS	3 HRS	7 HRS	12 HRS	24 HRS	
3.15	133	1000	0.1	> 15	> 15	7.7	4.5	2.2	EMIFB
		2500	0.25	> 15	> 15	> 15	8.8	4.4	
		5000	0.5	> 15	> 15	> 15	12.9	6.4	
3.15	16.94	1000	0.1	> 15	> 15	> 15	> 15	> 15	McASP
		2500	0.25	> 15	> 15	> 15	> 15	> 15	
		5000	0.5	> 15	> 15	> 15	> 15	> 15	

(1) Defective parts per million.

The Recommended Operating Conditions range for DVDD Supply Voltage, I/O, 3.3V, has been expanded to allow for a minimum voltage of 3.0V **for Silicon Revision 2.1 and 2.0 parts with the "3V" marking only** (for more details on the die symbolization and device revision codes, see Section 1.2, *Package Symbolization and Revision Identification*).

Recommended Operating Conditions

		MIN	NOM	MAX	UNIT
DVDD	Supply voltage, I/O, 3.3V	**3.0**	3.3	3.45	V

The Absolute Maximum Ratings requirement for the Input voltage ranges, V_I I/O, 3.3V (Steady State) has been modified.

Absolute Maximum Ratings Over Operating Case Temperature Range

Input voltage ranges	V_I I/O, 3.3V (Steady State)	–0.3V to DVDD + **0.350**V

See the IO Buffer Premature Aging Assessment Wiki page for more details on the implementation of the above workarounds.

4 Silicon Revision 2.0 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 2.0 of the *OMAP-L137*.

4.1 Usage Notes for Silicon Revision 2.0

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.

Silicon revision 2.0 applicable usage notes have been found on a later silicon revision. For more details, see Section 2.1 *Usage Notes for Silicon Revision 3.0.*

4.2 Silicon Revision 2.0 Known Design Exceptions to Functional Specifications

Some silicon revision 2.0 applicable advisories have been found on a later silicon revision. For more details, see

- Section 2.2, *Silicon Revision 3.0 Known Design Exceptions to Functional Specifications*
- Section 3.2, *Silicon Revision 2.1 Known Design Exceptions to Functional Specifications*

Table 4. Silicon Revision 2.0 Advisory List

Advisory 2.0.20 *Intermittent Boot Failures*

Revision(s) Affected 2.0 and earlier

Details

For affected silicon revisions, the DSP initiates the system boot sequence when the device is released from reset. To prepare the ARM for the user, the DSP will first initialize the ARM reset vector table with an infinite “idle” loop.

The ARM reset vector table is located in the ARM’s local RAM, however the ARM local RAM can only be accessed by two bus masters: ARM and PRU0. Therefore, the DSP must program PRU0 to copy the desired reset vector table into the ARM’s local RAM.

The PRU instructions are located inside of an instruction RAM (IRAM) which is initialized by the DSP during ROM boot (see Figure 4). After the instructions are stored to IRAM, the PRU is reset and enabled to execute its instructions. In this case, the PRU is instructed to initialize the ARM reset vector table.

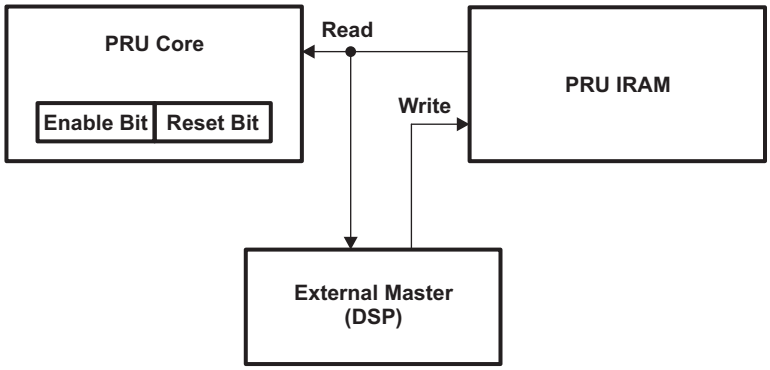


Figure 4. PRU and DSP Block Diagram

When the device is first powered-on, the read bus from the PRU IRAM is not initialized and will contain random values (see Figure 5). Under unpredictable circumstances, the random value on the read bus may resemble a reserved instruction which can be interpreted by the PRU when the core is reset and not enabled.

If the PRU core executes this reserved instruction, it will not be able to properly execute the first functional op-code in the PRU IRAM when the core is later enabled. In this fail state, the PRU will never acknowledge to the DSP that the reset vector table was successfully initialized and the DSP will be stuck in a polling loop waiting for the PRU to complete its task.

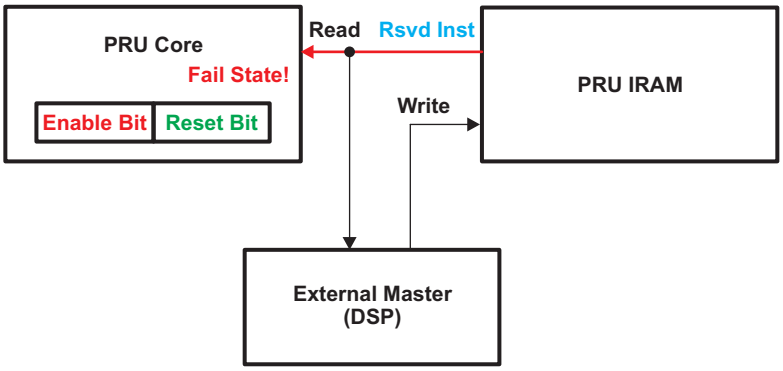


Figure 5. Boot Failure on Power-On

Although the PRU core execution is stuck, the PRU IRAM read bus is now initialized with a non-reserved instruction that was fetched from the IRAM by the PRU core (see

Figure 6). If a secondary reset is provided to the device (either POR or WARM), the PRU will be able to execute its functional instructions as expected.

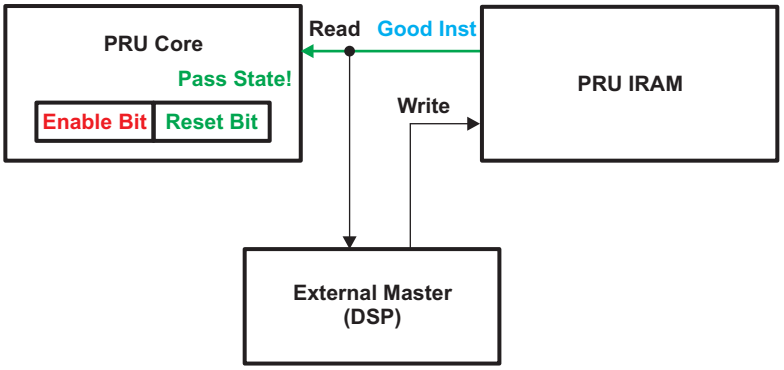


Figure 6. Secondary Reset

Note that in order to recover from this fail state with a secondary reset, the DSP must be allowed to execute its boot ROM up to the point where the PRU has fetched a known instruction from the PRU IRAM. The approximate count of 15k cycles into the boot ROM is sufficient.

The 15k clock cycle count does not include the 6192 clock cycles required to complete a device POR reset (see Figure 7). With a 24MHz crystal, the first RESET signal must be asserted high for at least 883us (or approximately 1ms).

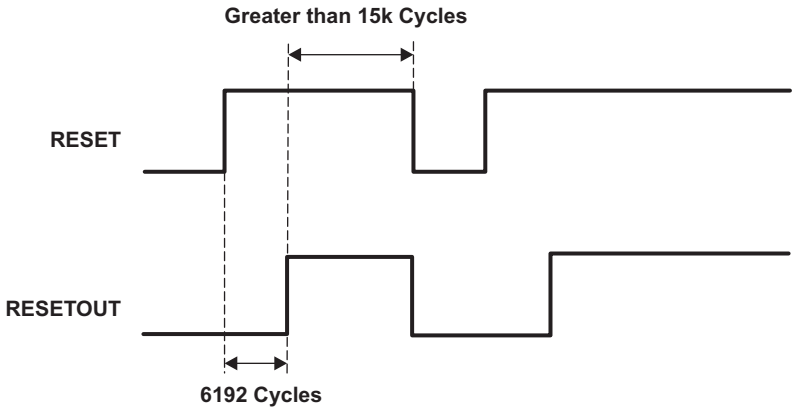


Figure 7. First POR Reset Timing

The long-term solution for this problem is to update the DSP boot ROM with a new PRU initialization sequence that is immune to the described fail mode (see Figure 8):

1. Before resetting the PRU, the DSP will perform a read-back-verify of the PRU IRAM so that the IRAM read bus will be initialized with a known and safe state.
2. The PRU will be reset and enabled in the same clock cycle by using a single register write so that the core does not have the opportunity to interpret reserved instructions.
3. The DSP will write additional, non-critical op-codes at the beginning of the PRU IRAM so that the PRU can self-recover even if it interprets a reserved instruction.

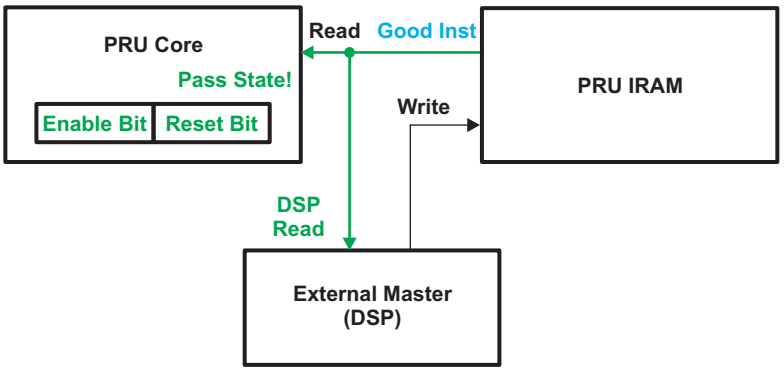


Figure 8. New Initialization Sequence

The following symptoms are all observable for this fail mode at power-on:

1. The RESETOUT signal toggles as expected 6192 cycles after the RESET signal is asserted high.
2. The device produces no boot-mode related activity and the user's boot program will not be loaded or executed.
3. Connecting to the DSP through JTAG emulation will show that the DSP is stuck in a loop inside of the DSP boot ROM (0x00700000 – 0x007FFFFF memory space).
4. A subsequent reset (either POR or WARM) which is initiated at least 1ms after the first POR reset is asserted high will always produce a successful boot.
5. Following a secondary reset, the device will function as expected without fail until the device is powered-off again.

Workaround(s)

Modify the target board so that the affected device is given a secondary reset on power-up as shown in Figure 6. Two example methods are described in the sections that follow.

Although secondary resets are compatible with future silicon revisions, they are not required for devices where the root cause has been fixed via an updated DSP boot ROM. In order to reduce BOM costs, board designers may want to route a reset signal bypass path so that the workaround circuit can be depopulated on future PCB builds.

- ❑ Use a reset supervisor device that includes a watchdog timeout function so that the reset supervisor will issue a secondary reset if the device fails to boot. The watchdog should be serviced with a device signal that is controlled by software. Options for servicing the watchdog timeout include GPIO, unused clock sources such as OBSCLK or a periodic output peripheral like TIMER and ePWM.

Some TPS382x reset supervisors include a watchdog function (shown in Figure 9).

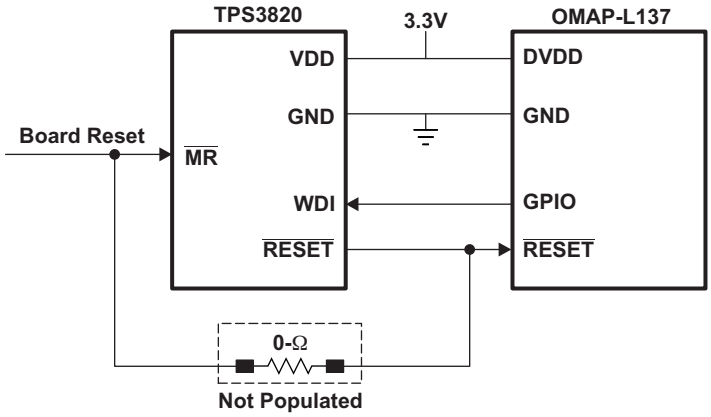


Figure 9. Reset Supervisor with Watchdog Function

The watchdog supervisor workaround is easy to implement, however the watchdog timeout period may exceed application boot-up time requirements. For example, the TPS3820 has a typical watchdog timeout period of 200ms. The second workaround can speed up the reset process.

- Implement a logic-based secondary reset circuit which is timed using RC components. For the circuit shown in Figure 10, a single board reset control signal can trigger three logic transitions in a dual XOR gate device.

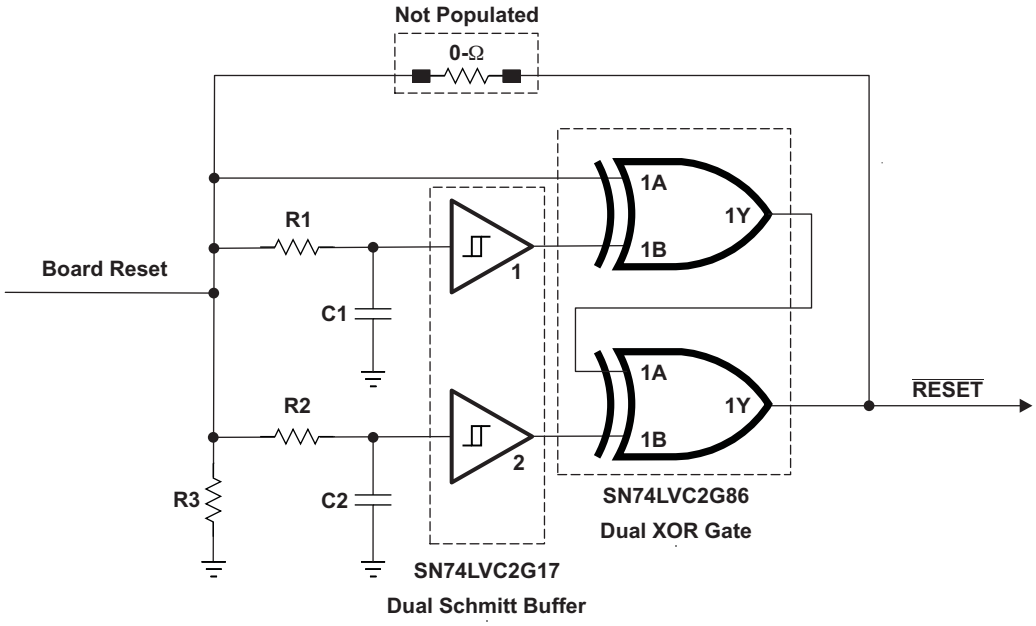


Figure 10. RC-Timed Secondary Reset

This is possible because each RC load connected to the board reset control signal can output a different rising-edge waveform. With increasing RC load, the resulting control signal will reach the Schmitt buffers' Vih level at a later point in time. Figure 11 shows the relationship between the board reset signal and the RESET signal produced by the circuit. The blue and green lines represent the voltage as seen by the Schmitt buffers. The output voltage of a charging RC circuit is defined as: Vo = Vi * (1 - e^[-t / RC])

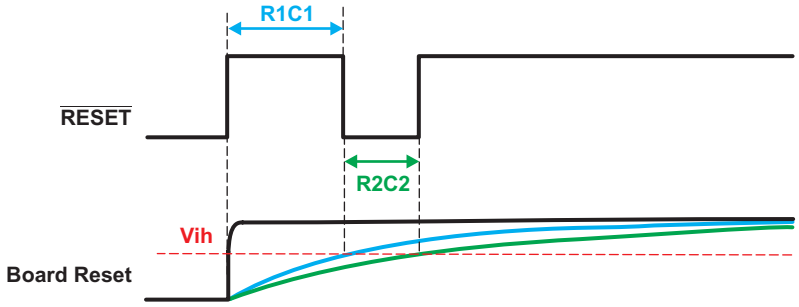


Figure 11. RESET Signal vs Board Reset

Given ideal conditions, a 3.3V board reset signal, and an input buffer Vih of 1.4V, the following set of component values would generate an initial RESET high period (R1C1 region) of approximately 2ms and a RESET low period (R2C2 region) of approximately 0.5ms:

- R1 = 36k, C1 = 100nF
- R2 = 45k, C2 = 100nF
- R3 = 450k

When implementing this workaround, some important aspects should be kept in mind:

(a) The dual Schmitt buffer is included because the dual XOR gate has an input rise-time requirement that is violated by the RC circuits.

(b) The Board Reset signal must meet the XOR gate input rise-time requirement and must provide enough output current to charge the RC circuits to the target Vih level.

(c) It is critical for the Vih level of the two input buffers to be very close together so only single-device buffers should be considered for this circuit (such as the 2-in-1 dual Schmitt buffer device used in this example).

(d) Variations in the electrical characteristics of the circuit components may produce waveforms that deviate from ideal calculations.

(e) The sole purpose of the R3 pull-down resistor is to discharge the RC components before the board reset signal is driven high. Therefore, the value selected for R3 should be sufficiently large enough to not interfere with the RC circuits as they are charging.

5 Silicon Revision 1.1 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 1.1 of the *OMAP-L137* device .

5.1 Usage Notes for Silicon Revision 1.1

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.

Silicon revision 1.1 applicable usage notes have been found on a later silicon revision. For more details, see Section 2.1, *Usage Notes for Silicon Revision 3.0*.

5.1.1 RTC Standby Power Consumption Is Elevated if the Module Is Not Configured Correctly

On Silicon Revision 1.1 and earlier, the RTC module is designed with the ability to keep time while the rest of the device is power cycled off and on. This ability is achieved by placing the RTC in its own power domain and isolating it from the device reset signal.

When the CVDD supply is powered down, the RTC_CVDD supply will experience elevated standby power consumption because of leakage between the RTC and core power domains. The RTC module includes circuitry that eliminates the leakage paths between the two domains when the SPLITPOWER bit is set to 1 in the control register (CTRL). The SPLITPOWER bit is a write-only bit that will always read back 0. Therefore, typical read-modify-write sequences should not be used when writing to the CTRL register because the SPLITPOWER bit will be cleared back to 0.

Also note that the SPLITPOWER bit has a default value of 0 after RTC module reset, and the only reset available to the RTC module is a software reset, therefore RTC is in an indeterminate state when the RTC_CVDD supply is first powered on. The RTC module should be reset, and the SPLITPOWER bit should be set to 1 before placing the device in a CVDD powered down standby state. The SPLITPOWER bit is permanently set to 1 inside the RTC module beginning with Silicon Revision 2.0 of the device.

5.1.2 SYSCFG: Possible Race Condition When Using KICK Registers

On Silicon Revision 1.1 and earlier, when two or more threads are simultaneously accessing the SYSCFG registers, there is the potential for one thread to lock the SYSCFG registers while another thread is still accessing them. There is no hardware semaphore to prevent this from occurring.

For example, the race condition can occur in the following situation

1. Thread 1 unlocks the SYSCFG register by writing to the KICK registers
2. An interrupt occurs and Thread 2 unlocks the SYSCFG registers as well
3. Thread 2 finishes and locks the SYSCFG registers
4. Thread 1 is locked out of the SYSCFG registers and is unable to complete its task

To prevent the SYSCFG lockout race condition, the application should unlock the SYSCFG registers via the KICK registers and leave them permanently unlocked.

Starting with silicon revision 2.0, the KICK registers will be disabled and the SYSCFG registers will be permanently accessible. Writes to the disabled KICK registers will have no effect.

5.2 Silicon Revision 1.1 Known Design Exceptions to Functional Specifications

Some silicon revision 1.1 applicable advisories have been found on a later silicon revision. For more details, see

- Section 2.2, *Silicon Revision 3.0 Known Design Exceptions to Functional Specifications*
- Section 3.2, *Silicon Revision 2.1 Known Design Exceptions to Functional Specifications*

- Section 4.2, *Silicon Revision 2.0 Known Design Exceptions to Functional Specifications*

Table 5. Silicon Revision 1.1 Advisory List

Advisory 1.1.1 *ARM Data Cache in Write-Back Mode Is Not Functional: Must Use Write-Through or Non-Cached Mode*

Revision(s) Affected 1.1 and earlier

Details The ARM926 subsystem allows data memory regions to be write-back cachable, write-through cachable, or non-cached. On this device revision, the *Write-Back* mode is not functional; therefore, *Write-Through* or *Non-Cached* mode must always be used.

Workaround(s) Only the *Write-Through* or *Non-Cached* mode can be used. *Write-Through* mode is preferred for better performance. The cache operation is controlled using the C and B bits in page or section descriptors. For operation in *Write-Through* mode, the C and B bits (bits 3:2 in the descriptor) must be set to a value of 10b.

The following is example code using a section descriptor to create a table entry for the first 1MB of external SDRAM on EMIFB as write-through cachable:

```
LDR  r1,  SDRAM0_ADDR            ; table offset for SDRAM0 region
LDR  r2,  SDRAM0_DATA            ; descriptor pattern for SDRAM0 region
STR  r2,  [r0, r1, LSL#2]        ; store the table entry at TTB base + table
offset * 4
SDRAM0_ADDR        .word    0x00000C00
SDRAM0_DATA        .word    0xC0000CFA
```

For more information on ARM data cache modes and how to configure them, refer to the *ARM926EJ-S™ Technical Reference Manual* available at www.arm.com/documentation. Chapter 4 of the *ARM926EJ-S™ Technical Reference Manual* provides details about cache operations on the ARM926EJ-S processor.

Section descriptor: A section descriptor provides the base address of a 1MB block of memory. Figure 12 shows the format of a section descriptor.

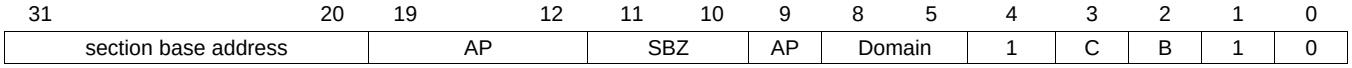


Figure 12. Section Descriptor

Table 6 shows the Section Descriptor bit assignments. Table 7 shows the Page Table C and B bit settings for the DCache.

Table 6. Section Descriptor Bits

BITS	DESCRIPTION
31:20	Form the corresponding bits of the physical address for a section.
19:12	Always written as 0.
11:10	Specify the access permissions for this section.
9	Always written as 0.
8:5	Specify one of the 16 possible domains, held in the domain access control register, that contain the primary access controls.
4	Should be written as 1, for backwards compatibility.
3:2	Indicate if the area of memory mapped by this section is treated as write-back cacheable, write-through cacheable, noncached buffered, or noncached nonbuffered.
1:0	These bits must be 10 to indicate a section descriptor

Table 7. Page Table C and B Bit Settings for the DCache

C BIT	B BIT	DESCRIPTION	ARM926EJ-S BEHAVIOR
0	0	Noncacheable, nonbufferable	DCache disabled. Read from external memory. Write as a nonbuffered store(s) to external memory. DCache is not updated.
0	1	Noncacheable, bufferable	DCache disabled. Read from external memory. Write as a buffered store(s) to external memory. DCache is not updated.
1	0	Write-through	DCache enabled: • Read hit - Read from DCache • Read miss - Linefill • Write hit - Write to the DCache, and buffered store to external memory • Write miss - Buffered store to external memory
1	1	Write-back	DCache enabled: • Read hit - Read from DCache • Read miss - Linefill • Write hit - Write to the DCache only • Write miss - Buffered store to external memory

Advisory 1.1.2 *USB0: CPPI Receive Starvation Interrupt*

Revision(s) Affected 1.1 and earlier

Details When an endpoint is enabled for receive transfer(s) that will be serviced via DMA and data has been received prior to allocating DMA resource, the DMA will generate a starvation interrupt to notify the application a lack of resource (starvation) in anticipation that the application will furnish the required resource. The CPPI DMA is supposed to generate a single interrupt. But, in this case it continues generating interrupt periodically, until application furnishes a resource. In some use cases, it has been observed that the application may desire to differ the time as to when to service the starvation request due to the CPU handling other urgent task(s). Since the DMA keeps on generating the starvation interrupt periodically and there exists no capability to mask the starvation interrupt at the USB controller level, the CPU is forced either to fully service the DMA interrupt as it is received or disable all USB interrupt at the CPU level. Disabling the entire USB interrupt might not be the desired option since the CPU needs to be aware of other USB interrupts that are more critical.

Workaround(s) The user can remove the associated channel number entry from the DMA scheduler Tables (scheduler array) for that endpoint when not expecting data from a host. Whenever data transfer is initiated by the host, the endpoint interrupt will be generated by the USB controller, which can be used as an indication for an application to secure required resources prior to adding the DMA channel entry for the endpoint onto the scheduler array.

Advisory 1.1.3 *SPI: Internally Generated SPI Clock Is Not 50% Duty Cycle*

Revision(s) Affected 1.1 and earlier

Details When the SPI is in master mode, the generated SPICLK signal is derived from the internal SPI module clock. This SPICLK signal duty cycle is not 50% when the SPIFMTn.PRESCALE is set to an even number.

With an even prescale value, the falling edge of the SPICLK is delayed 1-2 ns regardless of the SPICLK frequency. Therefore, the high side is wider than the low side.

Workaround(s) Use the SPIFMTn.PRESCALE with an odd value if possible.

Advisory 1.1.5 *Under Specific Conditions, SDMA Activity Can Corrupt the L1D Cache and L2 RAM*

Revision(s) Affected 1.1 and earlier

Details

Note: DMA refers to all non-CPU requests. SDMA refers to external system DMA master requests handled via the Slave Direct Memory Access port.

The C674x Megamodule uses a two-way set associative cache for L1D. This means that every physical memory location in the system has two possible set/way locations in the cache where it can reside. See TMS320C674x DSP Megamodule Reference Guide, Literature Number - SPRUFK5 for more information on the L1D cache architecture. Updated (dirty) values in L1D cache are not written back to external memory until cache activity evicts a cache-line (victim write-back) or a write-back is requested by the application.

An L1D cache-line corruption event occurs when all of the following conditions are met:

1. L1D cache evicts a dirty line (Line A) while allocating a new line (Line B) in the same set/way (cache Lines A and B consist of 64-bytes each). In order for this to happen, the following will have taken place:
   (a) Line A was previously read by CPU because L1D is a read-allocate cache,
   (b) Line A is dirty because its value was modified by CPU, and
   (c) Line B is read by CPU
2. Both Line A and Line B are associated with L2 RAM, and
3. While the original L1D victim write-back from condition (1) is in progress, the SDMA performs both:
   (a) a read or write operation to Line A in L2 RAM and
   (b) a write operation to Line B in L2 RAM.

If all of the above conditions are met, the L2 RAM data associated with the Line A victim writeback will become corrupt. Additionally, the Line B data originating from the SDMA write will also become corrupt in L1D cache. Figure 13 shows an example scenario of L1D cache and L2 RAM corruption.

	INITIAL STATE	STEP 1	STEP 2	STEP 3
C674x CPU Activity		CPU Request Line B	Request Pending	
SDMA Activity			SDMA Requests Writes to both LineA and LineB	
L1D Cache Line Status	Dirty Line A in L1D Cache	Dirty Line A in L1D Cache	IN-FLUX	Corrupt
Line A L2 SRAM	Old Data	Old Data	Old Data	Corrupt
Line B L2 SRAM	Old Data	Old Data	Old Data	NEW DATA

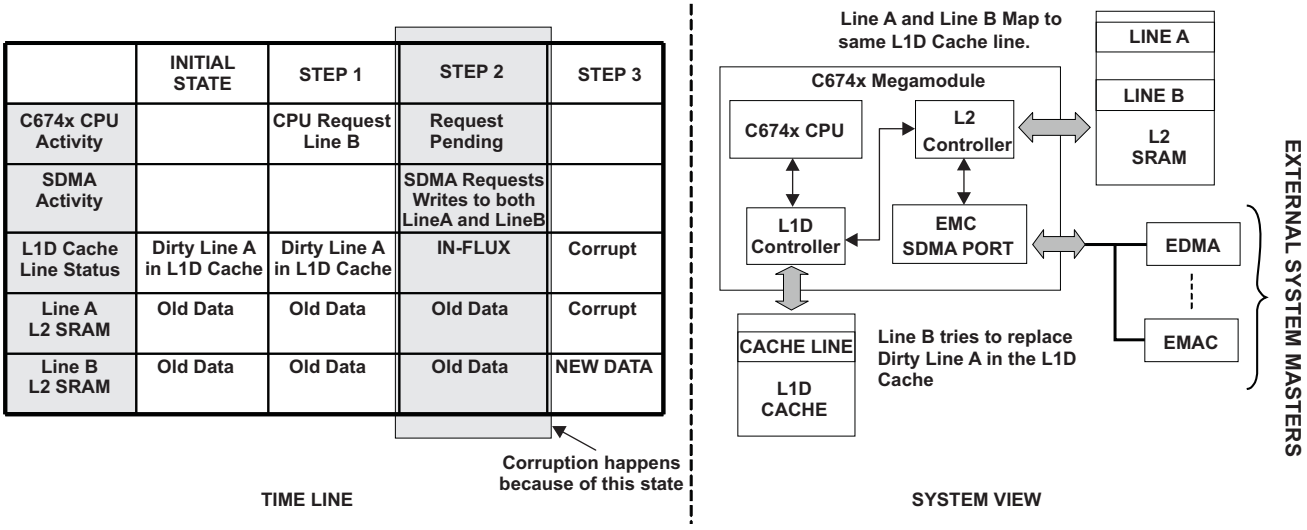


Figure 13. Example of L1D Cache and L2 RAM Corruption

Workaround(s)

Method 1

In cases where buffer access will not be shared between CPU and SDMA, unintended CPU/SDMA cache-line sharing can be avoided by aligning CPU and SDMA buffers to 64-byte boundaries. Aligning buffers to 64-byte boundaries will result in wasted space, however it ensures that the CPU and SDMA buffers will not have partial segments which overlap into the same L1D cache line.

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

            Uint8 *SDMA_BUFF, *CPU_BUFF;

            /* 64-byte aligned allocation Option 1 */
            SDMA_BUFF = malloc( (Int32) (( SDMA_BUFF_SIZE + 63)/64) * 64 );
            CPU_BUFF  = malloc( (Int32) ((CPU_BUFF_SIZE  + 63)/64) * 64 );

            SDMA_BUFF = (Uint8 *) ( (Int32) SDMA_BUFF & ~63 );
            CPU_BUFF  = (Uint8 *) ( (Int32) CPU_BUFF  & ~63 );

            /* 64-byte aligned allocation Option 2 with BIOS Call */
            SDMA_BUFF = MEM_alloc( IRAM, SDMA_BUFF_SIZE, 64 );
            CPU_BUFF  = MEM_alloc( IRAM, CPU_BUFF_SIZE,  64 );
```

Method 2

Manage access to a 64-byte boundary aligned buffer that is shared between CPU and SDMA by implementing a semaphore and forcing cache writeback operations after CPU writes. With this method, the semaphore ensures that there is clear ownership of the buffer between CPU and SDMA, and the CPU manages cache coherence by using explicit cache writeback operations.

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

            /* Example with EDMA as the external master */
            EDMA_ISR() {
                /* EDMA releases ownership of buffer */
                SEM_post(SyncSemaphore);
                return;
            }
  main() {
                while(COND) {
                    /* CPU waits for ownership of buffer */
                    SEM_pend(SyncSemaphore);

                    /**********************/
                    /*** CPU Processing ***/
                    /**********************/

                    /* Cache writeback for shared block */
                    /* Buffer must be 64-byte aligned              */
                    BCACHE_wbInv( blockPtr, blockSize, WAIT );

                    /* Initiate EDMA */
                    EDMA_Event_Generate();
                }
            }
```

Method 3

Configure the entire L2 RAM as cache. Critical peripheral data can be accessed in L1D RAM or L3 RAM instead of L2 RAM.

Method 4

Do not allow SDMA to access L2 RAM. SDMA can use buffers in L1D RAM or L3 RAM instead of L2 RAM.

Method 5

Do not configure L1D memory as cache - use the entire address space as RAM.

Advisory 1.1.8 *USB 1.1 Phy Does Not Have Internal Pull Down on DP/DM Lines Enabled*

Revision(s) Affected 1.1 and earlier

Details USB 1.1 requires a 15 KΩ pull down on the DP/DM lines. The USB 1.1 Phy does not include the internal pull down.

Workaround(s) The 15 KΩ pull downs should be added to the DP/DM lines externally to the device.

Advisory 1.1.9 *USB2.0 (USB0) and USB1.1 (USB1) Power Supply*

Revision(s) Affected 1.1 and earlier

Details There is the potential for the 1.8V and 3.3V USB power supplies to experience a latchup condition. The potential for latchup on these supplies is related to the power-on voltage slew rate and the likelihood for latchup with a rapid voltage supply slew rate gradually increases over the device lifetime.

Workaround(s) To prevent a latch up condition from occurring over the life of the device, the following recommendations should be followed:

- USB0_VDDA18 and USB1_VDDA18 power supply rails must have greater than 1ms (10%-90%) slew
- USB0_VDDA33 and USB1_VDDA33 power supply rails must have less than 100-mv pk-pk noise
- USB0_VBUS power supply rail must have less than 100-mv pk-pk noise sustained and must have greater than 1ms (10%-90%) slew

Advisory 1.1.10 *EMIFB: AC Timings Differ From Data Manual Specifications*

Revision(s) Affected 1.1 and earlier

Details The timing parameters in Table 8 and Table 9 differ from those specified in the *OMAP-L137 C6000 DSP+ARM Processor* data manual (literature number SPRS563 or later). Table 8 list the AC timing parameters that should be used on silicon revision 1.1 and earlier.

Workaround(s) During PCB board design and layout, the AC timings specified in Table 8 and Table 9 should be considered when designing interfaces to the EMIFB.

Table 8. Timing requirements Over Recommended Operating Conditions

NO.	PARAMETER		MIN	MAX	UNIT
19	$t_{su(DV-CLKH)}$	Input setup time, read data valid on EMB_D[31 :0] before EMB_CLK rising	1.26		ns

For the parameter $t_{OH(CLKH-DQMIV)}$, Table 9 and Table 10 are valid under the conditions described in their respective notes:

Table 9. Switching Characteristics Over Recommended Operating Conditions

NO.	PARAMETER		MIN	MAX	UNIT
6	$t_{OH(CLKH-DQMIV)}$	Output hold time, EMB_CLK rising to $\overline{\text{EMB\_WE\_DQM[3:2]}}$ invalid	0.8 (1) (2)		ns

(1) This timing requires CVDD = 1.15V ± 1% and DVDD = 3.3V ± 2%.
(2) This timing parameter ONLY applies to the signals $\overline{\text{EMB\_WE\_DQM[3:2]}}$; $\overline{\text{EMB\_WE\_DQM [1:0]}}$ timings are as shown in the device data manual.

Table 10. Switching Characteristics Over Recommended Operating Conditions

NO.	PARAMETER		MIN	MAX	UNIT
6	$t_{OH(CLKH-DQMIV)}$	Output hold time, EMB_CLK rising to $\overline{\text{EMB\_WE\_DQM[3:2]}}$ invalid	0.7 (1) (2)		ns

(1) This timing requires CVDD= 1.2V +/- 1% and DVDD= 3.3V +/- 2%.
(2) This timing parameter ONLY applies to the signals $\overline{\text{EMB\_WE\_DQM[3:2]}}$; $\overline{\text{EMB\_WE\_DQM [1:0]}}$ timings are as shown in the device data manual.

Advisory 1.1.11 *Electrostatic Discharge Charged-Device Model Performance*

Revision(s) Affected 1.1 and earlier

Details The ESD to Charge Device model (CDM) is rated as passing the 300V level instead of the 500V level. The OSCOUT pin passed 300V testing but failed starting at 400V. The RSV2 pin had marginality to 500V. All of the other pins are rated as passing the 500V level.

Advisory 1.1.16 *Vil on 3.3V LVCMOS Input Buffers*

Revision(s) Affected 1.1 and earlier

Details

The input buffers on the device have shown timing sensitivity to the logic-low input voltage that can cause changes to the AC input timings. Due to this issue, input voltages must be driven at or below 0.4V to limit impact on AC timings.

The timing effect on the input buffers is dependent on the Vil level:

- Case 1: For signals driven with **Vil < = 0.2V**, the input timings will be unaffected.
- For signals driven with **0.2V < Vil < = 0.4V**, there may be as much as 0.5 ns degradation to input timings.

This issue applies only to 3.3V LVCMOS inputs or IOs used as inputs. Signals operated at 1.8V are not affected.

Workaround(s)

Although there is no specific workaround, the following recommendations can be used to help prevent this issue:

- Minimize loads as much as possible, especially DC loads that could cause the Vil to rise. **Point-to-point (single-load) connections are unlikely to be affected.**
- Falling edges should transition as rapidly as possible (so the signal passes through the 0.2V point as early as possible). Heavily loaded nodes resulting in degraded fall times may require drivers to provide rapid input edges.

Advisory 1.1.17 *DSP SDMA/IDMA: Unexpected Stalling and Potential Deadlock Condition*

Revision(s) Affected 1.1 and earlier

Details

Note: This advisory is not applicable if DSP L2 memory is configured as 100% cache or L2 RAM is not accessed by IDMA or SDMA during run-time.

The C674x 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 C674x Megamodule.

The MDMA interface is typically used for CPU/cache accesses to memory beyond the Level 2 (L2) memory level. These accesses include cache line allocates, write-backs, and non-cacheable loads and stores to/from system memories. The cacheable memories external to the C674x Megamodule are listed in Table 11.

Table 11. Cacheable External Memory Resources

External Memory	Address Range
Shared Ram	0x8000 0000 – 0x8001 FFFF
EMIFA	0x4000 0000 – 0x67FF FFFF
EMIFB	0xC000 0000 – 0xCFFF FFFF

The SDMA interface allows other DMA master peripherals (listed in Table 12) to access Level 1 Data (L1D), Level 1 Program (L1P), and L2 RAM DSP memories.

Table 12. DMA Master Peripherals

Peripheral	Group
EDMA TC0 RD	A
EDMA TC0 WR	B
EDMA TC1 RD	C
EDMA TC1 WR	D
EMAC	E
USB1	E
USB0	F
UHPI	F
ARM	G

The C674x Megamodule has an L1D cache and L2 cache both implementing write-back data caches– it keeps updated values for external memory in cache for 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. 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 C674x Megamodule and its MDMA and SDMA ports, see the TMS320C674x Megamodule Reference Guide (literature number SPRUFK5).

Ideally, the MDMA (dashed-dotted line in Figure 14) and SDMA/IDMA paths (dashed lines in Figure 14) operate independently with minimal interference. Normally MDMA accesses may stall for extended periods of time due to expected system level delays (for example, bandwidth limitations). However, when using L2 as RAM, SDMA and IDMA accesses to L2/L1 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 device. 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 L2. Unexpected SDMA/IDMA stalls to L1 happen only when they are pipelined behind L2 accesses. Additionally, the deadlock scenario will be avoided if there are no SDMA accesses to L2.

Figure 14 is provided for illustrative purposes and is incomplete because of 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 DSP).

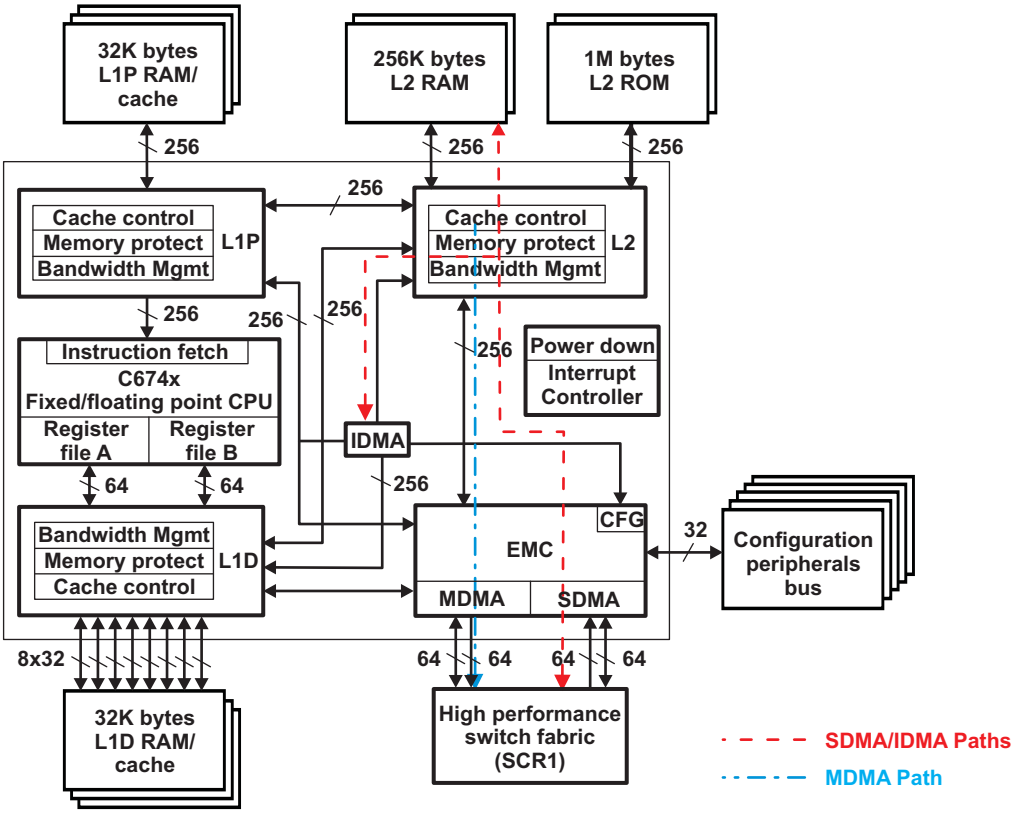


Figure 14. C674x Megamodule

The duration of the SDMA/IDMA stalls depend on the quantity/characteristics of the L1/L2 cache and the MDMA traffic in the system. Therefore, it is difficult to predict if stalling will occur and for how long.

IDMA/SDMA stalling and any system impact is most likely in systems with excessive context switching, L1/L2 cache miss/victim traffic, and heavy access to external memory.

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 poor peripheral throughput.

1. Determine if the transfer that is missing the real-time deadline is accessing L2 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 L2 memory (for example, EDMA transfer to/from L2 from/to a UART, HPI block transfer to/from L2). If there are no SDMA transfers to/from L2, 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.
   (b) If not all L2 SDMA transfers can be moved to L1D memory, temporarily direct some of the transfers to memory in Table 1 and keep the rest in L1D memory. There should be no L2 SDMA transfers.

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.

A deadlock situation may arise if the following sequence of events occurs:

Step 1: A DMA master from any group (listed in Table 12) issues a write command to the DSP's SDMA, and a DMA master from the same group issues a subsequent write command to cacheable memory outside of the C674x Megamodule (listed in Table 1). All write commands pass through Switched Central Resource 1 (SCR1). For more details on SCRs, see the device Technical Reference Guide SPRUH93.

Step 2: The DSP's SDMA asserts itself as not ready and is unable to accept the write data from Step 1, and a cache line writeback is initiated from DSP memory to the same cacheable memory from Step 1. The cache line writeback command also passes through SCR1.

With the above scenario, it is possible for SCR1 to order the write commands from Step 1 in front of the write commands from Step 2. Due to the MDMA/SDMA blocking behavior, the SDMA commands from Step 2 will be waiting for the MDMA traffic from Step 1 to finish, resulting in a deadlock situation at SCR1. Figure 15 is provided for illustrative purposes and is incomplete because of simplification.

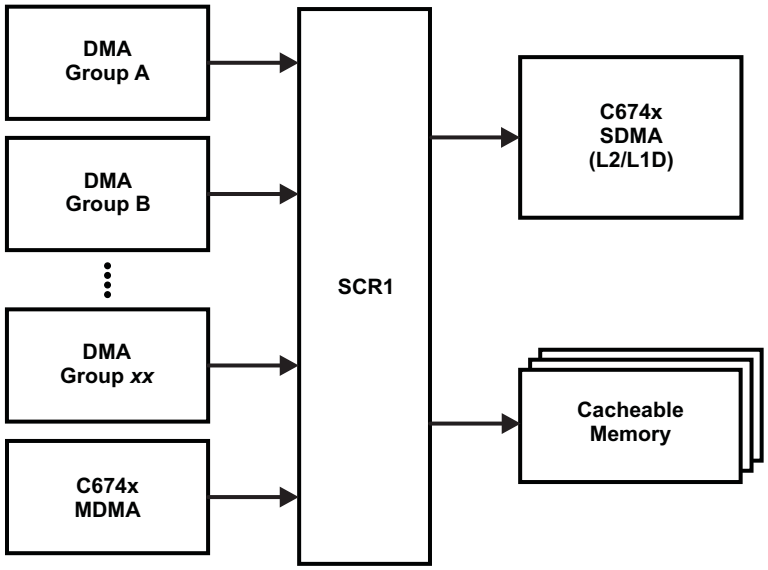


Figure 15. SCR1 System Interconnect

Workaround(s)

Entirely eliminate IDMA/SDMA stalling and potential for a deadlock condition using one of the following two methods:

1. Configure the entire L2 RAM as 100% cache (for example, move all data buffers from L2 to L1D or other memory). Note: Some throughput degradation is expected when the buffers are moved out to external memo
2. Eliminate all IDMA/SDMA access to L2 RAM when IDMA/SDMA stalling would have an impact by performing one of the following:
   (a) Constrain each DMA master group to perform writes to either DSP memory space or external memory space, but not to both, or

(b) Force each DMA master group to complete pending write commands to either DSP memory space or cacheable memory space before initiating writes to a different destination. Pending write commands from DMA masters are forced to complete when the DMA master initiates a read from the same destination memory. Note that in the case of off-chip memory, a read command only forces the completion of write commands within a 2KB-aligned window.

6 Silicon Revision 1.0 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 1.0 of the *OMAP-L137* device.

6.1 Usage Notes for Silicon Revision 1.0

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.

Silicon revision 1.0 applicable usage notes have been found on a later silicon revision. For more details, see Section 4.1, *Usage Notes for Silicon Revision 3.0* and see Section 5.1, *Usage Notes for Silicon Revision 1.1*.

6.2 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications

Some silicon revision 1.0 applicable advisories have been found on a later silicon revision. For more details, see

- Section 2.2, *Silicon Revision 3.0 Known Design Exceptions to Functional Specifications*
- Section 3.2, *Silicon Revision 2.1 Known Design Exceptions to Functional Specifications*
- Section 4.2, *Silicon Revision 2.0 Known Design Exceptions to Functional Specifications*
- Section 5.2, *Silicon Revision 1.1 Known Design Exceptions to Functional Specifications*

Table 13. Silicon Revision 1.0 Advisory List

Advisory 1.0.12 *SPI Communication Failure for Data Receipt*

Revision(s) Affected 1.0

Details There is a potential SPI communication failure on data receipt. When the communication failure occurs, the data received in the SPIBUF register is not correct. The data error pattern is not consistent between failures. The failure impacts both master and slave mode.

This issue is caused by un-matched delays between the clock and data paths of the shift register. The non-matched delays cause the clock edge to be detected earlier than the appropriate data. This results in inconsistent/incorrect receive data being copied into the SPIBUF register.

Workaround(s) There is no recommended workaround.

Adjusting the following settings may allow you to find a working region:

- Adjust the SPI settings of prescale, phase, and polarity
- Increase the core VDD to a range between 1.25V and 1.32V
- Decrease the CPU operating frequency

Increasing the core VDD or slowing down the CPU operating frequency is ideal to work across a variety of conditions. Although the adjustments recommended above may be used to find a working setup and allow further development to take place, these adjustments cannot be guaranteed to alleviate the issue across devices, hardware platforms, and temperature.

Revision History

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

This silicon errata revision history highlights the technical changes made to SPRZ291H revision to make it SPRZ291I revision.

Scope: Applicable updates relating to the device have been incorporated. Added Silicon Revision 3.0 device-specific information.

Revision History

SEE	ADDITIONS/MODIFICATIONS/DELETIONS
Section 2.1	Added the following new usage note to Usage Notes for Silicon Revision 3.0: • Section 2.1.1, McASP: Inactive Slot Usage Note Updated/Changed Advisory 3.0.18 text from "...System Reference Guide - SPRUG84..." to "Technical Reference Manual SPRUH92"
Section 2.2	Added the following new advisories to Silicon Revision 3.0 Known Design Exceptions to Functional Specifications: • Advisory 3.0.25, USB0: CPU gets Stale Receive Data from the Data Buffer located in External Memory • Advisory 3.0.26, USB0: Early DMA Completion in DMA Receive Mode and More Than One Endpoint is Transferring Data • Advisory 3.0.27, USB0: DMA Hung up in Frequent Teardowns
Section 5.2	Updated/Changed Advisory 1.1.17 text from "...System Reference Guide SPRUG84" to "...Technical Reference Manual SPRUH92"

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