# OMAP-L138 C6000 DSP+ARM® Processor Silicon Revisions 2.3, 2.1, 2.0, 1.1 and 1.0

# Silicon Errata

TEXAS INSTRUMENTS

Literature Number: SPRZ301M
June 2009–Revised March 2014

Silicon Errata
SPRZ301M – June 2009 – Revised March 2014

# OMAP-L138 C6000 DSP+ARM® Processor Silicon Revisions 2.3, 2.1, 2.0, 1.1 and 1.0

## 1 Introduction

This document describes the known exceptions to the functional specifications for the OMAP-L138 C6000 DSP+ARM® Processor . For more detailed information, see the *OMAP-L138 C6000 DSP+ARM® Processor* data manual (literature number: SPRS586).

### *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, **OMAPL138BZWT4**]. 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 Revision Identification

The silicon revision is identified by a code marked on the package. Please see the figure Device nomenclature in the data manuals Packaging information section for coding details. Table 1 describes the relationship between device revision code and silicon revision.

**Table 1. OMAP-L138 Device Revision Codes**

| DEVICE REVISION CODE (xx) | SILICON REVISION | COMMENTS |
|---|---|---|
| E | 2.3 | |
| B | 2.1 | Silicon Revision 2.1 can be differentiated from Revision 2.0 by a top side '21' marking on the package. See Figure 1 for details.<br>Additionally the ROM ID code may be used to distinguish Revision 2.1 from Revision 2.0 |
| B | 2.0 | - |
| A | 1.1 | - |
| (blank) | 1.0 | - |

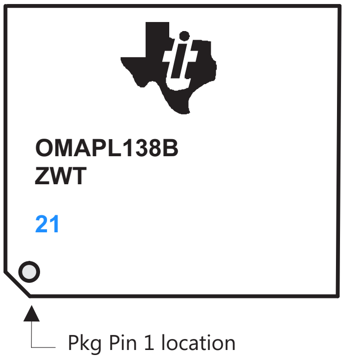


**Figure 1. Device Markings**

# 2 Silicon Revision 2.3 Usage Notes and Known Design Exceptions to Functional Specifications

The advisories may not always be enumerated in sequential order and hence some numbers may not appear in the document.

## *2.1 Usage Notes for Silicon Revision 2.3*

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 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 a 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, interrupt will be generated on each USB packets as opposed to receiving a single interrupt on the completion of a transfer.

### 2.1.2 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 interrupt 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.

### 2.1.3 LCDC: Underflow During Initialization

On all silicon revisions 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 DDR2/mDDR Controller 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.

The DDR2/mDDR Controller hardware automatically schedules refresh commands to the SDRAM memory. Therefore, it is not possible for the user/application code to schedule DDR2/mDDR Controller refresh commands to prevent them from being initiated during LCDC start up. This means that it is not possible to prevent a DDR2/mDDR Controller refresh occurrence during the start up of the LCDC.

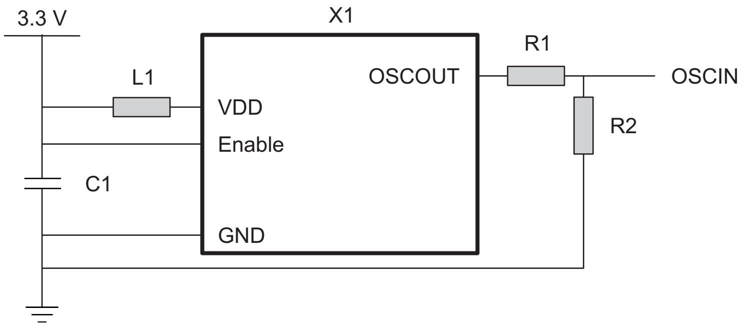
Software should poll the FUF bit field in the LCD_STAT register to check if an error condition has occurred or service the interrupt if FUF_EN is enabled when FUF occurs. If the FUF bit field has been set to 1, this will indicate an underflow condition has occurred and then the software should execute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

This problem may occur if the LCDC FIFO threshold size ( LCDDMA_CTRL[TH_FIFO_READY] ) is left at its default value after reset. Increasing the FIFO threshold size will reduce or eliminate underflows. Setting the threshold size to 256 double words or larger is recommended.

### 2.1.4 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 2 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 2. 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.
- Disable the DLL REFCLK signal in the DDR EMIF PHY. This prevents the DLL used by the DDR PHY from dynamically tracking glitches on the input clock. This can be done after normal DDR initialization by setting the following bit in the DDR PHY Control Register (0xB00000E4):

```
// Configure DDR PLL
   Set_DDRPLL_150MHz();
// Configure DDR timings
   DEVICE_DDR2Config(150);
// Minimum 600 MCLK cycle delay (allow master DLL to lock)
   Delay_600();
// Perform dummy DDR read
   volatile unsigned int k=0;
   ...
   k = *(volatile unsigned int*) (0xC0000000);
// Disable DLL REFCLK
   DRPYC1R |= 0x00002000;
```

- The processor should be provided as much power supply decoupling as is practical and placed as close to the processor as possible.
- Follow the entire DDR interface implementation requirements in the device datasheet.
- 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.

### 2.1.5 DDR2/mDDR Controller: mDDR Usage Note

On all silicon revisions, some mDDR memories designed with Status Register Read (SRR) support may be incompatible with the OMAPL138 device. These mDDR memories misinterpret certain initialization sequence commands sent by the DDR2/mDDR controller following a power-on-reset or a cold reset (by asserting the RESET pin) of the device. This results in the mDDR memory becoming unresponsive as it is stuck in the SRR state instead of returning to the IDLE state.

To ensure correct initialization of these types of mDDR memories, two consecutive mDDR initialization sequences have to be sent by the DDR2/mDDR controller to the mDDR memory after a power-on-reset or a cold reset.

If users rely on the ROM Bootloader (RBL) to perform DDR2/mDDR initializations, they can generate an Application Image Script (AIS) boot image with two consecutive DDR2/mDDR controller configuration sequences, with the help of the AIS GUI utility or the AIS command-line utility.

**Using the GUI utility (AISgen_d800k008.exe, version 1.13 or later):**
Use the GUI tool to generate the AIS boot image with the required configuration and choose the ‘mDDR with SRR support’ option under the Memory Type section.

**Using the command-line utility (HEXAIS_OMAP_L138.exe):**
With the command-line utility, users should manually add a second set of PLL1 and DDR2/mDDR controller configuration sequence at the end of the first DDR2/mDDR controller configuration sequence in the .ini script.

Note: The *Using the OMAP-L132/L138 Bootloader* application report (literature number: SPRAB41) provides a link to the install package for the AIS tool which includes the following in the install directory: prebuilt patch files, the GUI AIS generation tool, command-line AIS generation tool, and example .ini files.

If the RBL is not used to initialize the DDR2/mDDR memory, two consecutive PLL1 and DDR2/mDDR controller configuration sequences should be incorporated into the user code that configures the DDR2/mDDR controller.

Boot images generated using the above methods can be used for all compatible mDDR memories. However, it is recommended to check with the memory vendor before doing so.

Note: Not all mDDR memories designed with Status Register Read (SRR) support exhibit the stuck-in-SRR-state behavior.

The set of two consecutive mDDR initialization sequence is only required during after a power-on-reset or a cold reset. It is not required after a soft reset (from Power and Sleep Controller or Watch-Dog Timer) of the DDR2/mDDR controller.

### 2.1.6 McASP: Inactive Slot Usage Note

On all silicon revisions, in any McASP serializer configured a transmit serializer with an n-slot TDM, data transfer can fail if both of the conditions below are true:

1. one or more time slots within the n-slot TDM are configured as inactive and
2. EDMA is used to transfer data to McASP

If the conditions mentioned above exists, either the transmit operation may fail to start with the XDATA bit in the Transmit Status Register (XSTAT) set or if the transmit operation has started, random underrun errors may occur breaking the data transfer operation.

To ensure correct McASP transmit operation with EDMA triggered data transfers, all time slots in an n-slot TDM should be configured as active slots. 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, all five slots (0 to 4) should be configured as active in the Transmit TDM Time Slot Register (XTDM). In this example the remaining time slots (slot 5 onwards) can be configured as inactive. The EDMA configuration and user application should account for the transfer of extra data to the McASP for slots 3 and 4.

## 2.2 Silicon Revision 2.3 Known Design Exceptions to Functional Specifications

The advisories may not always be enumerated in sequential order and hence some numbers may not appear in the document.

**Table 2. Silicon Revision 2.3 Advisory List**

## Advisory 2.3.1 *DMA Access to L2 RAM Can Stall When DMA and C674x CPU Command Priorities Are Equal*

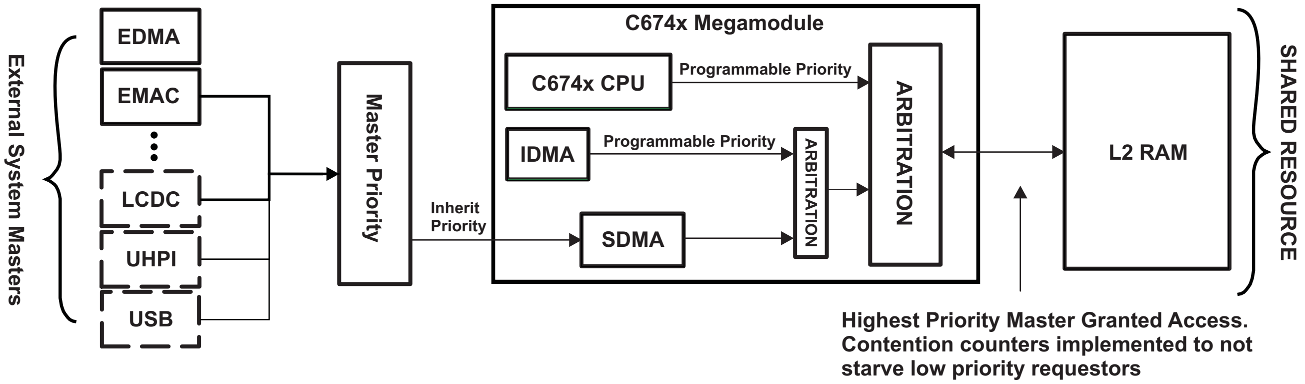
**Revision(s) Affected** 2.3 and earlier

**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 3 shows a high level diagram of the arbitration scheme used for L2 RAM requests.



Highest Priority Master Granted Access. Contention counters implemented to not starve low priority requestors

**Figure 3. Priority Arbitration Scheme for L2 RAM**

**Workaround(s)** Configure DMA and CPU requests to different priority levels. 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 2.3.3 *USB0: Extraneous RESET Interrupt*

**Revision(s) Affected** 2.3 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 2.3.4 *EMIFA: Asynchronous Memory Timeout Error Persistence*

**Revision(s) Affected** 2.3 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 EMIF 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 EMIF 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 EMIF 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 2.3.5 *A Single CHIPINTn Interrupt Event Will Register Multiple Times in the DSP Event Combiner Module (ECM)*

**Revision(s) Affected** 2.3 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] 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 2.3.6 *Potential USB2.0 Soft Reset Timing Violation*

**Revision(s) Affected** 2.3 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 through the Power and Sleep Controller (PSC)
2. Perform a soft USB reset
3. Wait for the USB soft reset bit to clear
4. Disable the USB controller module clock through the PSC
5. Configure the USB PHY parameters
6. Enable the PHY
7. Enable the USB controller module clock through the PSC

## Advisory 2.3.9 — *Vil on Dual-Voltage LVCMOS Input Buffers Operated at 3.3V*

**Revision(s) Affected** 2.3 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 to 0.5V or below on all dual-voltage LVCMOS input signals (signals associated with supplies DVDD1833_A, DVDD1833_B or DVDD1833_C).

Vil driven at or below 0.4V will cause no timing degradation. Vil driven up to 0.5V will cause up to 750 ps degradation in input timings.

The following datasheet parameters will be affected by Vil driven up to 0.5V. Their values adjusted for 0.75 ns degradation are shown.

### Table 3. Timing Requirements for the EMIFA SDRAM Interface

| NO. | PARAMETER | | 1.2V | | UNIT |
|---|---|---|---|---|---|
| | | | MIN | MAX | |
| 19 | $t_{\text{su(EMA\_DV-EM\_CLKH)}}$ | Input Setup time, read data valid on EMA_D[31:0] | 2.75 | | ns |

### Table 4. Timing Requirements for the EMIFA Asynchronous Memory Interface

| NO. | PARAMETER | | 1.2V | | UNIT |
|---|---|---|---|---|---|
| | | | MIN | MAX | |
| 12 | $t_{\text{su(EMDV-EMOEH)}}$ | Setup time, EMA_D[31:0] valid before $\overline{\text{EMA\_OE}}$ high | 3.75 | | ns |

### Table 5. Timing Requirements for McASP0

| NO. | PARAMETER | | | 1.2V | | UNIT |
|---|---|---|---|---|---|---|
| | | | | MIN | MAX | |
| 6 | $t_{\text{h(ACLKRX-AFSRX)}}$ | Hold time, AFSR/X input after ACLKR/X [1] | AHCLKR/X ext input | 1.15 | | ns |
| | | | AHCLKR/X ext output | 1.15 | | |
| 8 | $t_{\text{h(ACLKRX-AXR)}}$ | Hold time, AXR0[n] input after ACLKR/X [1] [2] | AHCLKR/X ext input | 1.15 | | |
| | | | AHCLKR/X ext output | 1.15 | | |

[1] McASP0 ACLKXCTL.ASYNC=1: Receiver is clocked by its own ACLKR0

[2] McASP0 ACLKXCTL.ASYNC=0: Receiver is clocked by transmitter's ACLKX0

### Table 6. Switching Characteristics for McASP0

| NO. | PARAMETER | | | 1.2V | | UNIT |
|---|---|---|---|---|---|---|
| | | | | MIN | MAX | |
| 13 | $t_{\text{d(ACLKRX-AFSRX)}}$ | Delay time, ACLKR/X transmit edge to AFSX/R output valid [1] | ACLKR/X ext input | 3.5 | | ns |
| | | | ACLKR/X ext output | 3.5 | | |
| 14 | $t_{\text{d(ACLKX-AXRV)}}$ | Delay time, ACLKX transmit edge to AXR output valid [1] | ACLKR/X ext input | 3.5 | | |
| | | | ACLKR/X ext output | 3.5 | | |
| 15 | $t_{\text{dis(ACLKX-AXRHZ)}}$ | Disable time, ACLKR/X transmit edge to AXR high impedance following last data bit | ACLKR/X ext | 3.5 | | |

[1] McASP0 ACLKXCTL.ASYNC=1: Receiver is clocked by its own ACLKR0

### Table 7. Timing Requirements for McBSP0

| NO. | PARAMETER | | | 1.2V | | UNIT |
|---|---|---|---|---|---|---|
| | | | | MIN | MAX | |
| 5 | $t_{su(FRH-CKRL)}$ | Setup time, external FSR high before CLKR low | CLKR ext | 5.5 | | ns |
| 7 | $t_{su(DRV-CKRL)}$ | Setup time, DR valid before CLKR low | CLKR ext | 5.5 | | |
| 10 | $t_{su(FXH-CKXL)}$ | Setup time, external FSX high before CLKX low | CLKR ext | 5.5 | | |

### Table 8. Switching Characteristics for McBSP0

| NO. | PARAMETER | | | 1.2V | | UNIT |
|---|---|---|---|---|---|---|
| | | | | MIN | MAX | |
| 4 | $t_{d(CKRH-FRV)}$ | Delay time, CLKR high to internal FSR valid | CLKR ext | 2.75 | | ns |
| 9 | $t_{d(CKXH-FXV)}$ | Delay time, CLKX high to internal FSX valid | CLKR ext | 2.75 | | |
| 13 | $t_{d(CKXH-DXV)}$ | Delay time, CLKX high to DX valid | CLKR ext | 2.75 + D1 [1] | | |

[1] Extra delay from CLKX high to DX valid applies only to the first data bit of a device, if and only if DXENA = 1 in SPCR.
If DXENA = 0, then D1 = D2 = 0
If DXENA = 1, then D1 = 6P, D2 = 12P

### Table 9. Switching Characteristics for McBSP1

| NO. | PARAMETER | | | 1.2V | | UNIT |
|---|---|---|---|---|---|---|
| | | | | MIN | MAX | |
| 4 | $t_{d(CKRH-FRV)}$ | Delay time, CLKR high to internal FSR valid | CLKR ext | 3.25 | | ns |
| 9 | $t_{d(CKXH-FXV)}$ | Delay time, CLKX high to internal FSX valid | CLKR ext | 3.25 | | |
| 13 | $t_{d(CKXH-DXV)}$ | Delay time, CLKX high to DX valid | CLKR ext | 3.25 + D1 [1] | | |

[1] Extra delay from CLKX high to DX valid applies only to the first data bit of a device, if and only if DXENA = 1 in SPCR.
If DXENA = 0, then D1 = D2 = 0
If DXENA = 1, then D1 = 6P, D2 = 12P

### Table 10. Timing Requirements for Universal Parallel Port (uPP)

| NO. | PARAMETER | | 1.2V | | UNIT |
|---|---|---|---|---|---|
| | | | MIN | MAX | |
| 4 | $t_{su(STV-INCLKH)}$ | Setup time, CHn_START valid before CHn_CLK high | 4.75 | | ns |
| 6 | $t_{su(ENV-INCLKH)}$ | Setup time, CHn_ENABLE valid before CHn_CLK high | 4.75 | | |
| 8 | $t_{su(DV-INCLKH)}$ | Setup time, CHn_DATA/XDATA valid before CHn_CLK high | 4.75 | | |
| 10 | $t_{su(DV-INCLKL)}$ | Setup time, CHn_DATA/XDATA valid before CHn_CLK low | 4.75 | | |
| 19 | $t_{su(WTV-INCLKL)}$ | Setup time, CHn_WAIT valid before CHn_CLK high | 4.75 | | |

### Table 11. Timing Requirements for Video Port Interface (VPIF)

| NO. | PARAMETER | | 1.2V | | UNIT |
|---|---|---|---|---|---|
| | | | MIN | MAX | |
| 1 | $t_{su(VDINV-VKIH)}$ | Setup time, VP_DINx valid before VP_CLKIN0/1 high | 4.75 | | ns |

**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 2.3.10 *ARM Interrupt Controller Vector Size Register (VSR) Initialization*

**Revision(s) Affected** 2.3 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 2.3.12 *A Single CHIPINTn Interrupt Event Can Register Multiple Times in the AINTC*

**Revision(s) Affected** 2.3 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 System Reference Guide 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 2.3.13 *Incorrect Masking of the C674x CSR:SAT Bit*

**Revision(s) Affected** 2.3 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 2.3.17 *SDMA Activity Can Corrupt L1D When L2 Is Configured as Mixed/Cache/SRAM*

**Revision(s) Affected** 2.3 and earlier

**Details**

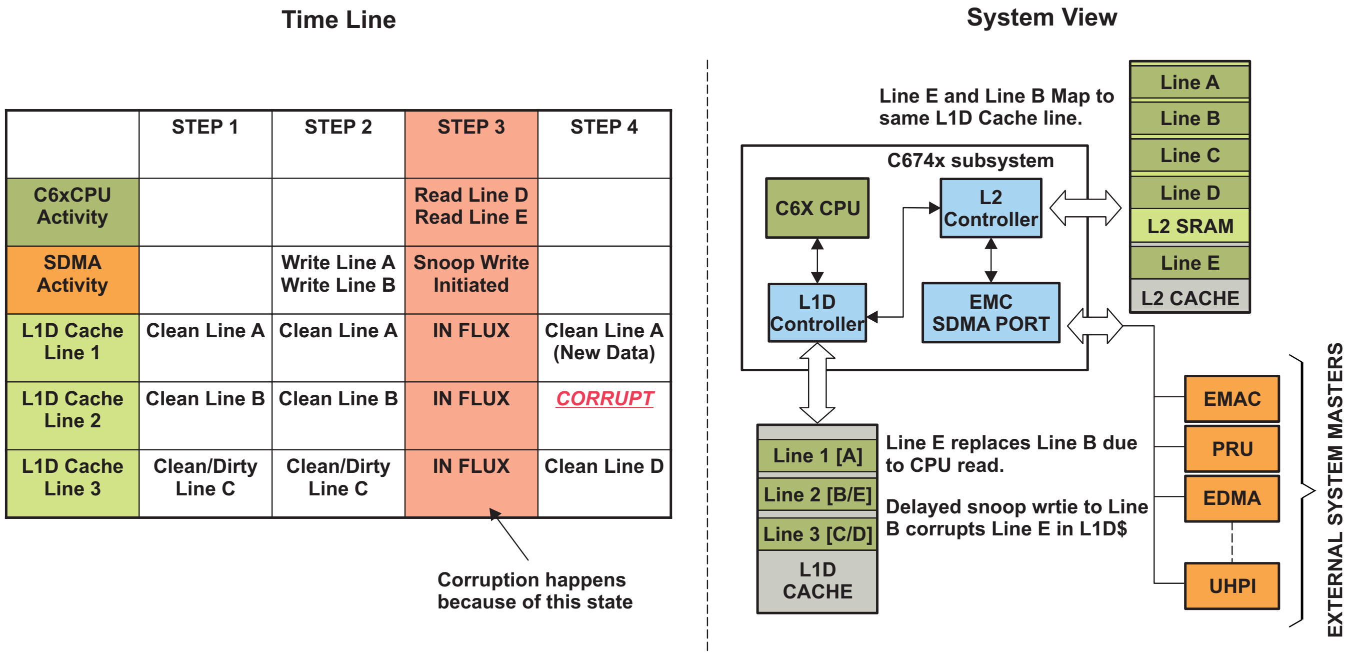
**Note:** SDMA refers to all non-CPU requests to the EMC SDMA (Slave Direct Memory Access) port (see Figure 4). SDMA requests are defined as external system bus master requests handled via this port.

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 4):

1. L1D cache Lines 1, 2, and 3 have the following characteristics:
   - Line 1 is associated with L2 SRAM (Line A in Figure 4), 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 4), 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.



| | STEP 1 | STEP 2 | STEP 3 | STEP 4 |
|---|---|---|---|---|
| C6xCPU Activity | | | Read Line D<br>Read Line E | |
| SDMA Activity | | Write Line A<br>Write Line B | Snoop Write Initiated | |
| L1D Cache Line 1 | Clean Line A | Clean Line A | IN FLUX | Clean Line A (New Data) |
| L1D Cache Line 2 | Clean Line B | Clean Line B | IN FLUX | CORRUPT |
| L1D Cache Line 3 | Clean/Dirty Line C | Clean/Dirty Line C | IN FLUX | Clean Line D |

**Figure 4. 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 2.3.18 *DVDD18 Can Pull Up to 2.7V When Using Dual-Voltage IOs at 3.3V*

**Revision(s) Affected** 2.3 and earlier

**Details** A condition can occur during the device power supply ramp in which the DVDD18 supply, which should be at 1.8V nominal, can be pulled up on-chip by the DVDD3318_A, DVDD3318_B or DVDD3318_C supplies operated at 3.3V. For the sake of this description, DVDD3318_x means any combination of DVDD3318_A, DVDD3318_B or DVDD3318_C.

**Important Note:** The 3.3V DVDD3318_x supplies must not be driven to 0V during zones A – E or high current capable of damaging the device may occur. The 3.3V DVDD3318_x supplies should not be driven during this time.

A normal example power-supply ramp is shown below:

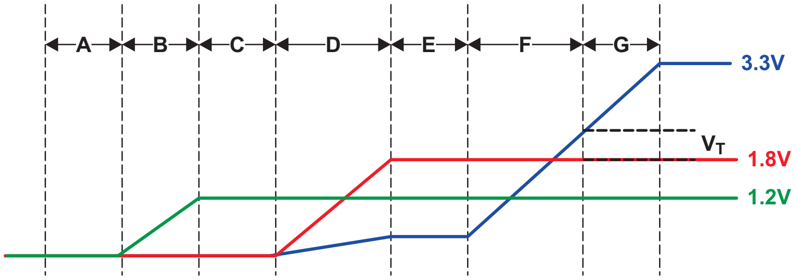


**Figure 5. Normal power-supply ramp**

- A. All device supplies are undriven
- B. The 1.2V supplies are ramped to their nominal levels
- C. Potential delay between supply ramps (not required)
- D. The 1.8V supplies (specifically DVDD18) are ramped to their nominal levels. During the DVDD18 ramp there will be some minor drift up in the undriven DVDD3318_x supplies. This is normal and is not a problem.
- E. Potential delay between supply ramps (not required)
- F. DVDD3318_x is ramping but is less than DVDD18 + $V_T$
- G. DVDD3318_x is ramping but is greater than DVDD18 + $V_T$

When the fault condition occurs, the supplies behave as shown below:

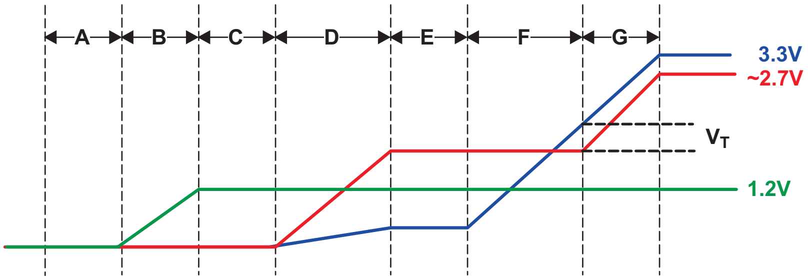
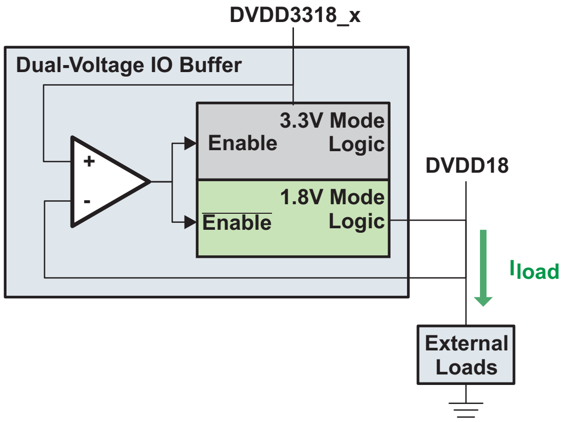


**Figure 6. Faulty power-supply ramp**

The behavior is the same until zone G. When the DVDD3318_x supply exceeds DVDD18 by a transistor $V_T$, the DVDD18 supply begins to be pulled up by the DVDD3318_x supply and follows the DVDD3318_x supply by DVDD3318_x- $V_T$. Since the DVDD18 supply is often connected to other 1.8V supplies in the design, these other supplies can be pulled up also.

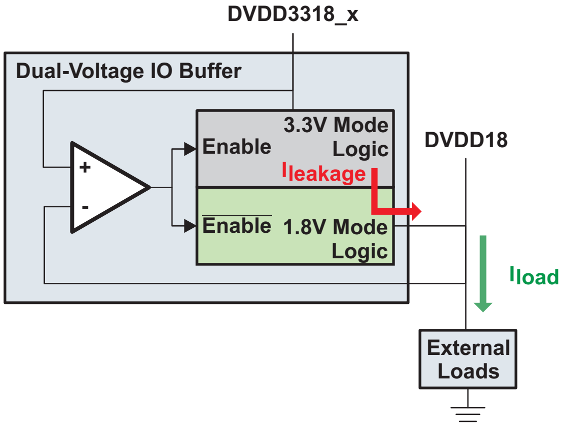
The condition occurs because the dual-voltage IO buffers have voltage detection circuitry that monitors DVDD3318_x during reset and determines whether the applied supply voltage is 1.8V or 3.3V. This detection circuitry then configures the IOs to operate at the appropriate voltage. As the DVDD3318_x supply ramps, there is a small range near DVDD18 + $V_T$ where the voltage detection circuit finds the state indeterminate and briefly turns on circuitry associated with both voltage options creating a current path between them. This current path can cause the DVDD18 supply to be pulled up. The mechanism for this behavior is explained below:



**Behavior of the IO buffer during Zone F:**

**As the DVDD3318_x supply ramps, but is not yet one $V_T$ above the DVDD18 supply, the comparator enables the 1.8V mode logic section and disables the 3.3V mode logic section. No unexpected current flows between the two sections.**

**Figure 7. Behavior of the IO buffer during Zone F**

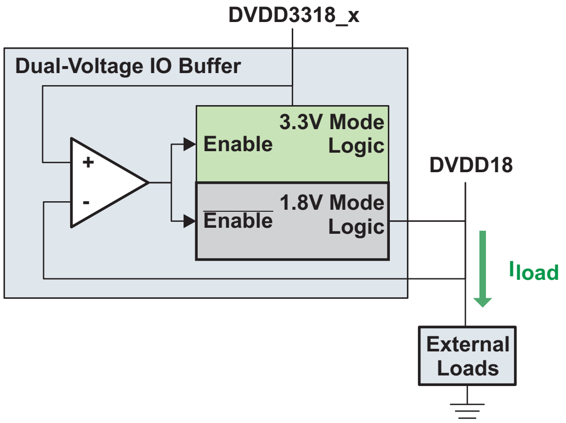


**Error State of the IO buffer during Zone G:**

**As DVDD3318_x exceeds DVDD18+$V_T$, some of the circuitry across the boundary of the two voltage domains will turn on and causes a leakage current (in red) to flow between the two voltage domains. This current pulls up the DVDD18 supply unless the power source providing that supply can oppose it or unless the load current is strong enough to counteract it.**

**If DVDD18 continues to rise as DVDD3318_x ramps, the comparator never sees enough difference between the two supplies to switch to 3.3V mode.**

**Figure 8. Error State of the IO buffer during Zone G**



**Recovered State of the IO buffer during Zone G:**

**If the load current is high enough or the DVDD18 power supply can oppose the leakage current, then the DVDD18 voltage stays low enough for:**
- **the comparator to recognize 3.3V mode**
- **the 1.8V mode logic is turned off**
- **the leakage stops**

**Figure 9. Recovered State of the IO buffer during Zone G**

This error condition may occur when either of the following conditions are present:

- The regulator used to control the DVDD18 supply can only regulate voltage up and relies on the load to pull the output voltage down
- There is low load on the DVDD18 supply during the DVDD3318_x power supply ramp

This condition will not occur in designs where:

- All DVDD3318_x supplies are operated at 1.8V, or
- The regulator used to supply DVDD18 has the capability to actively regulate (drive) its output voltage up or down and doesn't rely on system load to pull the voltage down, or
- The load on the DVDD18 supply during the DVDD3318_x power supply ramp is sufficient to oppose the leakage current.

**Workaround(s)**

1. Design simulation has indicated that under the worst-case process/voltage/temperature conditions the maximum leakage current into the DVDD18 supply due to this phenomenon will be 140 mA (the 140 mA is a cumulative current generated by all of the dual-voltage IOs). So this error state can be prevented by ensuring that the load on the DVDD18 supply during the DVDD13318_x supply ramp exceeds 140 mA. This can be achieved by any combination of the following as long as they cumulatively produce >140 mA load on DVDD18 during the DVDD3318_x ramp period:

   (a) Maintain sufficient bulk capacitance on the DVDD18 supply such that the charging current for these capacitors provides all or part of the required >140 mA. Bulk capacitance in this context means the total capacitance seen by the DVDD18 supply (filter capacitors, bypass capacitors, etc.). Capacitor charging current is defined as I = C*(dV/dt). So the ramp rate of the DVDD3318_x supply and the total bulk capacitance on the DVDD18 supply can be used to calculate the current produced. This solution provides additional power supply filtering and little current leakage after the supplies are ramped (depending on the type of capacitors used).

   The table below shows some examples of the bulk capacitance that would be required to use this solution alone:

**Table 12. Bulk Capacitance**

| DVDD3318_x ramp time for 3.3V | dV/dt (in V per second) | Required capacitance to generate 140 mA |
|---|---|---|
| 100 µs | 33000 | 4.2 µF |
| 250 µs | 13200 | 10.6 µF |
| 500 µs | 6600 | 21.2 µF |
| 1 ms | 3300 | 42.4 µF |
| 10 ms | 330 | 424.2 µF |

   (b) Use an additional shunt regulator to control the voltage at DVDD18. The shunt regulator is placed between DVDD18 and Vss. As an example, the TLVH431 can provide up to 70 mA of additional load current to help maintain the DVDD18 voltage. When the voltage drops back to the normal 1.8V range, the current flow through the shunt regulator drops into the <100 uA range. This solution requires the shunt regulator and two additional resistors to set the desired regulation voltage.

   (c) Use a resistor to provide additional load between DVDD18 and Vss. This solution is less desirable since it continues to draw power even after the supply ramp is completed but would likely be the lowest cost. An improvement is to use a FET/switch in series with the resistor between DVDD18 and Vss that can later be turned off (by the RESET signal or a GPIO signal for example).

2. Choose a regulator for the DVDD18 supply (or grouped 1.8V supplies) that is capable of actively regulating voltage up and down. Many high-efficiency switching regulators switch current into the load only when the voltage needs to be raised and rely on the load to pulldown the current. In this error condition, a regulator of this type will not be able to compensate for the leakage current described above. Synchronous buck regulators use external inductance to pulldown the regulated voltage when necessary.

## Advisory 2.3.19 *USB 2.0 On-The-Go (OTG) Session Request Protocol (SRP) Is Not Supported*

**Revision(s) Affected** 2.3 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 2.3.22 *SATA: Link Establishment Fails With SATA GEN3 Capable Targets*

**Revision(s) Affected:** 2.3 and earlier

**Details:** When connecting a SATA GEN3 capable target, for example a Hard Disk Drive (HDD), to a device with a SATA Host Subsystem (after power-up or reset) the speed negotiation fails between the two devices and no link is established.

Two different types of failure behaviors with the same results have been observed:

**Losing Synchronization:**

The Target (Device) always starts the speed negotiation at the highest speed supported, in this case GEN3, by sending an ALIGNp primitive data pattern to the Host SATA subsystem. The Host SATA subsystem sends a continuous D10.2 Tone at GEN1 speed (1.5 GBits/sec) and should ideally remain at this state until the Host recognizes the Targets' ALIGNp primitive data pattern [at GEN2 or GEN1 but not GEN3 speed]. While the Target (Device) is still at GEN3, due to aliasing, etc., the Host SATA subsystem falsely responds back to the Device with an ALIGNp primitive data pattern at a different speed (GEN2 speed). The Host completes the speed negotiation at GEN2 speed and transitions to a logical IDLE state (Non-ALIGNp primitive SYNCp) before the Target (Device) timeout period expires (54.6 µs). Once the timeout period expires for GEN3 speed, the Target (Device) starts sending an ALIGNp primitive data pattern at GEN2 speed, expecting an ALIGNp primitive data pattern from the Host which never happens because the Host is in a logical IDLE state at GEN2 speed. Another timeout period expires because the target (Device) did not receive the ALIGNp primitive data pattern at GEN2 speed. This forces the Target (Device) to drop its speed from GEN2 to GEN1 and attempt to establish a link at GEN1 speed. The Host still remains in a logical IDLE state at GEN2 speed. After the final timeout period expires, the Target (Device) requests a RESET (by sending COMINIT signal) to restart the link establishment process with the Host. This new link establishment results in the same outcome with the Host and Target always being out of sync.

**Unknown State/Lock-up:**

The Target (Device) always starts the speed negotiation at the highest speed supported, in this case GEN3, by sending an ALIGNp primitive data pattern to the Host SATA subsystem. The Host SATA subsystem sends a continuous D10.2 Tone at GEN1 speed (1.5 GBits/sec) and ideally should remains at this state until the Host recognizes the Targets' ALIGNp primitive data pattern [at GEN2 or GEN1 but not GEN3 speed]. While the Target (Device) is still at GEN3, due to aliasing, etc., the Host SATA subsystem falsely responds back to the Target (Device) with an ALIGNp primitive data pattern at a different speed (GEN2 speed) and remains at this state (sending the GEN2 ALIGNp primitive). The Target (Device) times out (54.6 µs) and starts sending an ALIGNp primitive data pattern at GEN2 speed. Target (Device) now recognizes the Host GEN2 ALIGNp primitive data pattern and responds with a logical IDLE state (Non-ALIGNp primitive SYNCp) completing the link establishment from the Target (Device) perspective. However, the Host is stuck in an unknown state sending GEN2 ALIGNp primitive and never completes the link establishment. Both the Host and Target (Device) remain at this state until a higher Host SATA Controller application (User S/W) performs a Reset.

**Note:** This issue does not apply to Target devices with maximum speed capability of GEN2 or GEN1 speed.

**Workaround(s):**

- Use GEN2 or GEN1 maximum speed drives to avoid the issue

  or

- Use GEN3 drives with jumper restricting capabilities to restrict their speed to GEN2

  or

- The Host Application S/W can continually perform Port resets to restart the link establishment eventually succeeding in establishing a link. **Note:** This is not a preferred method because an excessive amount of resets might be required to establishment the link.

## Advisory 2.3.23 *BOOT: Internal Pullup Resistors for BOOT[7:0] Pins Are Sometimes Enabled During Reset, Leading to Boot Failures*

**Revision(s) Affected** 2.3 and earlier

**Details:** The PUPD_SEL[29] register does not get initialized when the device is first powered on and in Reset. This register controls the internal pullup and pulldown resistors for the BOOT[7:0] pins. The contents of the PUPD_SEL[29] register at this state are unpredictable and may contain random values. These random values can result in the internal pullups being enabled for some or all of the BOOT[7:0] pins during reset after every power-on, which conflicts with the datasheet claim that the internal pulldowns are enabled during reset.

Internal pullups being enabled on the BOOT[7:0] pins may result in boot failures. If weak external pulldown resistors are used on the PCB to select the boot mode, they may not be able to overpower the internal pullups. This can result in the wrong boot mode being latched in the BOOTCFG[7:0] register when coming out of Reset.

Once the device is out of Reset, the PUPD_SEL[29] register is initialized, and the internal pulldown resistors for the BOOT[7:0] pins are all enabled. Issuing a second Power-On Reset ($\overline{\text{POR}}$) results in the intended boot mode being latched, since the pins are internally pulled down as expected.

Figure 10 shows the behavior of the PUPD_SEL[29] register before and after reset. Assuming weak or no external pull resistors are on the BOOT[7:0] pins, the BOOTCFG register will latch the unknown values after the device initially comes out of Reset. Note that, $\overline{\text{TRST}}$ ***must*** always be low in order to issue a $\overline{\text{POR}}$ and latch the boot pin values -- the boot pins are not latched after a Warm Reset.

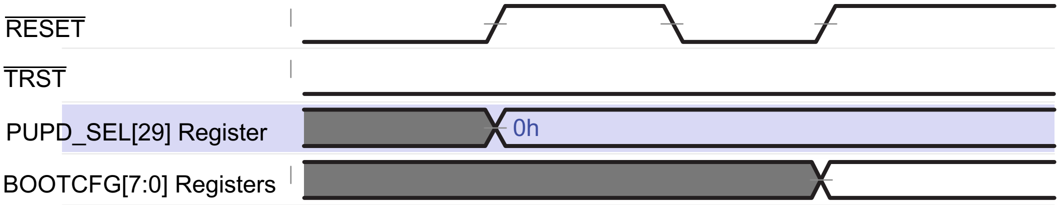


**Figure 10. Initialization of PUPD_SEL[29] and BOOTCFG[7:0] Registers with Weak or No External Pull Resistors on BOOT[7:0] Pins**

Other device pins with configurable internal pullup or pulldown resistors are always internally pulled down during reset and are not affected by this advisory.

**Workaround(s):** One of the following Workarounds must be implemented to ensure that the boot pins are always latched correctly:

**Method 1**

Use strong external pull resistors on BOOT[7:0] pins. Since either the internal pullup or pulldown resistors could be enabled after every power-on, the external resistors must be strong enough to oppose the internal pulls in either case.

Section 4, *Device Operating Conditions*, of the device datasheet shows the electrical characteristics information which can be used to calculate the maximum external pull resistance required. The value is dependent on the DVDD3318_C I/O supply level.

For BOOT pins which need to be sampled as logical low, the external pulldown resistance ($R_{PDmax}$) must be selected by assuming the internal pullup is enabled. The calculation is shown in Table 13.

### Table 13. Required Pulldown Resistance ($R_{PDmax}$) for Logical Low BOOT Pins

| DVDD3318_C | $I_I$ | $V_{IL}$ | $R_{PDmax}\left(\frac{V_{IL}}{I_I}\right)$ |
|---|---|---|---|
| 3.3 V | 310 µA | 0.80 V | 2.58 kΩ |
| 1.8 V | 310 µA | 0.80 V | 2.03 kΩ |

For BOOT pins which need to be sampled as logical high, the external pullup resistance ($R_{PUmax}$) must be selected by assuming the internal pulldown is enabled. The calculation is shown in Table 2.

### Table 14. Required Pullup Resistance ($R_{PUmax}$) for Logical High BOOT Pins

| DVDD3318_C | $I_I$ | $V_{IH}$ | $R_{PUmax}\left(\frac{DVDD3318\_C - V_{IH}}{I_I}\right)$ |
|---|---|---|---|
| 3.3 V | 270 µA | 2.00 V | 4.81 kΩ |
| 1.8 V | 270 µA | 1.17 V | 2.33 kΩ |

### Method 2

For applications that have already implemented the Secondary Reset Workaround described in Advisory 2.0.20, no additional modifications are required as a fix for this Advisory.

As shown in Figure 10, issuing a second $\overline{\text{POR}}$ will always latch the BOOT pins while the internal pulldown resistors enabled.

## Advisory 2.3.24 *Boot: ECC Data Error in Spare Area Causes NAND Boot Failure*

**Revision(s) Affected** 2.3 and 2.1 (ROM Versions D800K008)

**Details:** The ROM bootloader (RBL) reads a NAND page in segments of 512 bytes (user data) over the External Memory Interface A (EMIFA). The EMIF also reads the associated ECC data which is stored in the spare area of the flash (as shown in Figure 11). The ECC Correct function in the RBL can correct up to 4 bit errors in the user data and/or ECC data by using the syndrome generated from the ECC data and the parity of the user data calculated by the EMIFA module.

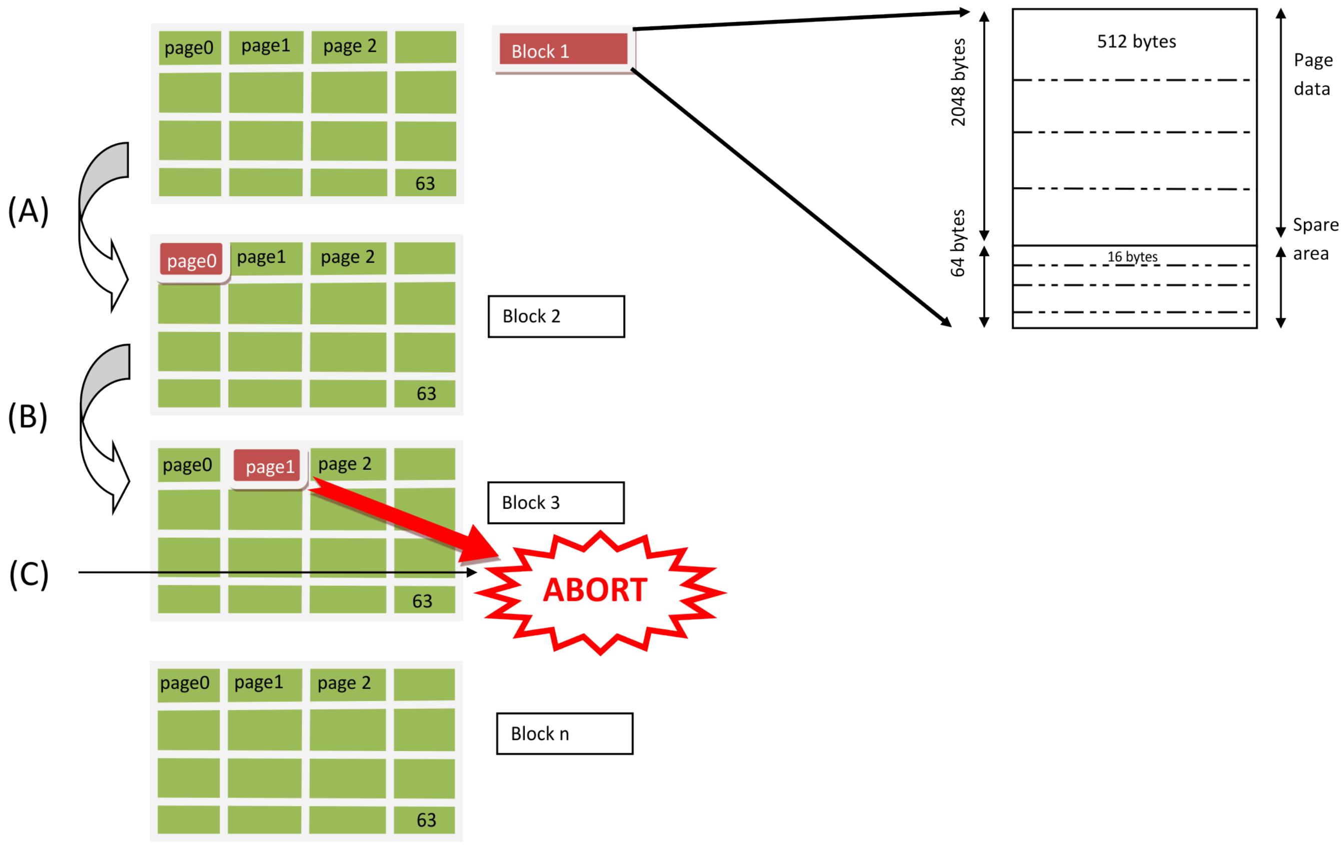
However, over the life span of the NAND flash, ECC data stored in the spare area can develop errors due to bit flips. When the calculated syndrome indicates an error in the ECC data, the ECC Correct and Read functions of the RBL abort the read process even though, it is possible to correct up to 4 bit errors combined in user and ECC data. Consequently the device fails to boot.

**Explanation of Current ROM Bootloader Behavior:**

The RBL implements a search mechanism to look for the boot image in NAND flash by using an Open function and a Read function. The Open function includes a bad block check where the RBL skips to the next block (shown as (A) in Figure 11) if the block under consideration is marked as "bad" (in the spare area). On finding a good block, the RBL attempts to read page0 (the first page) in that block.

If page0 of the good block has an ECC data error or an uncorrectable error (more than 4 bit errors combined), the RBL skips to the next block (shown as (B) in Figure 11). This bad block check mechanism enables the device to check up to the first 32 blocks in the NAND flash for booting (the boot process will fail if all 32 blocks have uncorrectable or ECC data errors in page0). Note that the RBL does not abort on detecting an error in ECC data on page0.

Once the RBL finds a good block with a good page0, it continues to read subsequent pages in that block. If an uncorrectable or ECC error is detected in subsequent pages, the RBL will abort with a boot error (shown as (C) in Figure 11). The workaround, described below, enables the boot process to continue for both types of errors (ECC data and uncorrectable errors).



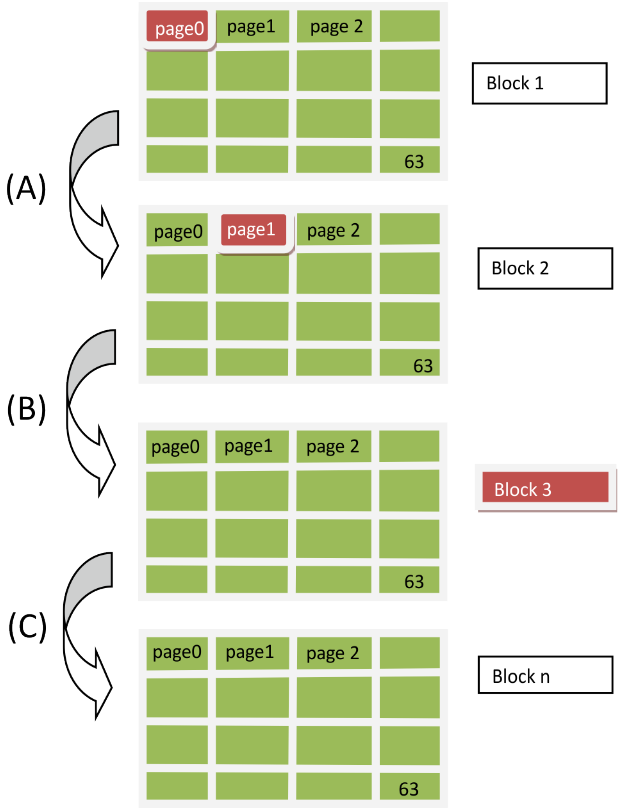
(1) All pages marked in **green** are good and all pages marked in **red** have "uncorrectable" errors.

**Figure 11. D800K008 ROM Behavior Before Application of Software Patch**[(1)]

**Workaround(s):** The workaround ignores errors in ECC data (these errors do not need to be corrected) so that the boot process can continue, correcting errors in the user data (up to 4 bit errors total), as necessary. The workaround also includes a mechanism to restart the NAND read process from the next good block if an uncorrectable error (more than 4 bit errors) is detected on any page of a good block.

This workaround is implemented using a software patch that is loaded in device RAM at boot time and is designed to change the default behavior of the ECC Correct and Read functions in the RBL. The patch binary replaces function pointers to the ECC Correct and Read functions in the ROM function table, (defined in device internal memory), during boot. The patch binary will reside in page0 of the NAND block and so will be applied only after the page0 of a good block has been read.

Once the patch is read, all subsequent page reads in the block will use the new ECC Correct and Read functions. ECC data errors on any page (other than page0) are ignored, any user data error (up to 4 bit errors total) is corrected and the boot process continues. If an uncorrectable error is detected on any page, the block is skipped and the boot process is restarted on the next good block. The behavior of the device boot from the NAND after application of the patch is shown in Figure 12.



(1) All pages marked in **green** are good and all pages marked in **red** have "uncorrectable" errors.

**Figure 12. D800K008 ROM Behavior With Software Patch Applied**(1)

The software patch is available as a pre-built file with the latest version of the AIS tool that is used to generate the NAND flash boot image. The *Using the OMAP-L132/L138 Bootloader Application Report* (Literature number: SPRAB41) provides a link to the install package for the AIS tool which includes the following in the install directory: prebuilt patch files, the GUI AIS generation tool (AISGEN.exe, version 1.11 or later), command-line AIS generation tool and an example INI file.

**Application of the Software Patch to NAND Boot Images:**

- For the GUI tool, AISGEN.exe version 1.11 or later (found in the install directory), the patch integrates the modified ECC Correct function into the user application file to generate one binary AIS file.
- For the Command line AIS tool, HEXAIS_OMAP_L138.exe users (found in the install directory), the patch can be integrated into the user application file to generate one binary AIS file by inserting lines below to the end of the INI configuration file.

The patch name is: **ARM_nand_ecc_patch_OMAP-L138.out**

```
[INPUTFILE] ; get the NAND ECC patch file
FILENAME=Patch_name.out

; patch the NAND ECC handling routine
[AIS_Jump]
LOCATION=_NAND_ECC_patchApply
```

- The patch will be burnt on page0, as it is in the head of AIS file. Once page0 has been read successfully, the AIS set command will overwrite the function pointer in the RBL with the modified function pointer and the modified function will be applied to the later pages read. Memory usage of the patch at boot time is as follows:
  - 500 bytes at location 0xFFFF 0B00

***Recommendations to Improve Robustness*:**

- Page0 should be stored on multiple blocks as backup to take advantage of the safety mechanism built into Silicon Revision 2.1 to cycle to the next block when a page0 read fails in a good block.
- Maintain backup boot images in multiple blocks. The patch to the Abort function reinitializes the boot process and forces the boot to cycle to the next good block and restart the boot process by reading from it.
- Silicon Revision 2.1 supports booting from Block 0 of the flash which many NAND manufacturers guarantee as a “more reliable” block than all other blocks. Hence, setting up boot to start from that Block 0 could help improve the reliability of boot. This is a hardware change, requiring the bootmode pins BOOT[6:5] = 1x.

  For more details on the bootmode pins, see the *Using the OMAP-L132/L138 Bootloader Application Report* (Literature number: SPRAB41), *NAND-Boot Modes*.

## Advisory 2.3.25 *USB0: CPU gets Stale Receive Data from the Data Buffer located in External Memory*

**Revision(s) Affected** 2.3 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 2.3.26 *USB0: Early DMA Completion in DMA Receive Mode and More Than One Endpoint is Transferring Data*

**Revision(s) Affected** 2.3 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 2.3.27 *USB0: DMA Hung up in Frequent Teardowns*

**Revision(s) Affected** 2.3 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 and earlier of the device.

## *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 2.0 applicable usage notes have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## *3.2 Silicon Revision 2.1 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 15. Silicon Revision 2.1 Advisory List**

Silicon revision 2.1 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## Advisory 2.1.21 *USB0 PLL Mean Frequency Can Drift Across Large Temperature Swings*

**Revision(s) Affected** 2.1 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 PLL drift occurs, the mean frequency will continue to drift outside the expected frequency and will eventually cause the PLL to lose lock 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 range of 0˚C to 90˚C. TI has identified a point-to-point device temperature range in which there is a very high confidence the compensation circuitry will properly compensate for all temperature variations, provided 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 */
}
```

# 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 device.

## *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 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## *4.2 Silicon Revision 2.0 Known Design Exceptions to Functional Specifications*

**Table 16. Silicon Revision 2.0 Advisory List**

Silicon revision 2.0 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## 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. Before the ARM can take control of the user boot mode, the DSP must first initialize the ARM reset vector table so that the ARM will execute from its boot ROM.

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 13). 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.

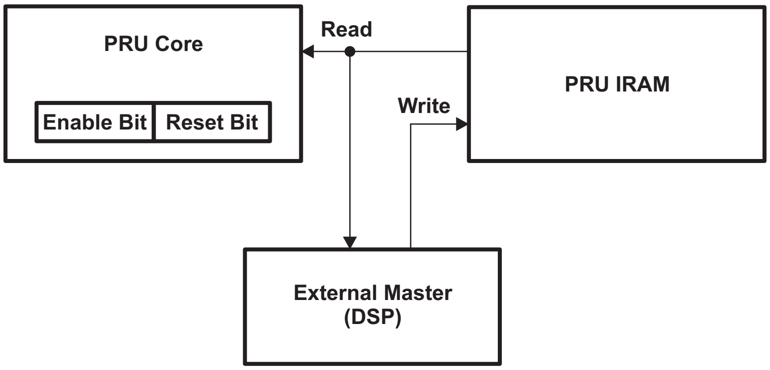


**Figure 13. PRU and DSP Block Diagram**

When the device is first powered-on, the read bus from the PRU IRAM is not initialized and will contain random values (see Figure 14). 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.

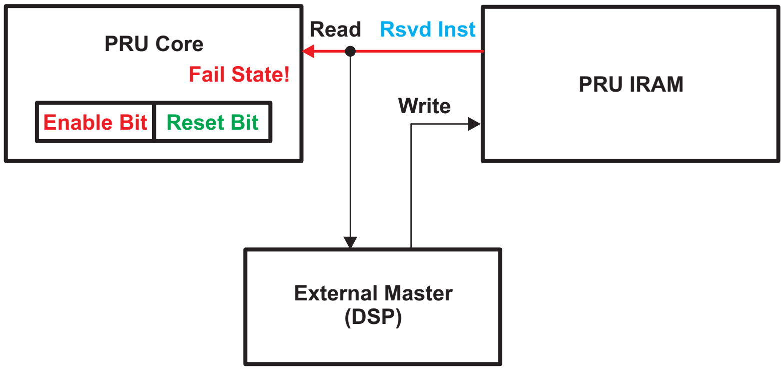


**Figure 14. Boot Failure on Power-On**

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 15). 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.

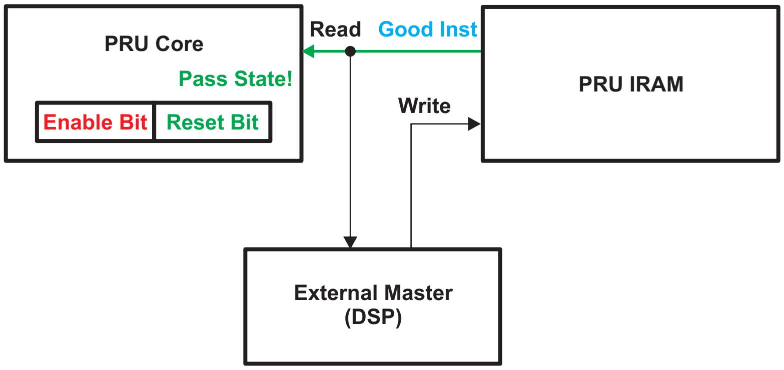


**Figure 15. Secondary Reset**

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 16). With a 24MHz crystal, the first RESET signal must be asserted high for at least 883us (or approximately 1ms).

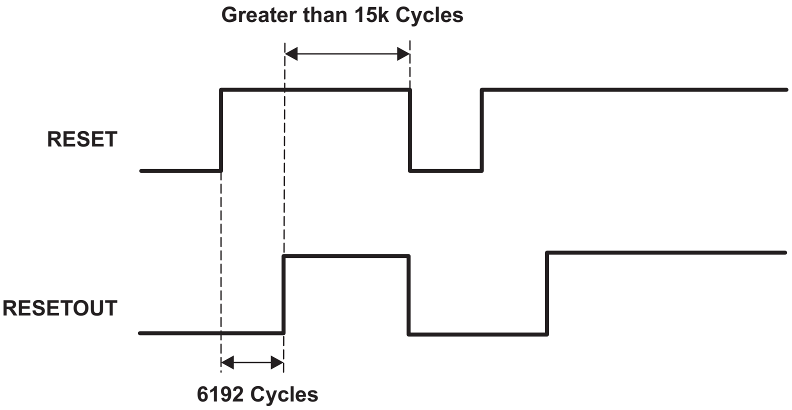


**Figure 16. First POR Reset Timing**

**Workaround(s)**

Modify the target board so that the affected device is given a secondary reset on power-up as shown in Figure 16. 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 CLKOUT or a periodic output peripheral like TIMER and ePWM.

Potential reset supervisors are TPS382x for 3.3V IOs (shown in Figure 17) and TPS312x for 1.8V IOs.

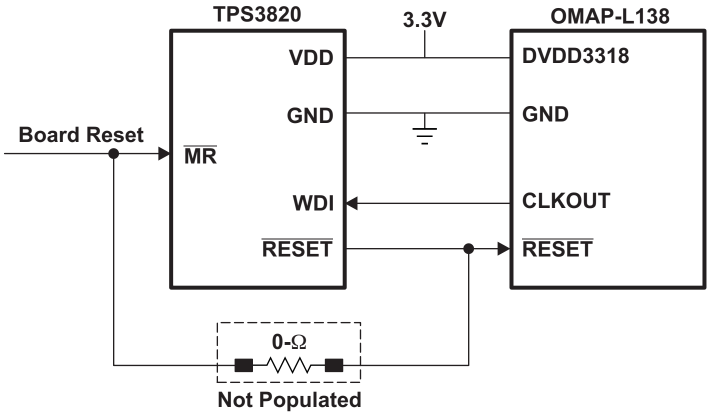


**Figure 17. Reset Supervisor with Watchdog Function**

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 18, a single board reset control signal can trigger three logic transitions in a dual XOR gate device.

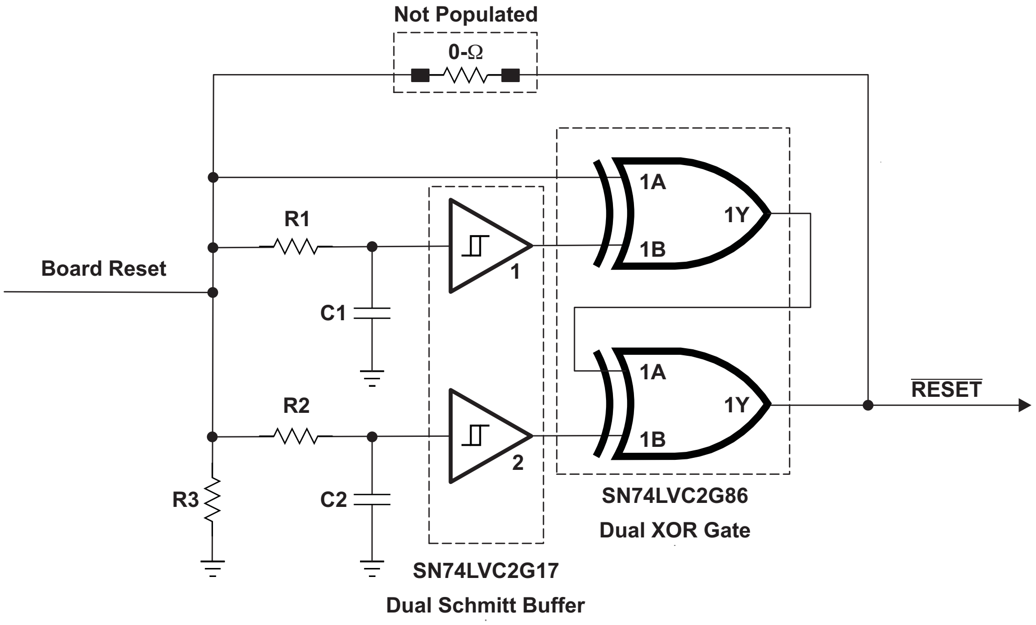


**Figure 18. RC-Timed Secondary Reset**

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 19 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])

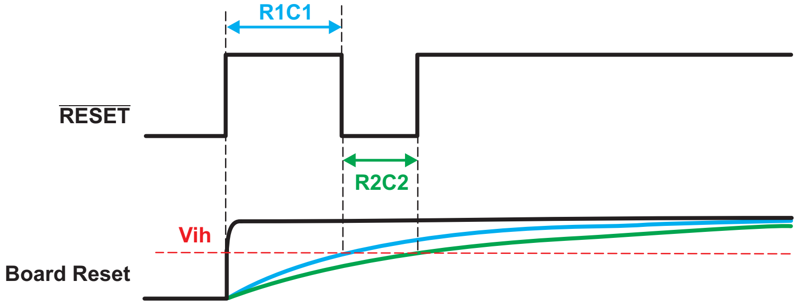


**Figure 19. RESET Signal vs Board Reset**

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, and

(e) The sole purpose of the R3 pulldown 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 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 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

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

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

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.

### 5.1.3 SATA & USB Digital Supplies Must Be Powered for Proper Device Operation

For silicon revisions prior to 2.0, the SATA and USB digital supply pins must be powered up. These supplies provide power to logic that is critical for device initialization during reset. A bad initialization sequence will cause the device boot process to fail. Another symptom of a bad initialization sequence is that the DIEIDRx registers (0x01C14008h – 0x01C14017h) will appear as all zeros.

Starting with silicon revision 2.0, the affected logic will be powered by the CVDD core supply instead of the USB and SATA supplies.

The following is a list of the supply pins affected:

**Table 17. Supply Pins Affected**

| Digital Supply | Pin |
|---|---|
| SATA_VDD | M2 |
| SATA_VDD | N4 |
| SATA_VDD | P1 |
| SATA_VDD | P2 |
| USB_CVDD | M12 |

## 5.2 *Silicon Revision 1.1 Known Design Exceptions to Functional Specifications*

Silicon revision 1.1 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 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 18. Silicon Revision 1.1 Advisory List**

**Title** **Page**

## Advisory 1.1.2 *Under Specific Conditions, SDMA Activity Can Corrupt the L1D Cache and L2 RAM*

**Revision(s) Affected** 1.1 and earlier

**Details**

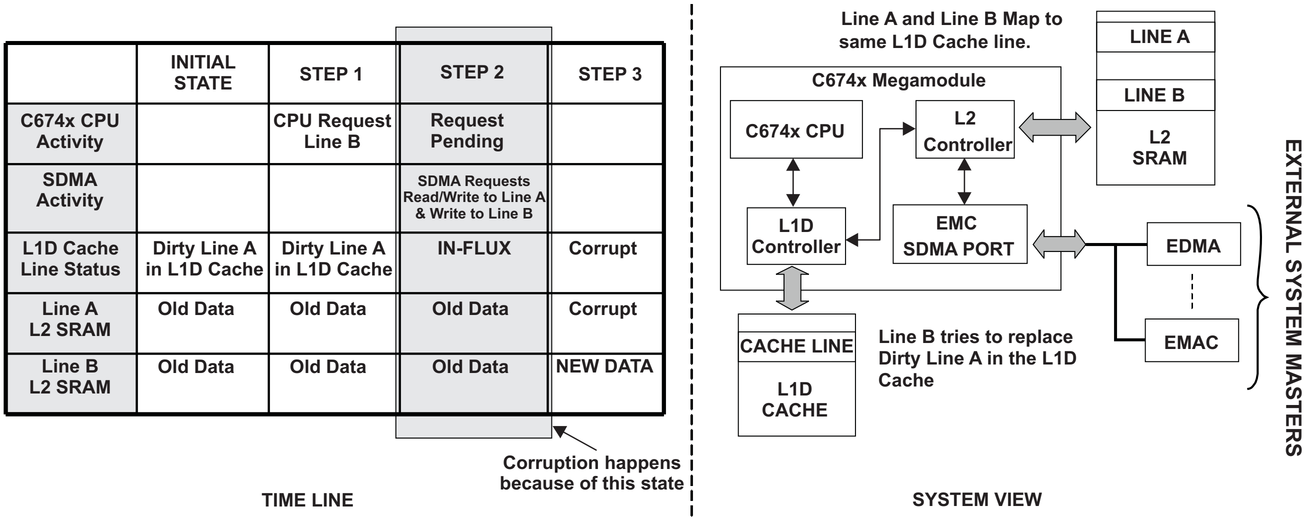
**Note:** DMA refers to all non-CPU requests. SDMA refers to external system DMA master requests handled via the Slave Direct Memory Access port.

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 20 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 Read/Write to Line A & Write to Line B | |
| 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 20. 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 1.1.7 *Low Voltage Operating Points (1.1V, 1.0V) Not Supported*

**Revision(s) Affected** 1.1 and earlier

**Details** The low-voltage operating points are not supported on revisions 1.1 and earlier. The support of these operating points is planned for revision 2.0.

**Workaround(s)** None

## Advisory 1.1.11 *Hardware RESET Reliability Lifetime*

**Revision(s) Affected** 1.1 and earlier

**Details**

A potential reliability issue has been discovered when hardware resets (RESET pin high-to-low transitions) occur. If any of the dual-voltage LVCMOS IOs on the device are being operated at 3.3V nominal and a hardware reset occurs, some 1.8V transistors in the IOs buffers are briefly exposed to 3.3V. This exposure to high voltage has the potential to degrade the gate oxide integrity of the transistors over repeated resets.

This issue does not occur during a power-up condition where the RESET pin is held low.

This issue does not occur with software resets or watchdog timer induced resets.

This issue does not occur if all dual-voltage LVCMOS IO buffers are being operated at 1.8V nominal.

Reliability estimates have been made to determine a budget for the number of resets in a product lifetime to maintain the reliability of the device under 50 FIT.

- At a junction temperature of 105°C, the total number of hardware resets in the product lifetime should not exceed 125,000 hardware resets.
- At a junction temperature of 90°C, the total number of hardware resets in the product lifetime should not exceed 250,000 hardware resets.

**Workaround(s)** Limit the number of hardware reset to within the limits listed above.

## Advisory 1.1.14 *SYSCFG SUSPSRC Bits Not Functional for McBSP0, McBSP1, TIMER2, and VPIF*

**Revision(s) Affected** 1.1 and earlier

**Details**

Device peripherals often include an emulation suspend function that gracefully halts peripheral activity. This function is activated when the target CPU is halted through emulator debug. While halted, the control and status registers for the module can be viewed and manipulated for debug purposes.

In a multicore device, it is desirable to choose a single CPU to master the emulation suspend function. Typically, the CPU that configures a module is chosen to be its suspend source CPU (ARM or DSP). The SUSPSRC register in the SYSCFG module allows the user to choose which CPU controls the emulation suspend function for each peripheral.

For the affected silicon revisions, the SUSPSRC fields MCBSP0SRC, MCBSP1SRC, TIMER64P_2SRC, and VPIFSRC are not functional. The ARM will always be the suspend source regardless of the bit settings. If the ARM is connected and halted through emulator debug, the affected peripherals will appear to be halted as observed by the DSP. Similarly, if the ARM is free-running, the affected peripherals will never halt when the DSP is halted.

The described condition is only present during emulation debug – the peripherals will function as expected when the device is free-running in an application.

**Workaround(s)**

When developing DSP software, the affected peripherals can be removed from halted states by either free-running the ARM CPU or by setting the peripheral-specific emulation mode to FREE. The affected peripherals can be placed into halted states by halting the ARM CPU.

Code Composer Studio (CCS) allows the user to set Global Breakpoints such that the DSP and ARM CPUs will both halt at the same time on a given breakpoint. CCS also includes the ability to simultaneously set debug states (such as RUN and HALT) across multiple processors. When the ARM and DSP both run and halt at the same time, the peripherals will appear as if their suspend source is the DSP.

## Advisory 1.1.16 *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 19.

**Table 19. Cacheable External Memory Resources**

| External Memory | Address Range |
|---|---|
| Shared Ram | 0x8000 0000 – 0x8001 FFFF |
| EMIFA | 0x4000 0000 – 0x67FF FFFF |
| DDR2/mDDR | 0xC000 0000 – 0xCFFF FFFF |

The SDMA interface allows other DMA master peripherals (listed in Table 20 ) to access Level 1 Data (L1D), Level 1 Program (L1P), and L2 RAM DSP memories.

**Table 20. DMA Master Peripherals**

| Peripheral | Group |
|---|---|
| EDMA0 TC0 RD | A |
| EDMA0 TC0 WR | B |
| EDMA0 TC1 RD | C |
| EDMA0 TC1 WR | D |
| EDMA1 TC0 RD | E |
| EDMA1 TC0 WR | F |
| UHPI | G |
| USB0 | G |
| USB1 | G |
| EMAC | G |
| PRU | H |
| SATA | I |
| UPP | I |
| VPIF | I |
| ARM | J |

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 21) and SDMA/IDMA paths (dashed lines in Figure 21) 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 21 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).

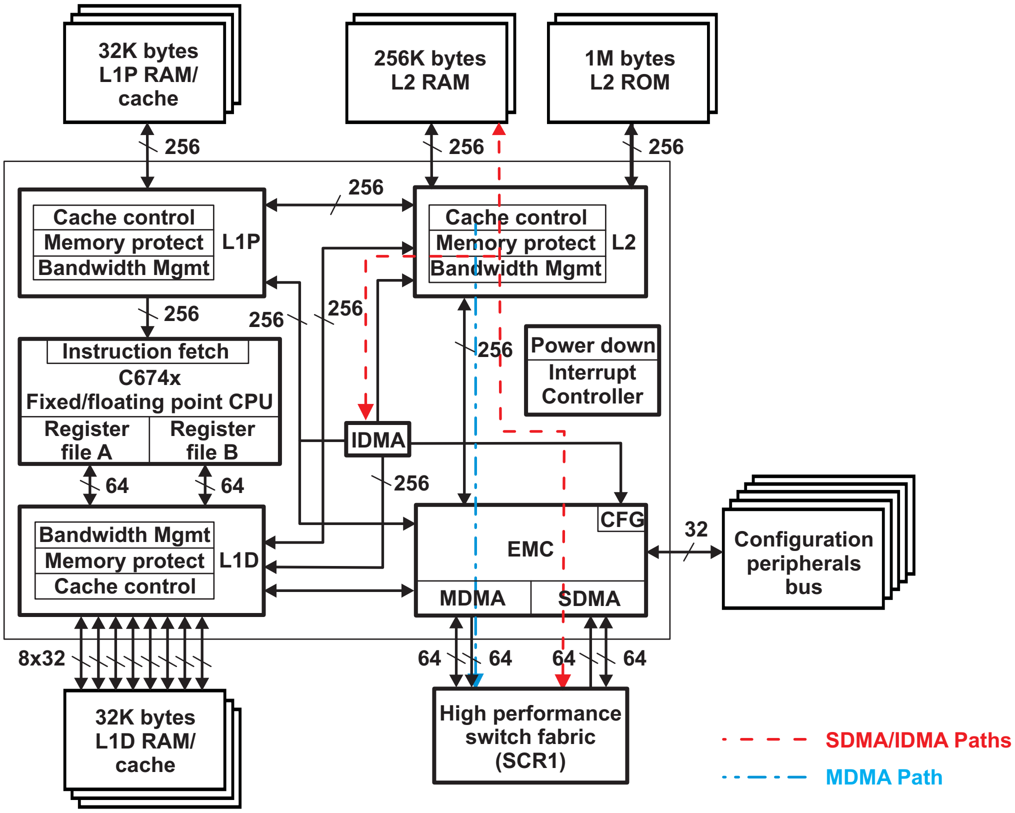


Figure 21. C674x Megamodule

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 2) 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 System Reference Guide SPRUG84.

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 22 is provided for illustrative purposes and is incomplete because of simplification.

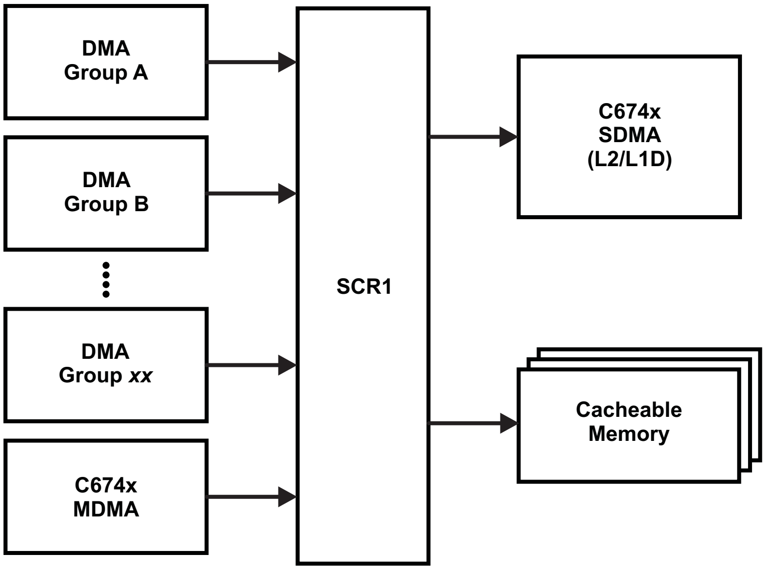


**Figure 22. SCR1 System Interconnect**

**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 devices.

## *6.1 Usage Notes for Silicon Revision 1.0*

Silicon revision 1.0 applicable usage notes have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## *6.2 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications*

Silicon revision 1.0 applicable advisories have been found on a later silicon revision. For more details, see Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications*.

## Revision History

This silicon errata revision history highlights the technical changes made to this document.

### Revision History

| ADDITIONS/MODIFICATIONS/DELETIONS |
|---|
| **Changes that were made from SPRZ301 to SPRZ301A**<br>• Added Silicon Revision 1.0 Usage Notes and Known Design Exceptions to Functional Specifications<br>• Added Silicon Revision 1.1 Usage Notes and Known Design Exceptions to Functional Specifications |
| **Changes that were made from SPRZ301A to SPRZ301B**<br>• Advisory 1.1.10 - ARM Interrupt Controller Vector Size (VSR) Initialization |
| **Changes that were made from SPRZ301B to SPRZ301C**<br>• Added Silicon Revision 2.0 Usage Notes and Known Design Exceptions to Functional Specifications |
| **Changes that were made from SPRZ301C to SPRZ301D**<br>• Advisory 2.0.6 - Potential USB 2.0 Soft Reset Timing Violation Update<br>• Advisory 2.0.17 - SDMA Activity Can Corrupt L1D When L2 Is Configured as Mixed/Cache/SRAM<br>• Advisory 2.0.18 - DVDD18 Can Pull Up to 2.7V When using Dual-Voltage IOs at 3.3V<br>• Advisory 2.0.19 - USB 2.0 On-The-Go (OTG) Session Request Protocol (SRP) Is Not Supported<br>• Usage Note: SATA & USB Digital Supplies Must Be Powered for Proper Operation |
| **Changes that were made from SPRZ301D to SPRZ301E**<br>• Advisory 2.0.20 - Intermittent Boot Failures |
| **Changes that were made from SPRZ301E to SPRZ301F**<br>• Advisory 2.0.20 - Intermittent Boot Failures Fixed in Silicon Revision 2.1<br>• Revision Identification Table Updated for Silicon Revision 2.1<br>• Device Marking Figure Added |
| **Changes that were made from SPRZ301F to SPRZ301G**<br>• Usage Note: System-Level ESD Immunity<br>• Advisory 2.1.21 - USB0 PLL Mean Frequency Can Drift Across Large Temperature Swings<br>• Advisory 2.1.22 - SATA: Link Establishment Fails With SATA GEN3 Capable Targets |
| **Changes that were made from SPRZ301G to SPRZ301H**<br>• Added Advisory 2.1.23 - BOOT: Internal Pullup Resistors for BOOT[7:0] Pins Are Sometimes Enabled During Reset, Leading to Boot Failures |
| **Changes that were made from SPRZ301H to SPRZ301K**<br>• Updated/Changed "OMAP-L138 C6-Intergra DSP+ARM® Low-Power Applications Processor" ***to*** "OMAP-L138 C6000 DSP+ARM® Processor" throughout the document [**Global**]<br>Section 2.1.4, System-Level ESD Immunity Usage Note:<br>• Added additional code to the "Disable the DLL REFCLK signal in the DDR EMIF PHY ..." bullet<br>Section 2.1.5, DDR2/mDDR Controller: mDDR Usage Note:<br>• Added *new*<br>Section 3.2, *Silicon Revision 2.1 Known Design Exceptions to Functional Specifications*:<br>• Advisory 2.1.21, *USB0 PLL Mean Frequency Can Drift Across Large Temperature Swings*<br>– Updated/Changed the "Once the onset of PLL drift occurs, ..." paragraph<br>• Advisory 2.1.24, *Boot: ECC Data Error in Spare Area Causes NAND Boot Failure*<br>– Added *new* |
| **Changes that were made from SPRZ301L to SPRZ301M**<br>Added new section: Section 2, *Silicon Revision 2.3 Usage Notes and Known Design Exceptions to Functional Specifications*<br>Section 2.1, *Usage Notes for Silicon Revision 2.3:*<br>• Moved all Silicon Revision 2.1 Usage Notes to Silicon Revision 2.3<br>• Added new Usage Note: Section 2.1.6, McASP: Inactive Slot Usage Note<br>Section 2.2, *Silicon Revision 2.3 Known Design Exceptions to Functional Specifications:*<br>• Moved all Silicon Revision 2.1 Advisories to Silicon Revision 2.3 ***except*** 2.1.21, USB0 PLL Mean Frequency Can Drift Across Large Temperature Swings<br>• Added new Advisory: Advisory 2.3.25, USB0: CPU gets Stale Receive Data from the Data Buffer located in External Memory<br>• Added new Advisory: Advisory 2.3.26, USB0: Early DMA Completion in DMA Receive Mode and More Than One Endpoint is Transferring Data<br>• Added new Advisory: Advisory 2.3.27, USB0: DMA Hung up in Frequent Teardowns |

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

## 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