TMS320DM816x DaVinci™ Digital Media Processors Silicon Revisions 2.1, 2.0, 1.1, 1.0

Silicon Errata

Literature Number: SPRZ329F

November 2010–Revised March 2015

Contents

List of Figures

List of Tables

Silicon Errata
SPRZ329F–November 2010–Revised March 2015

TMS320DM816x DaVinci™ Digital Media Processors Silicon Revisions 2.1, 2.0, 1.1, 1.0

1 Introduction

This document describes the known exceptions to the functional specifications for the DM816x DaVinci Digital Media Processors. The updates are applicable to the CYG package. For additional information, see the *TMS320DM816x DaVinci™ Digital Media Processors* data manual (SPRS614).

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

1.1 Device and Development Support-Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320DM8168CYG). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX — Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

TMP — Prototype device that is not necessarily the final silicon die and may not necessarily meet final electrical specifications.

TMS — Production version of the silicon die that is fully qualified.

Support tool development evolutionary flow:

TMDX — Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS — Fully-qualified development-support product.

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Production devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

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

1.2 Package Symbolization and Revision Identification

Figure 1 shows an example of the DM816x processor qualified device package symbolization. For TMX variants of this device, the symbolization shown in Figure 2 is used. The device revision can be identified by the markings on the top of the CYG package; the "C" between the device number and the package identifier indicates the device revision (2.1), as noted in the figures.

Table 1 lists the device revisions for the DM816x processor.

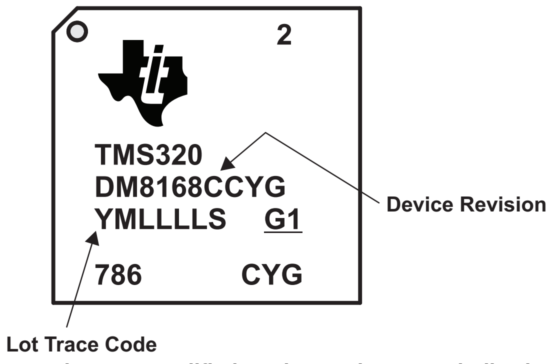

Figure 1. Qualified Device Package Symbolization

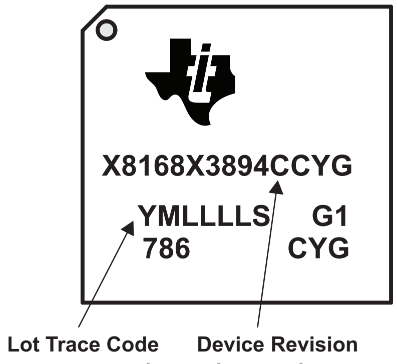

Figure 2. TMX Device Variant Package Symbolization

Table 1. Identifying Device Revision From Package Symbolization

DEVICE REVISION IDENTIFIER	DEVICE REVISION	COMMENTS
Blank	1.0	Initial device revision
A	1.1	Device revision 1.1
B	2.0	Device revision 2.0
C	2.1	Device revision 2.1

The VARIANT field of the JTAG ID Register (located at 0x4814 0600) changes between silicon revisions. Table 2 lists the contents of the JTAG ID Register value for each revision of the device. More details on the JTAG ID Register can be found in the *TMS320DM816x DaVinci™ Digital Media Processors* data manual (SPRS614).

Table 2. JTAG ID Register Variables

SILICON REVISION	JTAG ID REGISTER VALUE
2.1	0x3B81 E02F (VARIANT = 0011)
2.0	0x2B81 E02F (VARIANT = 0010)
1.1	0x1B81 E02F (VARIANT = 0001)
1.0	0x0B81 E02F (VARIANT = 0000)

1.3 ARM® Silicon Revision

Each DM816x silicon revision uses a specific revision of the Cortex®-A8 processor as shown in Table 3.

Table 3. ARM Silicon Revision

SILICON REVISION	ARM CORTEX-A8 VARIANT/REVISION
1.1 (revision A)	r3p2
2.0 (revision B)	r3p2
2.1 (revision C)	r3p2

2 Revision 2.1 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to revision 2.1 of the DM816x device.

2.1 Revision 2.1 Usage Notes

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

2.1.1 DDR2 and DDR3 Require Software Leveling

On all silicon revisions (except silicon revision 1.0), DDR2 and DDR3 require software leveling to tune the device IOs to the timing characteristics of a particular board design. Hardware leveling is not supported on these devices. For more information on software leveling, see the *TMS320DM816x/C6A816x/AM389x DDR3 Initialization With Software Leveling* application report (SPRABJ3) or click the following link to view the article in the TI Embedded Processors Wiki: http://processors.wiki.ti.com/index.php/DM816x_C6A816x_AM389x_DDR3_Init.

2.1.2 AVS Power Supply Requirements

The SmartReflex™ driver determines AVS voltage level at the DM816x CVDD power pin required to ensure proper operation. Along with the SR Driver, there are AVS power supply requirements for DM816x devices.

- The adaptive voltage scaling (AVS) is required for all DM816x devices.
- The AVS voltage must be within ±5% tolerance of the target voltage at all the BGA balls. This includes overshoot and undershoot due to load changes and broad spectrum AC noise.
- The power management IC (PMIC) maximum voltage should be considered as the sum of Vmax, worst case board DC voltage drop and low frequency regulator output AC ripple.
- Voltage remote sense for PMIC must be attached under the DM816x at CVDD and VSS.
- CVDD AVS Vmax operating range spec changes from 1.05V to 1.10V. For more detailed information on the CVDD Vmax voltage, see the TMS320DM816x DaVinci™ Digital Media Processors data manual (SPRS614).
- The PMIC maximum voltage must meet CVDD Vmax requirements depending on a speed grade of devices. For more detailed information on the CVDD Vmax voltage, see the TMS320DM816x DaVinci™ Digital Media Processors data manual (SPRS614).
- PMIC layout requirements must also be followed.

For more information on above power supply requirements, click the following link to view the article in the TI Embedded Processors Wiki: http://processors.wiki.ti.com/index.php/DM816x_Design_Resources

2.1.3 Initial Voltage Level Setting of CVDD AVS Rail Power Supplies Usage Note

Users are required to program their board CVDD AVS supply initial value to 1.00V or 1.05V on the device. The initial CVDD voltage at power-on will be 1.00V or 1.05V nominal and it must transition to AVS target value adjusted by a SR driver. This is required to maintain full power functionality and reliability targets specified by TI. For more detailed information on the CVDD Initial Startup voltage, see the *TMS320DM816x DaVinci™ Digital Media Processors* data manual (SPRS614).

2.1.4 PLL Programming Corrected to Operate Modules and Subsystems Within Rated Speed Without Exceeding the Maximum VCO Limit

The Main PLL configuration example listed in *Device Clocking and Flying Adder PLL* section in the "Chip Level Resources" chapter of the *TMS320DM816x DaVinci Digital Media Processors* Technical Reference Manual (SPRUGX8B) programmed the VCO to operate at 1593 MHz. Since this Main PLL configuration example did not support USB, customers have been configuring the PLL VCO to operate at 1728 MHz. The maximum rated speed for the PLL VCO is 1600 MHz. The Main PLL configuration in the *TMS320DM816x DaVinci Digital Media Processors* Technical Reference Manual (SPRUGX8) has now been revised to operate the PLL VCO at 1512 MHz so that it is within its rated range. This configuration also supports USB operation.

Similarly, PLL configurations which operated the VCO at 1728 MHz were implemented in the EZSDK and RDK example software. Again, this exceeds the maximum rated speed for the PLL VCO at 1600 MHz. Patch files have been created that correctly configure the PLLs to operate within their published ratings.

2.2 Revision 2.1 Known Design Exceptions to Functional Specifications

Table 4 lists the device revision 2.1 known design exceptions to functional specifications. Advisories are numbered in the order in which they were added to this document. If the design exceptions are still applicable, the advisories move up to the latest silicon revision section. If the design exceptions are no longer applicable or if the information has been documented elsewhere, those advisories are removed. Therefore, advisory numbering in this section may not be sequential.

Table 4. Revision 2.1 Advisory List

Table 4. Revision 2.1 Advisory List (continued)

Advisory 2.1.1 *Error Interrupt for PCIe EP Mode Not Generated*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: PCI Express® (PCIe®) errors observed on the device configured as endpoint (EP) do not result in interrupts to the local host (A8). As per the PCIe specification, a non-posted PCIe request from the initiator, which is interpreted as an unsupported request at the completer, results in an error indication being sent to the initiator. Currently, the device in EP mode does not generate an interrupt to the local host (A8) on receiving such an error. The same behavior is applicable for completer abort (CA) errors. This issue only impacts non-posted transactions. Memory writes do not have this issue since they are always posted.

Workaround: Software running on EP should check Received Master Abort and Received Target Abort bits in the STATUS_COMMAND register status after completing every request that would be initiated as a non-posted request over the PCIe link. This is particularly applicable to all PCIe reads initiated from A8 on EP.

To be specific, all reads initiated from A8 on EP over PCIe need to be followed by error status checks to ensure the sanity of read data. Note that this is not continuous polling and the error status should be checked only at the end of transaction. This should not impact performance in data transfer scenarios where EDMA is used to initiate transfers and the software running on EP relies on DMA completion interrupts.

Advisory 2.1.16 *Spurious RX_SOP_STARVATION Interrupt on the First CPPI DMA RX Descriptor Following USB Module Reset*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: A spurious RX_SOP_STARVATION interrupt (bit 0 of IRQ_STATUS_RAW) occurs on the first CPPI DMA RX descriptor fetch following a USB module reset. This issue is not dependent on the free queue number used or the RX DMA channel number. The spurious interrupt occurs only after the first CPPI DMA RX descriptor fetch and is not repeated.

Workaround: The software should ignore only the first RX_SOP_STARVATION interrupt.

Advisory 2.1.17 *GPMC Uses Bad Generator Polynomial in t=4 BCH Mode (t is Number of Correctable Errors)*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In mode t = 4, GPMC uses the wrong generator polynomial (0x14523043AB86A9) instead of a *good* generator polynomial (0x14523043AB86AB), where bit 1 is incorrect. This results in the following:

- On page write, it generates incorrect ECC parity.
- On page read, it generates an incorrect syndrome.

Workaround: There is no workaround for this issue. It is recommended to use the NAND flashes that need 8-bit or 16-bit ECC.

Advisory 2.1.18 *CPGMAC 1-Gbps Mode Does Not Work When EMAC_TXCLK is Not Running*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: Although EMAC_TXCLK is specific to the 10/100Mbps clock, if it is not running, then the 1-Gbps mode does not work.

In Ethernet boot, when the board is powered on, the Ethernet PHY chip auto-negotiates and establishes a link at either 10/100/1000 Mbps speed. If the link is established at 1 Gbps, the Ethernet boot does not work for PHY chips that do not provide the EMAC_TXCLK clock signal. According to the GMII specification, the EMAC_TXCLK signal is not required to be provided by the PHY for 1-Gbps mode of operation, hence some of the PHYs may not provide this clock. In these cases, the Ethernet boot fails.

Workaround: Use the PHY chip that outputs the transmit clock to MAC (EMAC_TXCLK pin) (for example, ET1011C PHY).

Ensure that the PHY does not auto-negotiate to 1 Gbps by default, until boot occurs. At a later stage, the second-level bootloader or OS driver can enable 1-Gbps mode in the PHY via MDIO and restart auto-negotiation to switch to gigabit mode. A PHY chip might provide an input pin to disable/enable 1-Gbps mode by default, which can be overridden by using MDIO register settings.

Software Workaround

EMAC requires a clock on EMAC_TXCLK only on initialization.

Enable EMAC as PHY (can be written only via MDIO), then disable auto-negotiation and force 100-Mbps full-duplex GMII copper mode so that the PHY starts outputting the clock on EMAC_TXCLK. Then restart the EMAC so it is reinitialized while the clock is running. After that, auto-negotiation is enabled by the generic driver and the Ethernet works in both U-Boot and Linux in all modes.

Advisory 2.1.19 *Kernel Crashes in Software While Accessing UART RHR Register*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In the software while accessing the UART receive holding register (RHR) register, the kernel crashes. If the RX FIFO is empty before a UART RHR read and the TX FIFO is full before a UART THR write, an error response results.

Workaround: The software should ensure that the RX FIFO is not empty before reading RHR and the TX FIFO not full before writing to THR.

The RHR read on receive FIFO empty can be avoided as follows:

1. Read the UART Line Statue Register (LSR).
2. Check to see if the Receiver Data Ready (DR) bit (bit 0) is set to 1.
3. If yes, proceed to read RHR, otherwise there is no data to be read and the RHR read should be skipped.

The THR write on transmit FIFO full can be avoided as follows:

1. Read the UART Line Statue Register (LSR).
2. Check to see if the Transmit Hold Register Empty (THRE) bit (bit 5) is set to 1.
3. If yes, proceed to write to THR, otherwise loop until THRE is set before writing to THR.
4. It may be desirable to implement a few milliseconds timeout for the loop to poll to avoid long busy waits for THR.

Advisory 2.1.20 *DDR3 Hardware Leveling Does Not Function Reliably*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In revision 1.0, DDR3 does not function.

In revision 1.1 and later, DDR3 hardware leveling does not function reliably.

Workaround: In revision 1.0, we recommend using DDR2 until the DDR3 issue is fixed in a future revision.

In revision 1.1 and later, DDR3 can be used with software leveling. For more details on using software leveling, see Usage Note, *DDR2 and DDR3 Require Software Leveling*.

Advisory 2.1.32 *RGB to YUV or YUV to RGB Inline Within HDVPSS VIP May Lead to VIP Path Lockup if DDR Bandwidth is Overconsumed*

Revision(s) Affected: 2.1, 2.0, 1.1

Details: Video capture of RGB data can be converted to YUV data within the HDVPSS VIP. RGB data is first converted to YUV444 data, then converted to YUV422 data using a 444-to-422 converter. Video capture of YUV422 data can be converted to RGB data within the HDVPSS VIP. YUV422 data is first converted to YUV444 data using a 422-to-444 converter, then converted to RGB data. The 422-to-444 converter requires a minimum of four pixels per line or it may lock up. The 444-to-422 converter requires a minimum of nine pixels per line or it may lock up.

Normal video does not have this few lines per frame, but it is possible that a momentary DDR bandwidth reduction can cause this lockup to occur. When a lockup occurs, it is always proceeded by a VIP_PARSER overflow event. This can only happen in single-channel capture modes when RGB is being converted to YUV inline, or YUV is being converted to RGB inline. Frequency is dependent on DDR loading. There are 4K bytes of buffering for captured video data. Overflow at the VIP_PARSER, which can lead to this issue, only occurs if this 4K-byte buffer overflows.

Workaround: Disable and re-enable the VIP port following this sequence:

1. Disable the VIP port being used.
2. Wait for completion of the current frame capture.
3. Put the VIP port being used in reset (bit 23 of port A and port B VIP registers).
4. Clean the VPDMA channels that are being used for capture by posting an ABORT descriptor and waiting for the list completion.
5. Bring the VIP port out of reset.
6. Start the VIP port.

Advisory 2.1.33 *When HDVPSS VIP is Configured to Write to Tiled Memory Space, Output Descriptors Generated Also Write to Tiled Space*

Revision(s) Affected: 2.1, 2.0, 1.1

Details: If HDVPSS VIP is configured to write frame data to tiled memory space, the descriptors it outputs are also written to tiled space. The descriptor does not overwrite video data. The only issue (if it is an issue) is that the descriptors are written to tiled DDR space. You cannot make HDVPSS VIP write video data to tiled space and the corresponding descriptors write to nontiled space.

Workaround: The application can read frame height and width from external decoders instead of depending on the hardware/driver to provide these values.

Advisory 2.1.34 *DEMMU May Hang When Used in Table-Walk Mode*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When an access is made through DEMMU toward the end of a virtual page and the page (corresponding to the next incrementing virtual address range) is not resident in the TLB, the DEMMU may hang. This hang may cause the requestor (C674x or EDMA) to hang. All subsequent requests that target the MMU also hang. Once a hang occurs, the only recovery mechanism is to reset the device.

The hang occurs because an internal counter for the address increments without accounting for potential stalls. Because of this, a request that was actually contained within a page spills into the second page, which is a miss. The DEMMU is not designed to handle this page crossing and causes a hang.

Workaround: One of the following workarounds must be used to avoid the hang situation:

1. Leave DEMMU in bypass mode. The DEMMU can be set to bypass mode by either of the following methods:
   - Set MMU_EN = 0 (default value is 1, enabled) in the MMU_CFG register.
   - Set MMUENABLE = 0 (default value is 0, disabled) in the MMU_CNTL register.

   Note that by default on RESET, the DEMMU is in bypass mode since the MMUENABLE bit in the MMU_CNTL register is 0.

   For potential issues with this workaround, see *DEMMU Bypass - GPMC Accessibility Limitations* below.
2. Lock all required entries within the MMU.

   The MMU consists of 32 entries, where each entry can map either a 4KB, 64KB, 1MB, or 16MB range. Thus, this workaround is straightforward if the DSP accesses, at most, 32 different 16MB (or smaller) ranges. It is recommended that the final 1KB at the end of a contiguous range is not used to avoid potential hardware pre-fetching crossing to an unmapped page boundary.

DEMMU Bypass - GPMC Accessibility Limitations

In order for the DSP to access GPMC directly, the DEMMU must be used to remap the GPMC physical address range to a different virtual address. Therefore, Workaround 1 cannot be used and also support direct DSP access to the GPMC addresses. If Workaround 1 is desired, then the DSP can access the GPMC via indirect means, such as by submitting an EDMA request to access the GPMC. Alternatively, Workaround 2 can be used.

The reason for this limitation is that the DSP views virtual addresses between 0x0000 0000 and 0x10FF FFFF as local addresses (mapped to DSP L1D, L1P, L2, and control registers). The L3 interconnect maps a part of GPMC addresses to the same range. In order for the DSP to directly access GPMC, the DEMMU must be used to remap a chosen virtual address (say 0x2000 0000) to the GPMC physical address of 0x0000 0000.

MMU Lockdown Example:

- Application usage needs from DDR:
  - 32M GPMC (0x0000 0000 - 0x007 FFFFF)
  - 64M DDR (0x8000 0000 - 0x80 FFFFFF)
- MMU Entries:
  - Entry 0: Super-section: GPMC: 0x0000 0000 - 0x003F FFFF
  - Entry 1: Super-section: GPMC: 0x0040 0000 - 0x007F FFFF
  - Entry 2: Super-section: PAD: 0x0080 0000 - 0x00FF FFFF
  - Entry 3: Super-section: DDR: 0x8000 0000 - 0x803F FFFF
  - Entry 4: Super-section: DDR: 0x8040 0000 - 0x807F FFFF

– Entry 5: Super-section: DDR: 0x8080 0000 - 0x80BF FFFF
– Entry 6: Super-section: DDR: 0x80C0 0000 - 0x80FF FFFF
– Entry 7: Super-section: PAD: 0x8100 0000 - 0x813F FFFF

NOTE:

1. The DEMMU has only 32 TLB entries. This may limit the usable address range. For example, using super-sections (16MB), the overall addressable range can be 32*16MB = 512MB. The DEMMU allows the TLB entry size to be super-section (16MB), section (1MB), large page (64KB) and page (4KB).
2. An OS (such as Linux) often requires full MMU functionality including the use of hardware table walks (for page misses) and small pages. In this case, crossing page boundaries is unavoidable. Therefore, if the DEMMU is used, this approach would require that any buffers shared between A8 and C674x need to be allocated from a contiguous memory rather than allocated in small pages.

Example Code:

```
#define __raw_readl(a)                  (*(volatile unsigned int *)(a))
#define __raw_writel(v, a)              (*(volatile unsigned int *)(a) = (v))
#define __raw_readw(a)                  (*(volatile unsigned short *)(a))
#define __raw_writew(v, a)              (*(volatile unsigned short *)(a) = (v))

#define SYS_MMU_BASE_ADDR                                   0x48010000
#define MMU_TTB     (SYS_MMU_BASE_ADDR + 0x4C)
/* IKD new defines for more MMU registers */
#define MMU_LOCK    (SYS_MMU_BASE_ADDR + 0x50)
#define MMU_LD_TLB  (SYS_MMU_BASE_ADDR + 0x54)
#define MMU_CAM     (SYS_MMU_BASE_ADDR + 0x58)
#define MMU_RAM     (SYS_MMU_BASE_ADDR + 0x5C)

#define MMU_LOCK_BASEVALUE_LSB     10
#define MMU_LOCK_CURRENTVICTIM_LSB 4

#define MMU_CAM_V 0x4
#define MMU_CAM_P 0x8

#define MMU_CAM_SECTION      0x0
#define MMU_CAM_LARGEPAGE    0x1
#define MMU_CAM_SMALLPAGE    0x2
#define MMU_CAM_SUPERSECTION 0x3

#define MMU_CAM_VATAG_MASK   0xfffff000
#define MMU_RAM_PHYADDR_MASK 0xfffff000
/* IKD */

#define TTB_ADDR                                         (0x80000000+54*1024*1024)

#define MMU_TTB_MASK          0xFFFFC000
#define MMU_SECTION_ADDR_MASK 0xFFF00000
#define MMU_CNTL     (SYS_MMU_BASE_ADDR + 0x44)

#define GPMC_PHYSICAL_ADDR    0x08000000
#define GPMC_VIRTUAL_ADDR     0x11000000
#define OCMC0_PHYSICAL_ADDR   0x40300000
#define OCMC0_VIRTUAL_ADDR    0x40300000
#define EMIF0_PHYSICAL_ADDR   0x80000000
#define EMIF0_VIRTUAL_ADDR    0x80000000

/* IKD */
#define SUPERSECTIN_SIZE      0x1000000
void mmu_setup_locked_entries();
void mmu_lock_section(int curr_entry, unsigned long va, unsigned long pa);
void main() {
                SYS_MMU_TWL();
}
void mmu_lock_section(int curr_entry, unsigned long va, unsigned long pa)
{
  unsigned long lock;

  /* point to the entry & lock entries till the same */
  lock =
    ((curr_entry + 1) << MMU_LOCK_BASEVALUE_LSB) |
    (curr_entry << MMU_LOCK_CURRENTVICTIM_LSB);
  __raw_writel(lock, MMU_LOCK);

  /* setup CAM and RAM */
  __raw_writel((va & MMU_CAM_VATAG_MASK) | MMU_CAM_V | MMU_CAM_P |
                    MMU_CAM_SUPERSECTION, MMU_CAM);

  __raw_writel((pa & MMU_RAM_PHYADDR_MASK), MMU_RAM);

  __raw_writel(1, MMU_LD_TLB); /* load an entry */

}
/*
 * lock some entries in the MMU so TWL does not have to happen
 */
void mmu_setup_locked_entries()
{
  int curr_entry = 0;
  int i;

  /* GPMC - one extra section at the end */
  for(i=0; i<2; i++){
    mmu_lock_section(curr_entry++,
                                    GPMC_VIRTUAL_ADDR + i*SUPERSECTIN_SIZE,
                                    GPMC_PHYSICAL_ADDR + i*SUPERSECTIN_SIZE);
  }

  /* DDR - one extra section at the end */
  for(i=0; i<9; i++){
    mmu_lock_section(curr_entry++,
                                    EMIF0_VIRTUAL_ADDR + i*SUPERSECTIN_SIZE,
                                    EMIF0_PHYSICAL_ADDR + i*SUPERSECTIN_SIZE);
  }

  /* OCMC */
  mmu_lock_section(curr_entry++, OCMC0_VIRTUAL_ADDR, OCMC0_PHYSICAL_ADDR);
}

/* IKD */
int SYS_MMU_TWL()
{
                /* IKD */
                mmu_setup_locked_entries();
                /* IKD */
                __raw_writel(0x2,MMU_CNTL);

                return 0;
}
```

Advisory 2.1.35 *On-Chip HDMI Does Not Operate in 8-/16-Bit Mode*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The HDVPSS VENC_D has 3 x 8-bit data buses [red (R), green (G), blue (B)] that connect to both VOUT1 and on-chip HDMI.

In 16-bit output mode, the G bus is used for Y data and the B bus is used for Cb/Cr interleaved data. VOUT1 connects to the G and B buses properly. The on-chip HDMI wrapper/PHY uses the G bus for Y correctly, but it uses the R bus instead of the B bus for Cb/Cr, so the HDMI never receives the Cb/Cr data. The HDMI display shows the wrong colors on the TV in 16-bit mode.

In 8-bit mode, Y as well as Cb/Cr data is sent over the G bus in interleaved format. The HDMI wrapper expects Y/Cb/Cr data on the G bus. However, HDMI needs to operate at 2x pixel clock since Y and C are interleaved on the same 8-bit bus. On-chip HDMI can operate only at 1x pixel clock; therefore, HDMI cannot operate in 8-bit mode.

On-chip HDMI has no issue to operate in 24-bit mode. This error mainly affects those use cases that want to operate both on-chip HDMI and VOUT1 simultaneously in 8-/16-bit modes to show the same video content.

Workaround: For those use cases that want to operate both on-chip HDMI and VOUT1 simultaneously to show the same video content:

1. Tie on-chip HDMI with on-chip HD_COMP DAC to show the same video content simultaneously.
2. Connect both an external HDMI transmitter and an external Video DAC to the VOUT0 output to show the same video content simultaneously.

Advisory 2.1.36 *SERDES Transit Signals Pass ESD-CDM Up To ±150 V*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The device meets all datasheet specifications associated with the SERDES high-speed functional pins to a Charged Device Model (CDM) threshold of ±150 V. Due to the sensitive nature of the SERDES high-speed pins, a robust ESD control program that mandates ESD safe handling methods is highly recommended during assembly and test operations.

The following is the list of pins that fall into this category:

- PCI Express (PCIe) transmit data pins:
  – PCIE_TXN0 (AB30)
  – PCIE_TXP0 (AB31)
  – PCIE_TXN1 (AB28)
  – PCIE_TXP1 (Y27)
- Serial ATA (SATA) transmit data pins:
  – SATA_TXN0 (T31)
  – SATA_TXP0 (T32)
  – SATA_TXN1 (U33)
  – SATA_TXP1 (V33)

Recommendations for ESD Safe Handling

An electrostatic discharge (ESD) control program is highly recommended for all processes that assemble and/or test ESD-sensitive devices. The industry accepted standards for ESD control are ANSI/ESD S20.20 (www.esda.org), JESD-625 (www.jedec.org) or IEC 61340-5-1 (www.iec.ch). In addition to the guidelines outlined in the aforementioned ESD control standards, the following additional ESD control methods are recommended:

1. Use of ionization on the printed circuit board (PCB) during assembly.
2. Use of grounded, conductive/dissipative suction cups on pick-and-place machines.
3. Use of dissipative materials for downholder pins and/or plastic covers as well as two-stage pogo-pins while performing in-circuit test.
4. Use of electrostatic voltmeter to measure voltages close to the IC package during handling.

Advisory 2.1.39 *Ethernet Boot May Function Unreliably*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When Ethernet boot mode is selected, and a power-on reset is applied, the ROM checks whether the link is up. The gigabit PHY on the EVM takes about a second to initialize but the ROM code waits only for 10 ms to check if the link is up. This causes the ROM to time out before the PHY is completely initialized. This results in the Ethernet boot mode failing. Some PHYs in the market can take up to 5 s to initialize.

Workaround: Use a boot mode where Ethernet follows UART. The UART attempts to boot first, and times out in 3 s (as there is nothing connected to the UART). This gives the PHY enough time to initialize before the Ethernet boot starts.

Advisory 2.1.40 *Link Speed Selection for Ethernet Boot is Not According to 802.3 Standard*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The ROM code relies on bit 6 of the control status register to determine whether the link speed was auto-negotiated to 1000 MHz or 100 MHz. However, the 802.3 specification states that, "it is not necessary for bits 0.6 and 0.13 to reflect the operating speed of the link when it is read."

While some PHYs update the link speed in these bits correctly, other PHYs do not update these bits as it is not mandatory according to the specification.

Workaround: Use PHYs that update bit 6 and 13 of the PHY control register correctly, based on the auto-negotiated link speed. Another option is to always use PHYs whose maximum speed is the speed of the network; i.e., use only 10/100 PHYs to connects to a 10/100 network and a gigabit PHY to connect to a gigabit network, and do not connect a gigabit PHY to a 10/100 network.

Advisory 2.1.41 *PCIe: PCIESS Slave Port Burst Destination Addresses Must be 16-Byte Multiples and 16-Byte Boundary Aligned*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: EDMA-to-PCIESS slave port write data is corrupted if 16-byte multiples and 16-byte boundary transfer rules are not met. Single-word (4-byte) accesses from any master (including EDMA) are not affected by this issue. Additional clarifications include:

- The Cortex-A8 supports only single-word access to the PCIESS and, therefore, any Cortex-A8 CPU access to the PCIe region does not cause this issue to occur.
- The EDMA can perform single-word (4-byte) access by programming ACNT=4. Note: In this case, the EDMA address ***must*** be 16-byte aligned.
- If the device DDR is the source and PCIe memory is the destination, then the EDMA source address ***need not*** be aligned, but the EDMA destination address ***must*** be aligned.
- If the device DDR is the destination and PCIe memory is the source, then the EDMA destination and source addresses ***need not*** be aligned.
- If this constraint is not met, then the system including the Cortex-A8 CPU may crash in unexpected ways.

Workaround: If the required EDMA 16-byte multiple and boundary rule is not desirable, then single-word accesses must be used.

Advisory 2.1.42 *PCIe: STATUS_COMMAND and SECSTAT Register Status Bits are Incorrect*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When a non-posted transaction is invoked between a requester (in RC mode) and a completer and the request packet is rejected by the completer for any reason, the completer transmits a completion TLP with an unsupported request as the reason for the rejection. This event is designed to be communicated to the user via both the status and command (STATUS_COMMAND) and secondary status and IO base/limit (SECSTAT) registers. However, the status bits within these registers are not updated to reflect the event status.

The "Received Master Abort" bit in the STATUS_COMMAND register and the "RX_MST_ABORT" bit in the SECSTAT register are not asserted when an RC port receives a "completion with unsupported request completion status".

The "Received Target Abort" bit in the STATUS_COMMAND register and the "RX_TGT_ABORT" bit in the SECSTAT register are not asserted when an RC port receives a "completion with completer abort status".

Workaround: Once the header portion of the completion transport layer packet (TLP) is parsed and placed within the PCIe header log registers, software can read the completion status field (HEADER_LOG1[bits 23:21]) to determine the captured error condition (for example, 001b for an *unsupported request (UR)* completion status and 100b for an *completer abort (CA)* status).

Software should read the completion status and this can be used for error detection.

Advisory 2.1.43 *DDR0/DDR1: LPDDR/DDR2/DDR3 Chip Select 1 ($\overline{\text{DDR[x]\_CS[1]}}$) Feature Not Supported*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: For both DDR0 and DDR1 modules, DDR[x]_CS[1] feature is not supported.

Workaround: Do not use the DDR[x]_CS[1] function. Configure the DDR0/DDR1 peripherals using DDR[x]_CS[0] *only*.

Advisory 2.1.44 *PCIe Gen2 Mode: PCIESS Corruption of Round-Trip Latency Time and Replay Time Limit Bits (PL_ACKTIMER Register)*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When the PCIe is operating in Gen2 mode (5-Gbps rate per PCIe link in each direction), writing to either of these bits, round trip latency time limit (RND_TRP_LMT) or the replay time limit (RPLY_LIMT), in the PL_ACKTIMER register causes the value of the bit field that was not being updated to also be modified, corrupting the register contents.

Workaround: Ensure that any updates to either the RND_TRP_LMT or the RPLY_LIMT bits in the PL_ACKTIMER register are made only when the PCIe is operating in Gen1 mode (2.5-Gbps rate per PCIe link in each direction).

Advisory 2.1.46 *HDVPSS: Enabling nd Disabling the VIP_PARSER Within the HDVPSS VIP in Single-Channel Capture and Discrete-Sync Modes Can Lead to VIP Overflow*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: This occurs only when in single-channel capture and discrete-sync modes.

In discrete-sync or single-channel capture mode, if the VIP_PARSER register within the HDVPSS VIP is disabled and re-enabled, processing elements following the VIP_PARSER register can lock up.

This error could cause the VIP port to lock up and stop capturing any data.

Workaround: Reset the VIP port by following this procedure:

1. Disable the VIP port.
2. Assert VIP reset in the HDVPSS CLKC register.
3. Assert the Async FIFO reset in the VIP parser register.
4. Make a list of abort descriptors for all VIP VPDMA channels.
5. Post this list and wait for it to complete.
6. De-assert the VIP reset in the HDVPSS CLKC register.
7. De-assert the Async FIFO reset in the VIP parser register.
8. Start the VIP capture port.

Advisory 2.1.47 *HDVPSS VOUT[x]_CLK: Does Not Support Positive-Edge Clocking*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: VOUT data transitions only on the negative (falling) edge of the clock.

The actual VOUT[x]_CLK to data/control delay times are now referenced to the falling edge and, therefore, timing parameters specified in the device-specific data manual have changed.

For more detailed information on the timing parameter changes, see the *TMS320DM816x DaVinci™ Digital Media Processors* data manual (SPRS614).

Workaround: There are two options for a workaround:

1. The external device connected to the HDVPSS interface can capture data on the rising edge.
2. External logic can be used to invert (*or delay*) the clock to provide adequate setup and hold times to meet the requirements of the external device.

Advisory 2.1.48 *SIGBUS Fault Under QNX When Accessing Last 48 Bytes of Physical Memory*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: SIGBUS faults have occurred while running the QNX operating system. If the last 48 bytes of physical memory are considered cacheable and an access is made to one of the bytes and it generates a cache miss, the subsequent cache line fill would cause a SIGBUS fault or *data abort* on the access.

Workaround: The data abort can be avoided if the specific cache line fill scenario can be avoided. The fundamental workaround is to ensure that the software executing on the Cortex-A8 does not make a cacheable access at the end of a local interconnect and synchronization agent (LISA) mapping [for architecture details, see the DMM Functional Description section of the *TMS320DM816x DaVinci Digital Media Processors* Technical Reference Manual (SPRUGX8)]. Options include configuring the high-level operating system (HLOS) not recognize to memory at the end of a LISA mapping or making the cumulative effect of the LISA mappings exceed the physical size of external memory, or a combination of both. The solution in workaround 4 is preferred as a general solution for systems that have less than 2GB of memory. It provides a LISA mapping that would exceed the amount of physical memory and it allows the remaining LISA mappings to precisely reflect actual memory. It also allows HLOS configuration to precisely reflect the actual memory. A variation of mappings that achieves the same effect as workaround 4 would be equally preferred. The solution in workaround 2 is required if the system has 2GB of memory.

1. **Ensure that the last MMU page of a LISA mapping is uncacheable.**

   This capability cannot be readily ensured by the HLOS since an application is typically free to allocate memory and declare it cacheable. A customized memory manager would be required. This solution could cover the entire 2-GB physical memory address space for DDR2/3 less the reserved MMU page(s).

2. **Ensure the last MMU page size of memory in a LISA mapping is reserved from the HLOS.**

   An HLOS provides a means for defining the location and size of all physical memory. Subsequently, the HLOS can be configured not to recognize a block of memory at the end of physical memory; therefore, a cache line access cannot be made to it. If the LISA mapping covers the entire physical memory, it will exceed the physical size recognized by the HLOS. The smallest block of memory that can be withheld should be aligned on an MMU page boundary. The smallest MMU page for the Cortex-A8 is 4KB.

   Note: The HLOS must ensure that an MMU mapping can never be made to the reserved space. This solution could cover the entire 2-GB physical memory address space for DDR2/3 less any reserved blocks of memory.

   Example: For 64 MB of physical memory, the last 4K bytes are withheld from the HLOS memory manager. The LISA mappings will cover the actual physical memory.

3. **Configure the LISA mapping to a size larger than the underlying physical memory.**

   If using 64M bytes of physical memory, then configure the LISA mapping for 128M bytes but only allow the HLOS to recognize/access the actual 64M bytes of physical memory. If the access is not to actual physical memory it is in error and the MMU would have to ensure that the access is caught and an exception issued. This solution can only cover up to 2GB less 128MB of physical memory.

   Example: For two EMIFs with 64MB of memory on each to be configured as two regions at 0x80000000 and 0xC0000000:

   - MAP_0 and _1 value would be 0x80300100 (128MB at 0x80000000).
   - MAP_2 and _3 value would be 0xC0300200 (128MB at 0xC0000000).

   The HLOS only recognizes 64MB at both 0x80000000 and 0xC0000000.

4. **Configure the lowest level LISA mapping (MAP_0) to cover the entire DDR2/3 external address range and force the EMIF to generate a legal data abort.**

   Higher-level LISA mappings will accurately describe actual physical memory while the lowest level mapping (MAP_0) will exceed the size of all physical memory and preclude the data abort scenario. MAP_0 will cover the full 2-GB space and directs any access to a reserved space within the EMIF. The DMM will first map an access according to MAP_1, _2 or _3 settings. These higher-level mappings only cover actual physical memory. If the access is not to actual physical memory it is in error and MAP_0 will cover it and map it to a reserved space in the EMIF which will generate a data abort. A data abort on an access to nonexistent memory is a legal fault and will generate an exception to the HLOS. This solution could cover a physical memory address space of 2GB less 128MB.

   Example: For two EMIFs with 64MB of memory on each to be configured as two regions at 0x80000000 and 0xC0000000:

   - MAP_0 value would be 0x80720100.
   - MAP_1 value would be 0x80200100 (64MB at 0x80000000).
   - MAP_2 and _3 value would be 0xC0200200 (64MB at 0xC0000000).

   The HLOS only recognizes 64MB at both 0x80000000 and 0xC0000000.

Advisory 2.1.49 *DMA Queue Priority for DSP SDMA*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The device does not support EDMA queue priority feature for C674x SDMA transactions due to an error in the mapping of open-core protocol (OCP) sideband signals during conversion to VBUS signals.

Workaround: There is no workaround for this issue. It is recommended not to use the EDMA queue priority feature.

Advisory 2.1.53 *Discrete Sync Capture Mode: When ACTVID Does Not Toggle and VIP Scaler is Used, Output is Affected*

Revision(s) Affected: 2.1, 2.0

Details: This discrete sync mode (ACTVID not toggling during blanking) was added to revision 2.0. In this mode, the capture port does not send the end of frame condition to the downstream modules until the start of the next frame. To the downstream modules, this appears as an end-of-frame condition immediately followed by a start-of-frame condition.

The VIP scaler resets itself on every frame that is passed through it. The reset begins when it receives the end-of-frame condition, and continues for ~2 lines due to its internal pipeline. While in this state, it stops requesting data from the capture port. The capture port has a very small FIFO to buffer data (32 pixels), which means that this FIFO overflows at the start of every frame when the capture port is in discrete sync mode and the VIP scaler is being used.

When the capture port is in the overflow state, it does not send pixel data to the scaler, so the scaler receives a short line at the beginning of each frame. The scaler generates a rectangular output regardless of the input data being fed to it. Because the first line is short, the beginning of the next line starts at the end of the first line, and this continues for the rest of the frame.

Depending on exactly what the capture port frequency is set to, an even or odd number of pixels could be dropped from the first line. Since the input source is YUV422, if the number of pixels dropped is odd, the Cb and Cr components of the video get swapped, resulting in color corruption.

Workaround:

- Use the discrete sync video mode where ACTVID toggles during blanking.
- If this mode is unavailable, use either the VIP_PARSER trimmer function to trim the last line from the captured video, or use the VIP scaler trimmer function to trim the last line from the captured video:
  - Using the trim function creates a separation between the end-of-frame condition and start-of-frame condition from the capture port, allowing the scaler to successfully reset.
  - If the VIP port is configured to capture a single stream and send only the downscaled version to memory, then either the VIP_PARSER or VIP scaler trimmer can be used. Because the scaler is reducing the number of vertical lines due to downscaling, trimming the last line has a negligible effect on the output.
  - If the VIP port is configured to capture a single stream and se nd both a downscaled and non-scaled version to memory, then use the VIP scaler trimmer. If the VIP_PARSER trimmer is used, then the non-scaled output will have one line missing, which may be undesirable.

Advisory 2.1.54 *VPDMA Line Limit Feature: Descriptor Reports All Fields as Even, but Captures 30 Even and 30 Odd Fields in Memory*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The VPDMA input descriptor can be set to only capture a specific height and width of input. This is typically used to keep random captured data that is too large from corrupting adjacent memory regions.

If this feature is used, the input video source is interlaced, and the size of the incoming video is larger than what was set, the field ID reported in the output descriptor from VPDMA is 0 for all fields that are larger than the specified height/width settings.

Workaround:

- If larger video is coming in than what the VPDMA was configured for, it is difficult to know if this is a software programming mistake or some other problem.
- The field ID (FID) of each input field is captured in the VIP_PARSER capture port. If software believes the source is interlaced and receives two fields with the same FID, it can query the VIP_PARSER to determine the correct FID. After one such check, it can auto-toggle the FID, or continue to check the VIP_PARSER.

Advisory 2.1.55 *VIP Inline Color Space Converter (CSC): In Interlaced Embedded or Discrete Sync Mode, Descriptor Reports All Fields as Even, but Captures 30 Even and 30 Odd Fields In Memory*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The CSC does not pass the field ID information correctly to the VPDMA. If the CSC is being used, the FID reported to the VPDMA and thus in the output descriptor is always 0. The CSC would be used to either:

1. convert interlaced RGB capture data (not a common format) to interlaced YUV422 or YUV420 format (not a common capture type)

   or
2. convert interlaced YUV422 capture data to interlaced RGB format (not a common format).

Workaround: The FID of each input field is captured in the VIP_PARSER capture port. If software believes the source is interlaced and receives two fields with the same FID, it can query the VIP_PARSER to determine the correct FID. After one such check, it can auto-toggle the FID, or continue to check the VIP_PARSER.

Advisory 2.1.56 *High-Quality Scaler Does Not Work For Images With Width Smaller Than 34 Pixels*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The high-quality scaler in the high-quality de-interlace path requires a minimum line width of 34 pixels to operate properly. If it is less than 34 pixels, it locks up.

Workaround: Ensure memory-to-memory or memory-to-display paths that use the high-quality scaler have a line width greater than 34 pixels.

Advisory 2.1.57 *Simultaneous Display of Internal HDMI and VOUT[1] Not Supported*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The same video encoder within the HDVPSS drives both the internal HDMI and the VOUT[1] output. HDVPSS is not properly connected to the internal HDMI when the HDVPSS video encoder is configured in YUV422 mode.

Because VOUT[1] is only pinned out as 16 bits, YUV422 is the only output mode it supports.

Because of the connection problem, it is not possible to drive 422 data from HDVPSS to both the internal HDMI and VOUT[1] at the same time.

RGB or YUV444 works properly between HDVPSS and the internal HDMI.

Workaround: Use VOUT[0] to simultaneously show the same contents as the internal HDMI.

Advisory 2.1.60 *Occassionally, During Connect/Disconnect and When Line or Width Limit Feature Not Used, Any Memory Areas Can Be Overwritten*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When connect/disconnect events occur, it is possible for any size frame to be captured by HDVPSS VIP. If the captured frame is corrupted by the connect/disconnect event such that a very long or very wide frame is captured, and the VPDMA maximum width/maximum height feature is not used (set to unlimited), memory areas outside of those allocated by software can be written to.

If adjacent memory areas are program code, this can cause random behavior depending on exactly what is written.

Workaround: Ensure the VPDMA maximum width/maximum height values are set to something other than *unlimited*. However, the maximum size to which these can be set is 1920x1080.

Advisory 2.1.62 *Discrete Sync Interlaced Output Mode: Vertical Sync Output For Odd Fields May Not Correctly Detect Video Signal*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In discrete sync interlaced output mode, the HDVPSS Video Encoder (VENC) outputs the vertical sync (VS) pulse at the end of every field output, just like in progressive format. VS is supposed to be output in the middle of the last line for the odd field in interlaced format.

Because it is not positioned correctly, some devices connected when the VENC is in this format cannot detect the video signal correctly.

Workaround: For any interlaced video output, use embedded sync output mode.

Advisory 2.1.65 *Watchdog Timer (WDT): Watchdog Timer Generates Reset When Enabled For First Time After Power-On Reset*

Revision(s) Affected: 2.1, 2.0

Details: As per device specification, the WDT default timeout is 2 ms. The ROM code that runs after a warm reset takes longer than 2 ms to execute. The WDT is stopped, by default, at reset time and the ROM code does not use the WDT functionality. Enabling the WDT can be done by writing to the control module register by writing WPOPDISABLE=0 in the CONTROL_SEC_CTRL register. Once the WDT is enabled, it issues a reset in 2 ms.

On reset, the ROM code executes again and this time as WPOPDISABLE=0, the watchdog is stopped. The application code can upload a new value into the watchdog and start it again.

Workaround: The reset happens only the first time the WDT is enabled after power-on reset. The WDT must be enabled once early on in the boot process, so that the system restarts again. The same WDT configuration code is run once more, and then works normally. This way the impact to boot time is minimized. If the WDT is initialized later by the operating system, a reset occurs immediately and then the whole system has to reboot itself, causing unwanted side effects.

Advisory 2.1.66 *PCI Express (PCIe): PCIe Boot Fails When Connected to Some PCs*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The ROM code cannot handle hot-reset or disable-link packets. It has been observed on some motherboards on some slots, that the PC BIOS issues hot-reset or disable-link transactions on the PCIe bus. When the device sees these transactions, the PCIe controller gets reset automatically.

This reset causes the PCIe controller to lose the configuration that was programmed by ROM and, as a result, the PCIe boot does not complete. This issue is seen only when interfacing with some PCs. It is not seen when one device acting as root complex tries to boot another device acting as endpoint.

Workaround: If possible, use another slot on the PC. If it is not possible to change slots, do not use PCIe boot. Boot from a SPI flash and the code running from the SPI flash can configure the PCIe and also handle reconfiguring the PCIe controller when a hot reset or disable link is detected.

Advisory 2.1.67 *Boot :Ethernet Boot ROM Code Sends an Incorrect Vendor Class identifier in BOOTP Packet*

Revision(s) Affected: 2.1, 2.0

Details: When using Ethernet boot, the device ROM code should send a BOOTP request with a unique identifier to distinguish itself from other devices on the same network. Instead, the ROM code sends the same identifier, "DM814x ROM v1.0", for all devices (i.e., DM812x, DM814x, AM387x, DM816x, AM389x, and AM335x, etc.); hence, the download host attempting to bootstrap the devices can no longer determine which device is requesting the code to be downloaded. Applications using the DM812x, DM814x, AM387x, DM816x, AM389x, AM335x, DM38x, DM813x, DM810x, and DMVA3/4 devices cannot coexist in the same network if they are booted from Ethernet.

Workaround: None.

For some applications, it might be necessary to uniquely identify and service BOOTP packets from a client. The recommended approach to uniquely identify clients is to use the MAC address. Every device comes with a unique MAC address. A list of MAC addresses and the device type can be made available to the host in advance, so that the host can take device-specific action when it receives a BOOTP packet from a MAC address on the Host's list.

Advisory 2.1.70 *USB: Attached Non-compliant USB Device that Responds to Spurious Invalid Short Packet May Lock Up Bus*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The integrated USB PHY (analog transceiver) has a timing error that turns on its receiver too early and occasionally detects the end of its own transmit data as receive data. This causes the USB controller to transmit an invalid short packet. Normally this invalid short packet would be ignored by the attached USB device and the data transmission would continue as expected.

At least one mass storage class USB device has been found to be non-compliant to the USB specification, by responding to this packet. This non-compliant response (NACK) to the invalid short packet violates USB protocol and causes the bus to hang. Poor signal integrity of the differential signal pair used to connect the attached USB device may contribute to this issue. Impedance discontinuities and mismatched terminations on the differential signal pair may cause reflections to propagate longer than expected, which allows the transceiver to detect these reflections of its own transmit data as receive data.

Workaround: There is no workaround for this issue.

To prevent an unexpected response to any invalid short packets, attach only USB devices that are compliant with the USB specification. To minimize reflections, it is also recommended that the USB DP and DM signals are routed as a 90-Ω differential pair transmission line with minimum impedance discontinuities and proper terminations.

Advisory 2.1.72 — *ROMCODE: ROM Code Does Not Support Booting From eMMC Devices of Size 4GB or More*

Revision(s) Affected: 2.1

Details: During eMMC initialization, the ROM code queries the card for operating conditions by issuing CMD1 with ARG = 0. The fixed pattern response expected from the MMC card, if the card capacity is greater than 4GB in size, is 0x**40FF** 8080. As stipulated in Section 7.4.2, *Operating voltage range validation*, of the JEDEC Embedded MultiMediaCard (e·MMC) industry standard (JESD84-A441).

eMMC devices currently available in the market only adhere to Section 7.4.3, *Access mode validation (higher than 2GB of densities)*, and do not adhere to Section 7.4.2, *Operating voltage range validation*, of the JEDEC eMMC standard. When the ROM code issues a CMD1 with ARG = 0, current eMMC devices respond with a fixed pattern of 0x**00FF** 8080, even if the eMMC card capacity is of 4GB or more. Because of this, the initialization fails.

Workaround: There are two workarounds for this issue:

1. If use cases permit, use an MMC device which is less that 4GB in size.
2. If capacity of 4GB or more is needed, use eSD instead of eMMC.

Advisory 2.1.76 UART: Extra Assertion of the FIFO Transmit DMA Request, UARTi_DMA_TX

Revision(s) Affected: 2.1

Details: When the UART transmit FIFO reaches 64, a TX_fifo_full signal is asserted. The UART TX_fifo_full signal is asserted asynchronously to the dma_request signal which is generated in the OCP clock domain. If a FIFO full condition is allowed, it is possible that an extra dma_request pulse can occur resulting in a TX sync error.

Workaround: This issue can be avoided by keeping the sum of the TX_THRESHOLD and TRIGGER_LEVEL ≤ 63 or 1 less than the FIFO size. If this TX_THRESHOLD sum condition is met, the transmit FIFO will not become full and the extra dma_request pulse will not occur.

TX_THRESHOLD + TRIGGER_LEVEL ≤ 63 (1 less than the FIFO size)

UART Registers Required

UART REGISTER	REGISTER DESCRITPION	UART OFFESET ADDRESS
EFR	Enhanced Feature Register	8h
LCR	Line Control Register and Register Portal	Ch
MCR	Modem Control Register	10h
SPR/TLR	Trigger Level Register	1Ch
SCR	Supplementary Control Register	40h
MDR3	Mode Definition Register 3	80h
TXDMA	TX_DMA_THRESHOLD Register	84h

Example setup for TX_THRESHOLD + TRIGGER_LEVEL = 63:

```
Set LCR[7:0] = BFh                // Enable EFR register access
Set EFR[4] = 1                          //Set ENHANCEDEN bit to enable writing to
MCR register

Set LCR[7:0] = 00h                // Enable FCR register access
Set SCR[6] = 1                         // Set TXTRIGGRANU1 bit to set TX trigger
level granularity to 1
Set MCR[6] = 1                // Enable TLR register access
Set TLR[3:0] =0 and FCR[5:4] = 1      // Set Trigger Level to 1
Set MDR3[2] = 1                                  // Set to enable TX DMA threshold
setting
Set TXDMA[5:0] = 3Eh            // Set TX_THRESHOLD = 62
```

Advisory 2.1.99 — *USB: CPU Gets Stale Receive Data From the Data Buffer Located in DDR Memory*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When the CPPI DMA completes a receive data transaction it posts a write to the RX data buffer located in memory, posts a write to update the descriptor located in memory, and raises an interrupt to the CPU. When the system load is high and the write is to the DDR, the posted writes to the 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 the DDR.

Workaround: Initialize the datalength descriptor field to zero. The CPPI DMA updates this field, after the completion of an RX DMA operation, with the actual number of bytes received. To ensure the DMA operation is completed, the software should poll this field (in the ISR or in a deferred call context), until it becomes a non-zero value to ensure that the data buffer has been updated with the 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.

Advisory 2.1.100 — *USB: In Generic RNDIS CPPI DMA Mode, Receive DMA may Flag Premature Completion Under Certain Conditions*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In generic RNDIS CPPI DMA mode, when a short packet is detected on a certain endpoint, the DMA incorrectly broadcasts the same status to all other RX endpoints in progress. This may result in premature completion of the other RX endpoint descriptors leading to missed data.

Workaround: There are two workarounds for this issue:

1. Do not use generic RNDIS CPPI DMA Mode.
2. If using generic RNDIS CPPI DMA mode, the workaround involves monitoring transfer data size before and after the transfer. On receiving the DMA completion interrupt for a particular endpoint, if software detects that all the data as per the original transfer size has not been received yet, then place a new request for the remaining data.

Advisory 2.1.101 *USB: Data May be Lost When USB Subsystem is Operating in DMA Mode and More Than One Endpoint is Transferring Data*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In generic RNDIS mode, data loss may occur due to a USB data toggle synchronization error that occurs when an internal data toggle counter is erroneously reset from the DATA1 state to the DATA0 state while the USB subsystem is actively receiving data from more than one endpoint. The erroneous reset of the data toggle counter occurs because the associated logic in the USB subsystem DMA contains an error that does not support the correct data toggle update with data transfers from multiple endpoints. If the DATA1 state is erroneously reset to the DATA0 state immediately following a USB transaction in which the PID is DATA0, the transmitter and receiver become desynchronized. This data toggle synchronization error causes the receiver, per the USB specification, to silently discard the non-synchronized packet, which causes the Packet and any data contain therein to be lost.

NOTE: For more information related to the definition of DATA0 and DATA1 PIDs and functional requirements of data toggle synchronization, see sections 8.4.4 and 8.6 of the *Universal Serial Bus Specification Revision* 2.0.

Workaround:

- *Operating in USB host mode -*
  - Option 1: The workaround involves detecting and correcting the data toggle mismatch by software after receiving each USB packet. In order to implement this workaround, the CPPI4.1 DMA must be configured to operate in transparent mode; generic RNDIS mode cannot be used. Software must save the previous data toggle value then compare the current data toggle value and the saved value to detect a data toggle mismatch. If a synchronization error is detected, it must be corrected by simultaneously writing 1b to the data toggle write enable (DATATOGWREN) and data toggle (DATATOG) bits in the respective RXCSR registers. Since software is constantly monitoring the DATAx state as part of this workaround, this can potentially have a performance impact. However, the exact implications on the performance will have to be evaluated in a given system context.
  - Option 2: Assign each stream to a different Endpoint. In this mode multiple EPs (max 16) can be used, as long as there is one-stream-to-one-Endpoint communication. Operating in transparent mode is not required.
- *Operating in USB device mode* - DO NOT use RNDIS mode

Advisory 2.1.102 *USB: DMA Hung up in Frequent Teardowns*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: DMA may potentially hang up if a teardown is initiated by software while the endpoint is still active.

Workaround: The following software sequence must be followed BEFORE initiating teardown:

1. Clear the DMAEN bit in the RXCSR register.
2. Wait for 250 µs

Advisory 2.1.103 *USB: Missed PID and CRC Error Interrupt Causes*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: During isochronous receive transfers, the DMA clears PID and CRC errors before the CPU can read the error status bits.

The error will manifest itself as a phantom interrupt to the CPU since the source of interrupt shall be cleared by the DMA before the CPU sees it.

Workaround: During isochronous transfers, ignore error interrupts with no error bits set.

Advisory 2.1.104 *USB: CPU Gets Premature TX Transfer Completion Interrupt*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: TX transfer completion interrupt from the CPPI DMA is asserted when the DMA copies transmit data into the intermediate CPPI FIFO rather than waiting until the data transmission has completed (sent over the wire). This causes the application to get false notification on status (success) when in reality there can be an error (like NAK timeout).

This results in the following undesirable events:

- The CPU must handle additional error interrupts that may occur after data is sent on the wire.
- The application could allocate the same hardware endpoint used in the previous transfer to begin a new transfer to another device while that specific hardware endpoint is still transmitting valid data because the DMA copied the data into the intermediate CPPI FIFO and returned prematurely. In this case, valid data from the previous transfer could be truncated.

Workaround: In the DMA TX completion ISR (actually in a deferred call context), the completion status and corresponding endpoints should be held until the TXCSR indicates FIFO empty status then status is returned to the application.

Advisory 2.1.105 *SATA: Gen2 Link Fails With Some of GEN3 Devices*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When connecting a SATA GEN3 – capable target [for example, a Hard Drive (HDD) or Solid State Drive (SSD)] to a device with a SATA Host subsystem (after power-up or reset), the speed negotiation fails connecting at GEN2 speed between the two devices but a link is established at GEN1 speed. It will take some time to finish a link connection when it fails the speed negotiation. This interoperability issue can be seen with some of the DM816x devices.

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

Workaround: Use GEN2 or GEN1 maximum speed drives to avoid the issue or use GEN3 drives with jumper restricting capabilities to restrict the speed to GEN2.

Advisory 2.1.106 *SDIO: SD_SDCD Pin Used as an GPIO Input Affects SDIO Operation*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: When the SD Card Detect Input (SD_SDCD) terminal function is configured as a GPIO input via pin multiplexing control, even though the peripheral function selected is GPIO, external stimulus on this IO is not blocked from getting into the SDIO module. Therefore transitions on the SD_SDCD pin impact the SDIO functional system, this can cause the SDIO system to hang unexpectedly.

Workaround: There is no workaround for this issue.

Whenever the SDIO interface is used, the SD_SDCD pin must either be used as a SDIO Card Detect input or it must be held at a constant input level. It cannot be used as a GPIO input or output. The pin-mux control for the SD_SDCD pin must select the SDIO Card Detect input mode when the SDIO module is in use to prevent unexpected behavior.

Whenever the SDIO interface is not used, the SD_SCDC pin can be left unconnected and the internal pulldown resistor will hold it low. If any track or circuitry is attached to the SD_SDCD pin when unused, the SD_SDCD pin should be held static low by a weak pulldown resistor (for example, 10-kΩ) to ground. It can also be driven static low by logic.

Advisory 2.1.107 *McASP to EDMA Synchronization Level Event can be Lost*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: The McASP FIFO events to the EDMA can be lost depending on the timing between the McASP side activity and the EDMA side activity. The problem is most likely to occur in a heavily loaded system which can cause the EDMA latency to increase and potentially hit the problematic timing window. When an event is lost, the McASP FIFO RX path will overflow or the TX path will underflow. Software intervention is required to recover from this condition.

The issue results due to a state machine boundary condition in the McASP FIFO logic. In normal operation, when “Threshold” (set by the RNUMEVT and WNUMEVT bit fields in the McASP RFIFOCTL register) words of data are read/written by the EDMA then the previous event would be cleared. Similarly, when “Threshold” words of data are written/read from the pins, a new event should be set. If these two conditions occur at the same exact time (within a 2 cycle window), then there is a conflict in the set/clear logic and the event is cleared but is not re-asserted to the EDMA.

Workaround: Since the McASP is a real-time peripheral, any loss of data due to underflow/overflow should be avoided by eliminating the possibility of EDMA read/write completing at the same time as a new McASP Event.

Software should configure the system to:

1. Maximize time until the deadline for the McASP FIFO.
2. Minimize EDMA service time for McASP related transfers.

In order to maximize time until deadline, the RNUMEVT and WNUMEVT should be set to the largest multiple of “number of serializers active” that is, ≤ 32 words. Since the FIFO is 64-Words deep, this which gives the maximum time to avoid the boundary condition.

In order to minimize EDMA service time for McASP related transfers multiple options are possible. For example, McASP buffers can be placed in OCMCRAM or L2SRAM of the DSP (since on-chip memories provide a more deterministic and lower latency path compared to DDR memory) In addition, a dedicated Queue/TC can be allocated to McASP transfers. At minimum, care should be taken to avoid any long transfers on the same Queue/TC to avoid head-of-line blocking latency.

Advisory 2.1.108 *PLL: Maximum Operating Frequencies Reduced for Each Speed Grade Device*

Revision(s) Affected: 2.1, 2.0, 1.1, 1.0

Details: In order to comply with the maximum operating frequency imposed by the PLL, the maximum frequency limits for the main modules and subsystems have been corrected from the previously provided values. These modules include the ARM Cortex-A8, C674x DSP, and HDVICP2, as well as other processing subsystems.

The maximum operating frequencies are limited by the PLL configuration as well as the period jitter created by the chosen PLL settings. These limits were stated correctly in the *PLLs* and *SYSCLKs* sections of previous data manual (revision E and earlier) but not reflected clearly throughout the rest of the documentation suite. The *PLLs* and *SYSCLKs* sections have now been updated to contain more explanation and the tables now clearly reflect the maximum operating frequencies that comply with PLL design specifications.

Operation of any modules or subsystem with a clock that violates the maximum operating frequency may lead to unpredictable results.

Workaround: There is no workaround for devices that are operated above the rated performance specifications.

The software must be modified to program the PLLs such that the maximum operating frequency is not violated for any module or subsystem.

For more detailed information on the maximum operating frequencies for all of the modules and subsystems and the associated SYSCLKs, see the *PLLs* and *SYSCLKs* sections of the TMS320DM816x DaVinci™ Digital Media Processors data manual (SPRS614,(revision F and later)). Additional explanation and PLL programming examples for each speed grade can be found in the *Device Clocking and Flying Adder PLL* section in the "Chip Level Resources" chapter of the *TMS320DM816x DaVinci Digital Media Processors* Technical Reference Manual (SPRUGX8).

The EZSDK and RDK example software previously available contains PLL configurations that operate some modules and subsystems above their rated speed. Patch files have been created that correctly configure the PLLs to operate all modules and subsystems within their published ratings.

3 Revision 2.0 Known Design Exceptions to Functional Specifications

Table 5 lists the device revision 2.0 known design exceptions to functional specifications. All other known design exceptions to functional specifications for device revision 2.0 still apply and have been moved up to Section 2, *Revision 2.1 Known Design Exceptions to Functional Specifications*, of this document. Therefore, advisory numbering in this section may not be sequential.

Table 5. Revision 2.0 Advisory List

Advisory 2.0.2 *VIP Parser Output FIFO Overflow Bit is Set/VIP Port Locks Up When Capture is Run With Other Drivers (e.g., SC, Display)*

Revision(s) Affected: 2.0, 1.1, 1.0

Details: When running capture with other HDVPSS drivers such as SC and Display, the VIP parser port sometimes gets locked or hung up and the output FIFO overflow bit in the VIP parser gets set (in the VINx_PARSER FIQ Status register).

This condition occurs when the VPDMA inbound and outbound descriptor priority for Capture, SC, or Display and other drivers is set to 0 (highest priority). Due to this issue, capture data does not get written to DDR memory because the capture VPDMA stalls and causes the output FIFO to become full. Once the output FIFO becomes full, the VPDMA drops the request to the VIP parser and causes the VIP port to lock up.

Workaround:

1. Detect VIP overflow using the FIQ_STATUS register in the VIP parser.
2. Stop the VIP port.
3. Keep the VIP port in reset.
4. Program abort descriptors for all VIP VPDMA channels.
5. Take VIP out of reset.
6. Start the VIP capture port.

All of the above steps are taken care of in the driver via IOCTLs:

- IOCTL_VPS_CAPT_CHECK_OVERFLOW (step 1)
- IOCTL_VPS_CAPT_RESET_AND_RESTART(steps 2-6)

This resets both Port A and Port B for that VIP instance. All the VIP blocks including CSC, SC, CHR_DS for that VIP instance are reset.

Ensure that no module is being accessed in that VIP instance while performing the VIP reset, either in M2M path or capture path, when this API is called.

To minimize VIP overflows, the descriptor priority for VIP data should be set to 0 (highest), and all other descriptor priorities should be non-0. In addition, the list number on which the VIP capture list is running should be the lowest numbered list (list 0).

Advisory 2.0.29 — *Occasionally, Video Frame Data is Not Written When Performing Video Capture Using HDVPSS VIP in Multichannel Mode*

Revision(s) Affected: 2.0, 1.1

Details: When capturing a video frame using the HDVPSS VIP port, a data descriptor is programmed to the VPDMA. A data descriptor must be programmed for every frame that needs to be captured. Occasionally, even though a data descriptor is programmed to the VPDMA, the frame data may not get written to the DDR memory. This is seen in the pixel multiplexed mode of the capture for a 16-channel D1 case.

The frequency of occurrence of this issue averages about one frame per channel in 24 hours. The frequency of this issue depends on the amount of loading that is present in the HDVPSS. This issue has ***not*** been seen in single-channel mode operation of the VIP port.

An additional cause of video frame drop is found in revision 2.0 which results in one frame per channel being dropped every 30 minutes.

Workaround: Currently, there is no known workaround, but to detect this error, the HDVPSS drivers can be programmed to put a 32-bit pattern at line 10 of the data buffer.

- If this 32-bit word is overwritten, then it is assumed that a frame was captured and the frame is returned to the user application.
- If this 32-bit word is not overwritten, then its assumed that a frame was not captured and the frame is not returned to the user application but, instead, submitted again to capture a subsequent frame.

This mechanism is used for all capture modes by the driver, except when outputting to tiled memory. The driver takes cache operations when using this logic. There is no impact to the CPU load due to this logic. The number of missing frames can be determined by calling FVID2 IOCTL IOCTL_VPS_CAPT_PRINT_ADV_STATISTICS, which prints the information to the console.

For revision 2.0, use VPDMA firmware revision 1B2.

Advisory 2.0.30 — *Occasionally, Video Frame Data Descriptor is Not Written Back When Performing Video Capture Using HDVPSS VIP*

Revision(s) Affected: 2.0, 1.1

Details: When capturing a video frame using the HDVPSS VIP port, a data descriptor is programmed to the VPDMA. A data descriptor must be programmed for every frame that needs to be captured. The VPDMA outputs this data descriptor to a driver-specified location in memory. The software driver uses this written back descriptor to get information about the current capture frame such as field ID, frame width, and frame height. Occasionally, it is seen that this descriptor is not written back to the memory even though the frame data itself is written to memory. This is seen in all modes of capture involving the HDVPSS VIP.

The frequency of occurrence of this issue averages about two descriptor write back misses per channel in one minute. The frequency of this issue depends on the amount of loading that is present in the HDVPSS.

An additional cause of descriptor drop is found in revision 2.0 in multichannel (pixel or line mux) mode which results in one output descriptor per channel not being written every 30 minutes.

Workaround: The written back descriptor is used only for informational purposes and does not affect the capture process itself. When the descriptor is not written back, the driver takes the following action:

- The frame width and frame height is read as the same as the last frame that was captured.
- If the user-specified expected input is of a progressive type, then the field ID is read as the same as the last frame field ID.
- If the user-specified expected input is of an interlaced type, then the field ID is read as the inverse of the last frame field ID.

Unless there was a change in the external source at the precise moment of the descriptor write back miss, this information is valid. If precise accuracy of frame width and height is important, then users are advised to use additional separate means, such as querying the external video decoder, to determine the frame width and frame height of the incoming frame.

Advisory 2.0.52 *Field ID Repeat Occurs in Multichannel Applications Along With Reduced Frame Rate After Some Period of Time*

Revision(s) Affected: 2.0

Details: During multichannel capture, if a new capture descriptor is loaded to the VIP capture channel just as the capture channel receives an end of frame from the external video source, and at the same time another client (memory to memory, display or capture) starts a line, ends a line or ends a frame, the VIP capture channel does not receive the new capture descriptor. The capture descriptor is sent correctly to another internal VPDMA module which software reads to determine if the descriptor has been accepted.

VIP capture channels hold a *current* descriptor and a *shadow* descriptor. The descriptor that is lost is the shadow descriptor.

When the current descriptor is completed, the client believes it does not have the shadow (next) descriptor, so that frame capture is dropped. Software believes it has loaded two descriptors, so this results in software thinking there are two descriptors loaded in the channel, but the client always has only one, leading to every other frame being dropped.

Workaround: Use VPDMA firmware revision 1B2. This firmware essentially causes the descriptors to be written twice to the VIP capture channels. If the channel has two descriptors, the additional descriptor load is ignored.

When the above described event occurs, there one frame is dropped; but after that, all frames are captured, rather than every other frame.

Advisory 2.0.58 *HDVPSS VIP Reset Sequence is Occasionally Unsuccessful if VPDMA is Writing Output Descriptors For VIP Captured Data*

Revision(s) Affected: 2.0, 1.1, 1.0

Details: The HDVPSS VIP reset sequence involves aborting the VIP clients within the VPDMA to reset them. Two registers in each client which should be reset on abort were not. These two registers are associated with the writing of the output descriptors from the HDVPSS VIP.

If any descriptor being used for VIP capture has the write descriptor bit set, and a VIP overflow occurs which requires reset on a frame that is writing a descriptor, and immediately after the reset is performed and a DDR-induced stall occurs during the first frame captured, the HDVPSS VPDMA hangs. Only a hard reset of the entire DSS can be performed to clear this hang.

A stall on the first frame after reset can only occur if the first frame after the reset is performed is sent to DDR. If the first frame after reset completes without stall to the end of frame, the two registers get set to the proper state.

Workaround: The first frame captured after completion of the VIP reset sequence should have both the drop data and write descriptor bits set, which keeps the contents of that frame from going to DDR, thus avoiding the potential stall condition.

Advisory 2.0.59 *Occasionally, Chip Lockup Occurs When HDVPSS VIP is Performing Single-Channel Capture, Sending Tiled Data to DDR and Connect/Disconnect Events are Occurring*

Revision(s) Affected: 2.0

Details: In tiled mode, the VPDMA VIP client multiplies the captured line width by either 2 or 4 (depending on tiled container size), and then a downstream CDMA module divides this by 2 or 4. An issue causes the VIP client to load the tiled mode status of the next descriptor before the complete frame is sent. If the next descriptor is not tiled, this causes the client to think it is not tiled, and if the captured width is less than either 2 or 4 128 byte words, the CDMA divides this number, producing a 0, which is sent as the OCP burst size, resulting in chip lockup disconnect.

The above condition can only occur under three circumstances:

1. The HDVPSS VIP scaler is scaling to a line width where the modulo 128 of the width is less than 2 or 4 bytes (depending on container size).
2. Connect/disconnect events are occurring such that line widths getting to the HDVPSS VPDMA VIP clients can have a width that matches the above condition.
3. Frame sizes are less than the sampling period of the descriptor pacing period. Normally, it is assumed that the sampling period is 2x of the frame period, but when receiving shorter frames caused by connect/disconnect, this is not true.

Workaround: Always keep the mode bit the same for all descriptors. If the software wants to change from tiled to non-tiled or non-tiled to tiled, then it must abort the client and give new descriptors with the new mode bit. If performing tiled transfers and drop data bit is set, the mode bit **must** be set. In this case the descriptor address must be converted to a tiled address and not use the normal L3 memory map.

Advisory 2.0.61 *In Single-Channel Capture, VIP Path Locks Up After 12-14 Hours With Good Pixel Data Which Can Lead To Luma/Chroma Tearing For 420 Capture, Requiring VIP Reset*

Revision(s) Affected: 2.0

Details: During single-channel capture, if a new capture descriptor is loaded to the VIP capture channel just as the capture channel receives an end of frame from the external video source, and at the same time another client (memory to memory, display, or capture) starts a line, ends a line or ends a frame, the VIP capture channel does not receive the new capture descriptor. The capture descriptor is sent correctly to another internal VPDMA module which software reads to determine if the descriptor has been accepted.

VIP capture channels hold a *current* descriptor and a *shadow* descriptor. The descriptor that is lost is the shadow descriptor. When the current descriptor is completed, the client locks up, leading to VIP overflow.

If the output format is selected as 420 (CHR_DS used), the luma and chroma components are sent to two separate clients. Either of these clients can reach this lock-up condition, which can lead to one of the clients capturing a frame of either luma or chroma, and the other locking up. This can lead to one luma or one chroma frame getting lost, but only when the lockup and resulting overflow occurs.

The actual time to event is determined by the amount of activity going on within the HDVPSS. 12-14 hours is with four camera sources being captured in 420 format, along with other memory-to-memory operations. The time to overflow is longer if fewer sources are being captured.

Workaround: Use the VIP reset sequence when this event occurs. This results in 2-3 frames being dropped on the channel where the lockup occurs.

Advisory 2.0.63 *HDVPSS VIP Single-Channel Capture Using Tiled Output Can Lead To VIP Lockup if Connect/Disconnect Events Occur*

Revision(s) Affected: 2.0

Details: When HDVPSS VIP is configured to output tiled mode, the VPDMA has to line buffer either two or four lines (depending on container size) of data to perform the tiling operation. The VPDMA assumes all lines have the same length. When a connect/disconnect events occur, any line can have any length. If the last of the set of lines (last of two or last of four) is either short or long compared to the previous lines, the VPDMA gets into a hung state, resulting in VIP overflow.

Workaround: Use the VIP reset sequence to clear the overflow. This will result in 2-3 frames lost on the channel where the event occurred.

Advisory 2.0.64 *SATA: Link Establishment Fails With SATA GEN3 Capable Targets*

Revision(s) Affected: 2.0

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 the 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 2.0.68 *Continuous Writes From Cortex-A8 Occasionally Starve Other Requestors for DDR Bandwidth*

Revision(s) Affected: 2.0, 1.1, 1.0

Details: A large number of continuous writes from the Cortex-A8 CPU can potentially consume all of the DDR bandwidth, starving all other requestors. In this state, writes from other masters may be starved, but reads are not. However, reads may be held off behind previous writes awaiting completion. These writes can be cache write backs or CPU writes. Priority mechanisms such as interconnect pressure and DDR priority (PEG PRIORITY in DMM) may not be usable when the situation occurs.

All the chip traffic to the DDR is routed via the Dynamic Memory Manager (DMM) and then the External Memory Interface (EMIF). The Cortex-A8 has a dedicated low-latency port (ELLA) to the DMM. All other requestors route the requests via the chip L3 interconnect to the two LISA ports of the DMM. In order to ensure lower latency for Cortex-A8 requests, the DMM always prioritizes Cortex-A8 requests over other requests and, other than prioritizing the Cortex-A8 ELLA port over LISA ports, no other prioritization is performed in the DMM. Writes and reads are prioritized independently, but no reordering of requests is performed in the DMM. The DMM also tags the requests to EMIF with the PEG PRIORITY. Requests in the EMIF may be prioritized as per the EMIF prioritization rules which include PEG PRIORITY [for more details, see the DMM and EMIF sections in the *TMS320DM816x DaVinci Digital Media Processors* Technical Reference Manual (SPRUGX8)]. In the error condition, the EMIF may be presented by the DMM with only the Cortex-A8 requests, thus starving all other requestors.

The issue may manifest as loss of bandwidth seen by requestors such as DSP or EDMA. Real-time requestors, such as HDVPSS, may sometimes lose real-time deadlines causing artifacts on the display, such as rolling.

Workaround: There is no single workaround for the issue. However, specific settings of the Cortex-A8 cache may help avoid this behavior in many systems:

1. Disabling the write allocate cache policy (bit 22) in the Cortex-A8 C9, L2 cache auxiliary control register is seen to change the observed behavior.
2. Setting the write allocate delay disable (bit 24) to disable (1) in the Cortex-A8 C9, L2 cache auxiliary control register is seen to change the observed behavior.

The workarounds may help by minimizing the chances of a long series of writes from the Cortex-A8.

Note that the L2 cache auxiliary control register cannot be entered by application/OS code; this must be programmed via ROM functions invoked via SMI. The following pseudo code is an example of how to program the L2 cache auxiliary control register:

```
l2_disable_wa:
        stmfd   sp!, {r0 - r12, lr}       @ Store registers r0 -
> r12 on the stack, ROM does not preserve values
        mrc     p15, 1, r0, c9, c0, 2     @ Read the L2 cache auxiliary control
register
        orr     r0, r0, #(1 << 22)        @ OR in the bit to disable write
allocate (bit #22)
        mov     r12, #0
        orr     r12, r12, #(1 << 8)
        add     r12, r12, #2              @ Program ROM Entry point into R12
(0x102 for L2 cache aux ctrl register)
                                          @ R0 contains the value the ROM will
write into the L2 cache aux ctrl register
        .word   0xe1600070                @ opcode of SMC/SMI instruction -
 jump into the ROM
        ldmfd   sp!, {r0 -
 r12, pc}       @ Restore registers from stack that may have been overwritten by
ROM code.
```

4 Revision 1.1 Known Design Exceptions to Functional Specifications

Table 6 lists the device revision 1.1 known design exceptions to functional specifications. Advisories are numbered in the order in which they were added to this document. If the design exceptions are still applicable, the advisories move up to the latest silicon revision section. If the design exceptions are no longer applicable or if the information has been documented elsewhere, those advisories are removed. Therefore, advisory numbering in this section may not be sequential.

Table 6. Revision 1.1 Advisory List

Advisory 1.1.3 *Spurious Descriptors are Output for Capture When No Descriptors are Programmed to VPDMA for Capture*

Revision(s) Affected: 1.1, 1.0

Details: The Capture driver works as follows:

- A list is programmed with two data descriptors for every capture channel.
- When a frame for given channel is captured, the frame data goes to the buffer address programmed in the descriptor and the programmed descriptor is written back to a specific memory location.
- The software then checks the specific memory location and, for every descriptor that is written in this memory location, it re-programs a new data descriptor.

When software does not program a descriptor, the expectation is that any new frame data arriving for that channel should be dropped **and** when software does not program a descriptor, there should be no descriptor output to the memory.

However when the software does not program a descriptor for a channel, the new frame data gets dropped since there is no descriptor and, therefore, no buffer address to output this data. But, the VPDMA generates a *spurious* descriptor and writes it to memory.

The spurious descriptor that gets written out has the following properties:

- The width x height reported in the descriptor is 1x1.
- The buffer address in the descriptor is same as the last received descriptor.

Workaround:

1. Ensure that spurious descriptors do not occur; i.e., make sure a capture VPDMA channel is never starved of descriptors. A capture VPDMA channel can be starved of descriptors in the following scenarios:
   - When the CPU is halted due to breakpoints.
   - When the CPU is halted due to a Code Composer Studio (CCStudio) printf.
   - When a high-priority task or ISR continuously consumes CPU MHz for more than 1 frame or field interval.
2. Thus, once a capture is started, the software should not perform any CCStudio printf (the software can perform a UART printf since the UART printf does not block the CPU).
   - Do not set any breakpoints.
   - The capture driver thread should be the highest priority.
3. If, after the above, spurious descriptors (that meet the above-described properties; i.e., 1x1 size and buffer address equals previously received descriptor buffer address), do not re-submit a new descriptor for this received descriptor since it is a spurious descriptor.

Advisory 1.1.4 — *Clear Descriptor Count in VPDMA Descriptor Status Control Register Does Not Perform as Expected*

Revision(s) Affected: 1.1, 1.0

Details: The Capture driver working mechanism is briefly explained in Advisory 1.1.3. In addition:

- Every few milliseconds, the capture driver software:
  - Examines this memory location for received descriptors.
  - Replaces the memory location with new descriptors.
  - Re-submits the new list of descriptors to the VPDMA.
- While capture is checking the received descriptors, new descriptors could be generated and, in order to make sure no descriptors output by VPDMA are missed, the capture driver sets the clear descriptor count bit in the VPDMA Descriptor Status Control register.
- This bit performs the following:
  - Switches the descriptor writing to a new memory location.
  - Tells the software, via the last descriptors written field in the Descriptor Status Control register, how many descriptors were received at the instant the clear descriptor count bit was set.

However, when the clear descriptor count bit is set, sometimes some descriptors were never written to memory. Due to this issue, the software does not replace those descriptors causing frame-rate drops for those channels.

Workaround:

- Do not use the clear descriptor count bit. Instead, use the Current Descriptor register in the VPDMA to keep track of how many descriptors were written since last check.
- The Descriptors Top and Descriptors Bottom registers are programmed to write the received descriptors in a circular manner in the buffer that starts at Descriptors Top and ends at Descriptors Bottom.
- The descriptor received at any given time is the *current descriptor* (the value of the current descriptor in the previous check). The software takes care to handle wraparound when reading descriptors in a circular manner.

Advisory 1.1.5 *Descriptor With Erroneous Frame Size 1x1 Written by VPDMA When a Descriptor is Posted For the Corresponding Channel For Capture*

Revision(s) Affected: 1.1, 1.0

Details: The Capture driver working mechanism is briefly explained in Advisory 1.1.3.

Sometimes, the VPDMA outputs descriptors that report the incorrect frame size 1x1 even though the descriptors were programmed to the VPDMA. Here the frame data, itself, is written to the correct programmed memory address.

The erroneous descriptor that gets written out has the following properties:

- The width x height reported in the descriptor is 1x1.
- The buffer address in the descriptor is the correct buffer address for the current frame.

Workaround: Detect this condition based on the above described properties of the descriptor and replace the received descriptors as usual.

Advisory 1.1.6 — *Camera Removal and Reconnection Causes VIP Parser to Lock Up*

Revision(s) Affected: 1.1, 1.0

Details: If the camera is removed and reconnected while capture is running, then the VIP parser locks up and causes the capture driver to hang up.

This issue occurs because when the camera is removed or reconnected, the F-bit and V-bit transitions are not consistent. For example, the V-bit transitions to a new frame every few lines and, thus, causes many short frames to be received at the VIP port which causes the VIP port to lock.

In the multichannel case, the internal hardware FIFO that is used for capture is common for all channels at a given port. So if one channel is removed or reconnected, the whole FIFO at the VIP port locks up and causes all channels at that port to lock up.

Workaround: For multichannel case, with TVP5158:

- Operate TVP5158 in pixel-mux mode.
- Load TVP5158 firmware patch v2.1.21 or above.
- Disable fast lock detect:
  - TVP5158 REG_FV_CTRL = 0x06.
- Disable auto-switch mode; i.e., be in force NTSC or force PAL mode:
  - TVP5158 REG_VID_STD_SELECT = 0x1 (NTSC) or 0x2 (PAL).
- Always decode F and V bits from line count (TVP5146 compatible):
  - TVP5158 REG_FV_DEC_CTRL = 0x03.
- TVP5158 REG_OP_MODE_CTRL = 0.

The above workaround does ***not*** work with the TVP5158 in line mux mode and is ***not*** specified to work with non-TVP5158 multichannel video decoders.

For single-channel case:

- The software should detect cable removal by querying the external video decoder.
- When sync loss is detected, the VIP port should be stopped.
- When the cable is reconnected, the software should:
  - Perform a VIP reset sequence, as mentioned in Advisory 1.1.8.
  - Start the VIP port.

Advisory 1.1.7 *CHR_DS Hangs When it Receives a Frame With an Odd Number of Lines*

Revision(s) Affected: 1.1, 1.0

Details: When CHR_DS receives a frame with an odd number of lines and then an EOF signal, it hangs up waiting for the even numbered line to arrive.

When using CHR_DS in memory-to-memory mode this is not an issue, since user software can ensure that CHR_DS always receives an even number of lines. However, when CHR_DS is used in capture path, this cannot be ensured.

Workaround: To resolve this issue:

1. Ensure that the external video decoder always gives even number of lines.
2. If Step 1 cannot be ensured, then use the SC to trim the input to an even number of lines and then send the lines to CHR_DS.
3. If Steps 1 and 2 cannot be performed (because SC is used in some other path), then do not use CHR_DS and use either YUV422S or YUV422I output mode.

If CHR_DS locks up, you can recover it by using the VIP reset sequence mentioned in Advisory 1.1.8.

This issue could also occur when the cable is disconnected. In this situation, refer to the workaround in Advisory 1.1.6 for single-channel case.

Advisory 1.1.8 *VPDMA List Stalls and Hangs When VIP Port is Re-enabled After Disabling With CHR_DS and/or SC and/or CSC Enabled in Non-Mux Mode*

Revision(s) Affected: 1.1, 1.0

Details: When capturing in non-mux mode with CHR_DS and/or SC and/or CSC enabled in the capture path, the capture runs correctly for the first run.

If capture is stopped by disabling the VIP port and then capture is started again with CHR_DS and/or SC and/or CSC in the path, then the VPDMA list for capture stalls and hangs up, causing the capture driver to hang up as well.

This is applicable for both embedded-sync mode as well as discrete-sync mode, except that in discrete-sync mode the hang occurs on the first run.

Workaround: For 8-/16-bit, YUV422 embedded-sync mode:

1. Disable the VIP port.
2. Assert VIP reset in the HDVPSS CLKC register.
3. Assert the Async FIFO reset in the VIP parser register.
4. Make a list of abort descriptors for all VIP VPDMA channels.
5. Post this list and wait for it to complete.
6. De-assert the VIP reset in the HDVPSS CLKC register.
7. De-assert the Async FIFO reset in the VIP parser register.

For 8/16/24-bit discrete-sync mode, in addition to the above workaround, the following additional sequence of descriptor programming **must** be performed in order to use discrete-sync mode with CSC, SC, and CHR_DS. For the first list that is posted when discrete-sync mode is started, perform the following:

1. To the list, add a data descriptor such that video data from the VIP parser goes directly to VPDMA, with DROP_DATA = 1, WRITE_DESC = 0 in the data descriptor (e.g., PORTB_RGB VPDMA channel).
2. To the list, add a sync-on-client descriptor with the VPDMA channel as the one set in the above data descriptor (e.g., PORTB_RGB VPDMA channel and event = EOF).
3. To the list, add a configuration descriptor with the actual required VIP mux settings which enables CHR_DS, CSC, and SC.
4. To the list, add the actual data descriptors as before (e.g., YUV420 output data descriptors).
5. Program the VIP mux such that data from the VIP parser goes directly to VPDMA.
6. Enable the VIP port so that data starts flowing.
7. Post the list that is prepared above.
8. The list stalls until it receives an end of frame. In the blanking period, the configuration descriptor programs the mux to use the CSC, SC, and CHR_DS and then capture continues as usual.

Advisory 1.1.10 *Discrete Sync Capture Style Line ACTVID Does Not Work*

Revision(s) Affected: 1.1, 1.0

Details: The VIP port supports a discrete sync capture mode where the ACTVID signal is used to determine the active region of the video and then the VPDMA writes this active video frame to memory. However, with some external video decoders, this mode does not work by default.

Workaround: The VIP parser requires that the ACTVID signal is active for all lines of the incoming data, including the vertical blanking region. The vertical blanking region, itself, is not written to memory by the VPDMA. Figure 3 illustrates this.

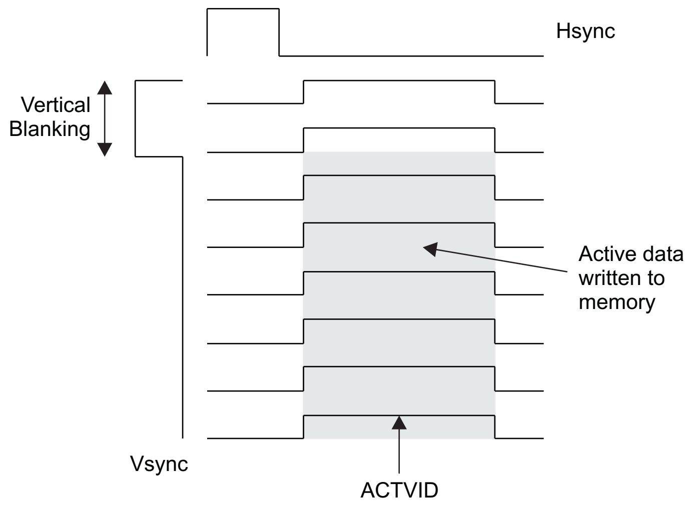

Figure 3. ACTVID Signal Active for All Lines of Incoming Data

Advisory 1.1.11 *In Single-Channel Capture No Descriptor May be Output Even Though a Descriptor was Previously Programmed*

Revision(s) Affected: 1.1, 1.0

Details: The Capture driver working mechanism is briefly explained in Advisory 1.1.3.

In single-channel non-mux mode, sometimes, even though a descriptor is programmed, it is not output to memory even though the frame itself is captured. Due to this issue, the software does not reprogram a descriptor for that VIP port and the capture frame rate drops.

Workaround: Program a dummy descriptor on one of the VPDMA MULT_SRCx channels for the port, even though no data is received on that channel. This always causes the VPDMA to output a descriptor to memory for the channel on which data is being received. In this case, Advisory 1.1.5 is also applicable.

Advisory 1.1.15 *USB Host Mode Cannot Perform Write Operation to TXFunction/TXHubPort/TXHubAdr Controller Addresses*

Revision(s) Affected: 1.1, 1.0

Details: The USB controller does not support devices connected through a hub when operating as USB host. Due to this, writing to the TXHubPort/TXHubAdr results in the write being ignored.

Workaround: There is no workaround for this issue.

Advisory 1.1.21 *UART Boot Generates Incorrect Baud Rate*

Revision(s) Affected: 1.1, 1.0

Details: The UART boot mode is not functional on the device due to generating incorrect baud rate. This occurs because the ROM code puts the DDR PLL in bypass instead of locking it. As a result, the UART module gets an input clock of 13.5 MHz. Therefore, the UART operates at 32452 baud. The expected baud rate in 115200.

Workaround: Use a host that can support the generated baud rate (32452 baud).

NOTE: Some PCs do not allow a non-standard baud rate to be set. On such PCs, there is no workaround.

Advisory 1.1.22 *SPI Boot Mode is Slow*

Revision(s) Affected: 1.1, 1.0

Details: SPI boot mode works at a lower speed than specified.

Workaround: There is no workaround for this issue.

Advisory 1.1.23 *Unable to Reset HDVPSS Through PRCM or Using CLKC Module in DSS*

Revision(s) Affected: 1.1

Details: The VPDMA is not being reset by the CLKC module although the CLKC module is resetting other modules.

Workaround: There is no workaround for this issue.

Advisory 1.1.24 *When Performing Cable Disconnect/Connect, Extra Lines are Being Output Even Though Max Width x Height Limit is Set to 720x288*

Revision(s) Affected: 1.1

Details: When performing cable disconnect/reconnect with capture, the frames being sent by the video decoder can be of non-fixed size. If the incoming frame height is greater than 2x of the maximum expected frame height that is set in the VPDMA, then the VPDMA parameter to apply the max frame height does not take effect. This results in extra lines overflowing the allocate buffer in DDR memory.

This issue affects modes of capture, such as non-multiplexed mode, multiplexed modes, discrete sync mode, embedded sync mode, and 8/16/24-bit capture modes.

Examples

If the VPDMA max height is set to 288 and an incoming frame of height 500 lines is received:

Since 500 < 2 x 288, the extra (500 - 288 = 212) lines do not overflow the allocated memory buffer in DDR memory.

If the VPDMA max height is set to 288 and an incoming frame of height 600 lines is receive:

Since 600 > 2 x 288, the extra (600 - (2 x 288) = 24) lines overflow the allocated memory buffer in DDR memory.

If the VPDMA max height is set to 288 and an incoming frame of height 1000 lines is receive:

Since 1000 > 2 x 288, the extra (1000 - (2 x 288) = 424) lines overflow the allocated memory buffer in DDR memory.

Workaround:

1. Ensure that, in no circumstance, the video decoder outputs > 2x max expected height.
2. If option 1 cannot be ensured, then, in the case of non-multiplexed capture, use SC in the capture path to crop the height.
3. If workaround 1 cannot be ensured and workaround 2 does not apply, then characterize the max number of lines that can be received from the video decoder by setting max height as unlimited and allocate extra buffer space in memory.

In TVP5158, specifically, it has been observed that frames of max height up to 2100 lines have been received; therefore, it is important to ensure that when a channel buffer is allocated, an extra 2100 lines are allocated at the end of the buffer.

Example

For a given channel, 6 buffers of size 720x288x2 bytes may be allocated; therefore, a contiguous memory of 720x288x2x6 + 720x2100x2 should be allocated. When memory is allocated this way, then during the cable disconnect/connect when the channel buffer overflows, it either overflows into its own channel buffer memory or in the extra 2100 lines of the buffer that was allocated at the end of the sixth buffer. This ensures that other channel data and/or code sections are not corrupted. For a 16-channel TVP5158 system, this means an extra 720x2100x2x16 bytes = 47MB of space is needed to accommodate the extra lines.

These workarounds need to be applied for other video decoders as well.

Note that this extra DDR buffer allocation needs to be handled in the application. The HDVPSS drivers do not allocate any buffer memory.

Advisory 1.1.26 *NAND Boot Cannot Detect Devices That are Not Fully ONFI Compatible*

Revision(s) Affected: 1.1, 1.0

Details: Some NAND flashes are not fully ONFI compatible. Specifically, in these flashes, the parameter pages do not contain the ONFI signature and the contents of the parameter page is not valid. This results in the following:

- NAND flash geometry is identified incorrectly.
- NAND boot fails.

Workaround: Choose a NAND flash that is fully ONFI compatible (i.e., the parameter page has an ONFI signature) or use NAND I2C boot, where the NAND geometry is read from an I2C EEPROM.

Advisory 1.1.27 *HDMI Audio May Not Inter-operate With All HDMI Audio Sinks, AVRs, Repeaters*

Revision(s) Affected: 1.1, 1.0

Details: The HDMI sends excessive ACR packets which leads to inter-operation issues and to CTS interval compliancy failures. This may cause the device HDMI audio not to inter-operate with all types of HDMI receivers, AVRs or repeaters.

Workaround: Currently, there is no workaround for this issue.

Advisory 1.1.28 *Timers May Lock Up if TCLKIN Pin is Tied High, Even When GPIO Module is Selected*

Revision(s) Affected: 1.1, 1.0

Details: Timers may sometimes lock up if the TCLKIN pin is tied high, even when the GPIO module is selected. If the TCLKIN pin is tied high (is not toggling/does not come to 0), TIMERs may lock up and may not be usable. This happens even if the TCLKIN pin is not selected by the pin control registers. The locks may be seen randomly on power up; i.e., it may be observed that some timers lock up and some do not.

Workaround: The TCLKIN pin must be driven to 0 on the board.

Advisory 1.1.31 *Luma and Chroma Become Out of Sync During VIP Capture of 420 Data*

Revision(s) Affected: 1.1

Details: When capturing a video frame using the HDVPSS VIP port, a data descriptor is programmed to the VPDMA. For 420 data, two data descriptors must be programmed for every frame that needs to be captured; one for the luma channel and one for the chroma channel. Occasionally, even though a data descriptor is programmed to the VPDMA, the frame data and descriptor for either luma or chroma may not get written to the DDR memory. This results in either luma or chroma data for the same frame to become out of sync with each other. This is seen in 420 capture, single-channel capture modes involving the HDVPSS VIP.

The frequency of occurrence of this issue averages about one frame, of either luma or chroma, missed per hour. The frequency of this issue depends on the amount of loading that is present in the HDVPSS.

Workaround: The driver checks for both luma and chroma writing by VPDMA to DDR and returns the buffer only when both are written. If the VPDMA has not written one of the luma or chroma data, the driver resubmits the descriptor to VPDMA after a few milliseconds. Most of the time, the VPDMA succeeds on resubmission.

If the second submission also fails, the driver clears the VPDMA channels in that capture instance using abort descriptors and the VPDMA recovers after that. This can cause, worst case, one frame loss versus 2-3 frames if a full VIP reset is used.

Advisory 1.1.37 *Watchdog Timer (WDT): Default Timeout Period of 2 ms is Too Short*

Revision(s) Affected: 1.1, 1.0

Details: As per device specification, the WDT default timeout is 2 ms. The ROM code that runs after a warm reset takes longer than 2 ms to execute. The WDT is disabled, by default, at reset time and the ROM code does not use the WDT functionality. Enabling the WDT can be done by writing to the control module register so that it is made available to use in the application. Once the WDT is enabled, the Timer starts ticking and before it reaches the 2-ms timeout period, the application software can modify the timeout value. However, once the WDT expires and issues a warm reset to the system, the WDT counter starts again with default timeout value of 2 ms. The time application software gains control only after the ROM execution. By this time, the WDT expires because the 2-ms timeout period is too short. This results in repeated WDT timeout resets within the interval of 2 ms. Application software cannot gain the system control after the first WDT timeout triggered warm reset. Regardless of any timeout value that may have been provided in software, the timeout value always reverts to 2 ms after a warm reset.

Workaround: A hardware workaround is to connect the WDT reset out to the $\overline{\text{POR}}$ input of the device. This loop on the board ensures that every time the WDT sends out a reset, the device receives a $\overline{\text{POR}}$ reset, which keeps the WDT in a frozen state. Application code can enable the WDT.

Advisory 1.1.38 *NEON Instructions Executed on a Non-Cached Memory-Mapped Address Result in Lockup*

Revision(s) Affected: 1.1, 1.0

Details: The AXI2OCP bridge causes device lockup when a NEON instruction is issued with load address targeting area of virtual memory mapped as strongly ordered or device type in the MMU. Cortex-A8 NEON™ instructions resulting in an ARLEN=0xf executed on a memory-mapped address that is not in the cache causes a hang condition. The AXI protocol (v1.0) specification allows the burst length, ARLEN[3:0], to be up to 16 data transfers long. The counter in AXI2OCP that maintains the count of the narrow burst transactions should compare five bits to properly track the maximum case of ARLEN[3:0] = 4’h0Fh (i.e.,16 transactions), but the comparator looks at the four LSBs instead of all five bits. This effectively leads the AXI2OCP to incorrectly signal the read response completion. The RLAST is never asserted and, therefore, the read burst is never cleanly completed. Once the AXI2OCP accepts the maximum number of outstanding requests for the ARID in question, the AXI bus hangs.

Workaround: Any of the following workarounds is an acceptable solution:

- Disable the compiler from using NEON instructions. The assumption is that majority of the software that uses NEON instructions is either hand-coded or from intrinsic library functions. Auto-detection of vectorizable code from the compiler is less prevalent.
- Avoid NEON SIMD element load instructions (VLD: i.e., VLDn.<size> {D0-D3} [RX]) to strongly ordered, device, or non-cacheable memory.
- NEON SIMD element load data accesses, if used, should only be done to data that is specified to exist in the A8 L1 or L2 caches, via use of the PLD instruction. This would avoid having to require any AXI read channel transactions to load data.

Advisory 1.1.45 *HDVPSS: VIP Scaler Causes Continuous VIP_PARSER Overflow if DDR Bandwidth is Over Consumed*

Revision(s) Affected: 1.1, 1.0

Details: This *only* occurs in single-channel capture cases where the VIP scaler is being used.

If the HDVPSS VIP scaler is being used to downscale external video inline with the VIP_PARSER and there is a momentary DDR bandwidth reduction that causes an overflow of the VIP_PARSER, the VIP scaler becomes out of sync with the external video, resulting in continuous overflows, garbled resulting video, but no lockups.

The VIP scaler is programmed with an input height/width and output height/width. If the actual input does not match the programmed input values, the design still tries to create the programmed output height/width. While it is "making up" the output for the missing input, it stops requesting data from upstream.

If the 4k-byte buffer for holding captured video data in the HDVPSS VPDMA becomes full due to DDR bandwidth restrictions (cannot get data out fast enough), this causes an overflow at the VIP_PARSER. This results in the VIP_PARSER dropping lines/pixels while it is in the overflow state. This results in a frame going to the VIP scaler which is smaller than the programmed size.

The VIP scaler detects the input is the wrong size, stops requesting data, and starts making up data for what is missing. External video data continues to come in, but because the VIP scaler has stopped requesting data, it causes another overflow of the VIP_PARSER.

While the VIP scaler is making up the missing data, it is running as fast as it can, and becomes out of sync with external video coming into the VIP_PARSER. The extra overflow it creates makes the next frame lose lines/pixels too, and this continues on.

The overflow could cause distortions on the data captured by the VIP port.

Workaround: Reset the VIP port by following this procedure:

1. Disable the VIP port.
2. Assert VIP reset in the HDVPSS CLKC register.
3. Assert the Async FIFO reset in the VIP parser register.
4. Make a list of abort descriptors for all VIP VPDMA channels.
5. Post this list and wait for it to complete.
6. De-assert the VIP reset in the HDVPSS CLKC register.
7. De-assert the Async FIFO reset in the VIP parser register.
8. Start the VIP capture port.

Advisory 1.1.50 *SATA: Activity LED Signal Not Generated for Port1 on SATA*

Revision(s) Affected: 1.1, 1.0

Details: Due to a connectivity issue, SATA_ACT0_LED and SATA_ACT1_LED ports are mapped to the internal Port0 of the device SATA module. The SATA Activity LED signal is not generated for Port1.

Workaround: There is no workaround for this issue.

This issue is corrected on revision 2.0, but with the output pins reversed. SATA_ACT1_LED is mapped to pin J32 and SATA_ACT0_LED to pin J33 (see Figure 4).

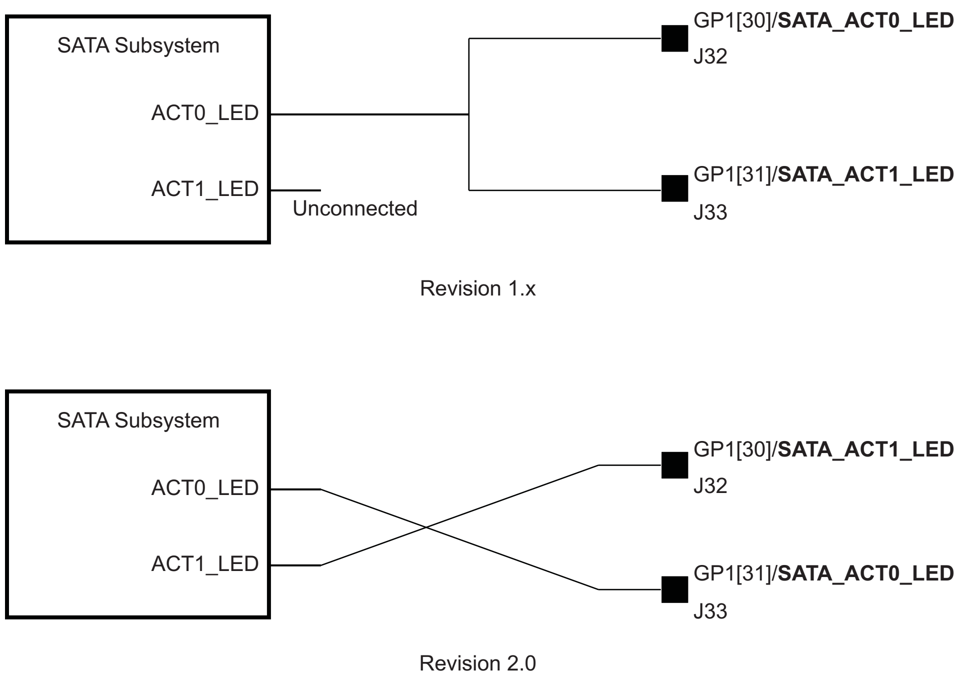

Figure 4. SATA Activity LED Connectivity

Advisory 1.1.51 *HDVPSS VIP Capture of One-Line Frame to VIP LO Port Immediately After HDVPSS Reset or VIP Client Abort Causes Lockup*

Revision(s) Affected: 1.1, 1.0

Details: As data is captured from the HDVPSS VIP to the VPDMA, it is broken into 128-byte segments for transfer to DDR. At the end of each line a *remainder* for the line is calculated, sent as a DDR transfer, and then the address incremented by the line stride. The VPDMA VIP LO port clients incorrectly send the previous line's remainder as the size of the last transfer of a line instead of the current line. If the line captured by the VIP LO port clients is one line, it uses the last remainder calculated from the previous frame.

If a VIP LO port client abort is performed, or immediately after reset, and a one-line frame goes to the VIP LO port clients as the first or second frame, the reset value of the remainder calculation registers is sent as the burst, and the reset value is 0. The remainder registers are ping-ponged and there are two per channel. This can occur in multichannel or single-channel configurations.

In the multichannel case, any camera source that sends a one-line frame can cause this to occur.

In a single-channel configuration, the VIP LO port captures luma and chroma components separately. If either component has one line immediately after reset, this can occur. This means if the CHR_DS module is used to convert 422 data to 420, a two- or three-line captured frame can also cause this condition, because the chroma component for a two- or three-line frame is downsampled to one line.

Workaround: For single-channel cases, the HDVPSS VIP scaler can be used to keep single line frames from reaching the VPDMA. The VIP scaler always generates the output size it is programmed to, regardless of the input. So long as the scaler is not set to output one line (or two or three if the output is going to be 420), this problem does not occur.

For multichannel cases, the source creating the multichannel input must ensure that one-line frames cannot come in. Occassionally, and only during connect/disconnect when a line or width limit feature is not used, any memory areas can be overwritten (memory corruption/memory flashthrough).

Advisory 1.1.111 *HDVPSS VIP Capture of One-Line Frame to VIP LO Port Immediately After HDVPSS Reset or VIP Client Abort Causes LockupAPCA-012 - VIP: Potential Buffer Overflow by VPDMA*

Revision(s) Affected: 1.1, 1.0

Details: VPDMA can accept two outbound descriptors – current and shadow descriptors. This means the VPDMA can consume two buffers for capture operation.

When the number of descriptors with the VPDMA underflows (becomes 0 or only one descriptor is provided to VPDMA) while capture is in progress, there is a corner condition in the VPDMA which can sometimes generate 2 descriptors without the software providing the descriptors to VPDMA.

The issue is related to how the VPDMA shifts data from the shadow register to the current register. When using only one descriptor, it should always load into the current descriptor but there are cases where the descriptor would load into the current descriptor and the shadow descriptor. Hence extra descriptors are generated.

When this condition happens and if the VIP parser keeps capturing data, this can lead to corruption of the previously provided buffers by the software because of the extra generated descriptor.

Two types of corruption can occur:

1. If the VIP parser generates the same size frame as previous frames, then the content of the previous buffer will be overwritten by the VPDMA while the software owns the buffer. This can corrupt the buffer within the limit of the buffer size.
2. If the VIP generates different size frames (as can happen in weak signal or connect/disconnect cases), the VPDMA can corrupt the buffer content beyond the limit of the allocated buffer (buffer overwrite).

To avoid this buffer overwrite and/or corruption, the software should adhere to the below mentioned workaround.

Workaround: To overcome this buffer corruption the following workarounds are recommended.

1. The software driver should ensure that the VPDMA descriptors don’t underflow by giving new descriptors to VPDMA for every descriptor completed by VPDMA i.e. for every frame captured.
   (a) **Capture Start:**
   In case of starting the capture, the driver should always provide two descriptors to the VPDMA and then enable the VIP parser.
   (b) **Capture in Progress:**
   When capture is in progress, for every frame captured, the driver should provide a new descriptor to the VPDMA.
   If no new frames (buffers) are available for programming, then the driver can set the drop data bit of the descriptor and provide a dummy buffer address. This will ensure that VPDMA descriptor doesn’t underflow even when there are no buffers to program.
   (c) **Capture Stop:**
   While stopping capture, the driver should abort all the current and shadow descriptors for a particular channel by programming the abort channel descriptor to the VPDMA. This will ensure that the VPDMA will stop further data capture and since the driver is not providing any more descriptors to the VPDMA, the already provided descriptors don’t get consumed and result in descriptor underflow.
2. To ensure that the VPDMA doesn’t overwrite the buffers beyond the expected frame size, VPDMA line limit feature of the descriptor can be used to limit the max amount of data that the VPDMA will capture.
   If the frame captured is less that the available limit sizes, then the software should allocate the buffer equal to the nearest max limit size.

5 Revision 1.0 Known Design Exceptions to Functional Specifications

Table 7 lists the device revision 1.0 known design exceptions to functional specifications. Advisories are numbered in the order in which they were added to this document. If the design exceptions are still applicable, the advisories move up to the latest silicon revision section. If the design exceptions are no longer applicable or if the information has been documented elsewhere, those advisories are removed. Therefore, advisory numbering in this section may not be sequential.

Table 7. Revision 1.0 Advisory List

Advisory 1.0.9 *Noise Filter (NSF) Outputting Frames of 64x32 Though Input Data is of Larger Size*

Revision(s) Affected: 1.0

Details: When the noise filter (NSF) is used along with capture and display after a few frames of execution, the NSF starts outputting frames of size 64x32, even though the input size is larger.

Workaround: Use the noise filter in external bypass mode.

Advisory 1.0.14 *When Using HDMI Port, HD_VENC_D_CLK Needs to be Doubled for Normal Operation*

Revision(s) Affected: 1.0

Details: HD VENC_D (DVO1) and HDMI should operate with the same pixel clock frequency since HD VENC_D supplies the video and control signals for HDMI.

The pixel clock selector (pixel clock for HDMI and HD VENC_D 1X clock), HD_VENC_D_CLK1X_SELECT, always selects HD_VENC_D_CLK / 2. This ensures that HD_VENC_D_CLK is always divided by 2; e.g., if the desired display is a resolution of 720p60, the HD_VENC_D_CLK would be 74.25 MHz. Since the pixel clock selector always selects HD_VENC_D_CLK / 2, the HDMI and HD VENC_D pixel clock would be 37.125 MHz.

Workaround: Clock HD_VENC_D_CLK at twice the required pixel clock and configure HD VENC_D to use 1X input; e.g., in the above example HD_VENC_D_CLK should be clocked at 148.5 MHz.

HDMI and HD VENC_D have been verified for HD_VENC_D_CLK input up to 297 MHz for 1080p60 resolution.

Revision History

Changes from February 19, 2014 to March 6, 2015 (from E Revision (February 2014) to F Revision) **Page**

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		