TMS320VC5505/VC5504
Fixed-Point Digital Signal Processor
Silicon Revision 1.4

Silicon Errata

Literature Number: SPRZ281C
June 2009–Revised January 2013

Contents

List of Figures

List of Tables

TMS320VC550x Silicon Revision

1 Introduction

This document describes the known exceptions to the functional specifications for the TMS320C55x devices (i.e., TMX320VC5505, TMX320VC5504, TMS320VC5505, and TMS320VC5504). For more detailed information on these devices, see the device-specific data manual:

- *TMS320VC5504 Fixed-Point Digital Signal Processor* data manual (Literature Number SPRS609)
- *TMS320VC5505 Fixed-Point Digital Signal Processor* data manual (Literature Number SPRS503)

Throughout this document, unless otherwise specified, TMS320VC550x and VC550x, refer to the TMX320VC5505, TMX320VC5504, TMS320VC5505, and TMS320VC5504 devices. For additional peripheral information, see the latest version of the *TMS320VC5505 DSP Peripheral Overview* Reference Guide (Literature Number SPRUFR9).

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

1.1 Device and Development-Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all DSP devices and support tools. Each DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g.,TMS320VC5505ZCH). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX — Experimental device that is not necessarily representative of the final device's electrical specifications.

TMP — Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification.

TMS — Fully-qualified production device.

Support tool development evolutionary flow:

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

TMDS — Fully qualified development-support product.

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

"Developmental product is intended for internal evaluation purposes."

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

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

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, ZUT), the temperature range (for example, "Blank" is the commercial temperature range), and the device speed range in megahertz (for example, "Blank" is the default [594-MHz DSP, 297-MHz ARM9]).

1.2 Revision Identification

Figure 1 provides an example(s) of the TMX320VC550x and TMS320VC550x device markings. The device revision can be determined by the symbols marked on the top of the package.

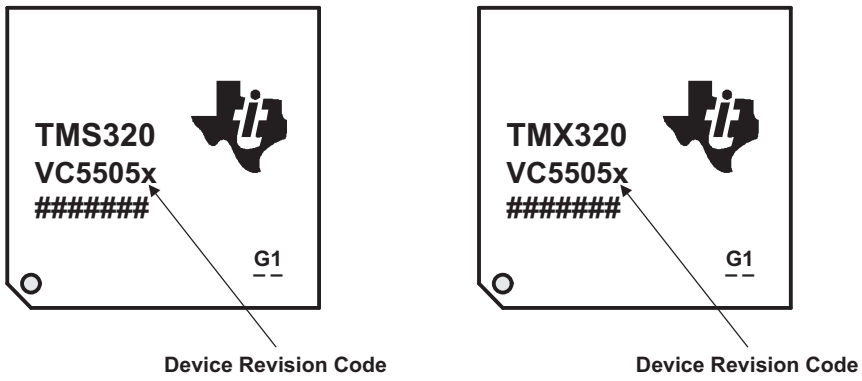


Figure 1. Example, Device Revision Codes for TMX320VC550x and TMS320VC550x (ZCH Package)

Silicon revision is identified by a device revision code marked on the package and through software. The code on the package is of the format VC5505x, where "x" denotes the silicon revision . If x is "D" in the device part number, it represents TMX devices and if "x" is *no letter* (blank) in the device part number, it represents TMS devices. Table 1 lists the information associated with each silicon revision.

Table 1. VC5505 and VC5504 Device Revision Codes

DEVICE PART NUMBER DEVICE REVISION CODE (x)	SILICON REVISION	PART NUMBERS/COMMENTS
(BLANK)	1.4	This silicon revision is available as TMS *only*. **TMS**320VC5505ZCH, **TMS**320VC5504ZCH
D	1.4	This silicon revision is available as TMX *only*. **TMX**320VC5505**D**ZCH, **TMX**320VC5504**D**ZCH

Through software, the user can read bits 15-12 of the I/O space Die ID Register 3 (DIEIDR3) [1C43h].

2 Silicon Revision 1.4 Usage Notes and Known Design Exceptions to Functional Specifications

This section describes the usage notes and advisories that apply to silicon revision 1.4 of the TMX320VC5505, TMX320VC5504, TMS320VC5505, and TMS320VC5504 devices.

2.1 Usage Notes for Silicon Revision 1.4

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

2.1.1 Master Clock Gating With WAKEUP, $\overline{\text{INT0}}$, or $\overline{\text{INT1}}$ Asserted

On silicon revision 1.4, the VC550x DSP can disable the Master Clock by setting bit 15 of the PCGCR register (0x1C02). Once the master clock is disabled, it can only be re-enabled by one of the following events:

- Hardware reset being asserted ($\overline{\text{RESET}}$ = low)
- An enabled RTC alarm or periodic interrupt occurring
- The $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ pins being asserted (low) (level-sensitive)
- The WAKEUP pin being asserted (high) (level-sensitive)

When the master clock is disabled, there are no clocks for edge detection and therefore the $\overline{\text{INT0}}$, $\overline{\text{INT1}}$, and WAKEUP pins are level-sensitive. This means that a low on either the $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ or a high on the WAKEUP pin will force bit 15 of the PCGCR register to "0", enabling the master clock. Attempting to write a "1" to bit 15 of the PCGCR register while $\overline{\text{INT0}}$, $\overline{\text{INT1}}$, or WAKEUP are asserted will be unsuccessful since re-enabling the clocks has a higher priority than disabling them.

Also, the WAKEUP pin can be configured as an input-pin or an output-pin. Regardless of the pin configuration, the WAKEUP pin's state must be low to disable the master clocks.

2.1.2 Serial Boot Modes Only Support 16-bit Address Mode

On silicon revision 1.4, the VC550x DSP supports the following boot modes: NOR Flash, NAND Flash, SPI EEPROM, and I2C EEPROM. The SPI and I2C EEPROM boot modes *only* support 16-bit address mode.

2.1.3 DMA Only Supports Single-Buffering

The VC550x DMA *only* supports single-buffer operations, the double-buffer (ping-pong) feature is ***not*** supported. To handle ping-pong buffers, the DMA needs to be reprogrammed by the CPU at the end of a transfer prior to firing the DMA for the next buffer. Reprogramming the DMA during an active frame transfer will affect the current active transfer.

2.1.4 Reserved Bits in the RTC Oscillator Register (RTCOSC) [0x192C]

For proper VC550x device operation on silicon revision 1.4, the "RESERVED" bits in the RTCOSC register (0x192C) should always be set to "zero".

2.1.5 Two 1149.1 JTAG Tap Controllers for JTAG Pins (TRST, TCK, TMS, TDI, TDO)

The VCC550x device's have two internal 1149.1 JTAG Tap controllers but only one set of corresponding JTAG pins (TRST, TCK, TMS, TDI, TDO). One TAP controller supports emulation and the other supports JTAG 1149.1 Boundary Scan. Only one of the two TAPs is internally connected to the pins at a time and it is the latched state of the EMU0 pin that determines which TAP is connected. The EMU0 pin is latched on the rising edge of TRST and from that time forward the selected tap is connected to the pins. If the latched state of EMU0 is "0", the boundary scan tap is selected and customers may perform boundary scan testing. If the latched state of EMU0 is "1", the DSP's emulation tap is selected and customers may perform emulation with TI's Code Composer Studio™ IDE Emulation Debugger.

Note: Because of the VCC550x device's internal (and recommended external) pullup on the EMU0 pin and the fact that the emulation pods (e.g., XDS560) do not drive the EMU0 pin while TRST is driven low-to-high, the emulation tap will normally be the one selected. However, customers who wish to do boundary scan testing will need to have an external pulldown, with sufficient strength to overcome the internal pullup, so that the VCC550x boundary scan tap is connected to the JTAG pins.

2.1.6 Bootloader Disables Peripheral Clocks

At hardware reset, all of the peripheral clocks are off to conserve power. The DSP boots into the bootloader code in ROM. During the boot process, the bootloader queries each peripheral to determine if it can boot from the peripheral. At that time, the required peripheral's clock will be enabled for the query and then disabled again when the bootloader is finished with the peripheral. By the time the bootloader releases control to the user code, all peripheral clocks will be off and all domains in the ICR, except the CPU domain, will be idled. After the boot process is complete, the user is responsible for enabling and programming the required clock configuration for the DSP.

For example on the VC5505 device, the bootloader disables both the MPORT and FFTHWA. To enable the MPORT and FFT HWA, write 0x000E to the ICR registers and issue an "idle" command.

Assembly Code Example:

```
*port(#0x0001) = #(0x000E)
        idle
```

C Code Example:

```
*(ioport volatile unsigned *)0x0001 = 0x000E;
        asm("    idle");    // must add at least one blank before idle in " ".
```

For example on the VC5504 device, the bootloader disables the MPORT. To enable the MPORT, write 0x0002E to the ICR registers and issue an "idle" command.

Assembly Code Example:

```
*port(#0x0001) = #(0x0002E)
        idle
```

C Code Example:

```
*(ioport volatile unsigned *)0x0001 = 0x0002E;
        asm("    idle");    // must add at least one blank before idle in " ".
```

2.1.7 HWAFFT Data and Scratch Buffers Must Reside at Data Locations Less Than 0x10000

On silicon revision 1.4, the HWAFFT uses two data buffers, data and scratch, to pass data to the FFT coprocessor, to store intermediate results, and to store the FFT output. The hwafft_Npts routines available in ROM contain pointers to the data and scratch buffers. These pointers are copied as 16-bit addresses instead of 23-bit addresses. When a buffer resides in word address 0x10000 or greater, 23 bits are required to address these memory locations. The hwafft_Npts routine incorrectly copies just the 16 least significant bits, leading to incorrect FFT results and potential corruption of data incorrectly addressed by the HWAFFT.

The user is responsible for following one of the following workarounds:

- Execute hwafft_Npts from RAM or from ROM with data and scratch buffers located entirely in word addresses less than 0x10000.
- Execute hwafft_Npts from RAM using the updated hwafft.asm with bug fixes. The hwafft.asm file included with SPRABB6.zip incorporates bug fixes for this errata. The HWAFFT routines stored in ROM cannot be updated and do not incorporate these bug fixes. For reference, hwafft_rom.asm is included in SPRABB6.zip and contains the HWAFFT routines exactly as they exist in the ROM of the affected revisions.
- Execute hwafft_Npts from RAM or from ROM while passing duplicate pointers to the scratch and data buffers. The data and scratch buffers can be located in word addresses greater than 0x10000, but the buffers must not cross 16-bit address boundaries (that is, address bits 22–16 must not change). Additionally, if hwafft_512pts is used, data and scratch must not reside at the beginning of a page boundary (that is, word address 0x10000 or 0x20000). This workaround initializes the most significant bits of internal registers XAR2 and XAR3 such that when the 16-bit address is copied from XAR0 and XAR1, and provided the assumptions above are satisfied, the full 23-bit address remains correct throughout HWAFFT execution.

  In the user program wherever hwafft_Npts is called, replace:

```
out_sel = hwafft_Npts(data, scratch, fft_flag, scale_flag);
```

  With:

```
out_sel = hwafft_Npts(data, scratch, scratch, data, fft_flag, scale_flag);
```

  In hwafft.h, replace:

```
Uint16 hwafft_Npts(
    Int32 *data,
    Int32 *scratch,
    Uint16 fft_flag,
    Uint16 scale_flag
);
```

  With:

```
Uint16 hwafft_Npts(
    Int32 *data,
    Int32 *scratch,
    Int32 *duplicate_scratch,
    Int32 * duplicate_data,
    Uint16 fft_flag,
    Uint16 scale_flag
);
```

2.2 Silicon Revision 1.4 Known Design Exceptions to Functional Specifications

Table 2. Silicon Revision 1.4 Advisory List

Advisory 1.4.1 *CPU, USB, and EMIF: Endianess Incompatibility*

Revision(s) Affected 1.4

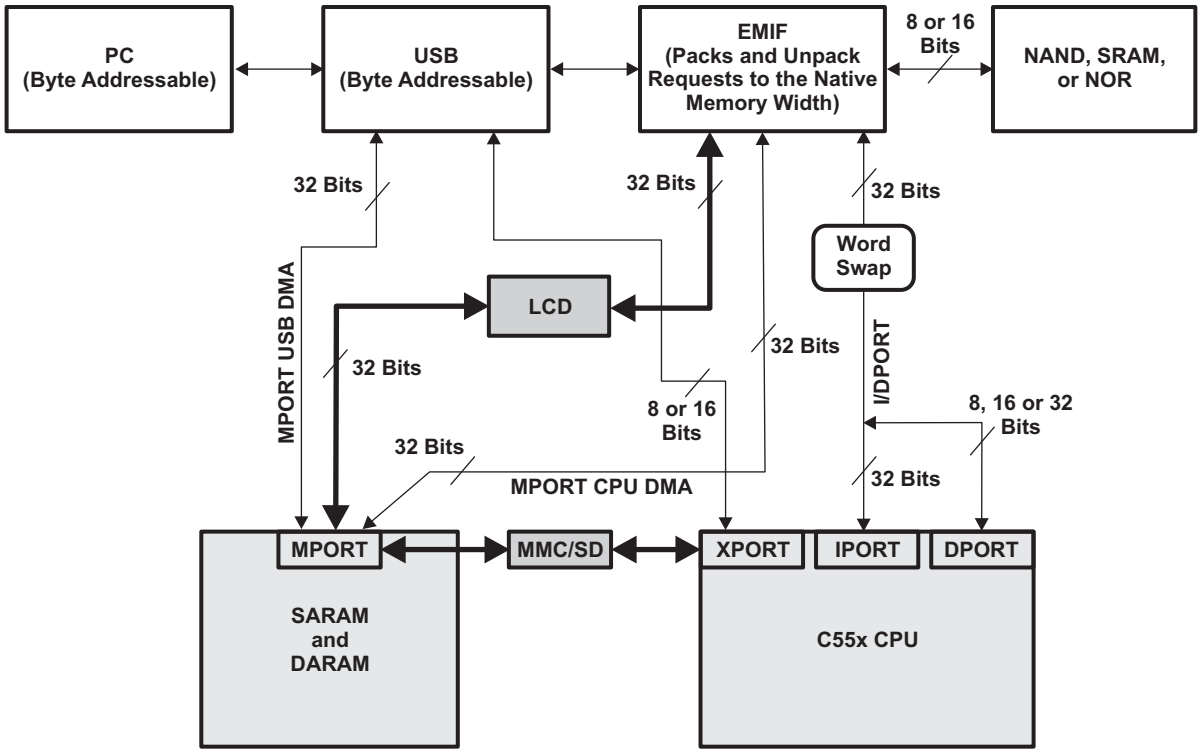
Details

The VC550x CPU is a word addressable and big endian architecture. The CPU interfaces to the rest of the system through several ports: MPORT, XPORT, DPORT, and IPORT. The DMA transfers data from peripherals to on-chip memory through the MPORT in 32-bit packets. The CPU accesses the peripherals through the XPORT in 8- or 16-bit packets. The CPU accesses external memory through the DPORT in the EMIF in 8-bit (configured through a system register setting) , 16-bit (CPU single-word access) or 32-bit (CPU dual-word access) packets. The CPU fetches code from external memory through the IPORT in the EMIF in 32-bit packets.

Accesses through the MPORT are big endian. Accesses through the XPORT are just 8-bit (for the USB through a system register setting) or 16-bits. 32-bit accesses through the DPORT are word swapped (MSW and LSW of a 32-bit access are swapped).

Some VC550x peripherals (e.g., EMIF, USB, LCD, and MMC/SD) have sensitivity to the data endianness. MMC/SD and LCD endianness is controlled through software. On the other hand, USB and EMIF, are hardcoded to little endian. Data transfers between incompatible endianess systems could result in swapped bytes/words based on the size of the data. For example, if the CPU writes data to external memory, the transfer occurs through the DPORT which swaps the MSW and LSW. If the DMA transfers the data from external memory to on-chip memory, the transfer occurs through the MPORT. The MPORT does not swap words; therefore, if the CPU compares the original values it wrote to external memory with those written by the DMA, they will be word swapped.

Figure 2 shows data paths that could create byte/word swaps.



A The "SARAM and DARAM" and C55x CPU blocks are big endian architecture.

B The "LCD" and "MMC/SD" blocks are software configurable for endianess.

C The "PC", "USB", "EMIF", and "NAND, SRAM, or NOR" blocks are little endian architecture.

Figure 2. CPU, USB, and EMIF Data Paths (A)(B)(C)

Example Case 1: Data transfer between the EMIF and CPU (see Figure 3 and Figure 4).

Data can be written to or read from the EMIF by the DMA or CPU. The DMA accesses on-chip memory through the MPORT and only performs a 32-bit read/write. The CPU performs a read/write as either 32-bit or 16-bit through the DPORT. The CPU can fetch instructions from the EMIF through the IPORT. Since IPORT and DPORT perform a word swap, while the MPORT does not perform a word swap, the word/byte swap occurs only if the DMA and CPU are combined in the reading and writing (writing with the DMA and reading with the CPU or vice-versa). In a case where only the DMA or only CPU is used for both writing and reading, no word/byte swap will occur.

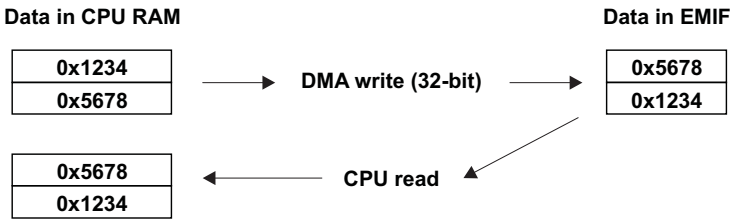


Figure 3. DMA Write and CPU Read

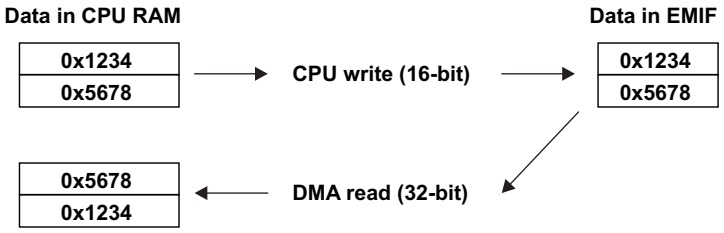


Figure 4. CPU Write and DMA Read

Example Case 2: Data transfer between the USB and CPU (see Figure 5 and Figure 6).

If the PC host (handles data in little endian mode) writes data to the VC550x through the USB, the USB peripheral transfers the data to on-chip memory through MPORT. Since MPORT does not swap data and the USB module is in little endian mode, when the CPU reads data from on-chip memory, the data will be byte and word swapped. If instead of using the USB peripheral's DMA to read the data, the CPU reads the data from the USB (using XPORT), the data is byte swapped at word boundaries.

Data can be read from the USB by the USB DMA or CPU. The USB DMA accesses on-chip memory through the MPORT and only performs a 32-bit read while the CPU can only perform a 16-bit read via the XPORT. The bytes/words of data will be out-of-order based on the size of original data.

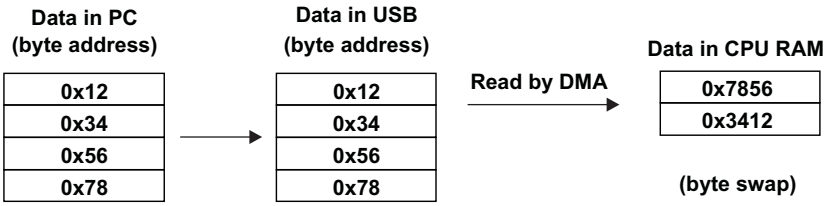


Figure 5. USB DMA Read

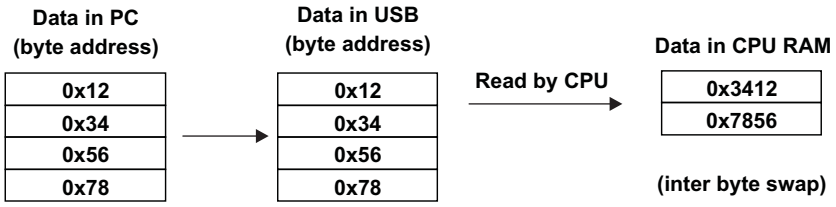


Figure 6. CPU Read From USB

Workaround(s)

To correct this issue, software is required to perform a word/byte swap on the data prior to processing on the C550x CPU. Two software examples are given for correcting word swapped data from/to the EMIF. For USB data read/write, it is recommended to do the correction in the PC side because the PC is byte addressable and offers more flexibility to order data.

Workaround #1 – Software example to correct word/byte swap between the EMIF and CPU (DMA Write and CPU Read):

```
Uint16 buff_in_RAM[N]; // buffer in CPU RAM
    Uint16 buff_in_EMIF[N]; // buffer in EMIF
     Uint16 I;
    ...
      // read from EMIF after writing data with DMA to EMIF
   for(I=0;i<N;i+=2) // locate even address (EMIF) to odd address (CPU)
   {
       buff_in_RAM [I+1] = buff_in_EMIF[i];
   }
   for(I=1;i<N;i+=2)  // locate odd addresses (EMIF) to even addresses (CPU)
   {
       buff_in_RAM [I-1] = buff_in_EMIF[i];
   }
```

Workaround #2 – Software example to correct word/byte swap between the EMIF and CPU (CPU Write and DMA Read):

```
Uint16 buff_in_RAM[N]; // buffer in CPU RAM
 Uint16 buff_in_EMIF[N]; // buffer in EMIF
  Uint16 I;
 ...
   // writing to EMIF. This data will be read by DMA
 for(I=0;i<N;i+=2) // locate even address (CPU) to odd address (EMIF)
 {
      buff_in_EMIF [I+1] = buff_in_RAM [I];
 }
 for(I=1;i<N;i+=2)  // locate odd addresses (CPU) to even addresses (EMIF)
 {
      buff_in_EMIF [I-1] = buff_in_RAM [I];
 }
```

Advisory 1.4.2 *RTC: RTC Positive Compensation – Multiples of Ten Values Do Not Work*

Revision(s) Affected 1.4

Details The RTC positive compensation only works for compensation values that are ***not*** multiples of 10. If a multiple of ten is needed, perform the compensation in two steps using non-multiples of ten. For example, to perform a compensation of 100, perform a compensation by 98 followed by a compensation of 2.

Workaround(s) Do not use compensation values that are multiples of 10.

Advisory 1.4.3 *I2S: Invalid I2S OUERRFL Error Report at First Frame*

Revision(s) Affected 1.4

Details After a hardware reset, the Overrun or Underrun (Error) condition flag (OUERRFL) in the I2SINTFL register (0x2810, 0x2910, 0x2A10, 0x2B10) is always set high when the corresponding I2S channel is enabled and the OUERR mask bit field in the I2SINTMASK register (0x2814, 0x2914, 0x2A14, 0x2B14) are enabled.

This bug occurs *only once* when I2S is enabled after a hardware reset.

Workaround(s) 1) Wait at least 5 frames before enabling the OUERR mask bit field in the I2SINTMASK register ***or*** 2) if the OUERR flag in the I2SINTMASK register is enabled from the beginning, ignore the first OUERR error.

Advisory 1.4.4 *I2S: I2S Internal Data Delay*

Revision(s) Affected 1.4

Details

The I2S module has an internal delay for the data transmit/receive path that varies depending on the settings of the Pack bit and Word Length in the I2S Control Register and the FSDIV in the Sample Rate Generator Register.

Table 3 shows the transmit path internal data delays. Feedback path refers to an I2S transmit pin that is externally connected to the I2S receive pin.

Table 3. Transmit Path Internal Data Delays

PACK	DATA DELAY
1	The first five transmit frames will be invalid data. On the sixth transmit frame, the data written to the transmit register will be shifted out on the DX pin.
0	The first three transmit frames will be invalid data. On the fourth transmit frame, the data written to the transmit register will be shifted out on the DX pin.

Table 4 shows the receive path internal data delays:

Table 4. Receive Path Internal Data Delays

WORD LENGTH	DATA DELAY
8-bit	The RX data registers will contain invalid data for the first two frames.
10, 12, 14, and 16-bit	The RX data registers will contain invalid data for the first three frames.
18, 20, 24, and 32-bit	The RX data registers will contain invalid data for the first two frames.

Table 5 shows the feedback path internal data delays:

Table 5. Feedback Path Internal Data Delays

PACK	WORD LENGTH	FSDIV BITS I2SSRATE.[5:3]	DATA DELAY
1	8-bit	000	Data received on the 7th sample
	8-bit	001 – 101	Data received on the 6th sample
	10-, 12-, 14-, and 16-bit	001	Data received on the 5th sample
	10-, 12-, 14-, and 16-bit	010 – 101	Data received on the 4th sample
0	8-, 10-, 12-, 14-, 16-, 18-, 20-, 24-, and 32-bit	000 – 101	Data received on the 3rd interrupt

Workaround(s) Invalid data should be Ignored.

Advisory 1.4.5 *USB: USB Queue Manager Reads Only 16-bit Address of USB Descriptors*

Revision(s) Affected 1.4

Details The VC550x has 23-bit address space but the USB Queue Manager (QMGR) is a 32-bit register that holds the address of the USB descriptors. A descriptor itself is a structure with information about the addresses of the source/destination data buffers and their sizes. The address of a particular descriptor is written to the QMGR register for a particular DMA endpoint. The CPU writes the address of a descriptor to the QMGR register for a for a DMA endpoint. The QMGR fires the USB DMA to read the descriptor at the address pointed to in the QMGR register and sets up the DMA endpoints for future transfers. When a USB host connects and performs a transfer, the QMGR copies the address of the descriptor to a completed queue. Upon receiving the USB interrupt, the USB driver should read the 32-bit descriptor address in the QMGR completion queue to determine which DMA endpoint has completed transferring data. Even though the CPU can write a 32-bit value into the QMGR register, it can only read the lower 16-bits of this register. Thus, the descriptor can only be allocated in the CPU memory map to the same lower 16-bit address and all descriptors must be placed in one contiguous block of 64K words in SARAM.

For example:

- USB descriptor A is located at 0x008000
- USB descriptor B is located at 0x018000

The descriptor A and B will be considered the same descriptor.

Workaround(s) The USB descriptors should be placed in ONE CONTIGUOUS BLOCK of 64K words (2^{16}) in memory.

Advisory 1.4.6 *DMA: In H/W Sync Mode, Auto Reload Bit Overrides Enable Bit*

Revision(s) Affected 1.4

Details When Hardware (H/W) Sync Mode (SYNCMODE bit = 1) and Auto Reload (AUTORLD bit = 1) are enabled, the Auto Reload bit overrides the Enable bit (EN). Therefore; if the SYNCMODE bit and AUTORLD bit in the DMACHmTCRn register are both set to "1", the designated hardware event for the selected DMAx channel will trigger a DMA data transfer even if the Enable bit (EN) is "0". For more detailed information, see Table 6.

Table 6. DMA Data Transfer Settings Matrix [H/W Sync Mode]

DMACHmTCR2 REGISTER		EVENT SOURCE REGISTER (DMAnCESRn)	DMA TRIGGERED BY
ENABLE BIT (EN)	AUTO RELOAD BIT (AUTORLD)		
0	0	Don't Care	None (DMA will not Start Data Transfer)
1	0	Any Event Assigned	The Assigned Hardware Event
0	1	Any Event Assigned	The Assigned Hardware Event
1	1	Any Event Assigned	The Assigned Hardware Event
Don't Care	Don't Care	No Event (Not Assigned)	None (DMA will not Start Data Transfer)

Workaround(s) In Hardware Sync Mode (SYNCMODE bit = 1), use the Auto Reload bit (AUTORLD) versus the Enable bit (EN) to enable the selected DMAx Channel m.

Advisory 1.4.7 *DMA: Hardware Event can Trigger DMA Data Transfer in S/W Control Mode*

Revision(s) Affected 1.4

Details

In Software (S/W) Control Mode (SYNCMODE bit = 0), the Enable bit (EN) triggers a DMA data transfer only when the Auto Reload bit (AUTORLD) is set to "0". If the Auto Reload bit (AUTORLD) is set to "1", the DMA data transfer is triggered by the designated hardware event for the selected DMAx channel, not by the Enable bit (EN). When the AUTORLD bit is set to 1, the EN bit is ignored. Note if the DMA data transfer is initiated by a hardware event in Software Control Mode, the DMA data transfer cannot be stopped by clearing *only* the EN bit; both the AUTORLD and the EN bits should be cleared (set to "0") to stop the DMA data transfer. For more detailed information, see Table 7.

Table 7. DMA Data Transfer Settings Matrix [S/W Control Mode]

DMACHmTCR2 REGISTER AUTO RELOAD BIT (AUTORLD)	EVENT SOURCE REGISTER (DMAnCESRn)	DMA TRIGGERED BY	TO STOP DMA
0	No Event (Not Assigned)	Enable Bit = 1	• After transferring the data amount into the transfer length register, the DMA will be stopped. • If EN bit is set to "0" in the middle of a DMA data transfer, the DMA will be stopped.
0	Any Event Assigned	Enable Bit = 1	• After transferring the data amount into the transfer length register, the DMA will be stopped. • If EN bit is set to "0" in the middle of the DMA data transfer, the DMA will be stopped.
1	No Event (Not Assigned)	None; DMA will not Start Data Transfer	N/A
1	Any Event Assigned	Hardware Event Triggers Data Transfer	• AUTORLD and EN bits both should be cleared (set to "0") to stop the DMA data transfer. The DMA will finish the current burst and then stop the data transfer.

Workaround(s)

Configure the Event Source Register to “no event” and make sure to clear the Auto Reload bit (AUTORLD) in the DMAnCESRn register before entering Software Control Mode to ensure that the DMA data transfer can *only* be triggered by the Enable bit (EN).

Advisory 1.4.8 *DMA: DMA Transfer Length Must be a Multiple of 4 x 2(Burst Mode)*

Revision(s) Affected

1.4

Details

If the transfer length register has a value that is zero or *not* a multiple of 4 x 2(Burst Mode) when the DMA transfer begins, it will cause an unexpected operation of DMA.

While the DMA transfers 32-bit words from a source address to a destination address, the value in the DMA transfer length register is the length in bytes. For example, if the total DMA transfer length is 4 32-bit words and the burst size = 1, 4 x 4 = 16 should be written to the DMA transfer length register. The burst size should also be considered. Burst size is the minimum data transfer size; therefore, the total DMA Transfer Length should be 4 x 2(Burst Mode). For more details, see Table 8

Table 8. DMA Transfer Lengths

BURST MODE	BURST SIZE (32-BIT WORD)	DMA TRANSFER LENGTH (BYTES)
0	1	Multiple of 4 (minimum 4)
1	2	Multiple of 8 (minimum 8)
2	4	Multiple of 16 (minimum 16)
3	8	Multiple of 32 (minimum 32)
4	16	Multiple of 64 (minimum 64)

Workaround(s)

Write only a multiple of 4 x 2(Burst Mode) to the DMA transfer register before the DMA starts.

Advisory 1.4.9 *Aggregated Interrupt (GPIO, DMA, and Timer)*

Revision(s) Affected 1.4

Details

This advisory applies to the interrupt aggregation registers of the VC5505:

- GPIO - **GPIO Interrupt Flag Registers (IOINTFLG1 and IOINTFLG2)**
- DMA - **DMA Interrupt Flag Register (DMAIFR)**
- Timer - **Timer Interrupt Aggregation Flag Register (TIAFR)**

The interrupt aggregation registers' purpose is to consolidate interrupts from multiple sources into one single interrupt signal to the CPU. When the CPU receives an interrupt from one of the aggregated sources, the corresponding flag in the CPU IFR sets. If that interrupt is not masked, then the CPU is interrupted and branches to the interrupt service routine (ISR). Entry into the ISR automatically clears the CPU's IFR flag, but the ISR must read the *aggregation IFR* to determine the exact source of the interrupt and the ISR must also write '1' to the aggregation IFR to clear its flag.

If there are still flags that remain set after writing a 1 to clear a flag in the aggregation register, then the CPU will be interrupted again as a normal behavior. However, the CPU will also be interrupted again if one of the following conditions is met:

1. If the ISR writes a 1 to the aggregation IFR flag *that was not already set* then the CPU will be re-interrupted.
2. If the ISR writes all 0s to the aggregation IFR register and the aggregation IFR register already had at least one flag set, then the CPU will be re-interrupted.

Workaround(s) Only attempt to clear the aggregation IFR flags which are already set to 1.

Advisory 1.4.10 *RTC BCD Value Error*

Revision(s) Affected 1.4

Details The binary coded decimal values of RTC Day and Month register incorrectly updates from 09 (0x0000 1001) to 1A (0x0001 1010) instead of correctly updating from 09 to 10. Once it occurs, the Day and Month registers increase incorrectly. For example 1A →1B → 1C → 1D → 1E → 1F → 10 → 11…

Workaround(s) RTC Day and Month register values must be reloaded with correct number 10 (0x00010000) when 1A (0x00011010) is read.

Advisory 1.4.11 *USB CPPI Receive Starvation Interrupt*

Revision(s) Affected 2.0

Details When an endpoint is enabled for receive transfer(s) that will be serviced via CPPI DMA and data has been received prior to allocating the DMA resource, the DMA will generate a starvation interrupt to notify the application a lack of resource (starvation) in anticipation that the application will furnish the required resource. Once the starvation occurs, the CPPI DMA continues generating interrupts periodically whenever the host tries to send data. It does not stop until the application furnishes a resource.

The USB starvation interrupt is always enabled and cannot be masked off at the USB controller level. Since the DMA continues to generate the starvation interrupt periodically and there exists no capability to mask the starvation interrupt at the USB controller level, the CPU is forced either to fully service the DMA interrupt as it is received or disable all USB interrupts at the CPU level. Disabling the entire USB interrupt might not be the desired option since the CPU needs to be aware of other USB interrupts that are more critical.

Workaround(s) Dedicated data receiving buffers are recommended. The data buffers should be allocated and available to the CPPI DMA during USB initialization. The required data buffer size is highly dependent on USB host applications. For Windows XP USB Mass Storage device driver, it is recommended to allocate at least 64 KB (128 descriptors; 128*512 = 64 KB) of data buffer space.

Advisory 1.4.12 *Aggregation Interrupts Can Be Missed*

Revision(s) Affected 1.4

Details

Under certain circumstances, CPU interrupts from the GPIO, DMA, and Timer can be missed.

The interrupt aggregation registers for GPIO, DMA, and Timers are physically implemented as 32-bit registers composed of a low-order 16-bit word at the even address and a high-order 16-bit word at the next higher odd address. The CPU accesses these registers as 16-bit values. When the CPU writes to either the low or high word, that write can block an interrupt coming in simultaneously (in the exact same clock cycle) on the opposite word. When the interrupt is blocked, the interrupt flag register will not get set and the interrupt will be missed.

For DMA Interrupt Flag Register, it is slightly different since the 32-bit physical register is composed of 16-bits of the DMA Interrupt Enable Register in the high-order word position and 16-bits of the DMA Interrupt Flag Register in the low-order word position. A write to the DMAIER register can block a simultaneous new DMA interrupt coming into the DMA Interrupt Flag Register. When the interrupt is blocked, the interrupt flag register will not get set and the interrupt will be missed.

For Timer Interrupt Aggregation Flag Register, it is also slightly different since the physical register is composed of 16-bits of reserved bits in the high-order word position and the low-order word is composed of 13-bits of reserved bits and 3-bits of timer interrupt flags. A write to the reserved register (high-order word position) can block a simultaneous new Timer interrupt coming into the Timer Interrupt Aggregation Flag Register. When the interrupt is blocked, the interrupt flag register will not get set and the interrupt will be missed.

Although it is possible, this situation is extremely unlikely to occur since writing to the aggregation interrupt register should occur concurrently (in the exact same clock cycle) with the new incoming interrupt.

Workaround(s)

- For GPIO: Do not use the interrupts of GPIO[15:0] and the interrupts of GPIO[31:16] at the same time. When the interrupt of GPIO[15:0] are enabled, disable the interrupt of GPIO[31:16] or vice versa.
- For DMA: Only start the DMA data transfer after the completion of the DMA configuration and do not attempt to write the DMA Interrupt Enable Register while DMA is running.
- For Timer: Do not write the reserved register at address 0x1C15.

Revision History

This silicon errata revision history highlights the technical changes made to the SPRZ281B revision to make it an SPRZ281C revision.

Scope: Applicable updates relating to the TMS320VC5505/VC5504 devices have been incorporated. Added device-specific information supporting the TMS320VC550x Silicon Revision 1.4 devices, which are now in the production data (PD) stage of development.

Revision History

SEE	ADDITIONS/MODIFICATIONS/DELETIONS
Section 2.1 Usage Notes for Silicon Revision 1.4	Added usage note: • Section 2.1.7, *HWAFFT Data and Scratch Buffers Must Reside at Data Locations Less Than 0x10000*

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

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

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

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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