TMS320C6202, TMS320C6202B

Digital Signal Processors

Silicon Errata

C6202 Silicon Revisions 1.0, 1.1, 1.2

C6202B Silicon Revisions 3.0, 3.1

SPRZ152H
November 2000
Revised February 2004

Texas Instruments

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPRZ152G revision to make it an SPRZ152H revision.

Scope: This document has been reviewed for technical accuracy; the technical content is up-to-date as of the specified release date.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
	None

Contents

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320C6202 silicon releases 1.2, 1.1, and 1.0, and for the TMS320C6202B silicon releases 3.0 and 3.1.

For additional information on the TMS320C6202 and TMS320C6202B devices, see the latest version of the *TMS320C6000 DSP Peripherals Overview Reference Guide* (literature number SPRU190) and the latest version of the *TMS320C6202, TMS320C6202B Fixed-Point Digital Signal Processors* data sheet (literature number SPRS104).

1.1 Quality and Reliability Conditions

TMX Definition

Texas Instruments (TI) does not warranty either (1) electrical performance to specification, or (2) product reliability for products classified as “TMX.” By definition, the product has not completed data sheet verification or reliability performance qualification according to TI Quality Systems Specifications.

The mere fact that a “TMX” device was tested over a particular temperature and voltage ranges should not, in any way, be construed as a warranty of performance.

TMP Definition

TI does not warranty product reliability for products classified as “TMP.” By definition, the product has not completed reliability performance qualification according to TI Quality Systems Specifications; however, products are tested to a published electrical and mechanical specification.

TMS Definition

Fully-qualified production device

1.2 Revision Identification

The device revision can be determined by the lot trace code marked on the top of the package. The location of the lot trace code for the GLS (C6202) and the GNZ (C6202B) packages is shown in Figure 1.

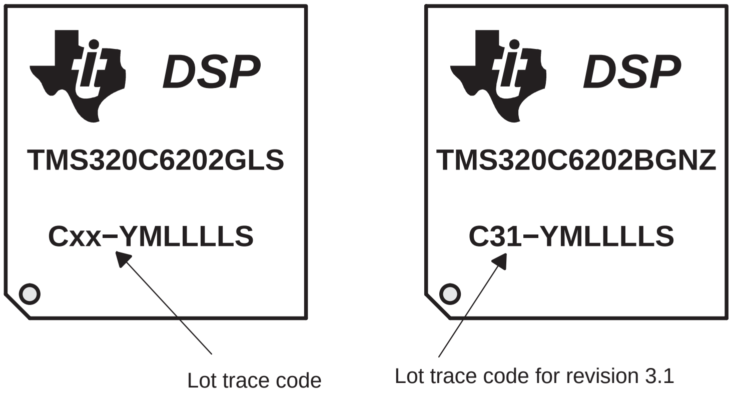


Figure 1. Example, Lot Trace Code for TMS320C6202 and TMS320C6202B

NOTE: Qualified devices are marked with the letters “TMS” at the beginning of the device name, while nonqualified devices are marked with the letters “TMX” at the beginning of the device name.

Table 1. Lot Trace Code Names

Lot Trace Code	Silicon Revision	Comments
31	3.1	TMS320C6202B (GNZ/GNY). There are ***no*** known design exceptions to functional specifications for TMS320C6203B silicon revision 3.1.
30	3.0	TMS320C6202B. All advisory issues presented on TMS320C6202 silicon revision 1.2 and earlier have been resolved.
12	1.2	TMS320C6202 Silicon Revision 1.2 is functionally the same as revision 1.1. It is optimized from revision 1.1 for yield improvement.
11	1.1	TMS320C6202
10	1.0	TMS320C6202

2 C6202B Silicon Revision 3.1 Known Design Exceptions to Functional Specifications and Usage Notes

There are no known design exceptions to functional specifications or usage notes for the TMS320C6202B silicon revision 3.1.

3 C6202B Silicon Revision 3.0 Known Design Exceptions to Functional Specifications and Usage Notes

Advisory 3.0.1 — *Memory: Potential Read After Write Error*

Revision(s) Affected: 3.0

Details: In the event that ALL of the following five conditions are met simultaneously, a data bit misread may occur in internal device memory:

1. Write immediately followed by a read
2. The write and read occur in back-to-back cycles without any stalls
3. Read must be in same 16-bit-wide bank as preceding write.
   (The addressing structure of the physical banks is described below.)
4. Read must access the same physical column as the preceding write (read offset from write by a multiple of 0x100 bytes)
5. Read value must be of a different state than that of the preceding write (e.g., if write value is 0x0, then read value must be different from 0x0)

Note: The read and/or write conditions can occur via the CPU, DMA, or a combination of the two.

- CPU Write, DMA Read
- CPU Write, CPU Read
- DMA Write, DMA Read
- DMA Write, CPU Read

Back-to-back accesses typically occur in tightly looped code. Tightly looped code rarely does both reads and writes to the same bank offset by a multiple of 0x100 bytes. In algorithms that do stride by this amount, typically loads are done for different iterations than stores so they are not offset in the software pipelined loop by this amount.

The C6202B data memory consists of 8 separate 16-bit-wide banks. These banks are broken down into 2 blocks of 4 banks. Data memory is addressed as follows:

Memory: Potential Read After Write Error (Continued)

ADDR[31]	→	Data Memory Select
ADDR[30..17]	→	Reserved
ADDR[16]	→	Block Select
ADDR[15..7]	→	Row Select
ADDR[6..3]	→	Column Select
ADDR[2..1]	→	16-bit Bank Select
ADDR[0]	→	Byte Enable

Only when a write followed by a read occurs and the Row Select bits (ADDR[15..7]) are different will this error potentially occur.

For example, the error may occur using the following sequence:

Write 0x12345678 to address 0x80010042

Read from address 0x80010542 ← Only Row Select changes

In this case, the read may come back with a different value than expected.

The error will ***not*** occur under the following sequences:

Write 0xFEDCBA98 to address 0x80000054

Read from address 0x80010554 ← Different Block

or

Write 0x55AA55AA to address 0x80010042

Read from address 0x80010562 ← Different column

Assuming all five conditions above have been met, this issue is more likely to be encountered at:

- 1.5-V, rather than 1.7-V core supply voltage. 1.7 V significantly reduces the likelihood of the error.
- Higher, rather than lower, device case operating temperature
- Higher, rather than lower, device operating frequency. 250-MHz operation significantly reduces the likelihood of error.

Note that this behavior is ***NOT*** a reliability risk. It should not degrade for a given set of operating conditions and code.

Workaround: The occurrence of all five conditions is application-specific. Code should be examined to determine if the five conditions above are met. If all conditions are met, insert at least one dead cycle between the write and read.

4 C6202 Silicon Revision 1.2 Known Design Exceptions to Functional Specifications

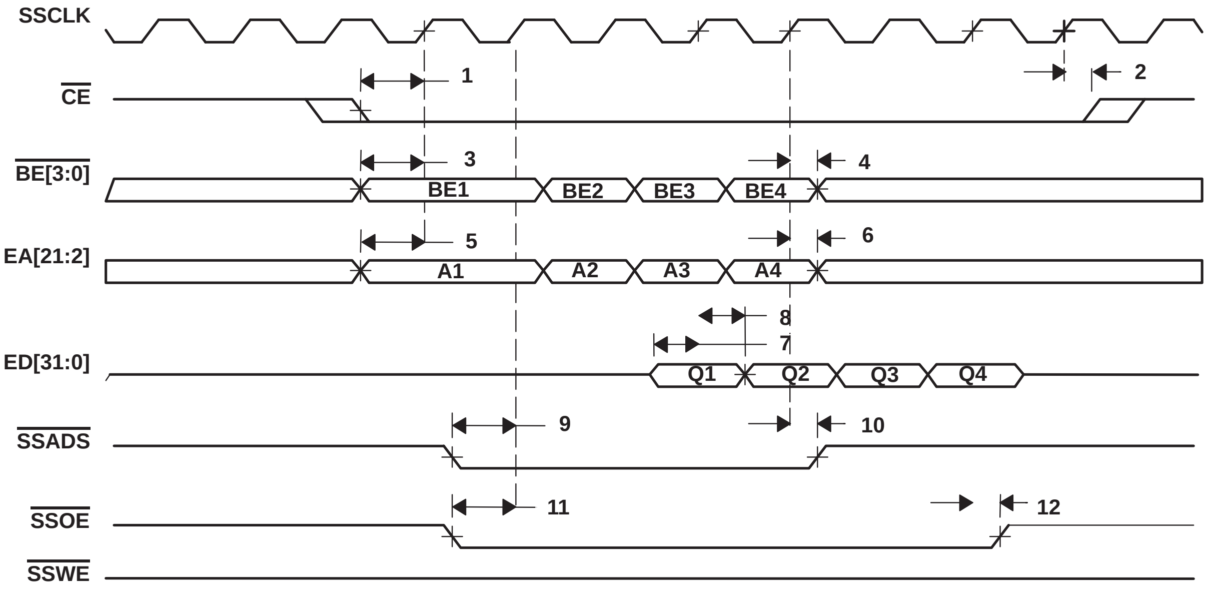


Figure 2. SBSRAM Read Timing

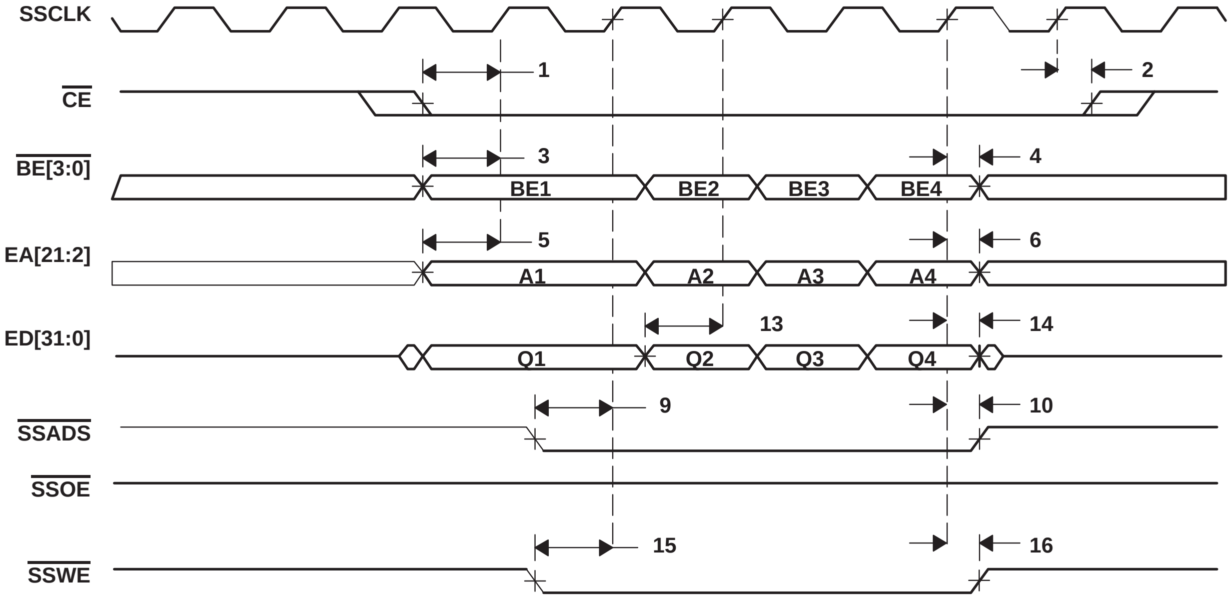


Figure 3. SBSRAM Write Timing[†]

[†] The CEx output setup and hold times are specified to be accurate relative to the clock cycle to which they are referenced, since these timings are specified as minimums. However, the CE output setup and hold time may be greater than that shown in the data sheet in multiples of P ns. In other words, for output setup time, the CEx transition from high to low may happen P, 2P, …, or nP ns before the time specified by the data sheet. Similarly, for output hold time, the CEx low-to-high transition may happen P, 2P, …, or nP ns after the time specified by the data sheet. This is indicated by the period of uncertainty for specs 1 and 2 in Figure 2 and Figure 3.

Advisory 1.2.1 — *EMIF: Invalid SDRAM Access to Last 1K Byte of CE3*

Revision(s) Affected: 1.2 and earlier

Details: If 16M bytes of SDRAM (2 64M-bit in a 1Mx16x4 organization) is used in CE3, you can have invalid accesses to the last 1K byte of CE3 (0x03FFFC00).

This occurs when the following is true:

- After a DCAB (Deactivate all pages) to all SDRAM CE spaces (forced by Refresh or MRS command)
- The first access to CE3 is to the last page of CE3 (0x03FFFC00).
- Then, a page activate will not be issued to CE3. Since the SDRAM in CE3 is in a deactivated state at that point, invalid accesses will occur. (Internal reference number C630280)

Workaround: Best Case: Avoid designing a board with a 64M-bit (1Mx16x4) SDRAM mapped into CE3.

Alternative: If a 64M-bit SDRAM is located in CE3, avoid using the last 1K byte in the CE3 memory map (0x03FFFC00).

Advisory 1.2.2 — *Cache During Emulation With Extremely Slow External Memory*

Revision(s) Affected: 1.2 and earlier

Details: If a program requests fetch packet “A” followed immediately by fetch packet “B”, and all of the following four conditions are true:

1. A and B are separated by a multiple of 64K in memory (i.e., they will occupy the same cache frame)
2. B is currently located in cache
3. You are using the emulator to single-step through the branch from A to B
4. The code is running off of an extremely slow external memory that transfers one 32-bit word every 8000 or more CPU clock cycles (CPU running at 200 MHz)

Then, A will be registered as a “miss” and B will be registered as a “hit”. B will not be reloaded into cache, and A will be executed twice. This condition is extremely rare because B has to be in cache memory, and must be the next fetch packet requested after A (which is not in cache memory). In addition, this problem only occurs if you single-step through the branch from A to B using the emulator, *and* when the code is located in an extremely slow external memory. (Internal reference number C630283)

Cache During Emulation With Extremely Slow External Memory (Continued)

Workaround:

- Do not single-step through the branch from A to B if the above conditions are true.
- Do not use an extremely slow external memory (transfers one 32-bit word every 8000 or more CPU clock cycles) if conditions 1, 2, and 3 are true.

Advisory 1.2.3 *XBUS: Corruption of Data Via Synchronous Host Port Mode*

Revision(s) Affected: 1.2 and earlier

Details: While the expansion bus is operating in Synchronous Host Port Mode, certain expansion Bus output signals may be corrupted. The corruption occurs on output signals only and there are no known issues with any other XBUS modes (Asynchronous Host Mode, Asynchronous I/O Mode, or Synchronous FIFO Mode). In Synchronous Host Port Mode, some or all outputs of the XD signal lines move from the next word into the current word being transferred. This corruption not only affects the XD signal lines, but the output control signals as well (XAS, XBLAST, etc.). Figure 4 shows what a typical Synchronous Host transfer should look like. Figure 5 shows the address (AD), as well as data word 2 (D2) being corrupted by the next word. In this figure, the address is corrupted by D0 and D2 is written over by D3. Figure 6 is an example showing D0 being completely corrupted by D1 and D3 being partially corrupted by D4. The transfer should be 0x0001, 0xFF02, 0x0003, ..., 0xFF1F to Address 0x8000.

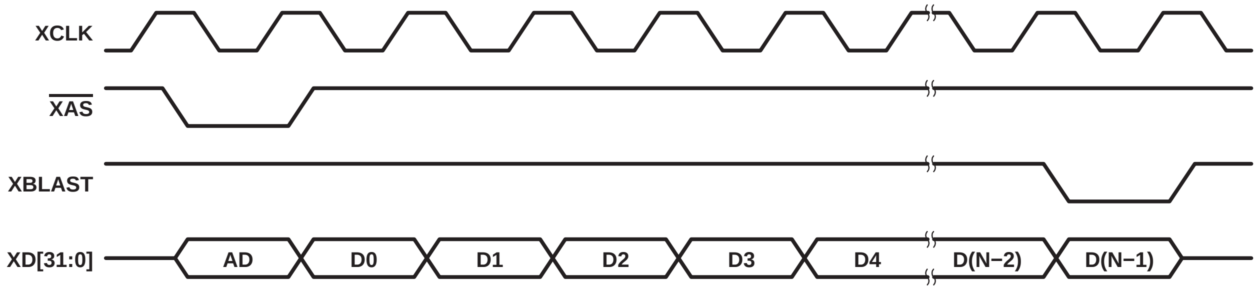


Figure 4. Correct Transfer of Address and Data via the Synchronous Host Port Mode of the Expansion Bus

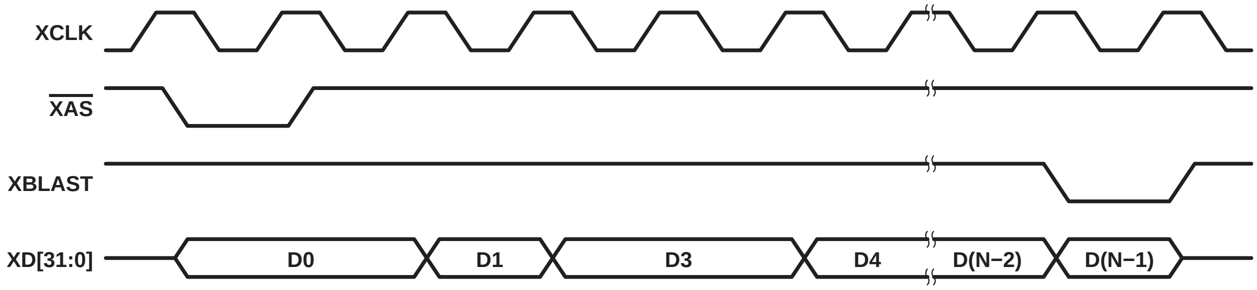


Figure 5. Incorrect Transfer via the Synchronous Host Port Mode of the Expansion Bus.

XBUS: Corruption of Data Via Synchronous Host Port Mode (Continued)

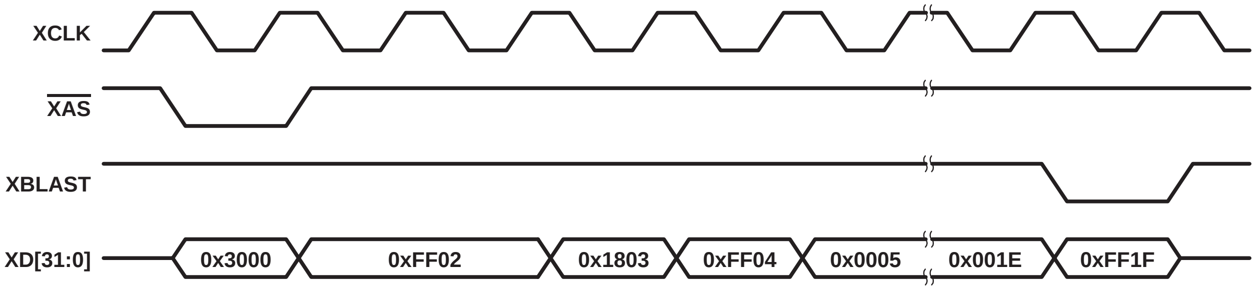


Figure 6. Example Showing Partial Corruption

Workaround: The best method of reducing the number of failures resulting from this issue is to increase the voltage and decrease the CPU Clock speed. For Rev. 1.2, 1.8 V − 250 MHz devices should operate correctly at 1.9 V − 225 MHz. This issue will be fixed in the next major revision.

Advisory 1.2.4 — *XBUS: XBUS and DMA Hang When Using XRDY During Asynchronous I/O*

Revision(s) Affected: 1.2 and earlier

Details: An XRDY synchronization problem causes incorrect operation on the expansion bus. When using the expansion bus asynchronous I/O interface with XRDY, the last word in a data transfer may not be captured. The XBUS interface will seem to act correctly but the last word will not be stored in memory and the direct memory acccess (DMA) will not complete. The XBUS and the DMA are stalled after the last access, not allowing any subsequent DMA or XBUS transfers to complete. This problem has not been reported internally or externally to date but it could affect the XBUS operation if the recommendations in the workaround are not followed.

Workaround: Do not use XRDY to stall the expansion bus. Instead lengthen the access parameters (setup, strobe, hold) to the maximum and ensure that the external device is ready in that time frame or only make XBUS requests after the external device is ready.

5 C6202 Silicon Revision 1.0 Known Design Exceptions to Functional Specifications

Advisory 1.0.3 *XBUS: Invalid Data Transfer in Slave Synchronous Host Port Mode When XARB = 1*

Revision(s) Affected: 1.0

Details: If the internal bus arbiter is enabled (XARB = 1) and the external host asserts the XHOLD to start a new transfer before the auxiliary DMA channel has finished the previous external host write transfer internally, the transfers get corrupted. (Internal reference number C630338)

Workaround: Use an external bus arbiter (XARB = 0).

Advisory 1.0.4 *XBUS: FIFO Data and Control Signals Skewed from XFCLK*

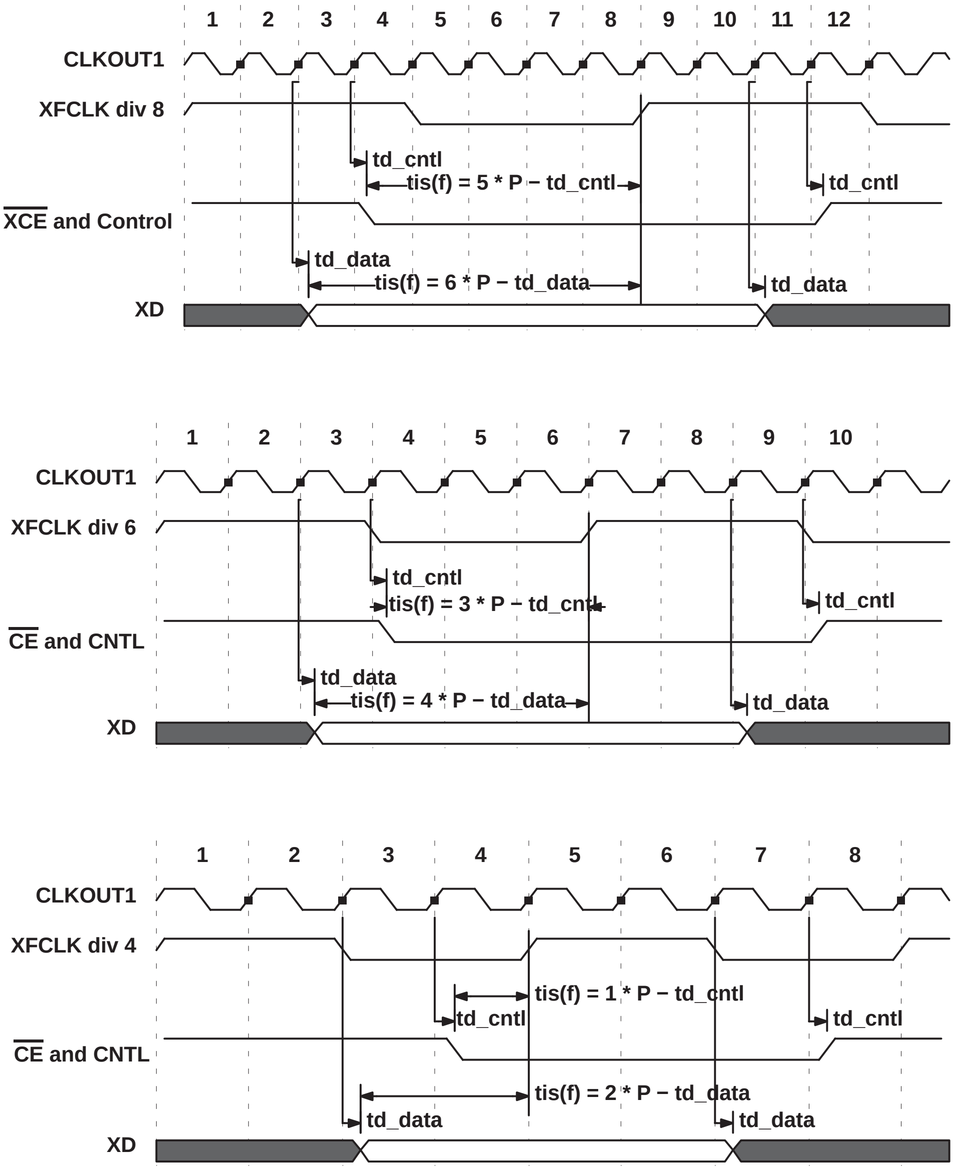
Revision(s) Affected: 1.0

Details: In XBUS Synchronous FIFO mode, output data is skewed by two CPU clocks, while output control signals are skewed by three CPU clocks, regardless of the FIFO clock rate (XFRAT). As a result, there may not be enough data and control signal setup time to the next rising edge of XFCLK, where the FIFO is expected to latch the control/data. As shown in the diagrams below, the impact of the control and data skew is less significant for the slower clock speeds.

For example, at XFCLK divided by 8, the control signal skew of three cycles leaves five CPU cycles of setup time before the next rising edge of XFCLK. For XFCLK divided by 6, the three CPU cycles of skew on the control bus leaves three CPU cycles of setup time to the next rising XFCLK edge. For XFCLK divided by 4, the three CPU cycles of skew on the control bus leave one CPU cycle of setup to the next rising XFCLK edge. Although not shown, the situation is the same for XFCLK divided by 2.

On reads, the FIFO control signal outputs are skewed exactly as in the write case. However, the XBUS latches data correctly on the rising edge of XFCLK for all XFCLK divide ratios. (Internal reference number C630489)

XBUS: FIFO Data and Control Signals Skewed from XFCLK (Continued)



Workaround: Select a slower FIFO clock rate (XFRAT) such that the FIFO setup time is met.

Advisory 1.0.5 *XBUS: Master Synchronous Host-Port Mode Transfers Corrupted*

Revision(s) Affected: 1.0

Details: During an XBUS Sync Host Master read/write from/to an external device, data transfer is skipped or overwritten when any of the following conditions are true, regardless if the XBUS internal bus arbiter is enabled or not (XARB):

- The XBUS output XWAIT = 0 and the XBUS input XRDY = 1, indicating that both the XBUS and the external devices are not ready. (Internal reference number C630391)
- Master bursts are longer than one word. (Internal reference number C630277)

Workaround:

- When interfacing to a PCI bridge, make sure that when the XBUS is not ready (XWAIT = 0), the PCI bridge is ready (XRDY = 0).
- Burst one word at a time in master mode.
- Other workarounds under investigation.

Advisory 1.0.6 *XBUS: Master Synchronous Host-Port Mode Incorrect Address*

Revision(s) Affected: 1.0

Details: In the Expansion Bus Master Synchronous Host-Port Mode, if an external device asserts XRDY low (indicating its ready status) after the START bit field of the XBHC register has been written but before the Expansion Bus asserts the output address strobe XAS, the Expansion Bus incorrectly increments the Expansion Bus External Register (XBEA) and decrements the XFRCT field in the Expansion Bus Host Port Interface Control register (XBHC) before the address phase. This incorrect XBEA and XFRCT modification occurs once every XCLKIN after XRDY is asserted low. In other words, the Expansion Bus, instead of monitoring the XRDY between the address strobe and the last transfer, starts monitoring the XRDY input after the START bit field of the XBHC register has been written. As a result, although the Expansion Bus responds to XRDY correctly during the active transfer period, the address presented during the address phase is incorrect. The problem occurs regardless of the value of XARB (Expansion Bus Arbiter). (Internal reference number C630517)

The DSP keeps monitoring the XRDY input even when it is the backoff state. The DSP that backed off keeps monitoring its XRDY signal, and keeps incorrectly incrementing the XBEA register and decrementing the XFRCT field in the XBHC register when XRDY indicates a ready status.

Workaround: Make sure (with the use of glue logic) the XRDY input signal is deasserted high except during an active transfer initiated by the XBUS. An active transfer refers to the time period between the address phase (XAS asserted) and when the XBLAST signal is asserted

Other workarounds under investigation.

Advisory 1.0.7 *XBUS: Slave Synchronous Host-Port Mode Transfers Do Not Wait for XCS*

Revision(s) Affected: 1.0

Details: When the XBUS is operating in Slave Synchronous Host-Port Mode, the XBUS starts driving the XRDY output after the external host device asserts the address strobe (XAS) input to the XBUS, regardless if the chip select input XCS is asserted. In addition, future accesses to the specific target DSP will not be recognized since the original access (with XAS) puts the XRDY state machine in an invalid state. (Internal reference number C621757)

Workaround: In a system with multiple slaves, the external host needs to generate a different address strobe signal (XAS) to each individual slave. The external host should only assert XAS to a particular slave when it needs to communicate to that slave.

Advisory 1.0.8 *DMA: Split-Mode Transfers Corrupted if Channels 1, 2, or 3 are Stopped*

Revision(s) Affected: 1.0

Details: There is a problem with stopping DMA channel 1, 2, or 3 when operating in split-mode transfers. If the DMA split-mode receive and transmit transfers are not in sync with one another when the channel is stopped, and then the same DMA channel is programmed for a new split-mode transfer, the new transfer will execute correctly but may not terminate completely. This problem does not exist in channel 0. (Internal reference number C621764)

Workaround: Do not stop DMA channels 1, 2, and 3 when they are operating in split mode or manually force the number of elements received and transmitted transfers to be equal. Split mode is most commonly used with the on-chip McBSPs. In typical McBSP applications, the transmit data is two elements ahead of the receive data. Therefore, to stop the serial transfer do the following:

- Reset the McBSP to prevent additional sync events
- Set RSYNC_STAT twice for the DMA channel to force two receive transfers
- Stop the DMA channel

To ensure that the same number of elements are transferred, the source and destination addresses can be checked.

Advisory 1.0.9 *XBUS: Slave Synchronous Host-Port Mode Writes Corrupted in a Multi-DSP System*

Revision(s) Affected: 1.0

Details: In a multi-DSP systems where a C6202 XBUS operating in Slave Synchronous Host-Port Mode is one of the slaves controlled by an expansion bus master, an expansion bus master write to the C6202 XBUS slave is corrupted if all of the following conditions are true:

- The expansion bus master terminates a write transfer to the C6202 XBUS slave by asserting XBLAST.
- Before the C6202 XBUS slave completes the master write transfer internally, the expansion bus master initiates a new transfer to another slave. The new slave responds by asserting XRDY low.
- The XRDY pins of the C6202 XBUS slave and the other slaves are connected together.

The C6202 XBUS slave detects the XRDY signal asserted by the other slave and assumes that more data has been written into the chip. Thus, the C6202 XBUS slave incorrectly increments the internal pipeline count and overwrites the data currently in the pipeline. (Internal reference number C630731)

Workaround: In a multi-DSP system, make sure (by the use of logic) the XRDY pin of the C6202 slave is gated off from the other slave devices.

Advisory 1.0.10 *XBUS: Internal Bus Arbiter Deasserts XHOLDA Before External XHOLD is Dropped*

Revision(s) Affected: 1.0

Details: When XARB = 1, the XBUS output XHOLDA can get deasserted while the XBUS input XHOLD stays asserted if an internal request to move data to the XBUS IO port or to start a master transfer is received.

The following bug takes place when the XBUS is operating in the Slave Synchronous Host-Port Mode with internal arbitration enabled (XARB = 1). The external host requests the XBUS to perform a transfer by asserting the XHOLD. When the DSP grants the bus by asserting the XHOLDA, the host can start the transfer. During the transfer, the DSP receives an internal request to move data to/from the XBUS IO port or an internal request to start a master transfer. After the external host moves the last data, the XBUS will deassert the XHOLDA to service the internal request even though the host keeps its XHOLD = 1. (Internal reference number C630760)

Workaround: Use an external bus arbiter (XARB = 0).

6 Documentation Support

For device-specific datasheets and related documentation, visit the TI web site at: **http://www.ti.com**.

The *TMS320C6202, TMS320C6202B Fixed-Point Digital Signal Processors* data sheet (literature number SPRS104) describes features of the C6202/C6202B device.

The *TMS320C6000 DSP Peripherals Overview Reference Guide* (literature number SPRU190) provides an overview and briefly describes the peripherals available on the TMS320C6000TM DSPs.

TMS320C6000 is a trademark of Texas Instruments.

IMPORTANT NOTICE

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

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

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

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

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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