TMS320C6201 SILICON ERRATA

The following is a list of problems on TMS320C6201 2.1 silicon or any lower revision. TI creates a new document revision when a new silicon bug is discovered. However, TI does NOT update previously edited files. For example, if you have silicon revision 1.0 and the latest silicon revision is 2.1, you should look at the latest silicon errata for 2.1, as it will also contain any problems found in silicon version 1.0.

Silicon revision is identified by a code in the lower left-hand corner of the chip. The code is of the format Cxx-yyww. If xx is 00 then the silicon is revision 1.0. If xx is 11, 20, 21 then the silicon is revision 1.1, 2.0, or 2.1 respectively.

Please also request the latest TMS320C6201 Peripheral Reference Guide and any Errata

Note:

- New items in this document are
  - Problem 2.1.19
  - Problem 2.1.20
- Problems in revision 2.0 silicon not fixed in revision 2.1 have been re-numbered as 2.1.x problems. This creates gaps in the 2.0.x problem numbering sequence.
- All remaining 2.0.x problems are fixed on revision 2.1.
- All 2.1.x problems will be fixed in revision 3.0.

List Of Bugs

REVISION 2.1 SILICON BUGS

Problem 2.1.1 EMIF: CE Space Crossing on Continuous Request Not Allowed

Any continuous request of the EMIF cannot cross CE address space boundaries. This condition can result in bad data read, or writing to the wrong CE. Internal Reference Numbers 2600 & 3421.

WORK-AROUNDS:

CPU Program Fetch: The simplest fix is for all external program to reside within a single CE space. Alternatively, program fetch flow should not occur across CE spaces. This can be accomplished by branching on chip in between executing from one CE to another CE.

DMA: All DMA block transfers without read or write synchronization should have all EMIF addresses within a frame to belong to one CE space. In other words, all read (src) addresses should belong to one CE space and should not cross CE boundaries. The same applies to write (dst) addresses within a frame. Note that the source can be in the same CE space or different CE space as the destination. DMA transfers with read and/or write synchronization together with CE boundaries crossed between frames are not affected by this bug.

CPU Data Access: External CPU data accesses cannot perform continuous requests and thus are not affected by this bug.

Problem 2.1.2 EMIF: SDRAM invalid access

An invalid SDRAM access occurs when each of the following is true:

- Two or more SDRAM devices in different CE spaces
- Each SDRAM device has a page activate
- One active page is in bank 0 and the other in bank 1
- Each CE space with SDRAM is accessed (alternating) without a page miss or refresh occurring (no Deactivate command).

OR

- Two or more SDRAM devices in different CE spaces
- A trickle refresh deactivates both devices
- Before refresh occurs, a request to access one CE space comes in. The refresh will wait until the first requestor has completed.
- If request to second CE space occurs before refresh occurs, then an invalid access takes place, since the controller neglects the fact that this space was deactivated.

Internal Reference Numbers 4139, 0335, and 0871.

WORKAROUND: Avoid use of multiple CE spaces of SDRAM within a single refresh period.

Problem 2.1.3 EMIF: DMA data block corrupted after pause

If a DMA channel is paused just as the last transfer in a block completes. When the same channel is restarted the first transfer in the next block may be corrupted. Internal reference number 0242.

WORKAROUND: Make sure that the transfer count is not near zero when pausing the DMA.

Problem 2.1.4 DMA: RSYNC cleared late for Frame Sync'd transfer

In a frame-synchronized transfer, RSYNC is only cleared after the beginning of last write transfer. It should occur after the start of the first read transfer in the synchronized frame. Internal reference number 0267.

WORKAROUND: Wait until end-of-frame (perhaps using DMAC pins for external status) to issue next frame synchronization.

Problem 2.1.5 McBSP: DXR to XSR copy not generated

If any element size other than 32 bits is written to the DXR of either serial port, then the register is not copied to the XSR.

The following work around is **applicable only for non-split mode DMA** transfers.

WORKAROUND:

(1) For little-endian mode:

Always write 32 bits to the DXR. When using the DMA, it is possible to perform word transfers, but increment or decrement the address by one or two bytes using one of the global index registers. If the serial port is transferring out 16-bit words, which are stored on consecutive half-word boundaries in memory (either internal or external), the DMA would need to be set up such that it performs word writes to DXR (ESIZE = 00b). The global index register used would need an element index of 0x0002 (2 bytes). If an 8-bit data transfer is desired, then element index would need to be 0x0001.

Please note that this workaround assumes that the receive justification, RJUST in the McBSP's SPCR is set for right justification (zero-fill or sign-extended). If left justification is chosen for receive data, the DMA receive src address pointing to DRR should be changed to DRR+3 (which is 0x018C0003 for McBSP0 and 0x01900003 for McBSP1) for byte-size elements and DRR+2 for half-word elements. This ensures packing data on byte or half-word boundaries for receive data.

Example:
Configure the DMA as follows:
(a) For half-word / byte-size accesses with right justification on receive data:

```
ch_A: /* for transmit */
src_address = mem_out; dst_address = DXR;
Element_size = WORD
Address_inc_mode = index
Index_reg_value = 2 /* change this to 1 for byte writes */

ch_B : /* for receive */
src_address = DRR; dst_address = mem_in;
Element_size = HALF /* change this to BYTE for 8-b element size */
Address_inc_mode =  inc_by_ element_size
/* inc_by_index whose value is as specified for ch_A above will also work */
```

(b) For half-word / byte-size accesses with left justification on receive data:
Same as (1)(a) above EXCEPT for:

```
ch_B : /* for receive */
src_address = DRR+3; /* for byte accesses */          OR
            = DRR+2; /* for half-word accesses */
```

(2) For big-endian mode:

Always write 32 bits to the DXR.
(a) For half-word accesses with right justification on receive data:
ch_A: /* for transmit */
src_address = mem_out;
dst_address = DXR+2; /* 0x018C0006 for McBSP0 or 0x01900006 for McBSP1 */
Element_size = WORD
Address_inc_mode = index
Index_reg_value = 2

ch_B : /* for receive */
src_address = DRR+2 /* 0x018C0002 for McBSP0 or 0x01900002 for McBSP1 */
dst_address = mem_in;
Element_size = HALF;
Address_inc_mode = = inc_by_ element_size
/* inc_by_index whose value is as specified for ch_A above will also work */

(b) For half-word writes with left justification on receive data:
Same as (2)(a) above EXCEPT for:
ch_B : /* for receive */
src_address = DRR;

(c) For byte-size writes with right justification on receive data:
ch_A: /* for transmit */
src_address = mem_out;
dst_address = DXR+3; /* 0x018C0007 for McBSP0 or 0x01900007 for McBSP1 */
Element_size = WORD
Address_inc_mode = index
Index_reg_value = 1

ch_B : /* for receive */
src_address = DRR+3 /* 0x018C0003 for McBSP0 or 0x01900003 for McBSP1 */
dst_address = mem_in;
Element_size = BYTE;
Address_inc_mode = = inc_by_ element_size
/* inc_by_index whose value is as specified for ch_A above will also work */

(d) For byte-size writes with left justification on receive data:
Same as (2)(c) above EXCEPT for:
ch_B : /* for receive */
src_address = DRR;

Problem 2.1.6 DMA Split-Mode End-of-frame Indexing

If a DMA channel is configured to do a multi-frame split-mode transfer, both the Receive and Transmit transfers will generate an end-of-frame condition. This will cause the FRAME COND bit to be set multiple times per frame in the Secondary Control Register of the channel.

Also, if DST_DIR = Index (11b), the end-of-frame condition by both the Receive and Transmit Transfers will cause a destination address to be incremented using Frame Index, rather than Element Index. The problem is that BOTH the last element in a frame for the Receive Read Transfer (split source to destination) AND the last element in a frame for the Transmit Write Transfer (source to split destination) will cause the destination address to be indexed using the frame index. This should only occur for the last element in a frame for the Receive Read Transfer. SRC_DIR = Index functions properly. Internal reference number 0559.

WORKAROUND: If the FRAME COND bit is used to generate an interrupt to the CPU and/or the frame index and the element index on the destination address are not the same for a split-mode transfer, use two DMA channels.

Problem 2.1.7 DMA Channel 0 Multi-frame Split-Mode Incompletion

If DMA Channel 0 is configured to perform a multi-frame split-mode transfer, it is possible for the last element of the last frame of the Receive Read to not be transferred. After the last element of the last frame of the Transmit Write Transfer, the element count is reloaded into the Channel 0 Transfer Counter Register, which may allow for the Transmit Read Transfer to be initiated. If the read synchronization and write synchronization are far enough apart in CPU cycles, then it is possible for the DMA to hang (due to the Transmit Read) before the Receive Write gets its sync event and completes the transmission. Internal reference number 0558.

WORKAROUND: If a multi-frame split-mode transfer is required, use DMA channel 1, 2, or 3.

Problem 2.1.8 Timer clock output not driven for external clock

When FUNC = 1 (TOUT is a timer pin), if CLKSRC = 0 (external clock source) the TOUT pin is not driven with TSTAT. The timer still functions correctly, but the output is not seen externally. Internal reference number 0568.

WORKAROUND: None. Timer functions correctly.

Problem 2.1.9 Power Down pin PD not set high for Power Down 2 mode

The power down pin, PD, only goes high (active) in power down mode 3, not in power down mode 2. Internal reference number 0537.

WORKAROUND: None. Power down modes function correctly.

Problem 2.1.10 EMIF: RBTR8 bit not functional

If RBTR8=1, a requester with continuous requests will not relinquish control of the EMIF even to a higher priority requester. Internal reference number 0432.

WORKAROUND: Leave RBTR8 set to the default of 0.

Problem 2.1.11 McBSP: Incorrect µLaw companding value

The C6201 McBSP u-Law/A-Law companding hardware produces an incorrectly expanded u-Law value. McBSP receives u-Law value 0111 1111, representing a mid-scale analog value. Expanded 16-bit data is 1000 0000 0000 0000, representing a most negative value. Expected value is 0000 0000 0000 0000. McBSP expands u-Law 1111 1111 (also mid-scale value) correctly. u-Law works correctly for all encoded values, except for 0x7f. Internal Reference Number 0651.

Problem 2.1.12 Cache: False cache hit – Extremely rare

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

- A and B are separated by a multiple of 64k in memory (i.e. they will occupy the same cache frame)
- B is currently located in cache

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). Internal Reference Number 4372.

WORKAROUND: The program should be re-linked to force A and B to not be a multiple of 64k apart.

Problem 2.1.13 EMIF: HOLD feature improvement on revision 3

This is documented as a difference between the 320C6201 revision 2.x (and earlier) and revision 3.0 (and later).

The HOLD feature of the ‘C6201 currently will not respond to a HOLD request if the NOHOLD bit is set at the time of the HOLD request, but is then cleared while the HOLD request is pending. In other words, for a HOLD request to be recognized, a high to low transition must occur on the HOLD input while the NOHOLD bit is not set. Future revisions of the device will operate as described below.

If NOHOLD is set and a HOLD request comes in, the C6x will ignore the HOLD request. If while the HOLD request is still asserted the NOHOLD bit is then de-asserted, the HOLD will be acknowledged as expected. Internal reference number 0101.

WORKAROUND: In order to recognize a pending HOLD request when the state of the NOHOLD bit is changed from 1 to 0, a pulse must be generated on the input HOLD line. This can be done by logically OR-ing a normally low general purpose output (DMAC can be used) with the HOLD request signal from the requestor, and creating a high pulse on the general purpose output pin.

Problem 2.1.14 EMIF: HOLD request causes problems with SDRAM Refresh

If the HOLD interface is used in a system with SDRAM, there are some situations that are likely to occur.

If the NOHOLD bit is not set and an external requestor attempts to gain control of the bus via the HOLD signal of the EMIF at the exact same time as the EMIF is issuing a SDRAM Refresh command, the HOLD request is never recognized. Even if the NOHOLD bit is set in the EMIF Global Control Register, SDRAM Refreshes are still disabled as long as the HOLD request is pending. A single Refresh after receiving the HOLD request is issued, but no additional Refreshes are issued until the HOLD request is removed. The C6x still owns the bus since the NOHOLD bit is set.

In addition, if an SDRAM burst is started just prior to a HOLD request, it is possible that the request will not be recognized until a refresh occurs. This will potentially allow for the HOLD request to be ignored for several micro-seconds. Internal reference number 0757 and 0777.

WORKAROUND: Do not allow a requestor to activate the HOLD line without acknowledging it for longer than the SDRAM refresh period. A workaround can be accomplished by keeping the NOHOLD bit set and software poll the HOLD bit of the EMIF Global Control Register. Software polling of the HOLD bit in the EMIF Global Control Register will indicate when a HOLD request has been received (this can be done in the SD_INT service routine or Timer interrupt service routine).

Upon detecting a HOLD request, SDRAM refreshes are disabled, NOHOLD bit is cleared, and a pulse is generated on the input HOLD signal (can use DMACx as a general purpose output pin in combination with the requestors HOLD signal). Then NOHOLD can be set and SDRAM refreshes enabled in anticipation of the next HOLD request.

Problem 2.1.15 DMA Priority Bit Ignored by PBUS

The CPU always has priority over the DMA when accessing peripherals. The DMA PRI bit is ignored and treated as “0”. Internal reference number 0540.

WORKAROUND: Leave sufficient gaps in CPU accesses to the PBUS to allow the DMA time to gain adequate access.

Problem 2.1.16 DMA Split-Mode Receive Transfer Incomplete After Pause

If the DMA is performing a split-mode transfer and the channel is paused after all Transmit Reads in a frame are completed but before the Receive Reads are completed, then the Receive Transfer will not complete after the channel is restarted. Internal reference number 0606.

WORKAROUND: Do not pause a split-mode transfer at the end of a frame unless the frame has completed.

Problem 2.1.17 DMA Multi-Frame Transfer Data Lost During Stop

If the DMA is stopped while performing an unsynchronized, multi-frame transfer, all of the read data may not be written. The data will be written when the channel is restarted. This case will only occur when the frame size (element count) is 10 or less and data elements from multiple frames are in the FIFO when it is stopped. Internal reference number 0789.

WORKAROUND: Keep frame size > 10, synchronize the frame (FS = 1), or do not stop the transfer.

Problem 2.1.18 Bootload: HPI boot feature improvement on revision 3

This is documented as a difference between the TMX320C6201 revision 2.x (and earlier) and revision 3.0 (and later).

Currently during HPI boot, all accesses to program memory are treated as writes by the PMEMC. This means that the host may not read the internal program memory space, as doing so will overwrite the memory space, usually with all zeros. The PMEMC will be changed to differentiate between reads and writes to program memory during boot. Internal reference number 0604.

Problem 2.1.19 PMEMC: Branch from external to internal

The program flow is corrupted after branching from external memory to internal program memory when the following are true:

- CPU is executing from external memory
- A CPU stall occurs that holds the CPU until all pending program fetches complete. CPU stalls may be caused by:
    - External data access
    - Multi-cycle NOPs
    - Prolonged data memory bank conflict with DMA
    - Multiple accesses to on-chip peripherals (not likely to cause this problem)
- A branch to internal program memory is taken before a new fetch packet is requested (i.e. during the same fetch packet that is executed when the CPU stalls.

The CPU will branch correctly to the internal memory location and correctly execute the code located there. When the branch is executed to return to external memory, the CPU will not complete the branch properly and the program will crash. Internal reference number 0958

WORKAROUND: There are several workaround options, depending on the situation that causes the failure. One or more of the following should be used to circumvent the problem:

- If the problem arises during an interrupt, move IST to external memory (same CE as code).
- If the problem occurs after a branch, delay the branch instruction with single-cycle NOPs or extend the delay slots to span multiple fetch packets (i.e. follow the branch instruction with parallel NOPs).
- If an external data access is causing the CPU stall, place data in internal data memory.

- If a multi-cycle NOP is causing the stall, change to multiple single-cycle NOPs.
- If stall is due to the CPU being starved, change the DMA priority to be lower than that of the CPU.

Problem 2.1.20 DMA: FIFO arbitration feature improvement on revision 3

This is documented as a difference between the 320C6201 revision 2.x (and earlier) and revision 3.0 (and later). Currently during a burst transfer by a low-priority channel, in which the DMA FIFO is being used, if an unsynchronized (or frame synchronized) high-priority channel begins it may not take possession of the FIFO for its transfer. This will result in an inability to burst to/from high-speed memories at its maximum rate. The FIFO arbitration logic has been modified to guarantee that the high-priority channel will always obtain the FIFO. This ensures that all burst transfers are performed at the maximum transfer rate possible. Internal reference number 0901.

REVISION 2.0 SILICON BUGS

Problem 2.0.1 Program Fetch: Cache Modes Not Functional

WORK-AROUND: Use internal program memory in mapped mode.

Problem 2.0.2 Bootload: Boot from 16-bit and 32-bit Asynchronous ROMs Not Functional

16-bit wide ROM mode and 32-bit wide asynchronous mode work in run time without bugs. The problem is only in boot. . Internal Reference Number 3088.

WORK-AROUND: Place all code in the lowest byte of the boot ROM.

Problem 2.0.3 DMA Channel 0 Split Mode Combined with Auto-initialization Performs Improper Re-Initialization

The source address (transmit read address) is reset too early when both split mode and auto-initialization are enabled. The bug exists on DMA channel 0 only. Internal Reference Number 3481.

WORK-AROUND: Substitute one of the other channels for channel 0 when this configuration is desired.

Problem 2.0.4 DMA/Program Fetch: Cannot DMA into Program Memory when Running Program From External

Performing a DMA transfer into program memory while running from off-chip can cause invalid program data to read by the CPU. Internal Reference Number 2978.

WORK-AROUND: DMA into program memory only when running from internal program memory.

Problem 2.0.5 Data Access: Parallel Read and Write Accesses to Same EMIF or Internal Peripheral Bus Location Sequenced Wrong

This bug occurs under the following conditions:

- A load and store are in the same execute packet. And Either
  - The addresses both point to off-chip memory through the EMIF, and the load has a destination register in side A (thus the store would have a source register in side B). Or
  - The addresses both point to the peripheral bus, and the load has a destination register in side B (thus the store would have a source register in side A).

When these conditions occur, the store occurs first rather than the load. In general, this will only cause an error if both the load and store addresses are the same. This bug DOES NOT occur if both accesses are to internal data memory. Internal Reference Number 3087.

WORK-AROUND: Avoid loading and storing the same address on the same cycle.

Problem 2.0.7 EMIF: Reserved Fields Have Incorrect Values

Fields in Bits 15:14 of EMIF CE Space control registers are writeable. They should be read only and have a 0 value. Bits 5:4 of EMIF SDRAM control register are 11b rather than 0. Internal Reference Number s 3248, 3283.

WORK AROUND: Mask these values if 0’s are expected and to only write 0’s to reserved fields.

Problem 2.0.8 EMIF: SDRAM Refresh/DCAB Not Performed Prior to HOLD Request Being Granted

SDRAM is left in the current state when an external HOLD is granted. SDRAM refresh/DCAB is necessary if an interface to a shared memory external SDRAM controller is desired. Internal Reference Number 3249.

WORK-AROUND: Make sure the external controller performs a refresh/DCAB before performing SDRAM accesses.

Problem 2.0.9 McBSP New Block Interrupt does not occur for Start of Block 0

When end-of-block interrupt is selected ((R/X)INTM=01b), does not occur at end of frame (i.e. before block 0). Internal reference number 4357.

WORK-AROUND: This interrupt is used when on-the-fly channel selection/enabling is being performed. A static channel selection/enabling avoids this.

Problem 2.0.11 DMA/Internal Data Memory: First load data corrupted when DMA in high priority

In the case of a single load from A side or B side followed by two loads in parallel from both sides, and in concert with a DMA high priority access to the same bank as the parallel load, the DMEMC provides corrupt data for that first load. Internal Reference Number 3858.

```
Example:     LDW    .D1    *A3, A4 ; A4 gets corrupt data due to the bug

             LDW    .D2    *B3, B4

          || LDW    .D1    *A6, A7
```

WORK-AROUND: Avoid high priority DMA transfers to/from internal data memory during these conditions.

Problem 2.0.12 McBSP: FRST Improved in 2.1 over 2.0

The following enhancements were made in 2.1.

When /FRST transitions to a 1, the first frame sync is generated after 8 CLKG clocks. The 2.0 implementation was such that the first frame sync was generated after FPER+1 number of CLKG clocks.

/FRST=1 is valid only when /GRST=1. In other words the user has to set /FRST=1 only after /GRST=1. If not, write to /FRST=1 is ignored or rather a zero is forced on /FRST by the logic.

During normal operation, when /FRST=1 and /GRST=1, and now the user puts the sample rate generator in reset (/GRST=0) without first clearing the /FRST bit to zero, then the logic will force a zero to /FRST bit before shutting down the sample rate generator.

Problem 2.0.13 McBSP: /XEMPTY stays low when DXR Written Late

/XEMPTY goes low and stays low when DXR was written on either the last bit or next to last bit of the previous word being transferred to DX. Internal Reference Number 3383.

Problem 2.0.14 EMIF: Multiple SDRAM CE Spaces: Invalid access after refresh

This bug exists only in those systems that have SDRAMs in more than one CE space. When there are two SDRAM accesses performed to two CE spaces, followed by a refresh, the pages in all CE spaces with SDRAM are de-activated. The first CE space to be accessed after the refresh gets activated correctly. The bug is that if the second CE space is accessed on the same page as before the refresh, it will not get activated before the read or write is attempted. Internal Reference Number 3952.

WORKAROUND: Avoid use of multiple CE spaces of SDRAM within a single refresh period.

Problem 2.0.18 DMA/Internal Data Memory: conflict data corruption

This bug occurs when the CPU has high priority and is accessing a bank with word access (load or store) followed by similar (load or store) halfword access, and the DMA is also accessing the same bank simultaneously with word accesses:

```
Example:    LDW   .D1    *A3, A4
            LDH   .D2    *A3, A5; A DMA to the bank containing never completes
                                ; but the DMA continues as if it did
```

The data transfer done by the DMA is corrupted in halfwords (or rather not updated) when the DMA transfer is complete. Internal Reference Number 4195.

WORKAROUND: When DMAing to/from internal memory with DMA in low priority, use half-word or byte element size transfers. Alternatively, avoid the above code sequence during DMA transfers.

Problem 2.0.19 EMIF: Data Setup Times

The data setup time for the external memory interface is listed in the February 21, 1998 Advanced Information TMSX320C6201 Data Sheet as 2 ns, 3ns, and 2ns for Full Rate SBSRAM, ½ Rate SBSRAM, and SDRAM respectively. In revision 2.0 of silicon, these values are to 4.8, 6.0, and 6.4ns respectively, from worst-case simulation data (low voltage, high temperature, worst case process conditions.)

WORKAROUND: In room temperature operation we have not seen these setup times affect operation except in the case of SDRAM where it may be limited to 80-95 MHz.

Problem 2.0.24 EMIF Extremely Rare Cases Cause an Improper Refresh Cycle to Occur.

If a trickle refresh is waiting for the EMIF, and the refresh timer counts down and makes the refresh urgent JUST AS the EMIF grants the request, then CE is held low for only 1/2 SDCLK cycle during the deactivate command before the refresh. This will result in an invalid deactivate command. Since the SDRAM did not deactivate the open page, the next activate command following the refresh will not be executed by the SDRAM. This will cause any subsequent accesses to go to the non-deactivated page. This will cause corrupt data read and writes if the page to be opened after the refresh was not the same page that was open before the refresh. Internal Reference Number 3453.

WORKAROUND: Increase the refresh period.

REVISION 1.0 BUGS

Problem 1.0.1 un-requested SBSRAM write generated

This problem only occurs after a block of SBSRAM reads is immediately followed by a write to on-chip data memory. This problem is limited to EMIF requests generated by the CPU and it does not affect requests originated by the DMA. EMIF requests to other types of memory than SBSRAM are also not affected.

WORK-AROUND: Insert a non-parallel NOP after any block read from SBSRAM that is immediately followed by a write to internal data memory.

Problem 1.0.2 Incorrect switch from SBSRAM and ASRAM cycles

Some data may be lost during switch from SBSRAM to ASRAM cycles. This happens during loads followed by stores, stores followed by loads and loads followed by loads. SBSRAM loads or stores can be sequential or non-sequential.

This problem does not affect other combinations of external memory accesses.

WORK-AROUND: insert at least two non-parallel NOPs between SBSRAM loads/stores and ASRAM loads/stores that immediately follow.

Problem 1.0.3 Cache fetching from off-chip followed by on-chip fetches

This problem occurs only if the second to last fetch packet from off-chip is not fully parallel (contains more than one execute packets). This condition can cause the processor to execute a corrupted execute packet with unpredictable results.

This problem does not occur during all external or all internal fetches. On-chip fetches followed by off-chip fetches are also O.K.

WORK-AROUND: Write code so that the second to last fetch packet fetched from off chip, preceding the on-chip fetches, contains more than one execute packets.

Problem 1.0.4 Cache mode not working

The fetch packets are still fetched from external memory when the cache mode is set. This problem does not result in incorrect execution, just slower operation.

Problem 1.0.5 Wrong address when reads cross from CE0 to CE1

When the external CPU read sequence crosses from CE0 to CE1 space, the first CE1 address in incorrect (same as the last CE0 address).

This problem is not caused by DMA accesses or program fetches.

WORK-AROUND: Do not cross the CE0/CE1 boundary during sequential reads or insert a non-parallel NOP at the boundary.

Problem 1.0.6 Write to ASRAM missed following SBSRAM writes

This problem only occurs when a sequence of writes crosses the memory type boundary.

WORK-AROUND: Insert a non-parallel NOP between sequential writes to different memory types.

Problem 1.0.7 Incorrect SDRAM timing

This problem is caused by a skew of CLKOUT2 relative to all of the SDRAM control signals. The SDRAM will not operate properly without a fix involving external components.

WORK-AROUND: Below is description of what is being done on the TI test card to overcome the CLKOUT2 skew. The jumpers will allow the same circuit to still be used with the upcoming revisions of the C6201 that will have this problem corrected.

```
                                          ^^
                              JP2  ====                    _______
                          _________o  o______             |SDRAM  |
 _____________           |                   |            |       |
|             |          |  |\74LVT240       |            |       |
|   C6201     |__clk02___|__| \o_____________|____sdclk___|>      |
|     A       |             | /UX2                        |       |
 -------------              |/o                            -------
    UX1                       |
                        1  2  |
                vcc o---o  o---
                        3  4  |
                gnd o---o  o---
                    JP1 ====
                         ^^
UX1: DSP
UX2: Inverting Buffer (delays: min=1 ns, nom=2.9ns, and max=4.1ns)
JP1: Jumper to enable/disable the buffer;
JP2: Jumper to bypass the buffer;
NOTES: (power to board is off during configuration changes)
  1. To use the C6201 clk02 directly ... JP1 on  1-2, off 3-4; JP2 on.
  2. To use the inverted C6201 clk02 ... JP1 off 1-2,  on 3-4; JP2 off.
```

Problem 1.0.8 Some signals not reaching full VOH

Affected signal terminals - EMU0, EMU1, TDO, INUMx, and IACK.

The above signals will reach the highest voltage of 1.8V at worst case process and fastest clock speed. At more nominal process they will be above 2V.

No other signals are affected besides the ones listed above. INUMx and IACK signals eventually reach VOH since they don’t switch on every cycle.

WORK-AROUND: At slower speed (CLKOUT1 > 10ns) the affected signals will reach full voltage rails.

Problem 1.0.9 Problem starting the Emulator

When bringing up the emulator, the C6201 must be held at reset until the emulator comes up. This condition only occurs the first time that the Emulator is started following power-up.

Problem 1.0.10 Debugger times-out while single-stepping

This problem can occur while the debugger is single-stepping the external memory. After the time-out error message the debugger may still work in some cases. This condition is rare and it only affects off-chip memory access.

WORK-AROUND: do not single-step through the external memory

Problem 1.0.11 Analysis events not recognized while single-stepping

---

Problem 1.0.12 EMU0 and EMU1 terminals need external pullups

Use 4.7kOhm external pull-ups with the EMU0 and EMU1 pins. The emulator will not work correctly without the pull-ups.

The future silicon revisions will have internal pull-ups on those signals.

WORK-AROUND: This problem has been corrected on the revision 1 test board, but customers designing their own board should add pullups.

Problem 1.0.13 On-chip program RAM may become corrupted during scan

This problem only occurs when the emulator is scanning data to/from the C6201 operating at speed less then 80MHz. This problem does not occur when the scan chain is not being accessed, or when the C6201 is operating at speeds faster then 80Mhz.

WORK-AROUND: Run the C6201 at CLKOUT1 frequency of 80MHz or above when debugging with the emulator.

Problem 1.0.14 Data not stored for sequential stores involving CPU sides A and B.

Serial stores involving both sides of the CPU (A and B) to SBSRAM and ASRAM may not occur. Memory still holds the old data.

WORK-AROUND: Insert a NOP or another instruction between the two stores.

Problem 1.0.15 Power-down mode doesn’t work with clock mode x1.

Pd3 (formerly idle mode 3) doesn’t shut off the internal clocks when the clock mode is set to x1. Pd3 still works for x2 and x4 clock modes.

WORK-AROUND: Do not use the x1 clock mode.

Problem 1.0.16 Internal program memory reads are failing when using the debugger.

This is caused by Problem 1.0.21.

Problem 1.0.17 INTA and INUM pins may randomly toggle during Emulation

WORK-AROUND: Do not use INTA/INUM pins, or do not use the emulator.

Problem 1.0.18 SDRAM bank select does not maintain valid bank value

The SDRAM bank select is generated on ED09 and for 1M x n x 2 banks also on EA2.0. EA11 does not maintain a valid bank value during all operations, and this causes part of the memory to be inaccessible.

WORK-AROUND: Use 1 x n x 2 type of SDRAM with the bank select of the SDRAM connected to EA18. This will provide valid bank selection for all SDRAM bus cycles.

Problem 1.0.19 External memory reads fail when preceded by an external write

This is probably a DMEMC problem since it affects all memory types.

WORK-AROUND: Prevent back-to-back store/load by inserting a NOP between the store and the load.

Problem 1.0.20 CPU external program fetches corrupt CPU internal data reads.

The corrupted DMEMC reads can also be triggered by DMA accessing the program memory.

WORK-AROUND: Keep the program internal and prevent the DMA from accessing internal program memory when the CPU is accessing the internal data memory.

Problem 1.0.21 Corrupted code when parallel fetch packets cross from external to internal memory.

This problem occurs when using Memory Map 0.

WORK-AROUND: Don’t use parallel code when crossing from internal to external program memory.

Problem 1.0.22 Missed SDRAM refresh cycles

Some refresh cycles may be missed if refresh is scheduled to occur during SDRAM read cycles in progress.

NOTICE: This Problem made the revision 1 SDRAM interface unreliable. In addition to fixing the revision 1 bugs, we have decided to completely redesign the SDRAM interface adding full burst functionality and other features. Our recommendation for revision 1 customers at this time is to not use the current revision 1 silicon to interface with SDRAM until the revision 2 silicon is released with the improved SDRAM interface.

Problem 1.0.23 PLL needs different PLLFREQ pin settings

There is a mismatch between bit settings on pins PLLFREQ(3-1) and the CLOCKOUT1 frequency ranges as specified in the Data Sheet. The new values are PLLFREQ(3-1)000=25-135MHz, 001=35-160MHz, 010=40-200MHz.

PLLFREQ(3-1) values 011 and 100 are not valid. If the desired frequency falls inside more then one specified frequency ranges, chose the 000 over 001 and choose the 001 over 010.

Problem 1.0.24 Emulator may not always come up

The correct state of EMU0 and EMU1 may not be latched correctly when bringing up the emulator resulting in emulator “timing out”.

This is a very infrequent occurrence that normally doesn’t happen unless some special emulation instructions are executed outside of normal boot procedures.

WORK-ROUND: Do system reset and try again

Problem 1.0.25 IACK pin not tested

The timing of the IACK pin is not tested for revision 1 of silicon.