ANALOG DEVICES

SHARC®
Embedded Processor

Silicon Anomaly List

ADSP-21060/ADSP-21060L

ABOUT ADSP-21060/ADSP-21060L SILICON ANOMALIES

These anomalies represent the currently known differences between revisions of the SHARC ADSP-21060/ADSP-21060L product(s) and the functionality specified in the ADSP-21060/ADSP-21060L data sheet(s) and the Hardware Reference book(s).

SILICON REVISIONS

A silicon revision number with the form "- x.x" is branded on all parts(see the data sheet for information on reading part branding). The silicon revision can also be electronically read by reading the bits 31-28 of the **MODE2** register either via JTAG or DSP code.

The following DSP code can be used to read the register:

```
<UREG> = MODE2;
```

Silicon REVISION	MODE2[31:28]
2.1	0110
3.0	0111
3.1	0111

ANOMALY LIST REVISION HISTORY

The following revision history lists the anomaly list revisions and major changes for each anomaly list revision.

Date	Anomaly List Revision	Data Sheet Revision	Additions and Changes
02/07/2008	A	E	Update of Silicon Anomaly List Format; Combine Anomaly items for Silicon Revisions 2.1, 3.0, 3.1.

NR003717A

SUMMARY OF SILICON ANOMALIES

The following table provides a summary of ADSP-21060/ADSP-21060L anomalies and the applicable silicon revision(s) for each anomaly.

No.	ID	Description	2.1	3.0	3.1
1	14000001	SPORT Multichannel TX with internal RCLK	x	.	.
2	14000002	DAG2 BITREV() modify instruction with non-zero modify	x	x	x
3	14000003	Hold Time Cycle Wait State and $\overline{\text{MSx}}$ pins	x	x	x
4	14000004	CJUMP instruction with alternate I4-I7	x	x	x
5	14000005	Sport Enable with External SCLK	x	.	.
6	14000006	Host Bus Grant not returned if ID=0 and $\overline{\text{CS}}$=1	x	x	x
7	14000007	Core external hold waitstate blocking of IOP access of EP	x	x	x
8	14000008	Sport RFS with LFS mode deasserts early	x	x	x
9	14000009	PX register read from external memory	x	.	.
10	14000010	DMA to Broadcast Memory Space	x	.	.
11	14000011	Data format specification for Sport Multichannel mode	x	.	.
12	14000012	Sport Transmit Underflow Status is not available for Multichannel mode	x	.	.
13	14000013	Host write failure in a multiprocessing system	x	.	.
14	14000014	Branch with ELSE instruction preceded by a held EP access	x	.	.
15	14000015	ACK during PROM Boot	x	.	.
16	14000016	$\overline{\text{BRx}}$ incorrectly asserted during DAG stalls	x	x	x
17	14000017	Glitch Rejection Circuits not applicable to currently shipping ADSP-2106x products	x	x	x
18	14000018	DAG2 register corruption when executing instructions from external program memory	x	.	.
19	14000019	Short word accesses (read or write) fail when following any stalled instruction	x	x	x
20	14000020	Link port transfers at the 2x clock rate may fail	x	x	x
21	14000021	Rn=MANT Fx results will be rounded if RND32 is enabled	x	x	x
22	14000022	Execution of instructions that modify interrupt latch registers may cause incoming interrupts to be ignored	x	.	.
23	14000023	DMA Request max queue depth of 6	x	x	x
24	14000024	Sharing of DMA channels for Sport1, EPB0/1 and LBUF0/1/4/5	x	x	x
25	14000025	Disabling active link port DMA transfers	x	x	x
26	14000026	Sport Transmit Underflow Status is incorrect when Sport generates TFS	x	x	x

Key: x = anomaly exists in revision
. = Not applicable

DETAILED LIST OF SILICON ANOMALIES

The following list details all known silicon anomalies for the ADSP-21060/ADSP-21060L including a description, workaround, and identification of applicable silicon revisions.

1. 14000001 - SPORT Multichannel TX with internal RCLK:

DESCRIPTION:
In Sport multichannel mode if internal RCLK and RFS are selected, then the transmitted data is corrupted (shifted by 1 bit position).

WORKAROUND:
Use external RCLK and RFS. See anomaly 14000005: "Sport Enable with External Sclk".

APPLIES TO REVISION(S):
2.1

2. 14000002 - DAG2 BITREV() modify instruction with non-zero modify:

DESCRIPTION:
The BITREV(Ic,<data24>) modify instruction where Ic is a DAG2 register and <data24> is a 24 bit modify constant behaves as if <data24>=0. Bit reversed indirect addressing mode (using I0 or I8) works properly. The BITREV(Ia,<data32>) modify instruction, where Ia is a DAG register, also works properly.

WORKAROUND:
Replace

```
BITREV(Ic,<data24>);
```

With

```
MODIFY(Ic,<data24>);
BITREV(Ic,0);
```

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

3. 14000003 - Hold Time Cycle Wait State and $\overline{\text{MSx}}$ pins:

DESCRIPTION:
Programmed wait states of 4,5,6 should automatically generate a hold time cycle for any read or write (pg. 5-36 to 5-40 of SHARC Users manual). This means that the address, $\overline{\text{MSx}}$, and data buses keep driving for 1 cycle after the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ goes high. A problem occurs if the $\overline{\text{RD}}$ or $\overline{\text{WR}}$ access is followed by a dead cycle (an internal access) on the external bus. The $\overline{\text{MSx}}$ lines are not held low during the hold cycle and are brought high as if the hold cycle was not programmed. The Address and Data buses are still properly held in this case, only the $\overline{\text{MSx}}$ is prematurely deasserted.

WORKAROUND:
If the hold time is needed, place a dummy external read or write after the access like this:

```
jump (pc,3) (db);/*delayed branch to make next 2 instructions non-interruptable*/
dm(hold_expected_address)=r0;/*this read or write requires a hold time cycle*/
r1=dm(dummy_ext_address); /*dummy ext RD or WR ensures the MSx will obey hold time*/
```

This ensures that a memory cycle will occur after the DM transfer and that the $\overline{\text{MSx}}$ line will be low during the hold time cycle. The dummy access can be a read or a write. This works even if a $\overline{\text{HBR}}$ causes a BTC (Bus Transition Cycle) and breaks up the two external accesses.

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

4. 14000004 - CJUMP instruction with alternate I4-I7:

DESCRIPTION:
If the alternate I0-I3 registers are enabled and the normal I4-I7 register are enabled and a CJUMP instruction is executed, then the CJUMP instruction will incorrectly use the alternate I6 and I7 instead of the normal set for the I6=I7 function of CJUMP. CJUMP is normally used only by the C compiler for function calls and since the C compiler does not split the normal and alternate DAG registers this is not a problem.

WORKAROUND:
When a CJUMP is executed, always have the I0-I7 registers all normal or all alternate.

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

5. 14000005 - Sport Enable with External SCLK:

DESCRIPTION:

When either receive or transmit serial ports are used with an external SCLK, if in the cycle that the sport is enabled the relationship between the external SCLK and the CLKIN edges falls in the range listed below (work around 3), the serial port may become locked such that it will not receive or transmit data as expected. The sport must be disabled to clear this condition.

WORKAROUND:

Work Around 1:
Use an internal RCLK or TCLK.

Work Around 2:
Delay the SCLK rising edge until 3 CLKIN cycles after the sport is enabled. This can be accomplished by gating SCLK with a flag output pin to hold SCLK low or high during the sport enable as in the TX example below. The flag output must be synchronized by a D-type flip-flop clocked by SCLK before it is gated with SCLK.

```
cycle N bit clr ASTAT FLG0; /*AND flag0 with TCLK to hold SCLK low for next 4 cycles*/
cycle N+1 dm(STCTL)=enable;
cycle N+2 nop;
cycle N+3 nop;
cycle N+4 bit set ASTAT FLG0; /*re-enable TCLK in the next cycle*/
```

Work Around 3:
Synchronize CLKIN with RCLK or TCLK so that:

```
If CKRE=1, CLKIN rising to T/RCLK rising = 14+5DT/16 ns max, -8-11DT/16 ns min.
If CKRE=0, CLKIN falling to T/RCLK rising = 14+5DT/16 ns max, -8-11DT/16 ns min.
```

These specifications define a narrow 3ns window where the SCLK should not rise if the sport may be enabled in that cycle. (DT=tCK-25ns).
If an external SCLK must be used, one of the processor's other TX or RX serial ports can create an external synchronized SCLK. If the serial port requiring the external SCLK input is set to CKRE=0 (Clock Rising Edge),then the SCLK can be directly looped back. If CKRE=1, then the looped back SCLK must be inverted and only even values (odd divisors) can be used in the XCLKDIV of the sport generating the SCLK. This will avoid the SCLK rising edge from falling within the window specified above.

Work Around 4:
Software Detect of Lockup and Sport Restart:
If your system can tolerate having the SHARC miss serial data being sent to it and/or tolerate the SHARC sending incorrect data, a software solution can be employed for this issue. The following steps would be used to enable the sport:

1. Enable the sport
2. Check to see if the serial ports are locked. If locked go to step 3, else continue with code.
3. Disable the sports and return to step 1.

If using DMA with the serial ports, the lock condition can be detected by checking the count register for the DMA channel. A lock has occurred if the count register has not decremented. This register should be checked at a time when it is known that at least one word has been transmitted from or received in the sport.
If using the core to read or write the sport buffers, the lock condition can be detected by the buffer status in the control register. A lock has occurred if the status does not change. The status should be checked at a time when it is known that at least one word has been transmitted from or received in the sport.

APPLIES TO REVISION(S):

2.1

6. 14000006 - Host Bus Grant not returned if ID=0 and $\overline{\text{CS}}$=1:

DESCRIPTION:
If $\overline{\text{HBR}}$ is asserted and the processor ID is 0, and the $\overline{\text{CS}}$ pin is deasserted, then, $\overline{\text{HBG}}$ will not be asserted.

WORKAROUND:
Change the ID to 1 and pull up or tie high $\overline{\text{BR2}}$-$\overline{\text{BR6}}$.

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

7. 14000007 - Core external hold waitstate blocking of IOP access of EP:

DESCRIPTION:
If the SYSTAT EBPR (external bus priority) is set to even priority, and the core continuously accesses an external address which is programmed for 4, 5 or 6 wait states (i.e., a hold cycle is enabled), then any IOP access of the external port will be blocked. The potential IOP accesses of the external port are "master mode", "paced master mode" , "handshake mode", and "external handshake mode" DMAs.

WORKAROUND:
If this situation can occur, set EBPR to "core priority" or "I/O processor priority".

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

8. 14000008 - Sport RFS with LFS mode deasserts early:

DESCRIPTION:
An internally generated RFS will deassert 1/2 RCLK cycle early if LAFS=1 (Late Frame Sync).For connection to other SHARCs this is not a problem because only the asserting edge of the frame sync is sampled on the transmit side.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

9. 14000009 - PX register read from external memory:

DESCRIPTION:
An external memory read to the PX register (i.e., PX=PM(<external address>);) can result in corrupted data. Reads to PX of internal memory operate correctly.

WORKAROUND:
To read the full databus bits DATA47-0 from external memory use an external port DMA with no packing enabled. Then read from the internal memory location to the PX register.

APPLIES TO REVISION(S):
2.1

10. 14000010 - DMA to Broadcast Memory Space:

DESCRIPTION:
DMA transfers where the destination is set to Broadcast Memory Space can cause the Master SHARC's external port to hang, specifically the master does not pre-charge ACK. Core driven transfers to Broadcast Memory Space do not cause this problem.

WORKAROUND:
If data needs to be sent to all processors either use the core to perform the transfers or perform separate DMA transfers to each processor.

APPLIES TO REVISION(S):
2.1

11. 14000011 - Data format specification for Sport Multichannel mode:

DESCRIPTION:
For multichannel operation, the companding selection and MSB-fill sections are not independent as indicated in the SHARC Users Manual. The DTYPE bits in the STCTLx and SRCTLx registers select the data format in this mode as follows:

```
If companding is selected on a channel:

DTYPE Data Formatting
x0 Compand using u-law; Right justify, sign-extend into unused MSBs
x1 Compand using A-law; Right justify, sign-extend into unused MSBs

If companding is not selected on a channel:

DTYPE Data Formatting
x0 Right justify, zero-fill into unused MSBs
x1 Right justify, sign-extend into unused MSBs
```

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1

12. 14000012 - Sport Transmit Underflow Status is not available for Multichannel mode:

DESCRIPTION:
The Transmit Underflow Status, TUVF bit in the STCTLx register, does not provide valid information in Sport Multichannel mode. This status bit should not be used in this mode.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1

13. 14000013 - Host write failure in a multiprocessing system:

DESCRIPTION:
Data from an asynchronous host write will be written to the wrong location under the following scenario:

a) A master SHARC writes data into the internal memory or EP (External Port) buffer of a slave SHARC such that the Direct Write FIFO or an EP Buffer (6 deep) fills up and data backs up into the Slave Write FIFO (2 deep) of a slave SHARC.,
b) A host performs an asynchronous write.

Under this scenario, the host data is written to the Direct Write FIFO if the master SHARC had been performing direct writes to memory, or the data will be written to the EP Buffer to which that master SHARC had been writing.

WORKAROUND:
Workaround 1:
Once the host gains bus mastership, it should perform a read of either an IOP location or memory before it performs any writes. The read will force the backed up writes to complete, thus removing the condition that causes this problem.

Workaround 2:
Assure that data never backs up into the Slave Write FIFO through software control.

APPLIES TO REVISION(S):
2.1

14. 14000014 - Branch with ELSE instruction preceded by a held EP access:

DESCRIPTION:
If an instruction type 9, 10 or 11 using the ELSE option is preceded by a held off EP (External Port) access, the held off access will not complete correctly ($\overline{RD}$ or $\overline{WR}$ is not asserted). The following is an example instruction combination which may result in the anomalous behavior:

```
a) R5 = dm(0x4000234); /* held off external port access*/
b) IF EQ jump (pc,5), ELSE R3=R1+R2; /* example instruction type 9,10 or 11 */
```

The external port access will be held off if any of the following conditions are true:

a) The processor is not the bus master or the bus is granted to the host.
b) $\overline{SBTS}$ is asserted.

WORKAROUND:
Workaround 1:
Place a NOP or other instruction with no EP access before the type 9, 10 or 11 instruction using the ELSE option (between line A and B in above example). To avoid the anomalous behavior, the destination of a delayed branch cannot be an instruction of type 9, 10 or 11 if the second slot of the delayed branch contains an EP access. (RTI instructions can only be avoided by disabling interrupts, thus an RTI db(Delayed Branch) with an EP access in the second delay slot can never be guaranteed to not precede an instruction of type 9, 10 or 11 which uses the ELSE.)

Workaround 2:
Do not use Branch with ELSE instruction if core external accesses can occur while the bus is granted to a host or another SHARC.

APPLIES TO REVISION(S):
2.1

15. 14000015 - ACK during PROM Boot:

DESCRIPTION:
In a multiprocessor system which uses PROM boot mode, a situation can occur in which ACK is driven low by a slave. This can cause PROM reads to hang. The situation is as follows:,
a) ID1 access of a slave's MMS (Multiprocessor Memory Space) address range is followed by a $\overline{\text{RESET}}$ cycle with no power cycling.
b) After the $\overline{\text{RESET}}$ is deasserted, ID1 will drive the MMS address of the slave previously accessed for 4 CLKIN cycles. Then ID1 will drive address 0x400000 and the PROM boot DMA will begin.
c) Normally the ACK line is pulled high by ID1 and kept high by the keeper latch in the bus master.
d) When the slave sees its MMS address, it drives ACK low. ACK is kept low by the keeper latch and the subsequent PROM boot DMA read will hang waiting for ACK.

WORKAROUND:
Workaround 1:
Drive ACK high with an tristate driver when $\overline{\text{BMS}}$ is asserted low. Tristate this driver when $\overline{\text{BMS}}$ is high.

Workaround 2:
Apply reset twice with at least 4 CLKIN cycles between them. This allows the PROM boot DMA address to replace the slave MMS address in ID1's address output latch.

APPLIES TO REVISION(S):
2.1

16. 14000016 - $\overline{\text{BRx}}$ incorrectly asserted during DAG stalls:

DESCRIPTION:
As documented on page 4-12 of the ADSP-2106x SHARC User's Manual [page 4-16 of the ADSP- 21065L SHARC User's Manual] certain instruction sequences involving data transfers to and from DAG registers will cause the insertion of a stall for a single processor cycle. An anomalous behavior occurs when such a stall cycle is caused by an indirect memory access to internal memory immediately following a register write in the same DAG (as shown below). During the stall cycle, the processor incorrectly assumes an external memory access and asserts the $\overline{\text{BRx}}$ line associated with its ID. (This occurs regardless of processor ID.) The instruction sequence will function correctly but this can potentially induce a BTC (Bus Transition Cycle), which may reduce bandwidth on the external cluster bus. (Note: the compiler can generate such code.)

```
L2= @buffer; // inst1
r0= DM(M4,I4); // inst2: I4 points to internal memory
```

WORKAROUND:
Insert an instruction that does not employ the stalled DAG (for example, nop;) between the instruction modifying the DAG register (inst1) and the instruction indirectly accessing memory (inst2), as follows:

```
L2= @buffer; // inst1
NOP; // workaround
r0= DM(M4,I4); // inst2
```

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

17. 14000017 - Glitch Rejection Circuits not applicable to currently shipping ADSP-2106x products:

DESCRIPTION:
Section 11.4.1 Glitch Rejection Circuits on page 11-17 of the ADSP-2106x SHARC User's Manual is not applicable to the currently shipping ADSP-2106x products. All critical signal lines such as CLKIN, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{DMAR}}$,, RCLKs, and TCLKs should carefully laid out to eliminate glitches. Section 11.4.2 Link Port Input Filter Circuits on page 11-17 of the ADSP-2106x SHARC User's Manual is not applicable to the currently shipping ADSP-2106x products. Link port data and clock lines should carefully laid out to eliminate glitches.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

18. 14000018 - DAG2 register corruption when executing instructions from external program memory:

DESCRIPTION:
When executing instructions from external memory, instructions that perform indirect memory transfers (DAG indexed) can corrupt DAG2 registers. The predominant failure mode causes a bit transition from (0->1) within registers (I15:8, M15:8). The failure mechanism is statistical in nature, and the likelihood of this corruption occurring is extremely low. Based on both board and production level testing, this failure is extremely difficult to consistently reproduce. This anomalous behavior is attributed to internal noise susceptibility within the DAG2 unit, which is independent of processor clock rate.

The failure has been observed primarily when using C source code. The register M13, which is initialized to 0x0 within the C run-time, is observed to fail by changing to a non-zero value. Again, this only occurs when executing instructions from external 48 bit wide memory.

WORKAROUND:
The only way to avoid this anomalous behavior is to execute exclusively from internal memory. If external execution is required, it is necessary to upgrade to revision 3.x silicon (which has been redesigned and is immune to this failure.)

APPLIES TO REVISION(S):
2.1

19. 14000019 - Short word accesses (read or write) fail when following any stalled instruction:

DESCRIPTION:

Any access (read or write) to short-word memory space can fail if it follows a stalled instruction or causes an instruction stall (see below for examples).
A DMA process cannot cause this anomaly. It is restricted to conditions set up in the core processor. Also a DMA process will not be impacted by this anomaly i.e. the DMA will function correctly even if the anomaly occurs.
Note: The occurrence of the failure will vary with temperature, voltage, and frequency.
An instruction can stall due to the following circumstances:

1) DAG Stall: An instruction that loads a DAG register followed by an instruction that uses the same DAG for a memory access will cause the second instruction to stall.Ex:

```
 L2= 8;
 DM(I0, M0) = R1; // Both L2 and I0 reside in DAG1 causing this instruction to stall
```

Failure can occur if I0 points short word space or if the above instruction sequence is followed by a short word access. See the ADSP-2106x SHARC User's Manual Second Edition Page 4-12, section 4.1.1.

2) PM Memory Data Access (Cache Miss): Any instruction that uses the PM bus to perform a data access will cause an instruction stall the first time it is executed. Ex:

```
 PM(I8,M8) = R1;
```

See the ADSP-2106x SHARC User's Manual Second Edition Page 3-38, section 3.10.

3) Memory Block Conflict: If an instruction requires two accesses to the same memory block, the instruction will stall.Ex:

```
 DM(I0, M0)=R0, PM (I8,M8) = R1; // a stall will occur when both address pointers point to the
same memory block.
```

See the ADSP-2106x SHARC User's Manual Second Edition Page 5-8, section 5.1.5.

4) Wait states for External Memory Accesses: An instruction that contains an external memory access where the waitstate setting for that memory bank is greater than 0 or the access is held off due to the ACK signal or through bus arbitration causes that instruction to stall. See the ADSP-2106x SHARC User's Manual Second Edition Page 5-39, section 5.4.4.

5) Multiple Bus Accesses to IOP Registers in the same IOP Register Group:
An instruction may be stalled due to multiple busses trying to access IOP registers in the same group. For this anomaly, the only applicable case is if an external host or processor accesses the same group of registers at the same time as the core.Ex:

```
 DM(MSGR0) = R1; // Instruction executed while the host or another processor writes to MSGR2.
```

See the ADSP-2106x SHARC User's Manual Second Edition Page E-8, section E.3.3

6) Executing instructions from external memory with wait states: Any instruction being executed from external memory where the waitstate setting for that memory bank is greater than 0 or the access is held off due to the ACK signal causes that instruction to be stalled.

WORKAROUND:

For the cases 1 to 5 above, insert a NOP between the stall-able instruction and the Short-word access or remove the stall condition. There is no work around for case 6. Note: Care must be taken when using the delayed branch (DB) option with jumps, calls, and returns to ensure that failing sequence does not occur. For example ensure that the 2 instructions in the RTI (DB) do not cause an instruction stall if returning to code that includes short word accesses.

APPLIES TO REVISION(S):

2.1, 3.0, 3.1

20. 14000020 - Link port transfers at the 2x clock rate may fail:

DESCRIPTION:
When link ports are configured for 2x clock operation, link data failures may occur when the rising edge of LACK from the receiving link port has a certain phase relationship with the CLKIN of the transmitting link port. This phase alignment results in a shortened first (LCLK) low pulse output (switching characteristic tLCKTWL) from the transmitting link port. Within normal operating voltage and temperature conditions, this first low LCK pulse can become so small that it violates the receiver's minimum pulse width (timing requirement tLCLKRWL) causing bad data to be received on the first nibble. The occurrence of the failure is dependent on temperature, voltage, and speed of the processor.

Failures have been observed on the 3.3v ADSP-21060L and ADSP-21062L products and on the 5.0v ADSP-21060 and ADSP-21062 products. These failures have been observed when operating the link ports in 2x mode at 40 MHz (tCK = 25 ns) clock rate in systems where the link ports connect clusters that do not share the same CLKIN. Failures have not been observed at 33 MHz (tCK = 33 ns).

WORKAROUND:
Use link port transfers at 1x CLK Speed Operation.

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

21. 14000021 - Rn=MANT Fx results will be rounded if RND32 is enabled:

DESCRIPTION:
The instruction Rn=MANT Fx was designed to work independently of the rounding mode, but it does not. For example, consider the following set of instructions:

```
R2=0x45678901;
F1=float R2;
R0=mant F1;
```

If rounding is enabled (RND32) the result in R0 would be R0=8ACF120000, but if rounding is not enabled the result would be R0=8ACF120200.

WORKAROUND:
If the desired result of the MANT instruction is unrounded, but rounding is enabled in the code, the user must disable rounding manually before executing the MANT instruction and then re-enable the instruction after the MANT instruction has been executed. Keep in mind that writes to MODE1 have a 2 cycle effect latency. The workaround implemented for the example presented above would be as follows:

```
Bit CLR MODE1 RND32;
R2=0x45678901;
F1=float R2;
R0=mant F1;
Bit SET MODE1 RND32;
Nop;
```

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

22. 14000022 - Execution of instructions that modify interrupt latch registers may cause incoming interrupts to be ignored:

DESCRIPTION:

When the execute phase of bit manipulation instruction that modifies an IRPTL (Interrupt Latch Register) is extended due to the core being held off, some of the interrupts that are latched during this period in the interrupt latch register can be lost. The core can be held off when fetching the next instruction from external memory, accessing data from external memory, reading from empty buffer, writing to full buffer or IOP register reads that take more than one core clock cycle.
The specific Group IV system register bit manipulation instructions that are affected are as follows:

```
BIT SET IRPTL <data32>; BIT SET IMASKP <data32>;
BIT CLR IRPTL <data32>; BIT CLR IMASKP <data32>;
BIT TGL IRPTL <data32>; BIT TGL IMASKP <data32>;
```

The interrupts that can be missed are IRQx, EMUI, TMZHI, TMZLI, VIRPT, LPxI, SPIRI, SPITI, EPxI. The LPxI, SPIRI and SPITI interrupts are affected only for DMA driven transfer mode.
This failure will occur under any of the following conditions:
1. When the bit manipulation instruction that modifies an interrupt latch register is executed from external memory.
2. When the bit manipulation instruction is executed from internal memory in a delayed branch to a JUMP or CALL to external memory. For example:

```
a)
JUMP/CALL ext_mem_location (db);
BIT CLR IRPTL <data32>;
NOP;
b)
JUMP/CALL ext_mem_location (db);
NOP;
BIT CLR IRPTL <data32>;
```

3. When the bit manipulation instruction is executed from internal memory and it immediately followed by an external memory data access. For example:

```
a)
BIT CLR IRPTL <data32>;
dm(ext_mem) = r0;
b)
BIT CLR IRPTL <data32>;
pm(ext_mem) = r0;
c)
BIT CLR IRPTL <data32>;
r0 = dm(ext_mem);
d)
BIT CLR IRPTL <data32>;
r0 = pm(ext_mem);
```

4. When the bit manipulation instruction is executed from the emulator (running or stepping.)

WORKAROUND:

1. The workaround for condition 1 is to place the bit manipulation operation in internal memory.
2. The workaround for condition 2 is to avoid the bit manipulation instruction from being within the delayed branch of a JUMP or CALL to external memory by placing it before the JUMP or CALL to external memory.
3. The workaround for condition 3 is to place a NOP; instruction directly following the bit manipulation instruction.
4. A compiler workaround for this issue will be available in a tools patch slated for June 2003.

APPLIES TO REVISION(S):

2.1

23. 14000023 - DMA Request max queue depth of 6:

DESCRIPTION:
Documentation note:
The DMA Request Counter can hold a maximum of 6 requests, not 7 as previously documented. Queuing up more than 6 requests will cause unpredictable results.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

24. 14000024 - Sharing of DMA channels for Sport1, EPB0/1 and LBUF0/1/4/5:

DESCRIPTION:
Documentation note:
Sport1 shares DMA channels with LBUF0/1, and EPB0/1 shares DMA channels with LBUF4/5. If simultaneous use of both the primary (Sport or EPB) and the secondary (LBUF) ports is required then the primary port must use DMA for transfers and the secondary port must be used with core based transfers.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

25. 14000025 - Disabling active link port DMA transfers:

DESCRIPTION:
Documentation note:
If aborting a link port DMA transmit operation before all words are transmitted, the LCTL LxTRAN bit must remain set to 1. For example do not abort a link transfer by writing a 0 to the link buffer control bit field, write an 8 so that the buffer is disabled but the LxTRAN bit can remain set. The LxTRAN bit can then be changed in subsequent cycles.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

26. 14000026 - Sport Transmit Underflow Status is incorrect when Sport generates TFS:

DESCRIPTION:
Documentation note:
If the transmitting sport generates TFS using late frame sync mode (ITFS, LAFS = 1 in STCTLx register) and data independent transmit frame sync is not selected (DITFS = 0 in STCTLx register), the transmitter will not underflow. However the Sport Underflow Status bit (TUVF) in the STCTLx register can be set incorrectly under these conditions. This status bit should be ignored in this mode.

WORKAROUND:
None

APPLIES TO REVISION(S):
2.1, 3.0, 3.1

ANALOG DEVICES

www.analog.com