ANALOG DEVICES

SHARC® Embedded Processor

Silicon Anomaly List

ADSP-21065L

ABOUT ADSP-21065L SILICON ANOMALIES

These anomalies represent the currently known differences between revisions of the SHARC ADSP-21065L product(s) and the functionality specified in the ADSP-21065L 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 31-25 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[31:25]
0.0	1100000
0.1	1100000
0.2	1100001
0.3	1100010

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	C	Update of Silicon Anomaly List Format

NR003720A

One Technology Way, P.O.Box 9106, Norwood, MA 02062-9106 U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

SUMMARY OF SILICON ANOMALIES

The following table provides a summary of ADSP-21065L anomalies and the applicable silicon revision(s) for each anomaly.

No.	ID	Description	0.0	0.1	0.2	0.3
1	14000002	DAG2 BITREV() modify instruction with non-zero modify	x	x	x	x
2	14000003	Hold Time Cycle Wait State and $\overline{\text{MSx}}$ pins	x	x	x	x
3	14000004	CJUMP instruction with alternate I4-I7	x	x	x	x
4	14000006	Host Bus Grant not returned if ID=0 and $\overline{\text{CS}}$=1	x	x	x	x
5	14000007	Core external hold waitstate blocking of IOP access of EP	x	x	x	x
6	14000008	Sport RFS with LFS mode deasserts early	x	x	x	x
7	14000015	ACK during PROM Boot	x	x	.	.
8	14000016	$\overline{\text{BRx}}$ incorrectly asserted during DAG stalls	x	x	x	x
9	14000021	Rn=MANT Fx results will be rounded if RND32 is enabled	x	x	x	x
10	14000022	Execution of instructions that modify interrupt latch registers may cause incoming interrupts to be ignored	x	x	x	x
11	14000031	The L_FIRST control bit is ignored for slave mode I2S	x	x	x	.
12	14000032	Early WS causes loss of L/R alignment in I2S slave mode	x	x	x	.
13	14000033	SDRAM access coincident with $\overline{\text{HBR}}$ assertion	x	x	x	.
14	14000034	SDRAM reads to sequential addresses can fail	x	x	x	.
15	14000035	Slave misinterprets EP address as MMS address	x	x	x	.
16	14000036	Wait states during PROM boot	x	.	.	.
17	14000037	Back to back short word accesses require NOP	x	x	x	.
18	14000038	Address hold from $\overline{\text{WR}}$ high	x	.	.	.
19	14000039	Sport I2S slave mode transmit TFS and RFS hold time	x	x	.	.
20	14000040	Timer interrupt not generated above 50MHz	x	x	.	.
21	14000041	BMAX-BTC failure in dual processor system with concurrent SDRAM accesses	x	x	x	x
22	14000042	Flushing EPx channel FIFO incorrectly clears the other EPx channel's internal DMAR counter	x	x	x	x
23	14000043	Glitch Rejection Circuits not applicable to currently shipping ADSP-21065L products	x	x	x	x
24	14000044	Enabling EP DMA (only Int. to Ext. with no packing) with $\overline{\text{HBG}}$ low corrupts data	x	x	x	x
25	14000045	DMA Read from BMS space ignores specified wait states.	x	x	x	.
26	14000046	DMA Write to BMS space ignores specified wait states.	x	x	x	x
27	14000047	Weak Keeper Latches exist on Address23-0 & Data31-0 pins	x	x	x	x
28	14000048	Erroneous SPORT receive interrupts when DMA chaining is enabled	x	x	x	x
29	14000049	Sport Tx data corruption in TDM mode when sport with DMAs is disabled and re-enabled	x	x	x	x
30	14000050	SPORT in TDM mode when enabled at edge of the frame sync causes data corruption	x	x	x	x
31	14000051	Sport Tx/Rx data corruption in I2S mode and DSP serial mode when sport with DMAs is disabled and re-enabled later	x	x	x	x
32	14000052	Word shift in Tx data when using SPORT in I2S slave mode	x	x	x	x
33	14000053	Reset state of EIEP0 for PROM Boot Mode	x	x	x	x
34	14000054	SWPD bit added to SYSTAT	x	x	x	x
35	14000055	$\overline{\text{DMARx}}$/$\overline{\text{DMAGx}}$ DMA channel assignments	x	x	x	x
36	14000056	For PROM booting EIEPx bit A23 must be set	x	x	x	x
37	14000057	Datasheet timing specification changes for revision 0.3	.	.	.	x
38	14000058	Updated VisualDSP 4.1 PROM kernel contains many changes from VisualDSP 4.0/4.01	x	x	x	x
39	14000059	Changes in the notes given in ADSP-21065L data sheet (rev B)	x	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-21065L including a description, workaround, and identification of applicable silicon revisions.

1. 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):
0.0, 0.1, 0.2, 0.3

2. 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):
0.0, 0.1, 0.2, 0.3

3. 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):
0.0, 0.1, 0.2, 0.3

4. 14000006 - Host Bus Grant not returned if ID=0 and $\overline{CS}$=1:

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

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

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

5. 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):
0.0, 0.1, 0.2, 0.3

6. 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):
0.0, 0.1, 0.2, 0.3

7. 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):
0.0, 0.1

8. 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):
0.0, 0.1, 0.2, 0.3

9. 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):
0.0, 0.1, 0.2, 0.3

10. 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):
0.0, 0.1, 0.2, 0.3

11. 14000031 - The L_FIRST control bit is ignored for slave mode I2S:

DESCRIPTION:
The L_FIRST control bit, used for selecting transmit and receive order, is only available for master mode I2S not slave mode. Because of this, in slave mode, there is no way to select if the first transmitted or received word at startup will align to the left or right I2S channel.

WORKAROUND:
Attach the WS pin (frame sync) to a FLAG input pin and in software wait until the detection of a transition to the desired left or right channel. Then enable the serial port.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

12. 14000032 - Early WS causes loss of L/R alignment in I2S slave mode:

DESCRIPTION:
The serial port in I2S slave mode will ignore a WS edge applied before the data word is fully transmitted or received. For interfacing with codecs or converters this is not a problem. For interfacing with SPDIF transceivers this may result in a loss of left/right alignment of the data samples. The revision 0.3 silicon will ignore two WS transitions in these situations so that an even number of words is transmitted or received and left/right alignment is maintained.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

13. 14000033 - SDRAM access coincident with $\overline{\text{HBR}}$ assertion:

DESCRIPTION:
If an SDRAM access is coincident with the transition of $\overline{\text{HBR}}$ low, and the ADSP-21065L does not have the bus locked, then the processor may enter an illegal state. Within this illegal state SDRAM accesses can occur even though $\overline{\text{HBG}}$ is asserted. SDRAM accesses initiated after $\overline{\text{HBR}}$ is low do not cause problems.

WORKAROUND:
Workaround 1:
Before the core accesses SDRAM or starts a DMA to SDRAM, it must signal thehost (via a FLAG output pin) to not assert $\overline{\text{HBR}}$ low if it is not already asserted. This work around gives the processor priority over the host. For a shared EP (External Port) multiprocessor system OR the FLAG outputs of both.

Workaround 2:
Before the core accesses SDRAM or starts a DMA to SDRAM, it must lock the bus by setting the BUSLK bit in the MODE2 register. After the core accesses or DMA is complete the bus lock must be cleared allow the host on the bus. This work around also gives the processor priority over the host. A drawback is that in a multiprocessor system one of the processors is locked off the bus for the duration of the bus lock.

Workaround 3:
This can be used if only the core accesses SDRAM, (no EP DMAs to SDRAM are done). The host asserts an IRQx pin N CLKIN cycles before asserting the $\overline{\text{HBR}}$ pin. Within the associated interrupt routine the core executes enough NOP instructions (or any non SDRAM access instructions) to ensure that no SDRAM access occurs when $\overline{\text{HBR}}$ transitions low. For a shared EP multiprocessor system the interrupt request must be made to both processors in parallel.

Workaround 4:
Use this work around to allow DMA and core access of SDRAM. The host asserts an IRQx pin N CLKIN cycles before asserting the $\overline{\text{HBR}}$ pin. Within the associated interrupt the core sets the external bus priority control bits in SYSCON (EBPR) to core priority (if not already set). Then the core executes enough consecutive reads of a non-SDRAM external location (dummy reads) to ensure that no SDRAM access occurs when $\overline{\text{HBR}}$ transitions low. For a shared EP multiprocessor system the interrupt request should be made to both processors in parallel. In the multiprocessor situation the interrupt routine can return early if that processor is not the bus master. One drawback to work around #4 is that the core will hang until the host releases the bus again.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

14. 14000034 - SDRAM reads to sequential addresses can fail:

DESCRIPTION:
The following sequence can cause the last read in the sequence to fail.
a) SDRAM read
b) sequential address SDRAM read
c) nop or any non SDRAM instruction which takes 1 core clk
d) on page SDRAM read
e) sequential address SDRAM read (fails, gets same data as in step d)

WORKAROUND:
Set the WAIT register EBxWS bits which correspond to the SDRAM bank to 4 and set the corresponding EbxWM bits to 2 (both internal and external acknowledge). This will insert one hold time cycle between SDRAM accesses and eliminate the problem (the wait state count does not affect SDRAM). The hold time cycle will affect SDRAM throughput. Writes will require one extra SDCLK cycle for all access types. Reads will always be non-burst and will take one extra SDCLK cycle. For example an on page read will take CL ($\overline{\text{CAS}}$ Latency)+ 3 SDCLK cycles, even if sequential. To avoid the write overhead EBxWS can be cleared before writes and set back to 4 before reads. Caution: if you are a using wait state of 4, make sure that you access SDRAM only after the powerup sequence is over, other wise system will hang. (If EBxWS is zero you can have an instruction access SDRAM during powerup because the controller will stall until power up is over).

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

15. 14000035 - Slave misinterprets EP address as MMS address:

DESCRIPTION:
In a shared EP multiprocessor system with 2 ADSP-21065Ls that can access external memory via $\overline{\text{MS3-1}}$ or $\overline{\text{BMS}}$, if the bus master accesses external memory, the slave may interpret it as an access to itself. This prevents access of the first 128K locations in external memory bank 1,2,3 and $\overline{\text{BMS}}$. External memory bank 0 and any memory bank mapped to SDRAM are not affected.

WORKAROUND:
Workaround 1:
This will work if there are no MMS accesses between the ADSP-21065Ls, i.e., they only share external memory. It also reduces the maximum memory bank size to 8M words. Disconnect A23 between the two processors and connect BMSTR(ID1) to A23(ID2) and BMSTR(ID2) to A23(ID1).

Workaround 2:
This will allow MMS accesses and shared memory but will limit the maximum bank size to 8M words. Disconnect the A23 pin between the two processors and gate together the ($\overline{\text{MS3-1}}$ and $\overline{\text{BMS}}$) memory selects that are used as shown below. Do not include the memory select mapped to SDRAM in this gating.

A23(ID1) = $\overline{\text{BMS}}$ + $\overline{\text{MS1}}$ + $\overline{\text{MS2}}$ + $\overline{\text{MS3}}$ + A23 (ID2), Tristated if BMSTR(ID1) is high.

A23(ID2) = $\overline{\text{BMS}}$ + $\overline{\text{MS1}}$ + $\overline{\text{MS2}}$ + $\overline{\text{MS3}}$ + A23 (ID1), Tristated if BMSTR (ID2) is high.

Contact DSP applications support for an example PALASM file for implementing this workaround on a PAL16V8.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

16. 14000036 - Wait states during PROM boot:

DESCRIPTION:
For PROM boot mode an external wait state circuit is required for slow PROMs. This is because the ADSP-21065L revision 0.0 initializes the WAIT register to 0x200D6B5A at reset. WAIT bits (21,20) correspond to the PROM boot mode wait state mode. These bits are set to 00 which corresponds to a wait state mode that ignores internally counted wait states and only depends on the ACK pin.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0

17. 14000037 - Back to back short word accesses require NOP:

DESCRIPTION:
Back to back accesses of short word memory space can result in incorrect data.

WORKAROUND:
Avoid back to back accesses from short word memory space by insertion of a NOP or another instruction without a short word memory space access.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

18. 14000038 - Address hold from $\overline{WR}$ high:

DESCRIPTION:
For revision 0.0 the switching characteristic tDWHA is -1.5ns instead of the desired 0ns. tDWHA is 0ns for revision 0.1 and later.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0

19. 14000039 - Sport I2S slave mode transmit TFS and RFS hold time:

DESCRIPTION:
For serial port I2S slave mode the TSF and RFS inputs normally should have a setup and hold time relative to the rising edge of TCK. Revision 0.0 and 0.1 silicon has a minimum hold time of 0ns from the falling edge of TCK.

WORKAROUND:
If the minimum hold time cannot be met add a transparent latch to TFS or RFS with TCK controlling an active low latch enable (transparent when TCK is low).

APPLIES TO REVISION(S):
0.0, 0.1

20. 14000040 - Timer interrupt not generated above 50MHz:

DESCRIPTION:
Above a 50MHz execution rate (CLKIN>25MHz) both timer 0 and timer 1 may fail to generate an interrupt. This applies to both the width count and the PWM event modes.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1

21. 14000041 - BMAX-BTC failure in dual processor system with concurrent SDRAM accesses:

DESCRIPTION:
A bus mastership timeout via BCNT register expiration can fail to force a Bus Transition Cycle (BTC) between the bus master and bus slave for 60/66 Mhz SDRAM transfers. This happens under the following condition:

• Dual ADSP-21065L multiprocessor system.
• Both processors require either executing code from SDRAM concurrently, performing core block transfers concurrently, or performing block DMA transfers concurrently.
• Both SHARC processors are arbitrating for the external bus at the same time in order to execute instructions, perform core transfers or DMA transfers.
• BMAX register is set to a value other than zero to allow bus mastership timeouts.

For SDRAM accesses, the BCNT counter expiry logic (initialized by programming the BMAX register) can fail to de-assert the $\overline{BRx}$ pin for the required one full CLKIN cycle (or 2x the core clock rate). This results in continued bus-mastership, causing both processors to drive $\overline{BRx}$ signals in a 'bus-lock' state. The current master SHARC will then continue to use the bus until a valid BCNT counter expiration and BTC occurs. The same problem may or may not happen in later BCNT expirations depending on when the BCNT counter expires relative to the CLKIN cycle.This is not applicable for non-SDRAM accesses because non-SDRAM external memory accesses occur with respect to CLKIN. The $\overline{BRx}$ signal in this case is guaranteed to be de-asserted at the CLKIN cycle boundary and hence will be driven out of the Bus Request Pin external to the processor. Since SDRAM accesses happen at the core clock rate this alignment is not guaranteed.
For SDRAM accesses between the two processors, it is recommended to set BMAX = 0 to disable the BCNT expiry logic.

WORKAROUND:
The following workarounds will allow a BTC by forcing the master DSP to periodically jump to internal memory. This jump to internal program memory will cause the current bus master to deassert it's $\overline{BRx}$ line and allow the slave processor to take control of the bus:

Workaround 1:
Insert a delayed-branch dummy subroutine call to an empty function resident in internal memory that simply returns with an RTS instruction. This subroutine call can be periodically called every desired number of instructions in the external SDRAM code to redirect the program sequencer to internal memory. This enables predictable bus mastership timeout control similar to writing to BMAX and allows the least overhead of 2-4 instructions. The code fragments below shows an example dummy routine using a delayed branch call and a delayed branch RTS instruction. For a delayed branch CALL instruction, the program sequencer will execute the user's next two instructions immediately following the CALL instruction, and the third instruction after the call is placed in the PC stack.
external_SDRAM_exec:

```
CALL dummy_subroutine(db);
user instruction;
user instruction;
user instruction; //(this user instruction is placed on the PC stack);
{----}
dummy_subroutine:
rts(db);
nop;
nop;
```

Worse case overhead is 4 cycles (1 for the call and three for returning. Note this overhead can be further reduced to 2 cycles if multiple dummy functions are created, and the programmer places two external SDRAM instructions (which follow the CALL instruction in SDRAM) in the delayed branch portion of the RTS(db) instruction.

Workaround 2:
In the SDRAM code, periodically insert an instruction that sets a bit in the IRPTL (Interrupt Latch) register which will force a jump to a user software interrupt (SFTxI), or any other unused interrupt.

Workaround 3:
Use a high or low priority timer interrupt when the timer counter expires, which will force the program sequencer to vector to internal memory. This method also provides timeout capability similar to setting the BMAX register but does not require extra external PM instructions to force the processor to jump to internal memory.

Workaround 4:
Apply an external hardware interrupt ($\overline{IRQx}$) by connecting an $\overline{IRQx}$ pin to either a host processor or the other SHARC (via a flag output

pin). The slave SHARC can also use a timer interrupt to periodically set the flag output pin.

Workaround 5:
The current bus master can periodically poll an input flag pin. The conditional flag test can then call a dummy subroutine resident in internal memory. An example assembly instruction would look like:

```
if FLAG3_IN CALL dummy_subroutine;
```

Workaround 6:
If a host processor can be used to resolve the deadlock, the host can write to the VIRPT register of the Bus Master to generate a Vector Interrupt, which would force the Program Sequencer to go to internal memory.

Workaround for concurrent core block transfers to SDRAM:
For core block transfers, insert NOP instructions periodically in the block read or write to force the DSP core to get off the bus, allowing a BTC. You must ensure that you have at least ($\overline{CAS}$ latency + 4 SDCLK) NOP instructions for reads and 3 NOP instructions if the access is a write. This recommended number of cycles is required to match the time the SDRAM controller can lock the bus until it is finished performing the access.

Workaround for concurrent DMA transfers to SDRAM:
Set BMAX = 0 to disable the BMAX/BCNT expiry logic. If the dual processor system cannot stand long bus-lock timeout periods, then break up DMA tasks between the two processors to smaller blocks. Once a processor becomes a bus master, it will retain bus-mastership until the DMA transfer completes.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

22. 14000042 - Flushing EPx channel FIFO incorrectly clears the other EPx channel's internal DMAR counter:

DESCRIPTION:
For handshake, external handshake and paced master mode DMA transfers, the external device can make up to six $\overline{DMARx}$ requests before the processor services the first request. These requests are stored in an internal working DMARx counter (refer to page 6-10 in the ADSP-21065L user's manual). Normally, the FLSH bit in the DMACx control register is used to clear extra requests. However, setting the FLSH bit in one DMACx register incorrectly flushes the other external port DMARx counter. For example, when the EPB0 channel is flushed by writing to the FLSH bit in DMAC0, the EPB0 status is correctly flushed, however, the DMAR counter of EPB1 is incorrectly flushed. The reverse is also true.

WORKAROUND:
a) If only 1 external port DMA channel is used, flush both external port DMA channels by setting the FLSH bit in both DMAC0 and DMAC1.
b) If both external port DMA channels are active, the user should determine when the FLSH bit should be set so that it does not affect operation of the other channel, depending on the situation. For example, if the EPB0 channel needs to be flushed, one can wait for the EPB1 DMA to complete, and then flush both the channels. This can be done upon entering the EPB1 DMA interrupt service routine, or the DSP core can poll the DMA channel status register (DMASTAT) to determine if the other EPB1 channel is completed.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

23. 14000043 - Glitch Rejection Circuits not applicable to currently shipping ADSP-21065L products:

DESCRIPTION:
Section 11.4.1 Glitch Rejection Circuits on page 12-41 of the ADSP-21065L SHARC User's Manual is not applicable to the currently shipping ADSP-21065L's. All critical signal lines such as CLKIN, $\overline{RD}$, $\overline{WR}$, $\overline{DMAR}$, RCLKs, and TCLKs should carefully laid out to eliminate glitches.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

24. 14000044 - Enabling EP DMA (only Int. to Ext. with no packing) with $\overline{\text{HBG}}$ low corrupts data:

DESCRIPTION:
Enabling an External Port DMA with no packing while $\overline{\text{HBG}}$ is asserted (low) will cause anomalous behavior. (If DMA enable happens concurrent with $\overline{\text{HBG}}$ rising or later, there is no failure.) If the DMA is a read (data moving from external memory to internal memory), the first data word will not make it to internal memory, the count will not reach zero, and so the DMA will never complete (and there will be no interrupt.) If the DMA is a write (internal memory to external memory), the second word written to the external memory is the same as the first word, the second data word will go to the third external memory location, and so on.

WORKAROUND:
Ensure $\overline{\text{HBG}}$ is not asserted before enabling the DMA transfer using the following steps:

1. Set Buslock (bit 6 in MODE2).
2. Wait for HSTM bit to clear (bit 0 in SYSTAT).
3. Enable DMA (bit 0 in appropriate DMACx).
4. Clear Buslock.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

25. 14000045 - DMA Read from BMS space ignores specified wait states.:

DESCRIPTION:
Reads from BMS address space (only possible via DMA) do not use the number of wait states specified in bits 22-24 of the WAIT register. Instead, the DSP mistakes the access as an MMS access and uses the number of waitstates specified for MMS accesses (bit 30 of the WAIT register). The default for MMS accesses is 1 waitstate. This failure will always occur if bit 23 of the external (BMS space) address is zero.

WORKAROUND:
To ensure that the programmed number of waitstates are used, bit 23 of the external address must have a value of (1) one (inconsequential because this bit is rarely connected to the boot PROM). This can be achieved simply by OR-ing the the desired external address with a value of 0x800000. For example:

```
#INCLUDE <def21065l.h>
#DEFINE BIT23 0x800000
USTAT1 = 0x4000|BIT23 ; // 0x4000 = desired ext. address
DM(EIEP0) = USTAT1; // set external address for an EP0 DMA
```

APPLIES TO REVISION(S):
0.0, 0.1, 0.2

26. 14000046 - DMA Write to BMS space ignores specified wait states.:

DESCRIPTION:
Writes to BMS address space (only possible via DMA) do not use the number of wait states specified in bits 22-24 of the WAIT register. Instead, the DSP mistakes the access as an MMS access and uses the number of waitstates specified for MMS accesses (bit 30 of the WAIT register). The default for MMS accesses is 1 waitstate. This failure will always occur if bit 23 of the external (BMS space) address is zero.

WORKAROUND:
Utilize the same workaround as Anomaly for DMA read: Bit-23 of the external address must be set to (1) one to prevent this failure.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

27. 14000047 - Weak Keeper Latches exist on Address23-0 & Data31-0 pins:

DESCRIPTION:
This keeper latch is used to maintain the input value on these pins at the value that it was last driven. The keeper latch can be described by the following specifications:

Parameter	Min	Max	Units
Keeper High Load Current IIKH	- 5	- 1	micro Amperes
Keeper Low Load Current IIKL	20	15	micro Amperes
Keeper High Overdrive Current IIKH-OD	-10	---	micro Amperes
Keeper Low Overdrive Current IIkL-OD	30	---	micro Amperes

This data augments the information available in ADSP-21065L datasheet.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

28. 14000048 - Erroneous SPORT receive interrupts when DMA chaining is enabled:

DESCRIPTION:
ADSP-21065L may generate unwanted SPORT receive interrupts. This happens under the following conditions:

- SPORTx (x = 0 or 1) Transmit and Receive are enabled in either internal/external loopback mode.
- Both Channels A and B of SPORTx Transmit and Receive are enabled in DMA chaining mode.
- SPORTx transmit and receive interrupts are unmasked in the IMASK register.
- PCI bit of chain pointer register is set for SPORTx transmit only.

In the situation mentioned above, whenever the count values of both the DMA channels go to zero at the same time, the receive DMA interrupt may be triggered in addition to the transmit DMA interrupt.

WORKAROUND:
1. Disable the SPORT receive interrupt in IMASK register.
2. Clear the PCI bit of chain pointer register for the SPORTx transmit.
3. Enable only channel A or channel B for both transmit and receive.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

29. 14000049 - Sport Tx data corruption in TDM mode when sport with DMAs is disabled and re-enabled:

DESCRIPTION:
When the SPORTs operating with the DMAs are disabled and then re-enabled, the Tx/Rx data may get corrupted on the SPORT channels. The errors encountered are corruption of the MSBs of the data words in the Rx/Tx slots.

WORKAROUND:
A work-around for this is to always disable the MCE/DMA in the following sequence:

a. Disable the Tx DMAs (DEN/CHEN etc.) by writing into STCTLx register.
b. Disable the Rx DMAs (DEN/CHEN etc.) by writing into SRCTLx register.
c. Disable the MCE (Multi-channel mode) by writing into SRCTLx register.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

30. 14000050 - SPORT in TDM mode when enabled at edge of the frame sync causes data corruption:

DESCRIPTION:
While enabling the SPORTs in multi-channel mode with external FS and CLK, the enabling of the sports should not coincide with the active edge of the FS. This would lead to the channel shift in the TDM Tx slots. This is independent of Tx/Rx.

WORKAROUND:
1. While enabling the SPORT in TDM mode, make sure that MCE bit is NOT set at the boundary of the frame sync pulse. This can be done either by connecting the FS to one of the FLAG/$\overline{\text{IRQx}}$ pins and polling it before enabling the serial ports.
OR
2. Do not use channel zero in multi-channel mode.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

31. 14000051 - Sport Tx/Rx data corruption in I2S mode and DSP serial mode when sport with DMAs is disabled and re-enabled later:

DESCRIPTION:
When the SPORTs with DMA operations are disabled and then re-enabled, the Tx/Rx data may get corrupted on the SPORT channels. The errors encountered are corruption of the MSBs of the data words and addition of unwanted channel in the Rx/Tx data.

WORKAROUND:
A work-around for this is to always disable the Serial Port/DMA in the following sequence of steps:

a. Disable serial port DMAs (by clearing the SDEN_X, SCHEN_X bits) in the STCTLx/SRCTLx registers.
b. Disable the sports (by clearing the SPEN_X bit) in the STCTLx/SRCTLx registers.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

32. 14000052 - Word shift in Tx data when using SPORT in I2S slave mode:

DESCRIPTION:
When the SPORT in I2S slave mode is enabled at an active edge of the frame sync, the Tx data may encounter a word shift.

WORKAROUND:
While enabling the SPORT in I2S slave, make sure that it is not enabled at an edge of the frame sync pulse. This can be done by connecting the FS pin to one of the FLAG/$\overline{IRQ}$ pins and polling it before enabling the serial ports.

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

33. 14000053 - Reset state of EIEP0 for PROM Boot Mode:

DESCRIPTION:
Documentation note:
Revision 0.0 and 0.1 ADSP-21065L initializes the EIEP0 (EP DMA channel 8 index) register after reset to 0x20000. All later revisions will initialize EIEP0 to 0x00000000. This feature allows PROMs larger than 128Kx8 to be directly connected to the address bus.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

34. 14000054 - SWPD bit added to SYSTAT:

DESCRIPTION:
Documentation note:
Revision 0.2 and later will have a new bit added to the SYSTAT register called SWPD (slave write pending data). This is bit#12 in the SYSTAT register. SWPD indicates that data is pending in the slave write FIFO. Revision 0.0 and 0.1 do not have this functionality.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

35. 14000055 - $\overline{DMARx}$/$\overline{DMAGx}$ DMA channel assignments:

DESCRIPTION:
Documentation note:
The preliminary ADSP-21065L datasheet assigns the $\overline{DMARx}$/$\overline{DMAGx}$ pins to incorrect DMA channels. The correct assingments are:

$\overline{DMAR1}$/$\overline{DMAG1}$ assigned to DMA channel 9 and EPB1
$\overline{DMAR2}$/$\overline{DMAG2}$assigned to DMA channel 8 and EPB0

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

36. 14000056 - For PROM booting EIEPx bit A23 must be set:

DESCRIPTION:
Documentation note:
For PROM booting, bit A23 must be set in the external DMA index register (EIEPx) that is used to for DMA PROM reads. This is normally done in the PROM boot loader kernel by adding 0x800000 to the EIEP0 address read from the PROM. There are two reasons why A23 must be set:

- This will ensure the proper number of wait states are inserted for DMAs with with the setting of bit BSO (boot select override) of the SYSCON (system configuration) register during the execution of the 256-word loader kernel (The March 22, 1999 anomaly list gave the recommendation to set bit A24, which is incorrect).
- This also ensures that in a dual ADSP-21065L system, the slave SHARC processor does not misinterpret the master SHARC's PROM access as an access to itself. NOTE: the fix for anomaly 14000035 in Revision 0.3 is only implemented for banked memory accesses using the $\overline{\text{MSx}}$ signals, and no fix for $\overline{\text{BMS}}$ space was added because the latest ADI supplied loader kernel sets bit A23 in the EIEP0 register.

If you are creating a custom loader kernel, you must set bit A23 in the EIEPx register to generate correct boot memory access waitstates and to work around anomaly 14000035 for $\overline{\text{BMS}}$ accesses. An updated boot loader kernel 065l_PROM.DXE can be obtained from dsp.support@analog.com.The SHARC VisualDSP version 4.1 contains this change (see 14000058).

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

37. 14000057 - Datasheet timing specification changes for revision 0.3:

DESCRIPTION:
Documentation note:
The following timing parameters have been have been documented to note differences in timing specification for silicon revision 0.2 and 0.3:

Specification Name	Datasheet Value (Rev. 0.2)	Datasheet Value (Rev 0.3)
tDGWRR (DMA Handshake)	min 1.0 ns	min 0.75 ns
tHDSDK (SDRAM Interface)	min 1.0 ns	min 1.25 ns
tSFSE (Serial Ports)	min 3.5 ns	min 4.0 ns
tACKTR (Synchronous READ/WRITE Bus Slave)	min 1.5 ns, max 7.0 ns	min 1.0 ns, max 6.0 ns
tACKTR (Three State Timing)	min 1.5 ns, max 7.0 ns	min 1.0 ns, max 6.0 ns
tACKEN (Three State Timing)	min 7.75 ns	min 7.5 ns

WORKAROUND:
None

APPLIES TO REVISION(S):
0.3

38. 14000058 - Updated VisualDSP 4.1 PROM kernel contains many changes from VisualDSP 4.0/4.01:

DESCRIPTION:
Documentation note:
An updated boot loader kernel 065l_PROM.DXE can be obtained from dsp.support@analog.com, which contains the following additions:

- DMA index register EIEP0 bit A23 is set (see 14000056).
- Detects the revision in MODE2 register to determine if bit A17 needs to be set in the EIEP0 (see 14000053).
- The VisualDSP 4.1 loader now correctly supports initialization of 48-bit external program instructions that will reside in 32-bit external memory. The programmer can now specify the WIDTH(48) for 48-bit external PM code segments in the Linker Description File, and not require use of the PACKING( ) command. Newer instructions have been added to the loader kernel so that the kernel now parses any 48-bit instructions in external memory into two 32-bit words. This only requires one 48-bit DMA by the loader kernel to initialize one 48-bit external PM instruction, resulting in a smaller .LDR file size. The previous method in release 4.0/4.01 required the programmer to declare external PM code with a WIDTH(32) for 32-bits under the MEMORY{} section in the Linker Description File. The PACKING( ) command in the external code section was required so that 48-bit words were parsed to two 32-bit data at link-time. The result of using this method is that the Elfloader utility stores these two 32-bit words as two 48-bit DMA words in PROM space, wasting 6 bytes of PROM storage for each instruction.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3

39. 14000059 - Changes in the notes given in ADSP-21065L data sheet (rev B):

DESCRIPTION:
Documentation note:
For the 21065L Datasheet (revision B) This following note replaces note 1 on pg. 22 (Multiprocessor Bus Request and Host Bus Request) and the * note on pg. 24(Asynchronous Read/Write--Host to ADSP-21065L) of the ADSP-21065L:

For first asynchronous access after $\overline{\text{HBR}}$ and $\overline{\text{CS}}$ asserted, ADDR23-0 must be a non-MMS value 1/2 tCK before $\overline{\text{RD}}$ or $\overline{\text{WR}}$ goes low or by tHBGRCSV $\overline{\text{HBGM}}$ goes low. This is easily accomplished by driving an upper address signal high when $\overline{\text{HBG}}$ is asserted. See the Host Processor Control of the ADSP-21065L section of the ADSP-21065L SHARC User's Manual, Second Edition.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.0, 0.1, 0.2, 0.3