ANALOG DEVICES

Blackfin Embedded Processor

Silicon Anomaly List — ADSP-BF523/BF525/BF527(C)

ABOUT ADSP-BF523/BF525/BF527(C) SILICON ANOMALIES

These anomalies represent the currently known differences between revisions of the Blackfin® ADSP-BF523/BF525/BF527(C) product(s) and the functionality specified in the ADSP-BF523/BF525/BF527(C) 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. The implementation field bits <15:0> of the DSPID core MMR register can be used to differentiate the revisions as shown below.

Silicon REVISION	DSPID<15:0>
0.2	0x0002
0.1	0x0000*

* - See anomaly 05000364

APPLICABILITY

Each anomaly applies to specific silicon revisions. See Summary or Detailed List for affected revisions. Additionally, not all processors described by this anomaly list have the same feature set. Therefore, peripheral-specific anomalies may not apply to all processors. See the below table for details. An "x" indicates that anomalies related to this peripheral apply only to the model indicated, and the list of specific anomalies for that peripheral appear in the rightmost column.

Peripheral	ADSP-BF527(C)	ADSP-BF525(C)	ADSP-BF523(C)	Anomalies
USB	x	x		05000346, 05000347, 05000380, 05000415, 05000420, 05000450, 05000456, 05000460, 05000465, 05000466, 05000467, 05000483, 05000510
Ethernet MAC	x			05000341, 05000423
CODEC	x	x	x	05000487

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
01/15/2016	K	D	Added Anomalies: 05000510, 05000511 Revised Anomaly: 05000265
03/14/2014	J	D	Added Anomalies: 05000503, 05000506 Removed Silicon Revision 0.0
05/23/2011	I	B	Added Anomalies: 05000490, 05000491, 05000494, 05000498, 05000501
04/29/2010	H	A	Added Anomalies: 05000119, 05000434, 05000473, 05000475, 05000477, 05000481, 05000483, 05000485, 05000487
08/25/2009	G	0	Added Anomalies: 05000450, 05000460, 05000461, 05000462, 05000465, 05000466, 05000467, 05000469

NR003469K Document Feedback

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

SUMMARY OF SILICON ANOMALIES

The following table provides a summary of ADSP-BF523/BF525/BF527(C) anomalies and the applicable silicon revision(s) for each anomaly.

No.	ID	Description	Rev 0.1	Rev 0.2
1	05000074	Multi-Issue Instruction with dsp32shiftimm in slot1 and P-reg Store in slot2 Not Supported	x	x
2	05000119	DMA_RUN Bit Is Not Valid after a Peripheral Receive Channel DMA Stops	x	x
3	05000122	Rx.H Cannot Be Used to Access 16-bit System MMR Registers	x	x
4	05000245	False Hardware Error from an Access in the Shadow of a Conditional Branch	x	x
5	05000254	Incorrect Timer Pulse Width in Single-Shot PWM_OUT Mode with External Clock	x	x
6	05000265	Sensitivity To Noise with Slow Input Edge Rates on External SPORT TX and RX Clocks	x	x
7	05000310	False Hardware Errors Caused by Fetches at the Boundary of Reserved Memory	x	x
8	05000313	PPI Is Level-Sensitive on First Transfer In Single Frame Sync Modes	x	.
9	05000328	Incorrect Access of OTP_STATUS During otp_write() Function	x	.
10	05000330	Host DMA Boot Modes Are Not Functional	x	.
11	05000337	Disallowed Configuration Prevents Subsequent Allowed Configuration on Host DMA Port	x	.
12	05000341	Ethernet MAC MDIO Reads Do Not Meet IEEE Specification	x	.
13	05000342	TWI May Not Operate Correctly Under Certain Signal Termination Conditions	x	.
14	05000346	USB Calibration Value Is Not Initialized	x	.
15	05000347	Preboot Routine Incorrectly Alters Reset Value of USB Register	x	.
16	05000355	Regulator Programming Blocked when Hibernate Wakeup Source Remains Active	x	.
17	05000357	Serial Port (SPORT) Multichannel Transmit Failure when Channel 0 Is Disabled	x	.
18	05000364	Incorrect Revision Number in DSPID Register	x	.
19	05000366	PPI Underflow Error Goes Undetected in ITU-R 656 Mode	x	x
20	05000368	Incorrect Default CSEL Value in PLL_DIV	x	.
21	05000371	Possible RETS Register Corruption when Subroutine Is under 5 Cycles in Duration	x	.
22	05000376	Authentication Fails To Initiate	x	.
23	05000380	Data Read From L3 Memory by USB DMA May be Corrupted	x	.
24	05000382	8-Bit NAND Flash Boot Mode Not Functional	x	.
25	05000385	Boot from OTP Memory Not Functional	x	.
26	05000386	bfrom_SysControl() Firmware Routine Not Functional	x	.
27	05000387	Programmable Preboot Settings Not Functional	x	.
28	05000388	CRC32 Checksum Support Not Functional	x	.
29	05000389	Reset Vector Must Not Be in SDRAM Memory Space	x	.
30	05000392	pTempCurrent Not Present in ADI_BOOT_DATA Structure	x	.
31	05000393	Deprecated Value of dTempByteCount in ADI_BOOT_DATA Structure	x	.
32	05000394	Log Buffer Not Functional	x	.
33	05000395	Hook Routine Not Functional	x	.
34	05000396	Header Indirect Bit Not Functional	x	.
35	05000397	BK_ONES, BK_ZEROS, and BK_DATECODE Constants Not Functional	x	.
36	05000398	SWRESET, DFRESET and WDRESET Bits in the SYSCR Register Not Functional	x	.
37	05000399	BCODE_NOBOOT in BCODE Field of SYSCR Register Not Functional	x	.
38	05000401	PPI Data Signals D0 and D8 do not Tristate After Disabling PPI	x	.
39	05000403	Level-Sensitive External GPIO Wakeups May Cause Indefinite Stall	x	.
40	05000404	Lockbox SESR Disallows Certain User Interrupts	x	.
41	05000405	Lockbox SESR Firmware Does Not Save/Restore Full Context	x	x

No.	ID	Description	Rev 0.1	Rev 0.2
42	05000407	Lockbox SESR Firmware Arguments Are Not Retained After First Initialization	x	.
43	05000408	Lockbox Firmware Memory Cleanup Routine Does not Clear Registers	x	x
44	05000409	Lockbox firmware leaves MDMA0 channel enabled	x	.
45	05000410	Incorrect Default Internal Voltage Regulator Setting	x	.
46	05000414	OTP_CHECK_FOR_PREV_WRITE Bit is Not Functional in bfrom_OtpWrite() API	x	.
47	05000415	DEB2_URGENT Bit Not Functional	x	.
48	05000416	Speculative Fetches Can Cause Undesired External FIFO Operations	x	x
49	05000417	SPORT0 Ignores External TSCLK0 on PG14 When TMR6 is an Output	x	.
50	05000418	PPI Timing Requirements tSFSPE and tHFSPE Do Not Meet Data Sheet Specifications	x	.
51	05000420	USB PLL_STABLE Bit May Not Accurately Reflect the USB PLL's Status	x	.
52	05000421	TWI Fall Time (Tof) May Violate the Minimum I^2C Specification	x	x
53	05000422	TWI Input Capacitance (Ci) May Violate the Maximum I^2C Specification	.	x
54	05000423	Certain Ethernet Frames With Errors are Misclassified in RMII Mode	x	.
55	05000424	Internal Voltage Regulator Not Trimmed	x	.
56	05000425	Multichannel SPORT Channel Misalignment Under Specific Configuration	x	.
57	05000426	Speculative Fetches of Indirect-Pointer Instructions Can Cause False Hardware Errors	x	x
58	05000429	WB_EDGE Bit in NFC_IRQSTAT Incorrectly Reflects Buffer Status Instead of IRQ Status	x	.
59	05000430	Software System Reset Corrupts PLL_LOCKCNT Register	.	x
60	05000431	Incorrect Use of Stack in Lockbox Firmware During Authentication	x	x
61	05000434	SW Breakpoints Ignored Upon Return From Lockbox Authentication	x	x
62	05000435	Certain SIC Registers are not Reset After Soft or Core Double Fault Reset	x	x
63	05000439	Preboot Cannot be Used to Alter the PLL_DIV Register	x	x
64	05000440	bfrom_SysControl() Cannot be Used to Write the PLL_DIV Register	x	x
65	05000443	IFLUSH Instruction at End of Hardware Loop Causes Infinite Stall	x	x
66	05000445	The WURESET Bit in the SYSCR Register is not Functional	x	x
67	05000450	USB DMA Short Packet Data Corruption	x	x
68	05000451	BCODE_QUICKBOOT, BCODE_ALLBOOT, and BCODE_FULLBOOT Settings in SYSCR Register Not Functional	x	x
69	05000452	Incorrect Default Hysteresis Setting for $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, and BMODE Signals	x	x
70	05000456	USB Receive Interrupt Is Not Generated in DMA Mode 1	x	x
71	05000457	Host DMA Port Responds to Certain Bus Activity Without $\overline{\text{HOST\_CE}}$ Assertion	x	x
72	05000460	USB DMA Mode 1 Failure When Multiple USB DMA Channels Are Concurrently Enabled	x	x
73	05000461	False Hardware Error when RETI Points to Invalid Memory	x	x
74	05000462	Synchronization Problem at Startup May Cause SPORT Transmit Channels to Misalign	x	x
75	05000465	USB Rx DMA Hang	x	x
76	05000466	TxPktRdy Bit Not Set for Transmit Endpoint When Core and DMA Access USB Endpoint FIFOs Simultaneously	x	x
77	05000467	Possible USB RX Data Corruption When Control & Data EP FIFOs are Accessed via the Core	x	x
78	05000469	PLL Latches Incorrect Settings During Reset	x	x
79	05000473	Interrupted SPORT Receive Data Register Read Results In Underflow when SLEN > 15	x	x
80	05000475	Possible Lockup Condition when Modifying PLL from External Memory	x	x
81	05000477	TESTSET Instruction Cannot Be Interrupted	x	x
82	05000481	Reads of ITEST_COMMAND and ITEST_DATA Registers Cause Cache Corruption	x	x
83	05000483	Possible USB Data Corruption When Multiple Endpoints Are Accessed by the Core	x	x
84	05000485	PLL_CTL Change Using bfrom_SysControl( ) Can Result in Processor Overclocking	x	x

No.	ID	Description	Rev 0.1	Rev 0.2
85	05000487	The CODEC Zero-Cross Detect Feature is not Functional	x	x
86	05000490	SPI Master Boot Can Fail Under Certain Conditions	x	x
87	05000491	Instruction Memory Stalls Can Cause IFLUSH to Fail	x	x
88	05000494	EXCPT Instruction May Be Lost If NMI Happens Simultaneously	x	x
89	05000498	CNT_COMMAND Functionality Depends on CNT_IMASK Configuration	x	x
90	05000501	RXS Bit in SPI_STAT May Become Stuck In RX DMA Modes	x	x
91	05000503	SPORT Sign-Extension May Not Work	x	x
92	05000506	Hardware Loop Can Underflow Under Specific Conditions	x	x
93	05000510	USB Wakeup from Hibernate State Requires Re-Enumeration	x	x
94	05000511	Lower 16 Bits of CNT_COUNTER Register Do Not Update Properly	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-BF523/BF525/BF527(C) including a description, workaround, and identification of applicable silicon revisions.

1. 05000074 - Multi-Issue Instruction with dsp32shiftimm in slot1 and P-reg Store in slot2 Not Supported:

DESCRIPTION:
A multi-issue instruction with dsp32shiftimm in slot 1 and a P register store in slot 2 is not supported. It will cause an exception.

The following type of instruction is not supported because the P3 register is being stored in slot 2 with a dsp32shiftimm in slot 1:

```
R0 = R0 << 0x1 || [ P0 ] = P3 || NOP;      // Not Supported - Exception
```

This also applies to rotate instructions:

```
R0 = ROT R0 by 0x1 || [ P0 ] = P3 || NOP;  // Not Supported - Exception
```

Examples of supported instructions:

```
R0 = R0 << 0x1 || [ P0 ] = R1 || NOP;
R0 = R0 << 0x1 || R1 = [ P0 ] || NOP;
R0 = R0 << 0x1 || P3 = [ P0 ] || NOP;
R0 = ROT R0 by R0.L || [ P0 ] = P3 || NOP;
```

WORKAROUND:
In assembly programs, separate the multi-issue instruction into 2 separate instructions. This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

2. 05000119 - DMA_RUN Bit Is Not Valid after a Peripheral Receive Channel DMA Stops:

DESCRIPTION:
After completion of a Peripheral Receive DMA, the DMAx_IRQ_STATUS:DMA_RUN bit will be in an undefined state.

WORKAROUND:
The DMA interrupt and/or the DMAx_IRQ_STATUS:DMA_DONE bits should be used to determine when the channel has completed running.

APPLIES TO REVISION(S):
0.1, 0.2

3. 05000122 - Rx.H Cannot Be Used to Access 16-bit System MMR Registers:

DESCRIPTION:
When accessing 16-bit system MMR registers, the high half of the data registers may not be used. If a high half register is used, incorrect data will be written to the system MMR register, but no exception will be generated. For example, this access would fail:

```
W[P0] = R5.H;   // P0 points to a 16-bit System MMR
```

WORKAROUND:
Use other forms of 16-bit transfers when accessing 16-bit system MMR registers. For example:

```
W[P0] = R5.L;   // P0 points to a 16-bit System MMR
R4.L = W[P0];
R3 = W[P0](Z);
W[P0] = R3;
```

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

4. 05000245 - False Hardware Error from an Access in the Shadow of a Conditional Branch:

DESCRIPTION:

If a load accesses reserved or illegal memory on the opposite control flow of a conditional jump to the taken path, a false hardware error will occur.

The following sequences demonstrate how this can happen:

Sequence #1:
For the "predicted not taken" branch, the pipeline will load the instructions that sequentially follow the branch instruction that was predicted not taken. By the pipeline design, these instructions can be speculatively executed before they are aborted due to the branch misprediction. The anomaly occurs if any of the three instruction slots following the branch contain loads which might cause a hardware error:

```
BRCC X [predicted not taken]
R0 = [P0];     // If any of these three loads accesses non-existent
R1 = [P1];     // memory, such as external SDRAM when the SDRAM
R2 = [P2];     // controller is off, then a hardware error will result.
```

Sequence #2:
For the "predicted taken" branch, the one instruction slot at the destination of the branch cannot contain an access which might cause a hardware error:

```
BRCC X (BP)
Y: ...
   ...
X: R0 = [P0];  // If this instruction accesses non-existent memory,
               // such as external SDRAM when the SDRAM controller
               // is off, then a hardware error will result.
```

WORKAROUND:

If you are programming in assembly, it is necessary to avoid the conditions described above.

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):

0.1, 0.2

5. 05000254 - Incorrect Timer Pulse Width in Single-Shot PWM_OUT Mode with External Clock:

DESCRIPTION:
If a Timer is in PWM_OUT mode AND is clocked by an external clock as opposed to the system clock (i.e., clocked by a signal applied to either PPI_CLK or a flag pin) AND is in single-pulse mode (PERIOD_CNT = 0), then the generated pulse width may be off by +1 or -1 count. All other modes are not affected by this anomaly.

WORKAROUND:
The suggested workaround is to use continuous mode instead of the single-pulse mode. You may enable the timer and immediately disable it again. The timer will generate a single pulse and count to the end of the period before effectively disabling itself. The generated waveform will be of the desired length.

If PULSEWIDTH is the desired width, the following sequence will produce a single pulse:

```
TIMERx_CONFIG = PWM_OUT|CLK_SEL|PERIOD_CNT|IRQ_ENA;  // Optional: PULSE_HI|TIN_SEL|EMU_RUN
TIMERx_PERIOD = PULSEWIDTH + 2;                      // Slightly bigger than the width
TIMERx_WIDTH  = PULSEWIDTH;
TIMER_ENABLE  = TIMENx;
TIMER_DISABLE = TIMDISx;
<wait for interrupt (at end of period)>
```

APPLIES TO REVISION(S):
0.1, 0.2

6. 05000265 - Sensitivity To Noise with Slow Input Edge Rates on External SPORT TX and RX Clocks:

DESCRIPTION:

A noisy board environment combined with slow input edge rates on external SPORT receive (RSCLK) and transmit clocks (TSCLK) may cause a variety of observable problems. When excessive noise occurs during high-frequency transitions on a slowly ramping RSCLK/TSCLK signal, it can cause an additional bit-clock with a short period due to high sensitivity of the clock input. A slow slew rate input allows any noise on the clock input around the switching point to cause the clock input to cross and re-cross the switching point. This oscillation can cause a glitch clock pulse in the internal logic of the serial port, which can result in numerous operational failures.

Problems which may be observed due to this glitch clock pulse include:

- In stereo serial modes, a frame sync may be missed, causing left/right data swapping.
- In multi-channel mode, multi-channel frame delay (MFD) counts may appear inaccurate or frames may be skipped.
- In normal (early) frame sync mode, received data words could be shifted right one bit. The MSB may be incorrectly captured in sign extension mode.
- In any mode, received or transmitted data words may appear to be partially right shifted if noise occurs on any input clocks between the start of frame sync and the last bit to be received or transmitted.

In Stereo Serial mode (bit 9 set in SPORTx_RCR2), unexpected high frequency transitions on RSCLK/TSCLK can cause the SPORT to miss rising or falling edges of the word clock. This causes left or right words of Stereo Serial data to be lost. This may be observed as a Left/Right channel swap when listening to stereo audio signals. The additional noise-induced bit-clock pulse on the SPORT's internal logic results in the FS edge-detection logic generating a pulse with a smaller width and, at the same time, prevents the SPORT from detecting the external FS signal during the next "normal" bit-clock period. The FS pulse with smaller width, which is the output of the edge-detection logic, is ignored by the SPORT's sequential logic. Due to the fact that the edge detection part of the frame sync logic was already triggered, the next "normal" RSCLK will not detect the change in RFS. In I^2S/EIAJ mode, this results in one stereo sample being detected/transferred as two left/right channels, and all subsequent channels will be word-swapped in memory.

In multi-channel mode, the MFD logic receives the extra sync pulse and begins counting early or double-counting (if the count has already begun). A MFD of zero can roll over to 15, as the count begins one cycle early.

In early frame sync mode, if the noise occurs on the driving edge of the clock the same cycle that FS becomes active, the FS logic receives the extra runt pulse and begins counting the word length one cycle early. The first bit will be sampled twice and the last bit will be skipped.

In all modes, if the noise occurs in any cycle after the FS becomes active, the bit counting logic receives the extra runt pulse and advances too rapidly. If this occurs once during a work unit, it will finish counting the word length one cycle early. The bit where the noise occurs will be sampled twice, and the last bit will be skipped.

While the above audio failures are possible signatures associated with this anomaly, numerous other failures are possible due to the internal logic being subjected to what amounts to an out-of-spec clock signal. Even though the external signal is within specification, the noise causes multiple transitions to be sensed where only one transition actually occurred, resulting in an out-of-spec clock being presented to the internal logic. This can lead to illegal logic states and/or incorrect advancement of state machines, which adversely affects the SPORT itself and synchronization with other logic units like the DMA engine. A number of failure scenarios may result from this, including misreported SPORT/DMA errors and unexpected DMA halts.

WORKAROUND:

1) Decrease the sensitivity to noise by increasing the slew rate of the bit clock or make the rise and fall times of serial bit clocks short, such that any noise around the transition produces a short duration noise-induced bit-clock pulse. This small high-frequency pulse will not have any impact on the SPORT or on the detection of the frame-sync. Sharpen edges as much as possible, if this is suitable and within EMI requirements.
2) If possible, use internally generated bit-clocks and frame-syncs.
3) Follow good PCB design practices. Shield RSCLK with respect to TSCLK lines to minimize coupling between the serial clocks.
4) Separate RSCLK, TSCLK, and Frame Sync traces on the board to minimize coupling which occurs at the driving edge when FS switches.

A specific workaround for problems observed in Stereo Serial mode is to delay the frame-sync signal such that noise-induced bit-clock pulses do not start processing the frame-sync. This can be achieved if there is a larger serial resistor in the frame-sync trace than the one in the bit-clock trace. Frame-sync transitions should not cross the 50% point until the bit-clock crosses the 10% of VDD threshold (for a falling edge bit-clock) or the 90% threshold (for a rising edge bit-clock).

To improve immunity to noise, optional hysteresis can be enabled for input pins by setting the appropriate bits in the PORTx_HYSTERESIS register.

APPLIES TO REVISION(S):

0.1, 0.2

7. 05000310 - False Hardware Errors Caused by Fetches at the Boundary of Reserved Memory:

DESCRIPTION:
Due to fetches near boundaries of reserved memory, a false Hardware Error (External Memory Addressing Error) is generated under the following conditions:

1) A single valid CPLB spans the boundary of the reserved space. For example, a CPLB with a start address at the beginning of L1 instruction memory and a size of 4MB will include the boundary to reserved memory.

2) Two separate valid CPLBs are defined, one that covers up to the byte before the boundary and a second that starts at the boundary itself. For example, one CPLB is defined to cover the upper 1kB of L1 instruction memory before the boundary to reserved memory, and a second CPLB is defined to cover the reserved space itself.

As long as both sides of the boundary to reserved memory are covered by valid CPLBs, the false error is generated. Note that this anomaly also affects the boundary of the L1_code_cache region if instruction cache is enabled. In other words, the boundary to reserved memory, as described above, moves to the start of the cacheable region when instruction cache is turned on.

WORKAROUND:
Leave at least 76 bytes free before any boundary with a reserved memory space. This will prevent false hardware errors from occurring.

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

8. 05000313 - PPI Is Level-Sensitive on First Transfer In Single Frame Sync Modes:

DESCRIPTION:
When the PPI is configured to trigger on a single external frame sync, all of the transfers require an edge on the frame sync except for the first transfer. For the first transfer only, the frame sync input is level-sensitive. This will make the PPI begin a transfer if the frame sync is at the active state, which can cause the PPI to start prematurely.

This anomaly does not apply when the PPI uses 2 or 3 frame syncs.

WORKAROUND:
When using a single external frame sync with the PPI, ensure that the frame sync is in the inactive state when the PPI is enabled.

APPLIES TO REVISION(S):
0.1

9. 05000328 - Incorrect Access of OTP_STATUS During otp_write() Function:

DESCRIPTION:
Firmware otp_write() function performs a 32bit access of the OTP_STATUS register. OTP_STATUS is a 16bit register, so a hardware error interrupt is generated (interrupt IVHW in the Event Vector Table and ILAT, IMASK, and IPEND registers).

The hardware error cause value of 0x02 is returned in the SEQSTAT register indicating that a system MMR error has occured. OTP write does not complete successfully when the hardware error interrupt is enabled and allowed to be serviced.

WORKAROUND:
When using otp_write function stored in L1 ROM firmware, mask (disable) hardware error interrupt (interrupt IVHW in the Event Vector Table and ILAT, IMASK, and IPEND registers) in order to successfully program the OTP.

OTP write does not complete successfully when the hardware error interrupt is enabled and allowed to be serviced.

APPLIES TO REVISION(S):
0.1

10. 05000330 - Host DMA Boot Modes Are Not Functional:

DESCRIPTION:
The host DMA boot modes are not functional.

WORKAROUND:
Use one of the other boot modes.

APPLIES TO REVISION(S):
0.1

11. 05000337 - Disallowed Configuration Prevents Subsequent Allowed Configuration on Host DMA Port:

DESCRIPTION:
When not enabled, the Enable Host Read (EHR) and Enable Host Write (EHW) bits in HOST_CONTROL prevent host read/write configurations sent to the Host DMA Port from being forwarded to the DMA Controller. After a disallowed configuration has been sent, the Host DMA Port does not forward subsequent allowed configurations to the DMA Controller.

WORKAROUND:
After the host has sent a disallowed configuration, it subsequently needs to send a Host IRQ control command. In the HIRQ interrupt service routine, reset the Host DMA Port by setting the HRST bit in the HOST_CONTROL register. A read/modify/write of HOST_CONTROL is necessary to prevent other settings from changing. Then, the ISR should set the handshake bit (HSHK) to let the host know that it's safe to configure again.

The host should read HOST_STATUS until HSHK has been set and ALLOW_CNFG = 1. The host can then proceed with the configuration as normal.

Example interrupt service routine code implementing the steps described above follows:

```
//Reset Host DMA Port
    P0.H = hi(HOST_CONTROL);
    P0.L = lo(HOST_CONTROL);
    R0=w[P0] (z);
    R1.L=0x0008;
    R0=R0 | R1;
    W[P0] = R0;
    ssync;

//Clear interrupt source and set HSHK bit
    P0.H = hi(HOST_STATUS);
    P0.L = lo(HOST_STATUS);
    R0=0x0050;
    W[P0] = R0;
    ssync;
```

APPLIES TO REVISION(S):
0.1

12. 05000341 - Ethernet MAC MDIO Reads Do Not Meet IEEE Specification:

DESCRIPTION:
When reading data from PHY registers via the Ethernet MAC (EMAC) MII interface, the EMAC latches the MDIO signal sourced by the PHY at the rising edge of the System Clock following the falling edge of MDC. The IEEE 802.3 standard stipulates that the PHY drive the MDIO at the rising edge of MDC with a 0 - 300ns output delay. In certain combinations of SCLK/MDC clocks and circuit routing, the EMAC may not be able to latch the correct MDIO data value.

WORKAROUND:
The following condition must be met to ensure the EMAC latches the correct MDIO data:

2 * Tprop + 300ns (or the actual PHY delay) < 0.5 * MDC period + tck

where:
- Tprop is the propagation time between the PHY and the Blackfin MII.
- tck is the delay between MDC falling edge and SCLK rising edge (-0.5 * SCLK period + 4ns).

To achieve this requirement, SCLK and MDC can be adjusted by programming the the PLL_DIV and EMAC_SYSCTL registers, respectively.

APPLIES TO REVISION(S):
0.1

13. 05000342 - TWI May Not Operate Correctly Under Certain Signal Termination Conditions:

DESCRIPTION:
Under the following operating conditions, the TWI controller may not operate correctly:

Vddext = Vbus = 1.8 V
Vddext = 1.8 V and Vbus = 2.5 V
Vddext = 1.8 V and Vbus = 3.3 V

Vbus here refers to the voltage rail to which the TWI signals are pulled up.

When the processor tristates the output driver in these scenarios, the TWI controller may not see the rising transition.

There is some marginality in the following cases below, but further characterization is ongoing:
Vddext = 3.3 V and Vbus = 5 V
Vddext = 2.5 V and Vbus = 3.3 V

WORKAROUND:
The TWI controller functions properly under the following operating conditions:

Vddext = Vbus = 3.3 V
Vddext = Vbus = 2.5 V

APPLIES TO REVISION(S):
0.1

14. 05000346 - USB Calibration Value Is Not Initialized:

DESCRIPTION:
After execution of the preboot routine, the USB_APHY_CALIB register may not be properly initialized. The USB peripheral may not function properly without proper initialization.

WORKAROUND:
It is recommended that users program the USB_APHY_CALIB register with a nominal value of 0x6510 in their application code if they wish to utilize the USB peripheral.

```
#define SYSMMR_BASE         0xFFC00000

...

P1.H = HI(SYSMMR_BASE); // P1 Points to the beginning of SYSTEM MMR Space
P1.L = LO(SYSMMR_BASE);

...

r0 = 0x6510(z);  // recommended value to be set
w[p1 + lo(USB_APHY_CALIB)] = r0;
ssync;
```

APPLIES TO REVISION(S):
0.1

15. 05000347 - Preboot Routine Incorrectly Alters Reset Value of USB Register:

DESCRIPTION:
After executing the preboot routine, the value of a USB control register (USB_APHY_CNTRL) is not properly restored to its default value. This causes the USB peripheral to not function properly out of reset.

WORKAROUND:
The user application must restore the value of the register prior to using the USB peripheral. This can be done with the following statements:

```
#define SYSMMR_BASE         0xFFC00000

...

P1.H = HI(SYSMMR_BASE); // P1 Points to the beginning of SYSTEM MMR Space
P1.L = LO(SYSMMR_BASE);

...

r0 = 0x0(z);            // restore default value
w[p1 + lo(USB_APHY_CNTRL)] = r0;
ssync;
```

APPLIES TO REVISION(S):
0.1

16. 05000355 - Regulator Programming Blocked when Hibernate Wakeup Source Remains Active:

DESCRIPTION:
After the processor is placed into the hibernate state, a subsequent peripheral wakeup event can take the part out of hibernate. If that wakeup source remains asserted throughout the initiated reset sequence, writes to the VR_CTL register will not get latched into the regulator when the **IDLE** sequence is executed. Latching into the regulator circuit will be blocked until the wakeup source de-asserts.

The write will effectively be lost, however, the written value will go to the VR_CTL register's physical memory-mapped address. If no further writes to VR_CTL are performed, the "lost" write will be latched into the regulator hardware when the next **IDLE** instruction is executed.

WORKAROUND:
Ensure that the peripheral hibernate wakeup source de-asserts before attempting to program the regulator via the **IDLE** sequence required after the write to VR_CTL.

APPLIES TO REVISION(S):
0.1

17. 05000357 - Serial Port (SPORT) Multichannel Transmit Failure when Channel 0 Is Disabled:

DESCRIPTION:
When configured in multi-channel mode with channel 0 disabled, DMA transmit data will be sent to the wrong SPORT channel if all of the following criteria are met:

1) External Receive Frame Sync (IRFS = 0 in SPORTx_RCR1)
2) Window Offset = 0 (WOFF = 0 in SPORTx_MCMC1)
3) Multichannel Frame Delay = 0 (MFD = 0 in SPORTx_MCMC2)
4) DMA Transmit Packing Disabled (MCDTXPE = 0 in SPORTx_MCMC2)

When this specific configuration is used, the multi-channel transmit data gets corrupted because whatever is in the channel 0 placeholder in non-packed mode gets sent first, even though channel 0 is disabled. The result is a one-word data shift in the output window, which repeats for each subsequent window in the serial stream. For example, if the non-packed transmit buffer is {0, 1, 2, 3, 4, 5, 6, 7}, and the window size is 8 channels with channel 0 disabled and channels 1-7 enabled to transmit, the expected data sequence in a series of output windows is:

1234567--1234567--1234567--1234567

With this anomaly, the output looks like this instead:

0123456--7012345--6701234--5670123

WORKAROUND:
There are several possible workarounds to this:

1) Disable Multichannel Mode
2) Use Internal Receive Frame Syncs
3) Use a Multichannel Frame Delay > 0
4) Use a Window Offset > 0
5) Enable DMA Transmit Packing
6) Do not disable Channel 0

APPLIES TO REVISION(S):
0.1

18. 05000364 - Incorrect Revision Number in DSPID Register:

DESCRIPTION:
The DSPID register does not contain the correct silicon revision information.

WORKAROUND:
The following ROM value difference below should be used to distinguish between the revisions with the same DSPID<15:0> value:

Silicon REVISION	ROM Address	Value
0.1	0xEF000014	0x2796

APPLIES TO REVISION(S):
0.1

19. 05000366 - PPI Underflow Error Goes Undetected in ITU-R 656 Mode:

DESCRIPTION:
If the PPI port is configured in ITU-R 656 Output Mode, the FIFO Underrun bit (UNDR in PPI_STATUS) does not get set when a PPI FIFO underrun occurs. An underrun can happen due to limited bandwidth or the PPI DMA failing to gain access to the bus due to arbitration latencies.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1, 0.2

20. 05000368 - Incorrect Default CSEL Value in PLL_DIV:

DESCRIPTION:
The reset value of CSEL in the PLL_DIV register is incorrectly set to 0x1, which yields a CCLK of VCO/2.

WORKAROUND:
Initcode may be used to raise the frequency of the processor during boot time.

APPLIES TO REVISION(S):
0.1

21. 05000371 - Possible RETS Register Corruption when Subroutine Is under 5 Cycles in Duration:

DESCRIPTION:

The RTS instruction can fail to return correctly if placed within four execution cycles of the beginning of a subroutine. For example:

```
CALL STUB_CODE;
...
...
...
STUB_CODE:
   RTS;
```

When this happens, potential bit failures in RETS will cause the processor to vector to the wrong address, which can cause invalid code to be executed.

WORKAROUND:

If there are at least four execution cycles in the subroutine before the RTS, the CALL and RTS instructions can never align in the manner required to encounter this problem. Since a NOP is a 1-cycle instruction, the following is a safe workaround for all potential failure cases:

```
CALL STUB_CODE;
...
...
...
STUB_CODE:
   NOP;      // These 4 NOPs can be any combination of instructions
   NOP;      // that results in at least 4 core clock cycles.
   NOP;
   NOP;
   RTS;
```

Branch prediction does not factor into this scenario. Conditional jumps within the subroutine that arrive at the RTS instruction inside of 4 cycles will not result in the scenario required to cause this failure. Asynchronous events (interrupts, exceptions, and NMI) are also not susceptible to this failure.

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):

0.1

22. 05000376 - Authentication Fails To Initiate:

DESCRIPTION:
The Secure State Machine fails to transition from Open Mode into Secure Entry Mode due to improper sensing of Program Counter addressing in the hardware monitor. As a result, the digital signature authentication process fails to properly initiate.

WORKAROUND:
This workaround only applies to rev 0.1 silicon.

Set bit 0 in the TBUFCTL register equal to 1 before initiating the digital signature authentication process.

Users can optionally reset bit 0 in TBUFCTL any time after authentication is complete to marginally reduce dynamic power consumption due to parts of trace buffer logic switching enabled when bit 0 in TBUFCTL was set to 1.

Example Code to set bit0 of TBUFCTL:

```
/* example C code */
*pTBUFCTL = 0x1;
ssync();


/* equivalent assembly code */

r0 = 0x0001 (z);
p1.h = hi(TBUFCTL);
p1.l = lo(TBUFCTL);
[p1] = r0;
ssync
```

APPLIES TO REVISION(S):
0.1

23. 05000380 - Data Read From L3 Memory by USB DMA May be Corrupted:

DESCRIPTION:
Data read from L3 memory by the USB DMA Controller may have repeated or missing values.

WORKAROUND:
Use a Memory DMA to move data from L3 to L1 memory before accessing the data with the USB DMA controller.

For USB DMAs to/from L1:

If the Core and USB DMA controller are accessing different L1 data banks, or different sub-banks within the same L1 data bank, there won't be any further elements to this workaround. However, if the Core and the USB DMA controller are performing accesses within the same L1 data bank and sub-bank, make sure to use 32-bit-aligned addresses (e.g., the start address of the USB DMA transfer to/from L1 memory should be 0x0, 0x4, 0x8, 0xC in the least significant nibble). Note that there is no restriction on the packet size transferred that is imposed by this workaround.

APPLIES TO REVISION(S):
0.1

24. 05000382 - 8-Bit NAND Flash Boot Mode Not Functional:

DESCRIPTION:
The 8-bit NAND flash boot mode (BMODE[3:0] = b#1100 and b#1101) is not functional.

WORKAROUND:
Use one of the other boot modes.

APPLIES TO REVISION(S):
0.1

25. 05000385 - Boot from OTP Memory Not Functional:

DESCRIPTION:
The "Boot from OTP Memory" boot mode is not functional.

WORKAROUND:
Use one of the other boot modes.

APPLIES TO REVISION(S):
0.1

26. 05000386 - bfrom_SysControl() Firmware Routine Not Functional:

DESCRIPTION:
The *bfrom_SysControl()* firmware function facilitates changes to dynamic power management configurations in the hardware, such as changing voltage levels, power modes, and system or core clock frequencies. *bfrom_SysControl()* is not functional.

WORKAROUND:
Dynamic power management changes must be made manually by accessing the PLL_CTL, PLL_DIV and VR_CTL registers directly.

APPLIES TO REVISION(S):
0.1

27. 05000387 - Programmable Preboot Settings Not Functional:

DESCRIPTION:
User-programmable Preboot settings are not functional. OTP pages 0x14 to 0x1F must be kept blank.

WORKAROUND:
For some of the preboot functionality, such as PLL, voltage regulator and SDRAM controller programming, the initcode feature provides an alternative. See *System Reset and Booting* chapter in the ADSP-BF52x Hardware Reference manual.

APPLIES TO REVISION(S):
0.1

28. 05000388 - CRC32 Checksum Support Not Functional:

DESCRIPTION:
The CRC32 wrapper routine (*bfrom_Crc32Callback()*) is not functional.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

29. 05000389 - Reset Vector Must Not Be in SDRAM Memory Space:

DESCRIPTION:
The reset vector is provided by the TARGET ADDRESS field of the first block header in the boot stream. If this address points to SDRAM memory space, the boot kernel does not properly solve a speculative fetch situation, as the SDRAM controller has not yet been initialized at this point in the boot process.

WORKAROUND:
Ensure the reset vector points to any memory location in L1 or external asynchronous space. If necessary, place code in one of these memory locations which jumps to SDRAM.

APPLIES TO REVISION(S):
0.1

30. 05000392 - pTempCurrent Not Present in ADI_BOOT_DATA Structure:

DESCRIPTION:
The variable *pTempCurrent* is not present in the firmware *ADI_BOOT_DATA* structure, and the rest of the structure elements' addresses after the insertion point are off by four bytes. The affected elements in the structure are *dTempByteCount*, *dBlockCount*, and *dClock*.

WORKAROUND:
Do not use *pTempCurrent*, and be sure to respect the memory alignment defined above when accessing the structure elements using offsets.

APPLIES TO REVISION(S):
0.1

31. 05000393 - Deprecated Value of dTempByteCount in ADI_BOOT_DATA Structure:

DESCRIPTION:
The *dTempByteCount* variable in the firmware *ADI_BOOT_DATA* structure defaults to 0x100 (256 bytes) instead of 0x200 (512 bytes), and the *pTempBuffer* variable points to 0xFF907F00 rather than 0xFF907E00.

WORKAROUND:
None required. The setting for 256 bytes is built into the firmware and functions as designed. It was later modified to 512 bytes to accommodate NAND boot modes.

APPLIES TO REVISION(S):
0.1

32. 05000394 - Log Buffer Not Functional:

DESCRIPTION:
The firmware Log Buffer functionality within the boot kernel provides a log of each block parsed during the boot sequence. This log buffer feature is not functional, therefore the firmware *ADI_BOOT_DATA* structure does not contain the *pLogBuffer*, *pLogCurrent*, and *dLogByteCount* variables.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1

33. 05000395 - Hook Routine Not Functional:

DESCRIPTION:
The firmware Hook routine functionality provides users with the ability to overwrite default values in the *ADI_BOOT_DATA* structure. This Hook routine feature is not functional. Consequently, the *BFLAG_HOOK* flag is not present in the *dFlags* word, and the *bfrom_MemBoot()*, *bfrom_TwiBoot()* and *bfrom_SpiBoot()* API functions do not expect the related argument.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1

34. 05000396 - Header Indirect Bit Not Functional:

DESCRIPTION:
The firmware *BFLAG_HDRINDIRECT* (header indirect) bit in the *dFlags* word is not functional.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1

35. 05000397 - BK_ONES, BK_ZEROS, and BK_DATECODE Constants Not Functional:

DESCRIPTION:
The firmware constants *BK_ZEROS*, *BK_ONES* and *BK_DATECODE* are not functional.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1

36. 05000398 - SWRESET, DFRESET and WDRESET Bits in the SYSCR Register Not Functional:

DESCRIPTION:
The SWRESET, DFRESET and WDRESET bits in the SYSCR register are not functional and read as zeros.

WORKAROUND:
This information can be found in SWRST[15:13] instead.

APPLIES TO REVISION(S):
0.1

37. 05000399 - BCODE_NOBOOT in BCODE Field of SYSCR Register Not Functional:

DESCRIPTION:
The BCODE_NOBOOT option in the BCODE field of the SYSCR register is not functional. Therefore, the processor cannot skip the boot kernel coming out of reset.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

38. 05000401 - PPI Data Signals D0 and D8 do not Tristate After Disabling PPI:

DESCRIPTION:
The PPI data signals D0 and D8 do not get tristated when PPI is disabled after a transmit operation. The rest of the PPI data signals are unaffected.

WORKAROUND:
Disable PPI as normal. Then reconfigure the PPI settings for receive mode. Re-enable the PPI port and then disable it again. D0 and D8 will be tristated.

APPLIES TO REVISION(S):
0.1

39. 05000403 - Level-Sensitive External GPIO Wakeups May Cause Indefinite Stall:

DESCRIPTION:
When level-sensitive GPIO events are used to wake the processor from the low-power sleep mode of operation, the processor may stall indefinitely if the width of the wakeup pulse is too short. When this occurs, the PLL begins transitioning from the sleep mode due to the level sensed on the GPIO pin, but then reverts back to the sleep mode if the trigger level is removed before the core has had sufficient time to break the idle state to resume execution.

As a result, the processor does not wake up properly, at which point only a hardware reset can exit the resulting stall condition.

WORKAROUND:
There are two ways to avoid this anomaly:

1) Use edge-sensitivity for the pin(s) being used to generate the wakeup event.
2) Ensure that the edge on the wakeup signal is clean and held at the trigger level for at least 3 system clock (SCLK) cycles.

APPLIES TO REVISION(S):
0.1

40. 05000404 - Lockbox SESR Disallows Certain User Interrupts:

DESCRIPTION:
When the processor enters Secure Entry Mode, interrupts are shut off (i.e. IMASK is cleared except for bits [4:0] because they are hardwired). The user can select interrupts to be unmasked via an argument to the SESR. The IVTMR (core timer) and IVHW (HW error) interrupts cannot be unmasked inside the SESR Therefore, IVTMR and IVHW cannot be serviced during the digital signature authentication process (Secure Entry Mode).

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

41. 05000405 - Lockbox SESR Firmware Does Not Save/Restore Full Context:

DESCRIPTION:
Embedding `asm("raise 2;");` to call authentication in C code is not recommended. Registers R0-R3 and P0-P2 are not saved in the SESR. The compiler will assume that any C code after `asm("raise 2;");` will still be able to use these registers safely, although they are overwritten inside the SESR.

WORKAROUND:
The following C code instruction, which informs the compiler that it is not safe to use the registers R0-R3 and P0-P2, may be used to begin authentication:

```
asm("raise 2;":::"R0", "R1", "R2", "R3", "P0", "P1", "P2");
```

When using assembly code, the user must save the registers R0-R3 and P0-P2, issue the `raise 2;` instruction, then restore the registers R0-R3 and P0-P2.

APPLIES TO REVISION(S):
0.1, 0.2

42. 05000407 - Lockbox SESR Firmware Arguments Are Not Retained After First Initialization:

DESCRIPTION:
Lockbox SESR firmware arguments must be reset in subsequent digital signature authentication attempts as the arguments are not retained after first initialization.

WORKAROUND:
Lockbox SESR arguments must be programmed upon every authentication attempt.

If digital signature authentication is to be attempted following a failure or abortion of the authentication process, users must reset their SESR arguments for subsequent authentication attempts.

APPLIES TO REVISION(S):
0.1

43. 05000408 - Lockbox Firmware Memory Cleanup Routine Does not Clear Registers:

DESCRIPTION:
The security firmware memory clear routine does not clear processor registers. It only clears on-chip SRAM memory.

Sensitive information may remain in processor registers upon exiting from Secure Mode, so users must not rely on the firmware memory clear routine to clear registers.

WORKAROUND:
Users should clear out processor registers prior to exiting Secure Mode. The security firmware memory clear routine may be called to clear on-chip SRAM or users may substitute their own memory clear routine instead.

APPLIES TO REVISION(S):
0.1, 0.2

44. 05000409 - Lockbox firmware leaves MDMA0 channel enabled:

DESCRIPTION:
During the digital signature authentication process, the security firmware uses the MDMA0 channel, but does not disable the channel after use. If the customer application made use of that same MDMA0 channel, the customer may encounter problems configuring it.

WORKAROUND:
Customer applications should first disable the MDMA0 channel used by the security firmware before attempting to reconfigure it. This is only necessary following a successful authentication process.

APPLIES TO REVISION(S):
0.1

45. 05000410 - Incorrect Default Internal Voltage Regulator Setting:

DESCRIPTION:
The default VLEV setting in the VR_CTL register was incorrectly set to 1.0V. It should have been set to 1.1V.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

46. 05000414 - OTP_CHECK_FOR_PREV_WRITE Bit is Not Functional in bfrom_OtpWrite() API:

DESCRIPTION:
OTP_CHECK_FOR_PREV_WRITE is a flag which is provided to enable the check for unprogrammed OTP bits in the firmware OTP write operation.

If the *OTP_CHECK_FOR_PREV_WRITE* bit is set in the *dFlags* argument in the *bfrom_OtpWrite()* API, the OTP data bits are first checked for zero (unprogrammed) values before the write is allowed to proceed. This error detection for a write to a previously programmed page causes dedicated error messages and the write does not occur. If the *OTP_CHECK_FOR_PREV_WRITE* bit is cleared (default), the write proceeds without first performing this check.

The *OTP_CHECK_FOR_PREV_WRITE* bit is not functional in firmware and therefore this feature cannot be configured by the user. The firmware OTP write operation always performs the check for unprogrammed OTP bits to the data page target of the write operation. If the page is detected as being previously programmed (i.e., page contains a bit whose value is '1'), the write will not occur.

WORKAROUND:
None.

APPLIES TO REVISION(S):
0.1

47. 05000415 - DEB2_URGENT Bit Not Functional:

DESCRIPTION:
The DEB2_URGENT bit[12] in the USB_PLLOSC_CTRL register is not functional. Therefore, USB accesses to the EBIU are always lower priority than core EBIU accesses. For a complete description of the DEB2_URGENT bit please refer to the BF52x Hardware Reference manual.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

48. 05000416 - Speculative Fetches Can Cause Undesired External FIFO Operations:

DESCRIPTION:
When an external FIFO device is connected to an asynchronous memory bank, memory accesses can be performed by the processor speculatively, causing improper operations because the FIFO will provide data to the Blackfin, and the data will be dropped whenever the fetch is made speculatively or if the speculative access is canceled. "Speculative" fetches are reads that are started and killed in the pipeline prior to completion. They are caused by either a change of flow (including an interrupt or exception) or when performing an access in the shadow of a branch. This behavior is described in the Blackfin Programmer's Reference.

Another case that can occur is when the access is performed as part of a hardware loop, where a change of flow occurs from an exception. Since exceptions can't be disabled, the following example shows how an exception can cause a speculative fetch, even with interrupts disabled:

```
CLI R3;                             /* Disable Interrupts */
LSETUP( loop_s, loop_e) LC0 = P2;
   loop_s: R0 = W[P0];              /* Read from a FIFO Device  */
   loop_e: W[P1++] = R0;            /* Write that Generates a Data CPLB Page Miss */
STI R3;                             /* Enable Interrupts */
RTS;
```

In this example, the read inside the hardware loop is made to a FIFO with interrupts disabled. When the write inside the loop generates a data CPLB exception, the read inside the loop will be done speculatively.

WORKAROUND:
First, if the access is being performed with a core read, turn off interrupts prior to doing the core read. The read phase of the pipeline must then be protected from seeing the read instruction before interrupts are turned off:

```
CLI R0;
NOP; NOP; NOP;  /* Can Be Any 3 Instructions */
R1 = [P0];
STI R0;
```

To protect against an exception causing the same undesired behavior, the read must be separated from the change of flow:

```
CLI R3;                             /* Disable Interrupts */
LSETUP( loop_s, loop_e) LC0 = P2;
   loop_s: NOP;                     /* 2 NOPs to Pad Read */
           NOP;
           R0 = W[P0];
   loop_e: W[P1++] = R0;
STI R3;                             /* Enable Interrupts */
RTS;
```

The loop could also be constructed to place the NOP padding at the end:

```
LSETUP( .Lword_loop_s, .Lword_loop_e) LC0 = P2;
   .Lword_loop_s: R0 = W[P0];
                  W[P1++] = R0;
                  NOP;          /* 2 NOPs to Pad Read */
   .Lword_loop_e: NOP;
```

Both of these sequences prevent the change of flow from allowing the read to execute speculatively. The 2 inserted NOPs provide enough separation in the pipeline to prevent a speculative access. These NOPs can be any two instructions.

Reads performed using a DMA transfer do not need to be protected from speculative accesses.

APPLIES TO REVISION(S):
0.1, 0.2

49. 05000417 - SPORT0 Ignores External TSCLK0 on PG14 When TMR6 is an Output:

DESCRIPTION:
When SPORT0 is set for an external TSCLK0 from PG14 and TMR6 is an output, SPORT0 will receive the TMR6 signal that is looped back internally. The external clock connected to PG14 will be ignored.

WORKAROUND:
Assign TSCLK0 to a different signal or disable TMR6.

APPLIES TO REVISION(S):
0.1

50. 05000418 - PPI Timing Requirements tSFSPE and tHFSPE Do Not Meet Data Sheet Specifications:

DESCRIPTION:
The data sheet PPI timing specifications tSFSPE and tHFSPE are not being met. Instead, use the following data:

Specification	Min	Unit
tSFSPE	2.1	ns
tHFSPE	7.0	ns

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1

51. 05000420 - USB PLL_STABLE Bit May Not Accurately Reflect the USB PLL's Status:

DESCRIPTION:
The PLL_STABLE bit in the USB_PLLOSC_CTRL register does not automatically update when there is a change in the PLL's status.

WORKAROUND:
Read the value of USB_PLLOSC_CTRL, and write it back to the register. Immediately read USB_PLLOSC_CTRL again and the correct status of PLL_STABLE will be reflected.

APPLIES TO REVISION(S):
0.1

52. 05000421 - TWI Fall Time (Tof) May Violate the Minimum I²C Specification:

DESCRIPTION:
TWI Fall Time (Tof) may be less than the minimum I²C specification. This is not a functional issue, but may have an impact on signal integrity.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1, 0.2

53. 05000422 - TWI Input Capacitance (Ci) May Violate the Maximum I²C Specification:

DESCRIPTION:
TWI input capacitance (Ci) may be may be more than the maximum I²C specification. This is not a functional issue, but may have an impact on signal integrity.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.2

54. 05000423 - Certain Ethernet Frames With Errors are Misclassified in RMII Mode:

DESCRIPTION:
In RMII-mode at 100Base-T, some RX frames with FCS errors are incorrectly classified as having alignment errors in the EMAC_RX_STAT register.

WORKAROUND:
Ensure that all error frames are properly discarded.

APPLIES TO REVISION(S):
0.1

55. 05000424 - Internal Voltage Regulator Not Trimmed:

DESCRIPTION:
The internal voltage regulator tolerance is not factory trimmed. Therefore, internal voltage regulator accuracy may vary.

WORKAROUND:
Use the code below to trim the internal voltage regulator from a VLEV offset stored in OTP memory. If using VisualDSP++ 5.0 (Update 3 or later) or CrossCore Embedded Studio, the bfrom.h header file must be included in order to read from OTP. Note that upon power-up the default VLEV setting is not trimmed.

This example sets the regulator to 1.2V with the proper trim value:

```
// 16-bit data elements to temporarily store VR_CTL and VLEV field
u16 VR_reg, VLEV_field;

// 32-bit data elements to store return code and SIC_IWRx registers
u32 return_code = 0, SIC_IWR0_reg, SIC_IWR1_reg;

// 64-bit data element to store the data read from OTP memory
u64 VLEV_offset = 0;

// Save information to local data
SIC_IWR0_reg = *pSIC_IWR0;
SIC_IWR1_reg = *pSIC_IWR1;

VLEV_field = 0xB;                          // 0xB = 1.2V in VLEV Field

// Read OTP Memory Page 8
return_code = bfrom_OtpRead(8,OTP_UPPER_HALF, &VLEV_offset);
if ( (return_code && (1 << OTP_MASTER_ERROR)) != 0)
   exit (1);    // Exit or investigate what error has occurred and take actions
VLEV_offset &= 0x0000000000000F0;          // Mask out everything but VLEV_offset

switch (VLEV_offset) {
   case 0x10:                              // If the offset is 1, add 1 to VLEV
      VLEV_field += 1;
      break;

   case 0xF0:                              // If the offset is -1, subtract 1 from VLEV
      VLEV_field += -1;
      break;
   // If there is no offset then proceed
}

VLEV_field <<= 4;           // Shift VLEV field into proper position
VR_reg = *pVR_CTL;          // Read VR_CTL into temporary data
VR_reg &= 0xFF0F;           // Clear out current VLEV field data
VR_reg |= VLEV_field;       // Deposit new VLEV field into VR_CTL word
*pSIC_IWR0 = 0x1;           // Enable only the PLL wakeup to break idle
*pSIC_IWR1 = 0x0;           // Disable all other possible wakeups
*pVR_CTL = VR_reg;          // Write new value to VR_CTL
idle();                     // Drain pipeline, enter idled state, wait for PLL wakeup
*pSIC_IWR0 = SIC_IWR0_reg;  // Restore original SIC_IWR0 register value
*pSIC_IWR1 = SIC_IWR1_reg;  // Restore original SIC_IWR1 register value
```

APPLIES TO REVISION(S):
0.1

56. 05000425 - Multichannel SPORT Channel Misalignment Under Specific Configuration:

DESCRIPTION:
When using the Serial Port in Multi-Channel Mode, the transmit and receive channels can get misaligned if a very specific configuration for the SPORT is met, as follows:

1) Window Offset (WOFF) = 0.
2) Window Size is an odd multiple of 8 (i.e., WSIZE is an even number > 0).
3) The time between RFS pulses is exactly equal to the window duration.

Note: The anomaly does NOT apply when WSIZE = 0.

When this exact configuration is used, the multi-channel mode channel enable registers are mislatched after the first window concludes, which results in the TDV signal being driven according to incorrect channel assignments and receive data being sampled on the wrong channels. So, the first window will send and receive properly, but all windows after the first will be misaligned, and data sent and received will be corrupted.

This error occurs for external and internal clocks and RFS.

WORKAROUND:
There are several workarounds possible:

1) Use a window offset other than 0.
2) Use a window size that is an even multiple of 8.
3) For internal RFS, make sure that SPORTx_RFSDIV is at least equal to the window size (# of enabled channels * SLEN).

APPLIES TO REVISION(S):
0.1

57. 05000426 - Speculative Fetches of Indirect-Pointer Instructions Can Cause False Hardware Errors:

DESCRIPTION:
A false hardware error is generated if there is an indirect jump or call through a pointer which may point to reserved or illegal memory on the opposite control flow of a conditional jump to the taken path. This commonly occurs when using function pointers, which can be invalid (e.g., set to -1). For example:

```
    CC = P2 == -0x1;
    IF CC JUMP skip;
    CALL (P2);
  skip:
    RTS;
```

Before the IF CC JUMP instruction can be committed, the pipeline speculatively issues the instruction fetch for the address at -1 (0xffffffff) and causes the false hardware error. It is a false hardware error because the offending instruction is never actually executed. This can occur if the pointer use occurs within two instructions of the conditional branch (predicted not taken), as follows:

```
    BRCC X [predicted not taken]
    Y: JUMP (P-reg);  // If either of these two p-regs describe non-existent
       CALL (P-reg);  // memory, such as external SDRAM when the SDRAM
    X: RTS;           // controller is off, then a hardware error will result.
```

WORKAROUND:
If instruction cache is on or the ICPLBs are enabled, this anomaly does not apply.

If instruction cache is off and ICPLBs are disabled, the indirect pointer instructions must be 2 instructions away from the branch instruction, which can be implemented using NOPs:

```
    BRCC X [predicted not taken]
    Y: NOP;          // These two NOPs will properly pad the indirect pointer
       NOP;          // used in the next line.
       JUMP (P-reg);
       CALL (P-reg);
    X: RTS;
```

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

58. 05000429 - WB_EDGE Bit in NFC_IRQSTAT Incorrectly Reflects Buffer Status Instead of IRQ Status:

DESCRIPTION:
The NAND Flash Controller (NFC) "write buffer edge detect" (WB_EDGE) interrupt is generated when the 4-word-deep write buffer of the NFC becomes empty. The functional generation of the interrupt is correct. The functional description of the WB_EDGE bit is described as "The write buffer edge detect (WB_EDGE) is set when the write buffer transitions from 'not empty' to 'empty.'" This correctly describes the behavior through which the interrupt is generated. However, the status bit for this interrupt does not align with the functional behavior of the interrupt.

The WB_EDGE bit of NFC_IRQSTAT register currently behaves as a "write buffer empty" status bit instead of as an actual IRQ status bit. The WB_EDGE bit is set when the NFC write buffer is empty and cleared when the buffer is not empty. The IRQ is cleared correctly after the W1C operation, but the bit will remain set. This does not reflect the IRQ status, and the behavior of this bit is identical to the WB_EMPTY bit in the NFC_STAT register.

The NFC_IRQSTAT default reset value of 0x0004 indicates that an IRQ is already active. The intended default reset value is 0x0000.

WORKAROUND:
Do not depend upon the WB_EDGE bit of NFC_IRQSTAT register to determine status of the write buffer.
The status of the write buffer should be determined by checking the WB_EMPTY bit or WB_FULL bit in the NFC_STATUS register.

If software requires an interrupt handler that services WB_EDGE interrupts, the following procedure is advised:

1) Only unmask the WB_EDGE interrupt prior to performing NFC operations that are issued through the write buffer. Otherwise mask off the interrupt.
2) From within the interrupt handler, service all interrupts except for the WB_EDGE interrupt and clear them by performing the appropriate W1C operation to the NFC_IRQSTAT register.
3) Service the WB_EDGE interrupt only when the SIC_ISRx register indicates an NFC interrupt is still latched and the only bit set in the NFC_IRQSTAT register is the WB_EDGE bit.
4) Mask off the WB_EDGE interrupt within the WB_EDGE interrupt handler if the handler does not issue any NFC write buffer transactions. If the WB_EDGE IRQ handler does issue NFC write buffer transactions, ensure the current WB_EDGE interrupt is cleared before performing the transactions.

APPLIES TO REVISION(S):
0.1

59. 05000430 - Software System Reset Corrupts PLL_LOCKCNT Register:

DESCRIPTION:
Software reset (`raise 1;`, core double fault, watchdog reset, *bfrom_SysControl()* call with *SWRST* flag) will return with PLL_LOCKCNT register value set to 0x0007 (instead of intended reset value of 0x0200). This may result in failure to subsequently re-program PLL and Voltage Regulator control registers.

Hardware reset will result in proper reset value loaded into PLL_LOCKCNT and no action is required.

WORKAROUND:
Following any software reset (`raise 1;`, core double fault, watchdog reset, *bfrom_SysControl()* call with *SWRST* flag) users cannot rely upon the default reset value within the PLL_LOCKCNT register and must explicitly write a value of 0x0200 or larger to PLL_LOCKCNT register before making subsequent changes to PLL or Voltage Regulator control registers.

Do not use the Preboot mechanism for setting PLL or Voltage Regulator control registers if power settings will be changed before entering software reset. It is recommended to use application code or init code to explicitly write to PLL_LOCKCNT prior to changing the PLL or Voltage Regulator control registers as stated above.

APPLIES TO REVISION(S):
0.2

60. 05000431 - Incorrect Use of Stack in Lockbox Firmware During Authentication:

DESCRIPTION:
Some of the cryptographic routines utilized by the security firmware use the stack in ways that do not conform to the run-time execution model by using stack locations that reside both inside and outside the currently allocated stack frame. This causes problems when an interrupt occurs during authentication and the interrupt handler pushes data onto the stack thus possibly overwriting live data. When live data is overwritten by the interrupt handler, the behavior of the security firmware is undefined upon return from interrupt.

This anomaly can manifest itself as a processor hang. The corruption of the stack when an interrupt is serviced during authentication can result in the Program Counter not being properly returned from the Lockbox SESR firmware. There is no evidence of this anomaly resulting in a security compromise. Issuing reset may be required to recover from this anomalous behavior.

WORKAROUND:
There are several workarounds possible:

1) Interrupt handlers that occur during authentication should wrap themselves with the following two instructions:

```
/* Start of original interrupt handler */
SP += -60;
<body of interrupt handler>
SP += 60;
RTI;
/* End of original interrupt handler */
```

2) When using the Device Driver/System Services Libraries (DD/SS) distributed with VisualDSP++, the DD/SS libraries will have to be rebuilt from source with the fix presented above implemented. For example, to implement the fix in the version of the DD/SS distributed with VisualDSP++ 5.0 update 3, the following steps must be taken:

   a) In /Blackfin/lib/src/services/int/adi_int_module.h, add the *SP += -60;* \ instruction immediately after the __STARTFUNC(Name) entry point for both ADI_INT_ISR_FUNCTION and ADI_INT_EXC_FUNCTION. For example:

```
#define ADI_INT_ISR_FUNCTION(Name,IVG,Nested) \
   __STARTFUNC(Name) \
      SP += -60; \                          // <-- ADD THIS INSTRUCTION
      [--SP] = R0; \
      [--SP] = P1; \
```

   In the same file, increase the length of the adi_int_ISR_Entry array from 16 to 18.

   b) In /Blackfin/lib/src/services/int/adi_int_asm.asm, the complementary stack modification must be made to the end of the handlers. The *SP += 60;* \ instruction must be inserted before the RTI in ADI_INT_ISR_EPILOG and before the RTX in ADI_INT_EXC_EPILOG. For example:

```
#define ADI_INT_EXC_EPILOG(Nested) \
   unlink;\
   SP +=  12;\
      .
      .
      .
   SP += 60; \                              // <-- ADD THIS INSTRUCTION
   RTX;\
   NOP;\
   NOP;\
   NOP;
```

   c) Rebuild the DD/SS library with these changes and use them instead of the prebuilt libraries distributed with VisualDSP++.

3) When using the Device Driver/System Services Libraries (DD/SS) distributed with VisualDSP++, the DD/SS libraries will have to be rebuilt from source with the fix presented above implemented. For example, to implement the fix in the version of the DD/SS distributed with VisualDSP++ 5.0 update 8, the following steps must be taken:

   a) In /Blackfin/lib/src/services/int/adi_int_module.h, add the *SP += -60;* \ instruction immediately after the __STARTFUNC(Name)

entry point for both ADI_INT_ISR_FUNCTION and ADI_INT_EXC_FUNCTION. For example:

```
#define ADI_INT_ISR_FUNCTION(Name,IVG,Nested) \
   __STARTFUNC(Name) \
      SP += -60; \                          // <-- ADD THIS INSTRUCTION
      [--SP] = R0; \
      [--SP] = P1; \
```

b) In the same file, the complementary stack modification must be made to the end of the handlers. The *SP += 60;* \ instruction must be inserted before the RTI in ADI_INT_ISR_EPILOG and before the RTX in ADI_INT_EXC_EPILOG. For example:

```
#define ADI_INT_EXC_EPILOG(Nested) \
   unlink;\
   SP +=  12;\
      .
      .
      .
   SP += 60; \                              // <-- ADD THIS INSTRUCTION
   RTX;\
   NOP;\
   NOP;\
   NOP;
```

c) Rebuild the DD/SS library with these changes and use them instead of the prebuilt libraries distributed with VisualDSP++.

4) Do not enable interrupts to be serviced during authentication.

APPLIES TO REVISION(S):

0.1, 0.2

61. 05000434 - SW Breakpoints Ignored Upon Return From Lockbox Authentication:

DESCRIPTION:

Upon returning from a failed Lockbox authentication attempt, software breakpoints are not able to halt the debugger until after the fourth breakpoint is executed.

This anomaly occurs when Condition #1 AND Condition #2 OR Condition #3 are met:

1) The code initiating the authentication by executing instruction "RAISE 2;" is executing from L1 Code Cache memory configured as SRAM.

and

2) The failure from authentication is due to a message-digital signature-message size mismatch.

or

3) The failure from authentication is due to fact that the public key is not programmed and the firmware reads back all 0's.

Note that if the combination of Condition 1) and either Condition 2) or Condition 3) are not met, the anomaly will not be encountered.

WORKAROUND:

There are several workarounds possible:

1) Use a hardware breakpoint instead of a software breakpoint to break right after returning from authentication.
2) Do not initiate authentication from L1 Cache Code area of memory
3) Place multiple (at least four) NOPs after the return point of authentication and place a regular software breakpoint at each NOP. The first four will not execute as expected but the fifth (5th) breakpoint will trigger the debugger to halt. NOPs may be replaced with non-critical code if desired.
4) Do not link the calling routine that executes the instruction "RAISE 2;" that initiates authentication into L1 Cache Code space.

APPLIES TO REVISION(S):

0.1, 0.2

62. 05000435 - Certain SIC Registers are not Reset After Soft or Core Double Fault Reset:

DESCRIPTION:
SIC_IMASK1, SIC_IAR4, SIC_IAR5, SIC_IAR6, and SIC_IWR1 do not get reset to their default values after a soft or double fault reset. This may cause the boot kernel to hang, depending on the boot mode, if the following bits in SIC_IWR1 are been modified from their default value of b#1:

Interrupt Name	Bit Number	Boot Modes Affected
MDMA0	10	All
MDMA1	11	All
NFC status	16	NAND Flash
HOSTDP status	17	Host DMA
Host read done	18	Host DMA

Watchdog and hardware resets are not affected.

WORKAROUND:
1. If possible, do not change SIC_IWR1 bits affecting the boot mode(s) in use from their default settings.
2. This workaround only applies to rev 0.2 silicon: If it is necessary to change the bits in SIC_IWR1 that affect the boot mode(s) in use, the following workaround may be used before altering SIC_IWR1:
   a. Set the BCODE field of the SYSCR register to BCODE_NOBOOT so that the processor will begin executing out of L1 Instruction Memory after a software or double fault reset:

```
*pSYSCR |= BCODE_NOBOOT;
```

   b. Place the code below at the beginning of L1 instruction memory (0xFFA00000).

```
*pSIC_IWR1=0xFFFFFFFF;
asm ("raise 1;");
```

   The processor will then boot normally.

APPLIES TO REVISION(S):
0.1, 0.2

63. 05000439 - Preboot Cannot be Used to Alter the PLL_DIV Register:

DESCRIPTION:
The preboot routine must not be used to alter the value of the PLL_DIV register. Doing so may result in the processor ceasing to execute instructions.

WORKAROUND:
There are two possible workarounds for this issue.

1. The initcode feature described in the Hardware Reference Manual's *System Reset and Booting* chapter can be used to change PLL_DIV during boot time. The initcode must be located in L1 memory and must access PLL_DIV directly rather than using the *bfrom_SysControl()* routine.
2. This workaround only applies to rev 0.2 or higher silicon: The default SCLK and CCLK frequencies can still be changed with the preboot routine by modifying the default MSEL value in the PLL_CTL register. Program OTP_SET_PLL to '1' and OTP_PLL_CTL to the desired value in the PBS00L page. However, the field OTP_PLL_DIV must be programmed with the default PLL_DIV value of 0x0004. See the *System Reset and Booting* chapter in the Hardware Reference Manual for details on how to customize power management using the preboot routine.

APPLIES TO REVISION(S):
0.1, 0.2

64. 05000440 - bfrom_SysControl() Cannot be Used to Write the PLL_DIV Register:

DESCRIPTION:
The *bfrom_SysControl()* firmware function cannot be used to program the PLL_DIV register. Doing so may result in the processor ceasing to execute instructions.

WORKAROUND:
PLL_DIV must be changed by modifying the register directly rather than using *bfrom_SysControl()*.

APPLIES TO REVISION(S):
0.1, 0.2

65. 05000443 - IFLUSH Instruction at End of Hardware Loop Causes Infinite Stall:

DESCRIPTION:
If the IFLUSH instruction is placed on a loop end, the processor will stall indefinitely. For example, the following two code examples will never exit the loop:

```
P1 = 2;
LSETUP (LOOP1_S, LOOP1_E) LC1 = P1;
LOOP1_S: NOP;
LOOP1_E: IFLUSH[P0++];

LSETUP (LOOP2_S, LOOP2_E) LC1 = P1;
LOOP2_S: NOP; NOP; NOP; NOP;       // Any number of instructions...
LOOP2_E: IFLUSH[P0++];
```

WORKAROUND:
Do not place the IFLUSH instruction at the bottom of a hardware loop. If the IFLUSH is padded with any instruction at the bottom of the loop, the problem is avoided:

```
LSETUP (LOOP_S, LOOP_E) LC1 = P1;
LOOP_S: IFLUSH[P0++];
LOOP_E: NOP;                    // Pad the loop end
```

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

66. 05000445 - The WURESET Bit in the SYSCR Register is not Functional:

DESCRIPTION:
The WURESET bit in the SYSCR register is not functional and always reads back as b#0. This prevents the boot kernel from detecting a wake from hibernate condition and performing a quickboot.

WORKAROUND:
The SCKELOW bit in VR_CTL can be used to determine if the processor has come out of hibernate.

If quickboot is required, initcode can be used to detect a wake from hibernate and to initiate a quickboot by setting the BFLAG_WAKEUP bit in the *ADI_BOOT_DATA* structure. If a wake from hibernate is not detected, simply return to the boot kernel. The following pseudo-code can be used:

```
Read VR_CTL Register;
Check SCKE_LOW Bit;
if b#1
   Set BFLAG_WAKEUP in ADI_BOOT_DATA;
Return to boot kernel;
```

APPLIES TO REVISION(S):
0.1, 0.2

67. 05000450 - USB DMA Short Packet Data Corruption:

DESCRIPTION:
When DMA mode 0 is used for the entire transfer, the short packet can be corrupted if double buffering is enabled for the TX Endpoint FIFO.

When DMA mode 1 is used, large transfers generate a single interrupt at the end of the entire transfer. The transfer itself is split up into packets with a length specified in the Maximum Packet Size field for that endpoint. If the transfer size is not an integer multiple of the Maximum Packet Size, a short packet will be present at the end of the transfer, thus making it susceptible to the same short packet corruption.

WORKAROUND:
Use DMA mode 1 to transfer (n* Maximum Packet Size) and schedule DMA mode 0 to transfer the short packet.

For example, if the transfer size is 33168 bytes and the Maximum Packet Size is 512, schedule [33168 - (33168 % 512)] in DMA mode 1 and the remainder (33168 % 512) in DMA mode 0.

When using only DMA mode 0 for the entire transfer, there are two workarounds:
1) Disable double buffering.
2) Ensure that the short packet's length is a multiple of 4 bytes.

APPLIES TO REVISION(S):
0.1, 0.2

68. 05000451 - BCODE_QUICKBOOT, BCODE_ALLBOOT, and BCODE_FULLBOOT Settings in SYSCR Register Not Functional:

DESCRIPTION:
The BCODE_QUICKBOOT, BCODE_ALLBOOT, and BCODE_FULLBOOT settings in the BCODE field of the SYSCR register are not functional.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1, 0.2

69. 05000452 - Incorrect Default Hysteresis Setting for $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, and BMODE Signals:

DESCRIPTION:
The setting b#00 for the NMI_RST_BMODE_SE field in the NONGPIO_HYSTERESIS register should enable hysteresis such that it is on by default for the $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, and BMODE signals. Instead, the setting b#00 incorrectly disables hysteresis for those signals.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1, 0.2

70. 05000456 - USB Receive Interrupt Is Not Generated in DMA Mode 1:

DESCRIPTION:
Whether the USB is used in host or device mode, the USB receive interrupt may not be generated when DMA Mode 1 is used.

For DMA Mode 1 host mode receive operations where the transfer size is an integer multiple of MaxPacketSize, extra "in" tokens are sent out by the USB controller at the end of the DMA transfer. This causes the slave device to send an additional data packet back to the Blackfin processor, where it is received in the USB FIFO, but no USB RX interrupt is generated. Taking the Mass Storage Class as a specific example, this causes the devices to send a status packet, which should generate a USB RX interrupt, however, this interrupt may be lost.

For DMA Mode 1 device mode receive operations where the transfer size is unknown, the Short Packet Interrupt must be relied upon to indicate the end of the transfer. However, this anomaly prevents the USB controller from issuing an RX interrupt for the corresponding endpoint when a short/null packet is received.

This anomaly does not apply to device mode when the size of the receive transfer is known in advance, as the DMA Completion Interrupt is generated at the end of the transfer and the endpoint receive interrupt is not used.

This anomaly also does not apply to transmit operations.

WORKAROUND:
In all affected cases described above, use DMA Mode 0 instead of DMA Mode 1.

APPLIES TO REVISION(S):
0.1, 0.2

71. 05000457 - Host DMA Port Responds to Certain Bus Activity Without HOST_CE Assertion:

DESCRIPTION:

The Host DMA Port (HOSTDP) responds to certain bus activity even without the assertion of its chip enable, HOST_CE. If the HOSTDP is the only slave on the bus (meaning that HOST_WR and HOST_RD are asserted by the host only for communicating with the HOSTDP), this anomaly will not be observed. There are two states in which the HOSTDP responds to bus activity (assertion of HOST_WR or HOST_RD) without HOST_CE being asserted:

1) HOSTDP is Enabled and Waiting for the Host to Configure - While waiting for configuration in both acknowledge and interrupt modes, assertions of HOST_WR for other slaves can cause the HOSTDP to be erroneously configured.

2) HOSTDP is Configured for Data Reads - While waiting for data reads in both acknowledge and interrupt modes, assertions of HOST_RD for other slaves can cause the HOSTDP to subsequently return data out of order.

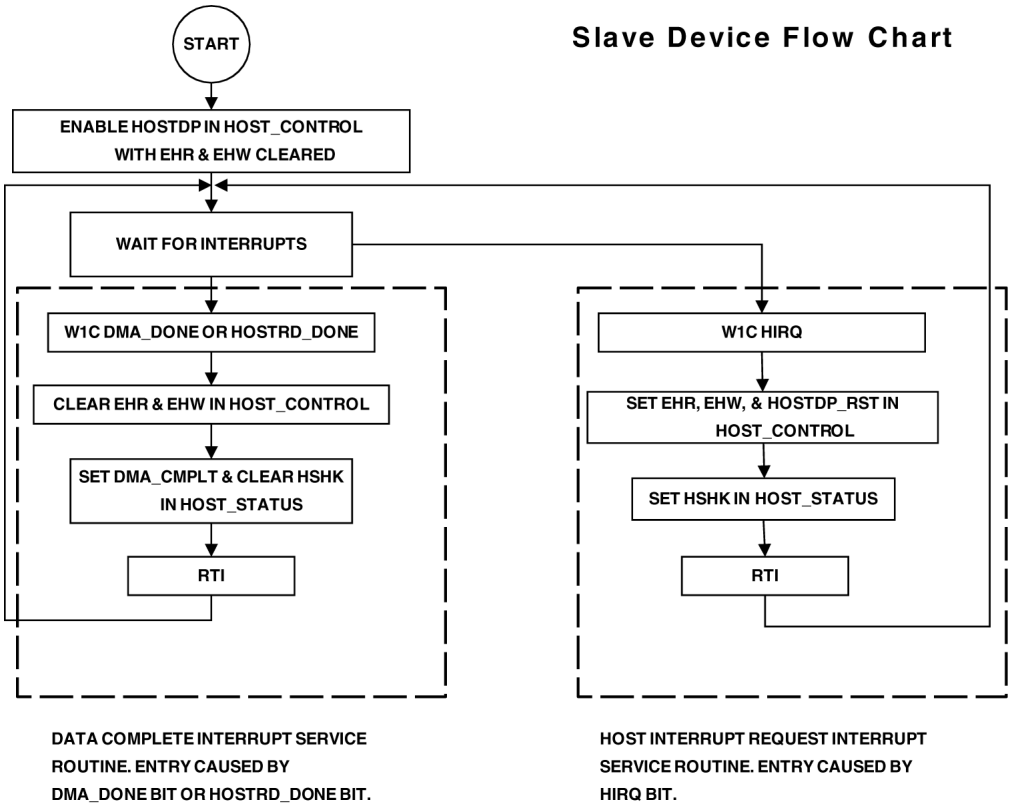
WORKAROUND:

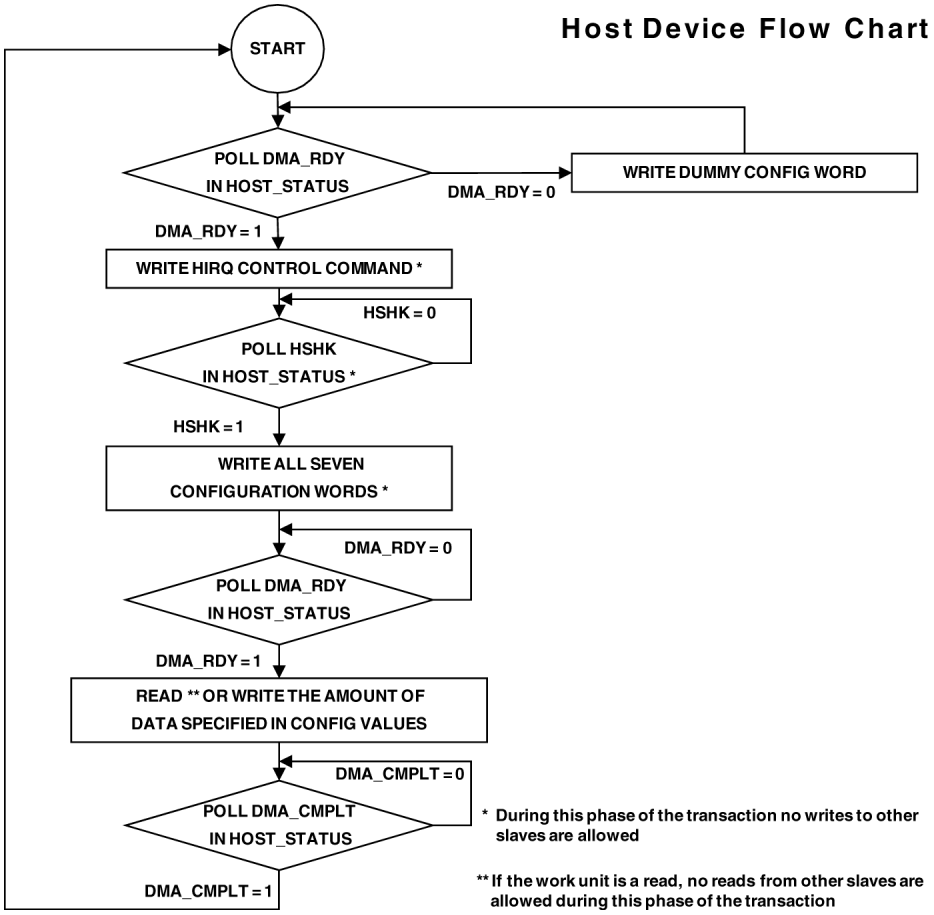
The following workarounds can be used for state #1:

1) Add additional logic - Connect to HOST_WR through a 2-input OR gate where one input of the OR gate is the host's write signal and the other is the host's chip select.

2) Time bus accesses by the host processor - Do not write to other slaves on the bus between the time that the HOSTDP is enabled and the DMA_RDY bit in HOST_STATUS is set.

3) Change the software on both the host and the slave as shown in the following flowcharts:





The following workarounds can be used for state #2:

1) Add additional logic - Connect to $\overline{\text{HOST\_RD}}$ through a 2-input OR gate where one input of the OR gate is the host's read signal and the other is the host's chip select.

2) Time bus accesses by the host processor - Do not read from other slaves on the bus between the time that the final configuration word (YMODIFY) is written to the HOSTDP and the DMA_CMPLT bit in HOST_STATUS is set.

APPLIES TO REVISION(S):
0.1, 0.2

72. 05000460 - USB DMA Mode 1 Failure When Multiple USB DMA Channels Are Concurrently Enabled:

DESCRIPTION:
When multiple USB DMA channels are enabled/active at the same time and any enabled channel uses DMA Mode 1, one of the channel's DMA Address registers may be corrupted, resulting in either a DMA hang or data corruption.

WORKAROUND:
Use DMA Mode 0 if the application requires multiple USB DMA channels to be concurrently enabled.

APPLIES TO REVISION(S):
0.1, 0.2

73. 05000461 - False Hardware Error when RETI Points to Invalid Memory:

DESCRIPTION:

When using CALL/JUMP instructions targeting memory that does not exist, a hardware error condition will be triggered. If interrupts are enabled, the Hardware Interrupt (IRQ5) will fire. Since the RETI register will have an invalid location in it, it must be changed before executing the RTI instruction, even if servicing a different interrupt. Consider the following sequence:

```
P2.L = LO (0xFFAFFFFC);  // Load Address in Illegal Memory to P2
P2.H = HI (0xFFAFFFFC);
CALL(P2);                // Call to Bad Address Generates Hardware Error IRQ5
....

IRQ5_code:               // Hardware Error Interrupt Routine
RAISE 14;                // (1)
RTI;                     // (2)

IRQ14_code:
[--SP] = ( R7:0, P5:0 ); // (3)
[--SP] = RETI;           // (4)
....
```

When the hardware error occurs, the program counter points to the invalid location 0xFFAFFFFC, which is loaded into the RETI register during the service of the IRQ5 hardware error event. When the RTI instruction (2) is executed, a fetch of the instruction pointed to by the RETI register, which is an illegal address, is requested before hardware sees the level 14 interrupt pending. This fetch causes another hardware error to be latched, even though this instruction is not executed. Execution will go to IRQ14 (3). As soon as interrupts are re-enabled (4), the pending hardware error will fire.

WORKAROUND:

1) Ensure that code doesn't jump to or call bad pointers.
2) Always set the RETI register when returning from a hardware error to something that will not cause a hardware error on the memory fetch.

APPLIES TO REVISION(S):

0.1, 0.2

74. 05000462 - Synchronization Problem at Startup May Cause SPORT Transmit Channels to Misalign:

DESCRIPTION:
When the SPORT is configured in multichannel mode with an external SPORT clock, a synchronization problem may occur when the SPORT is enabled. This synchronization issue manifests when the skew between the external SPORT clock and the Blackfin processor's internal System Clock (SCLK) causes the channel counters inside the SPORT to get out-of-sync. When this occurs, a "dead" channel is inserted at the beginning of the window, and the rest of the transmit channels are right-shifted one location throughout the active window. The last channel data will be sent as the first enabled transmit channel data in the second window after another "dead" channel is inserted. All data will be sent sequentially and in its entirety, but it is transmitted on the wrong channels with respect to the frame sync and will never recover.

WORKAROUND:
When this error occurs, the SPORT must be restarted and checked again for this error. The failure is extremely rare to begin with, so the probability of seeing consecutive restarts showing the failure is infinitesimally small.

A software solution is possible based on the timing of the SPORT interrupt. In the SPORT ISR, the CYCLES register can be set to zero the first time the interrupt occurs and then read back the second time the interrupt occurs. This will provide a time reference in core clocks for the frequency of the SPORT interrupt itself. If the value read the second time exceeds the duration of the multichannel window (in core clocks), then a "dead" channel was inserted into the stream, and the SPORT must be restarted.

Hardware workarounds are going to be heavily dependent on how the multichannel mode SPORT is configured. In multichannel mode, TFS functions as a Transmit Data Valid (TDV) signal and will always be driven to the active state (as governed by the LTFS bit in the SPORTx_TCR1 register) during transmit channels. Therefore, the TDV signal can be routed to one of the GPIO pins configured to generate an interrupt upon detection of the TDV pin changing states, based upon how the application configures the channels within the active frame, to detect the "dead" channel. If all the channels in the window are configured as transmit channels and there is no window offset and no multichannel frame delay, then TDV should go active as soon as the RFS pulse is received. If the period of the RFS pulse is exactly the window size (i.e., there are no extra clocks after the active window before the next RFS is detected), then TDV will remain active throughout operation. Therefore, if TDV goes inactive while the SPORT is on, the failure happened and the SPORT must be restarted and run again with this test in place until the failure is not detected.

For applications that have a window offset, a multichannel frame delay, extra clocks between the end of the active window and the next frame sync, and/or non-transmit channels inside the active window, the first TDV assertion would need to be tracked manually to detect the "dead" channel. One idea might be to do the following:

1) Connect TFS (TDV) to a GPIO interrupt and configure the interrupt to occur when TDV goes active.
2) Connect RFS to a GPIO interrupt and configure the interrupt to occur when RFS goes active.
3) Connect the SPORT receive clock to a TMRx pin configured in EXT_CLK mode.

When the GPIO interrupt for the active RFS pulse signifying the start of the window occurs, enable the Timer that is being used to track the SPORT receive clock. When the GPIO interrupt for the TDV signal transition occurs, check the TIMERx_COUNTER register to determine how many SPORT clocks have passed since the frame started. If it is one channel's worth over the expected value, the error occurred and the SPORT must be restarted and tested again. The GPIO interrupts should also be disabled if the startup condition is not detected.

APPLIES TO REVISION(S):
0.1, 0.2

75. 05000465 - USB Rx DMA Hang:

DESCRIPTION:
USB Rx DMA hangs if the endpoint FIFOs are configured in double buffer mode (this is the case when MaxPacketSize in USB_EP_NIx_RXMAXP is equal or less than half the endpoint FIFO size) When double buffering is enabled, there is the possibility of a race condition where RxPktRdy is set and cleared in the same cycle. When this happens RxPktRdy will remain cleared, thus preventing the USB DMA from unloading the FIFO, resulting in a Rx DMA hang.

WORKAROUND:
Use DMA mode 0 with double buffering disabled.

APPLIES TO REVISION(S):
0.1, 0.2

76. 05000466 - TxPktRdy Bit Not Set for Transmit Endpoint When Core and DMA Access USB Endpoint FIFOs Simultaneously:

DESCRIPTION:
TX DMA data can be lost when both the USB DMA and the core access two different USB endpoint FIFOs at the same time. The DMA pointer does not increment correctly for the last two bytes of the DMA accessed FIFO, thus preventing the USB controller from setting TXPKTRDY when AUTOSET is enabled. If TXPKTRDY is set manually, data will be sent on the bus but the last two bytes will be missing.

WORKAROUND:
Do not mix concurrent DMA and core accesses to the USB TX endpoint FIFOs.

APPLIES TO REVISION(S):
0.1, 0.2

77. 05000467 - Possible USB RX Data Corruption When Control & Data EP FIFOs are Accessed via the Core:

DESCRIPTION:
Received data may be corrupted if the RX FIFO is accessed via the core under the following conditions:

1) Control and data USB endpoints are enabled.
2) Data has been received at the control endpoint.
3) Data is being received or has been received at the data endpoint.
4) Core reads an ODD number of bytes from the control endpoint, EP0.
5) Subsequent core read of the data endpoint's RX FIFO will return corrupted data.

WORKAROUND:
Use DMA to read data from the streaming (data) endpoint FIFO.

APPLIES TO REVISION(S):
0.1, 0.2

78. 05000469 - PLL Latches Incorrect Settings During Reset:

DESCRIPTION:
The PLL can latch incorrect SSEL and CSEL values during reset when VDDINT is powered before VDDEXT. In this case, the PLL_DIV register will show the correct default value when read, but SSEL in the PLL will actually be set to 0xF and CSEL will be set to 0x3. This results in lower frequency CCLK and SCLK after reset than what the default values would imply.

WORKAROUND:
There are several possible workarounds:

1) Write PLL_DIV with a value other than the default. If the default value is desired, first write 0x003F and then 0x0004. Make sure to use an SSYNC after each write to PLL_DIV.
2) Issue an extra hardware reset of any duration.
3) Ensure that VDDEXT ramps up fully before turning on VDDINT.
4) Use the EXT_WAKE signal to enable the voltage regulator that powers VDDINT.
5) Use the internal voltage regulator.

APPLIES TO REVISION(S):
0.1, 0.2

79. 05000473 - Interrupted SPORT Receive Data Register Read Results In Underflow when SLEN > 15:

DESCRIPTION:
A SPORT receive underflow error can be erroneously triggered when the SPORT serial length is greater than 16 bits and an interrupt occurs as the access is initiated to the 32-bit SPORTx_RX register. Internally, two accesses are required to obtain the 32-bit data over the internal 16-bit Peripheral Access Bus, and the anomaly manifests when the first half of the access is initiated but the second is held off due to the interrupt. Application code vectors to service the interrupt and then issues the read of the SPORTx_RX register again when it subsequently resumes execution after the interrupt has been serviced. The previous read that was interrupted is still pending awaiting the second half of the 32-bit access, but the SPORT erroneously sends out two requests again. The first access completes the previous transaction, and the second access generates the underflow error, as it is now attempting to make a read when there is no new data present.

WORKAROUND:
The anomaly does not apply when using valid serial lengths up to 16 bits, so setting SLEN < 16 is one workaround.

When the length of the serial word is 17-32 bits (16 <= SLEN < 32), accesses to the SPORTx_RX register must not be interrupted, so interrupts must be disabled around the read. In C:

```
int temp_IMASK;

temp_IMASK = cli();
RX_Data = *pSPORT0_RX;
sti(temp_IMASK);
```

In assembly:

```
P0.H = HI(SPORT0_RX);
P0.L = LO(SPORT0_RX);

CLI R0;
R1 = [P0];
STI R0;
```

APPLIES TO REVISION(S):
0.1, 0.2

80. 05000475 - Possible Lockup Condition when Modifying PLL from External Memory:

DESCRIPTION:
Synchronization logic in the EBIU can get corrupted if PLL alterations are made by code that resides in external memory. When this occurs, an infinite stall will occur, and the part will need to be reset. The lockup is dependent on what the original ratio was, what the new ratio is, and other factors, thus making it impossible to specify any cases where this is safe.

WORKAROUND:
The CCLK::SCLK ratio should not be changed via the external interface, whether it's from asynchronous memory or SDRAM. Only make modifications to the PLL_CTL and PLL_DIV registers from code executing in on-chip memory.

APPLIES TO REVISION(S):
0.1, 0.2

81. 05000477 - TESTSET Instruction Cannot Be Interrupted:

DESCRIPTION:
When the TESTSET instruction gets interrupted, the write portion of the TESTSET may be stalled until after the interrupt is serviced. After the ISR completes, application code continues by reissuing the previously interrupted TESTSET instruction, but the pending write operation is completed prior to the new read of the TESTSET target data, which can lead to deadlock conditions.

For example, in a multi-threaded system that utilizes semaphores, thread A checks the availability of a semaphore using TESTSET. If this original TESTSET operation tested data with a low byte of zero (signifying that the semaphore is available), then the write portion of TESTSET sets the MSB of the low byte to 1 to lock the semaphore. When this anomaly occurs, the write doesn't happen until TESTSET is re-issued after the interrupt is serviced. Therefore, thread A writes the byte back out with the lock bit set and then immediately reads that value back, now erroneously indicating that the semaphore is locked. Provided the semaphore was actually still free when TESTSET was reissued, this means that the semaphore is now permanently locked because thread A thinks it was locked already, and any other threads that subsequently pend on the same semaphore are being locked out by thread A, which will now never release it.

WORKAROUND:
The TESTSET instruction must be made uninterruptible to avoid this condition:

```
CLI R0;
TESTSET(P0);
STI R0;
```

There is no workaround other than this, so events that cannot be made uninterruptible, such as an NMI or an Emulation event, will always be sensitive to this issue. Additionally, due to the need to disable interrupts, User Mode code cannot implement this workaround.

This workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

82. 05000481 - Reads of ITEST_COMMAND and ITEST_DATA Registers Cause Cache Corruption:

DESCRIPTION:
Reading the ITEST_COMMAND or ITEST_DATA registers will erroneously trigger a write to these registers in addition to reading the current contents of the register. The erroneous write does not update the read state of the register, however, the data written to the register is acquired from the most recent MMR write request, whether the most recent MMR write request was committed or speculatively executed. The bogus write can set either register to perform unwanted operations that could result in:

1) Corrupted instruction L1 memory and/or instruction TAG memory.
and/or
2) Garbled instruction fetch stream (stale data used in place of new fetch data).

WORKAROUND:
Never read ITEST_COMMAND or ITEST_DATA. The only exception to this strict workaround is in the case of performing the read atomically and immediately after a write to the same register. In this case, the erroneous write will still occur, but it will be with the exact same data as the intentional write that preceded it.

APPLIES TO REVISION(S):
0.1, 0.2

83. 05000483 - Possible USB Data Corruption When Multiple Endpoints Are Accessed by the Core:

DESCRIPTION:
When using core transfers to fetch data from endpoint FIFOs with multiple endpoints enabled, data corruption can occur when the core is reading data from one endpoint FIFO and a change in code flow occurs immediately prior to this read committing in the pipeline. If this code accesses a different endpoint FIFO, the core will read the data from the different endpoint FIFO; however, when the application resumes with the read access to the previous endpoint FIFO that did not commit, the read is corrupted.

Note that this change in code flow could be due to a different endpoint interrupt, or the read could be speculatively executed in the shadow of a conditional branch. Exceptions can also cause this problem, but only in the unusual case where an access to an endpoint FIFO takes place in the exception handler. The most likely scenario for this corruption to occur is when the core is reading data from one endpoint and gets interrupted to service a different endpoint.

WORKAROUND:
1) Use the USB DMA to read from the Endpoint FIFOs.
2) When multiple Endpoint FIFOs are enabled, disable interrupts around reads from an endpoint FIFO.
3) Ensure that Endpoint FIFO reads do not occur in the shadow of a conditional branch by placing three NOPs between the branch instruction and the read.

APPLIES TO REVISION(S):
0.1, 0.2

84. 05000485 - PLL_CTL Change Using bfrom_SysControl( ) Can Result in Processor Overclocking:

DESCRIPTION:
When bfrom_SysControl() is called with both the SYSCTRL_PLLCTL and the SYSCTRL_PLLDIV flags set in dActionFlags, and the new PLL_CTL value has either the PDWN or the STOPCK bit set, then MSEL gets updated in the PLL before CSEL/SSEL, which can lead to a situation where the processor is overclocked (depending on the new MSEL value).

WORKAROUND:
If setting either the PDWN or STOPCK bits in the new value of PLL_CTL passed into bfrom_SysControl(), the possible workarounds to avoid overclocking are:

1) Set the new value of MSEL equal to the old value of MSEL,
or
2) Do not set the SYSCTL_PLLDIV flag in the same call to bfrom_SysControl().

APPLIES TO REVISION(S):
0.1, 0.2

85. 05000487 - The CODEC Zero-Cross Detect Feature is not Functional:

DESCRIPTION:
The DAC zero-cross detect feature is not functional. Therefore, bit 7 of registers 2 and 3 should always be set to b#0.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1, 0.2

86. 05000490 - SPI Master Boot Can Fail Under Certain Conditions:

DESCRIPTION:
Master Mode SPI Booting can fail under certain combinations of SPI_BAUD, CCLK::SCLK ratio, boot block size, and SDRAM Refresh Rate. The root cause for this problem is described in anomaly 05-00-0501. This is the manifestation of that anomaly due to the boot ROM sequence, which does not incorporate the software workaround to the underlying hardware problem of a stuck RXS bit in the SPI_STAT register.

When the RXS bit gets stuck as a result of anomaly 05-00-0501, a subsequent re-enabling of the SPI port results in DMA requests to a FIFO that has not yet been populated. This causes bogus data retrieved at the end of the previous block to be interpreted by the boot ROM as an invalid block header for the next block, which causes the boot to abort. There are two places in the boot ROM where the device is susceptible to this, manifesting in one of three ways:

1) When a bootable image block size exceeds 64K, it is broken into multiple DMA work units. In the DMA handler invoked between the work units, the anomaly can be encountered.

2) When SPI_BAUD = 2, the maximum SPI baud rate of SCLK/4 aligns exactly with the boot ROM execution frequency, which allows for the SPI disable to align exactly with a word being received (as a result of the SPI's behavior to continue issuing clocks even after the RX DMA is completed). At this particular baud rate, the SPI issues exactly 40 clocks between when the DMA completes and when the SPI is disabled in the ROM. This equates to exactly 5 additional received bytes, which completely fills the 4-deep SPI RX FIFO and the shift register, which asserts the RXS bit as the SPI is shut down.

3) When system timing parameters allow for any single word to get transferred from the shift register to the SPI FIFO exactly as the SPI port is being shut down, the anomaly can theoretically be encountered, though it has not been observed on silicon or in simulations. All of the system timing parameters mentioned above must combine to cause the timing that triggers the anomaly.

WORKAROUND:
For case 1), do not allow block sizes over 64K.

For case 2), using any SPI_BAUD setting other than 2 avoids the timing required to encounter the anomaly.

For case 3), if the problem were to occur, the behavior would be consistent and repeatable. Changing any of the SPI_BAUD, CCLK::SCLK ratio, block sizes, and/or SDRAM refresh rate will alter the timing (as aligned to boot ROM execution) such that the problem can be avoided.

A workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, CrossCore Embedded Studio, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):
0.1, 0.2

87. 05000491 - Instruction Memory Stalls Can Cause IFLUSH to Fail:

DESCRIPTION:
When an instruction memory stall occurs when executing an IFLUSH instruction, the instruction may fail to invalidate a cache line. This could be a problem when replacing instructions in memory and could cause stale, incorrect instructions in cache to be executed rather than initiating a cache line fill.

WORKAROUND:
Instruction memory stalls must be avoided when executing an IFLUSH instruction. By placing the IFLUSH instruction in L1 memory, the prefetcher will not cause instruction cache misses that could cause memory stalls. In addition, padding the IFLUSH instruction with NOPs will ensure that subsequent IFLUSH instructions do not interfere with one another, and wrapping SSYNCs around it ensures that any fill/victim buffers are not busy. The recommended routine to perform an IFLUSH is:

```
SSYNC;          // Ensure all fill/victim buffers are not busy
LSETUP (LS, LE)
LS:  IFLUSH;
     NOP;
     NOP;
LE:  NOP;
SSYNC;          // Ensure all fill/victim buffers are not busy
```

Since this loop is four instructions long, the entire loop fits within one loop buffer, thereby turning off the prefetcher for the duration of the loop and guaranteeing that successive IFLUSH instructions do not interfere with each other.

APPLIES TO REVISION(S):
0.1, 0.2

88. 05000494 - EXCPT Instruction May Be Lost If NMI Happens Simultaneously:

DESCRIPTION:

A software exception raised by issuing the EXCPT instruction may be lost if an NMI event occurs simultaneous to execution of the EXCPT instruction. When this precise timing is met, the program sequencer believes it is going to service the EXCPT instruction and prepares to write the address of the next sequential instruction after the EXCPT instruction to the RETX register. However, the NMI event takes priority over the Exception event, and this address erroneously goes to the RETN register. As such, when the NMI event is serviced, program execution incorrectly resumes at the instruction after the EXCPT instruction rather than at the EXCPT instruction itself, so the software exception is lost and is not recoverable.

WORKAROUND:

Either do not use NMI or protect against this lost exception by forcing the exception to be continuously re-raised and verified in the exception handler itself. For example:

```
EXCPT 0;
JUMP -2;  // add this jump -2 after every EXCPT instruction
```

Then, in the exception handler code, read the EXCAUSE field of the SEQSTAT register to determine the cause of the exception.
If EXCAUSE < 16, the handler was invoked by execution of the EXCPT instruction, so the RETX register must then be modified to skip over the JUMP -2 that was inserted in the workaround code:

```
R2 = SEQSTAT;
R2 <<= 0x1A;
R2 >>= 0x1A;   // Mask Everything Except SEQSTAT[5:0] (EXCAUSE)
R1 = 0xF (Z);
CC = R2 <= R1; // Check for EXCAUSE < 16
IF !CC JUMP CONTINUE_EX_HANDLER;
R2 = RETX;
R2 += 2;       // Modify RETX to Point to Instruction After Inserted JUMP -2;
RETX = R2;
JUMP END_EX_HANDLER;

CONTINUE_EX_HANDLER: // Rest of Exception Handler Code Goes Here
   .
   .
   .
END_EX_HANDLER: RTX;
```

In this fashion, the JUMP -2 guarantees that the soft exception is re-raised when this anomaly occurs. When the NMI does not occur, the above exception handler will redirect the application code to resume after the JUMP -2 workaround code that re-raises the exception.

A workaround may be built into the development tool chain and/or into the operating system source code. For tool chains and operating systems supported by Analog Devices (VisualDSP++, VDK, the GNU Tool Chain, and the Linux kernel), please consult the "Silicon Anomaly Tools Support" help page in the applicable documentation and release notes for details.

For all other tool chains and operating systems, see the appropriate supporting documentation for details.

APPLIES TO REVISION(S):

0.1, 0.2

89. 05000498 - CNT_COMMAND Functionality Depends on CNT_IMASK Configuration:

DESCRIPTION:
The counter's "Zero Once" mode is only functional if at least one of the CZMZIE (Counter Zeroed by Zero Marker Interrupt Enable), CZMEIE (Zero Marker Error Interrupt), and/or CZMIE (CZM Pin Interrupt Enable/Push-Button Interrupt) bits in the CNT_IMASK register is set.

WORKAROUND:
If the counter is to be reset only on the first active level on the CZM pin, do all of the following:

1) Clear the ZMZC bit in the CNT_CONFIG register.
2) Set the W1ZMONCE bit in the CNT_COMMAND register.
3) Set the CZMZIE, CZMEIE, and/or CZMIE bits in the CNT_IMASK register.

As long as the SIC_IMASK register doesn't enable the counter interrupt, no action will be taken by the processor as a result of enabling one of these counter interrupts. If an alternate interrupt from the same counter is desired, software must ignore the extra interrupts resulting from the enable bit being set. To this regard, the CZMZIE interrupt is the most convenient to choose for this workaround.

Note that the Zero-marker-zeros-counter (ZMZC) mode is not affected by this anomaly.

APPLIES TO REVISION(S):
0.1, 0.2

90. 05000501 - RXS Bit in SPI_STAT May Become Stuck In RX DMA Modes:

DESCRIPTION:
When in SPI receive DMA modes, the RXS bit in SPI_STAT can get set and erroneously get stuck high if the SPI port is disabled as hardware is updating the status of the RXS bit. When in RX DMA mode, RXS will set as a word is transferred from the shift register to the internal FIFO, but it is then automatically cleared immediately by the hardware as DMA drains the FIFO. However, there is an internal 2 system clock (SCLK) latency for the status register to properly reflect this. If software disables the SPI port in exactly this window of time before RXS is cleared, the RXS bit doesn't get cleared and will remain set, even after the SPI is disabled. If the SPI port is subsequently re-enabled, the set RXS bit will cause one of two problems to occur:

1) If enabled in core RX mode, the SPI RX interrupt request will be raised immediately even though there is no new data in the SPI_RDBR register.
2) If enabled in RX DMA mode, DMA requests will be issued, which will cause the processor to DMA data from the SPI FIFO even though there is actually no new data present.

In master mode, the SPI will continue issuing clocks after RX DMA is completed until the SPI port is disabled. If any SPI word is received exactly as software disables the SPI port, the problem will occur.

In slave mode, the host would have to continue providing clocks and the chip-select for this possibility to occur.

WORKAROUND:
Reading the SPI_RDBR register while the SPI is disabled will clear the stuck RXS condition and not trigger any other activity. If using RX DMA mode, be sure to include this dummy read after the SPI port disable.

APPLIES TO REVISION(S):
0.1, 0.2

91. 05000503 - SPORT Sign-Extension May Not Work:

DESCRIPTION:
In multichannel receive mode, the SPORT sign-extension feature (RDTPYE=b#01 in SPORTx_RCR1) is not reliable for channel 0 data when configured for MSB-first data reception. This is regardless of any channel offset and/or multichannel frame delay.

WORKAROUND:
1) If possible, use receive bit order of LSB-first.
2) Do not use channel 0.
3) Ignore channel 0 data.
4) Use software to manually apply sign extension to the channel 0 data before processing.

APPLIES TO REVISION(S):
0.1, 0.2

92. 05000506 - Hardware Loop Can Underflow Under Specific Conditions:

DESCRIPTION:
When two consecutive hardware loops are separated by a single instruction, and the two hardware loops use the same loop registers, and the first loop contains a conditional jump to its loop bottom, the first hardware loop can underflow. For example:

```
P0 = 16;
LSETUP(loop_top1, loop_bottom1) LC0 = P0;
   loop_top1:    nop;
                 if CC JUMP loop_bottom1;
                 nop;
                 nop;
   loop_bottom1: nop;

nop;                          // Any single instruction

LSETUP(loop_top2, loop_bottom2) LC0 = P0;
   loop_top2:    nop;
   loop_bottom2: nop;
```

If a stall occurs on the instruction that is between the two loops, the top loop can decrement its loop count from 0 to 0xFFFFFFFF and continue looping with the incorrect loop count.

WORKAROUND:
There are several workarounds to this issue:

1) Do not use the same loop register set in consecutive hardware loops.
2) Ensure there is not exactly one instruction between consecutive hardware loops.
3) Ensure the first loop does not conditionally jump to its loop bottom.

APPLIES TO REVISION(S):
0.1, 0.2

93. 05000510 - USB Wakeup from Hibernate State Requires Re-Enumeration:

DESCRIPTION:
Applications may choose to place the processor in hibernate state when the USB suspend is seen on the bus. Logic was added at the PHY level to maintain the state of the USB data lines so that the processor will appear to still be connected to the host during the USB suspend state. This would prevent re-enumeration when the USB bus was subsequently resumed. However, when in hibernate, the USB controller also loses its logic state because power is removed. As a result, re-enumeration is unavoidable when coming out of the hibernate state.

WORKAROUND:
None

APPLIES TO REVISION(S):
0.1, 0.2

94. 05000511 - Lower 16 Bits of CNT_COUNTER Register Do Not Update Properly:

DESCRIPTION:
If the CNT_COUNTER register is read in the same SCLK cycle that the counter is being incremented by hardware, the MMR read may incorrectly return the previous value in the lower 16 bits. Under most circumstances, this is nearly negligible, as the value of the 32-bit read would be only one less than the correct value. However, in the case of a 16-bit overflow, the value read from the counter will be off by 0xffff from the expected value, as the upper 16 bits do get properly updated. For example, if the current counter value is 0x0000ffff, the next counter increment should result in CNT_COUNTER containing the value 0x00010000. When this anomaly manifests, the value read will incorrectly be 0x0001ffff.

WORKAROUND:
There is no workaround for this anomaly when it occurs any time other than when the counter wraps at 16 bits. For the case where the value in the lower 16 bits is 0xFFFF, a second read of the CNT_COUNTER register will ensure that the correct value is read.

APPLIES TO REVISION(S):
0.1, 0.2