a **Preliminary**

ADSP-2192-12 Anomaly List for Revision 0.0

June 13th, 2001

This document lists the anomalies known to be in the revision 0.0 ADSP-2192-12. These anomalies represent differences between this revision of silicon and the functionality specified in the preliminary ADSP-2192-12 data sheet dated October 2000 and the preliminary ADSP-219x/2192 Hardware Reference Manual dated June 2001.

Summary of Anomalies in ADSP-2192-12 rev. 0.0:
1) Status stack pops too early if too many stalls/dma in rti (db)
2) XTAL doesn't wake up on PCI I/O access
3) ROM code does not cleanly handle unexpected resets
4) PCI Slave Reads abort DSP core memory writes.
5) PCI Reads fail when used with PCI to PCI bridge chips.
6) The command "jump parse_command" is not located at 0x14000 as originally intended on both DSP0 and DSP1 ROM code
7) Type 32a instruction with DMA accesses
8) DMA Interrupts fail during I/O access

ANOMALIES

1) Status stack pops too early if too many stalls/dma in rti (db):
If it so happens that a DMA running in the background causes a number of stalls, while the core is in the process of executing the instructions in the delay slots of an RTI (db), the second instruction in the delayed branch may execute with the incorrect status conditions. This is because the Status stacks are popped too early.

Work Around: If this happens, manually pop and push status stack immediately prior to the RTI (db). This will cause the hardware interlocks in silicon to
e.g.,
```
POP STS;
PUSH STS.
RTI (db);
<instr>;
<instr>;
```

2) XTAL doesn't wake up on PCI I/O access
When the chip is powered down with the XTAL stopped, a PCI I/O access does not restart the XTAL and wake the chip as intended. (PCI memory space accesses do wake the chip, however.)

Work Around: (a) wake up with a memory write instead of an I/O access.
(b) Keep XTAL running by writing the CMSR:XON bit to a 1 before powering down.

3) ROM code does not cleanly handle unexpected resets
Resets in unexpected places can leave the ROM code command monitor without interrupts enabled and the timer values properly set.
Work Around: Program EEPROM patch code to modify reset routines to enable timer and interrupts for monitor code.

4) PCI Slave Reads abort DSP core memory writes.
A PCI read from DSP memory can cause a core write to memory to be aborted. The read completes normally but the write never occurs. This happens if the write is ongoing during the initiation of the PCI read.

Work Around: Use I/O space to access DSP memory from the PCI bus.

5) PCI Reads fail when used with PCI to PCI bridge chips
When the ADSP-2192 attempts to communicate with a host PCI interface via a PCI to PCI bridge chip, some PCI reads may cause infinite retries on the PCI bus. This occurs if the bridge chip performs speculative prefetch during host memory read operations; the access is disconnected by the chip, and the bridge does not restart the next read at the same address.

Work Around: There are three possible workarounds:

(a) Use I/O space to access DSP memory from the PCI bus.
(b) Use system without a PCI to PCI bridge chip.
(c) Program the bridge chip configuration registers to disable speculative prefetching from the chip.

6) The command "jump parse_command" was intended to be placed at 0x14F00 in both DSP0 and DSP1 ROM code. It was not.
.
Work Around: The location of the parse_command routines on the two cores are 0x14315 for DSP0 and 0x140BF for DSP1. The user needs to execute a "LJUMP 0x14315" or "LJUMP 0x140BF" to jump to the corresponding functions.

7) If a DMA process reads from DSP memory while a delayed or aborted Type 32a instruction is in the pipeline, the DMA will complete but either the DAG modify register or the DMA data will get corrupted.

Work Around: Use an equivalent series of DAG instructions instead of the Type 32a. It is also important, if using DMAs, to never allow the Type 32a instruction into the pipe at all. For this reason, code segments should be padded with at least 2 NOP instructions at the end, so that an adjacent data segment doesn't accidentally provide a Type 32a instruction.

8) DMA interrupts may not be latched while an I/O instruction is being executed. The DMA transaction will occur, but the interrupt generated when the count reaches zero will not be latched in IRPTL.

Work Around: Avoid simultaneous DMA and I/O instructions.

DOCUMENTATION NOTES
Known anomalies are intended to be fixed in the next tapeout.