



CAC User's Note

DSP32C R35/F35 Bugs

Two anomalies have been discovered in the new, F35 and R35 revisions of the Lucent DSP32C processors. One pertains to writing data to memory through the PIO port and the other involves acknowledging external interrupts.. Users should note that these problems only occur when the DSP32C is run at clock rates above 50 MHz and the occurrence of these problems is intermittent. These problems and the work-arounds to avoid them are discussed in this bug report

PIO Transfer Problem

DESCRIPTION: This problem involves missed PIO transfers from the PIO port to the DSP's memory bus. Data is stored in the PDR register and the PDF flag is set but a DMA transfer from the PIO port to memory is not initiated.

WORK-AROUND: The hardware for the VME6U6 and VME9U12 boards has been modified and the host libraries for the VME6U6 / VME9U12 and SB32C2 boards have been modified.

- The VME6U6 and VME9U12 VME interface was modified to ignore the DSP's PDF signal for PIO transfers. This avoids VME bus timeouts if the PIO port error occurs. This modification is included in versions 24 and higher of the FPGA configuration PROMs. Existing boards with older DSP32C revisions or running at 50 MHz do not require upgrading of the PROM.
- The host libraries for the VME6U6, VME9U12 and SB32C2 boards were modified to check the lower portion of the DSP's PAR register after the download operation is complete to ensure that all of the writes to the PDR register were properly transferred to memory. Should an error be detected, the entire download operation is retried until it is completed successfully or a maximum number of retries is exceeded. In the latter case, the

CAC DSP32C Bug Report

download function will return an error status with the global error variable set to EIO.

- The functions effected by this modification are listed below. The function names shown are for the VME6U6/VME9U12 library. The equivalent functions in the SB32C2 library are also effected. Their names begin with "**sbdsp_**" instead of "**fdsp_**".

fdsp_init_chip()	fdsp_run()	fdsp_halt()
fdsp_dl_i8b()	fdsp_dl_i16b()	fdsp_dl_i32b()
fdsp_dl_a8b()	fdsp_dl_a16b()	fdsp_dl_a32b()
fdsp_dl_fp()	fdsp_dl_ieee()	fdsp_dl_dma()
fdsp_dl_exec()		

The testing for the error involved added overhead for these functions, especially when the target of the function is the broadcast resource on VME6U6 or VME9U12 boards.

Systems using only 50 MHz VME6U6, VME9U12 and/or SB32C2 boards do not require the download checking and a future revision of the hardware for these boards may eliminate the need for this checking on 74 MHz boards. For systems in which the board types and revisions are known not to exhibit the problem, it may be desirable to eliminate the error checking. There are two methods by which installations may turn off the download error checking and remove the added overhead:

- 1) At runtime an application may turn off the error checking by setting the value of a global variable to 0. For VME6U6 or VME9U12 boards the variable is named ***dsp_check_dl_err***. For SB32C2 boards the variable is named ***sbdsp_check_dl_err***.
- 2) For installations in which only 50 MHz boards will ever be used the checking may be turned off permanently by first setting the value of the maximum retry macro to 0 and then recompiling the library and applications. This method also avoids the overhead of checking the global variable used in method 1.

For VME6U6 and VME9U12 boards the macro is named **DSP_MAX_DL_RETRIES** and is found in the file ***\$/CAC/include/dsputil.h***. For SB32C2 boards the macro is named **SBDSP_MAX_DL_RETRIES** and is found in the file ***\$/CAC/include/sbdsputil.h***.

CAC DSP32C Bug Report

Interrupt Acknowledge Problem:

DESCRIPTION: This problem involves the DSP32C failing to acknowledge an external interrupt when it enters the interrupt handling routine. Without the external acknowledge signal the interrupt request signal to the DSP32C will remain active.

WORK-AROUND: There is no hardware or library work-around for this problem. Although the problem may go away with a future revision of the circuit board.

Avoiding the problem requires additional code in the DSP32C application's interrupt handling routines, as illustrated in the example below. The code involves testing the interrupt request flag at the beginning of the routine and, if still set, returning from the routine. The program will see the interrupt set again and the interrupt routine will be re-entered giving the DSP32C another chance to acknowledge the interrupt and clear the request. The sample code, below, tests the **ireq1_hi** flag which, if true, means that the interrupt request is not set.

```
/* routine to handle external int req 1 */
ireq1_handle()
{
    asm("ireq1_hand:");
    asm("nop"); /* delay to allow intack to clear intreq */

    /* test interrupt request state */
    asm("if (ireq1_hi) pcgoto intl_ok"); /* intreq clear ? */
    asm("nop"); /* pipeline delay */

    /* return if intreq not cleared */
    asm("ireturn");

    /* intreq was cleared so handle the interrupt */
    asm("intl_ok:");

    .
    .
    .

    /* normal handler return */
    asm("ireturn");
}
```