Engineer To Engineer Note EE-35

Notes on using Analog Devices' DSP, audio, & video components from the Computer Products Division
Phone: (800) ANALOG-D or (781) 461-3881, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com

ADSP-218x EZ-ICE Tips and Tricks

Last Modified: 4/30/98

Introduction:

This Engineer To Engineer note will discuss troubleshooting issues using the latest version of the ADSP-218x ez-ice. These troubleshooting tips are an addendum to what is included in chapter 6 of your ADSP-218x Family EZ-ICE User's Guide and Reference Manual. This information ***ONLY*** pertains to the single-board, windows based emulator, ***NOT*** the dual-board MS-DOS based emulator
Please note: the latest version of the 218x ezice software (version 2.0.0.2) is windows 95 only, since the software uses DLL's and does not support legacy windows 3.x INI files.

Finding your Software/Firmware revision:

The first thing to do is find out what version of the emulator software you are using. Currently, the latest version that is available is rev 2.0.0.2. You can find this information out by using the "help/about…" pulldown menu in the ez-ice, or by viewing the readme.txt file which is located in the directory with your ez ice software (the default directory is C:\EZICE8x.) If you are not using the latest version of the emulator software or are still experiencing problems with the emulator software, please contact Analog Devices Technical Support at 1-800-ANALOGD or 1-781-461-3672 for an update.

The latest version of the emulator firmware with release 2.0.0.2 is version V1.0i. You can find out your firmware version by running the emulator software and then by selecting the "help" then "about EZICE 218x" pulldown windows. Make sure that you have all of the "child" windows in the ez ice application closed to be able to view the firmware version. Both the emulator software and the firmware versions should be displayed here. To update your firmware, again run the ez-ice windows application making sure that all of the "child windows are closed, then select "options/update firmware".

How do I perform an IDMA boot sequence?

Many users may experience problems bringing up the emulator software to a slave 218x DSP processor configured in IDMA boot mode (BMODE=1 on 128 pin 218x processors or mode pins A=1 and C =1 on 100 pin 218x processors). The following is a workaround which will allow you to access your target processor using the ez-ice.

The first step that ***must*** be performed is to ***hold*** the target processor in reset while bringing up the emulator software. Continue holding the DSP in reset until the emulator software initializes itself and the DSP. At this point you can bring the target processor out of reset. (Also make sure that the host processor is ***not*** initiating any IDMA accesses yet.) Next, write the instruction "jump 0x0000" in the program disassembly window at PM memory location 0x0000. Then hit the F5 key or the "run" icon to begin program execution. At this point, you can now initiate an IDMA boot sequence from your host processor.

Emulating 100-pin processors:

Some customers may experience problems using the emulator with 100-pin 218x variants. First, make sure that the mode pins of the DSP are pulled high/low with a 1KΩ or 2KΩ resistor. Weaker resistor values may cause the processor to come up in an unknown state during emulation, due to reverse leakage current issues.

Issuing the chip reset command during emulation causes the DSP to perform a full chip reset, including a reset of its memory mode. Therefore, it is vital that the mode pins are set correctly ***prior*** to issuing a chip reset command from the emulator user interface.

If you are using a passive method of maintaining mode information (by tying the mode pins to a specific logic level) then it does not matter that the mode information is latched by an emulator reset. However, if you are using the /RESET pin as a method of setting the value of the mode pins, then you have to take into consideration the effects of an emulator reset.

One method of ensuring that the values located on the mode pins are those desired is to construct a circuit like the one shown in figure 1. The circuit in this figure forces the value located on the Mode A pin to logic low; regardless if it latched via the /RESET or the /ERESET pin. (Please refer to the appropriate DSP processor data sheet for the mode pin assignments for the specific processor in your design.)

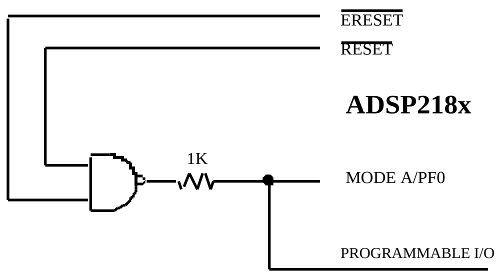

Figure 1: Active Mode Pin Configuration Example

Single Stepping Restrictions

Because of the operation of the emulator/processor interface, there are some limitations when performing single step executions in the emulator.

- Single stepping ignores the state of the target DSP's /BR and /RESET signals.
- Single stepping either of the last two instructions of a DO loop causes incorrect program sequencing. In this instance, the resulting "Warning Message 159 Unexpected Step Response" can be ignored.
- Single stepping influences the contents of the TCOUNT register. As you single step, the emulator makes a transition to and from User Space. On exiting User Space, the DSP's timer continues to run for a few additional cycles. As a result, the contents of the TCOUNT register "when single stepping" differ from the contents of that register "when running" your executable.
- If program execution during a single step does not reach a "step breakpoint" in time, a "step time-out" occurs. When this happens, the emulator forces an interrupt in order to return control to the user. This step time-out error doesn't always indicate a coding error or a hardware problem.

Single Stepping With "Hot" Interrupts

If your target has an unmasked pending interrupt, the DSP vectors to the corresponding interrupt service routine (ISR) on a single step. The instruction pointed to by the program counter (PC) prior to the ISR is pushed onto the top of the PC stack and is not executed until you return from the ISR.

Serial Port Autobuffering

When using an ez-ice emulator, there are several limitations on using autobuffering. These limitations relate to interrupting program execution (hitting a breakpoint or stopping the target processor during *Run* for example) while autobuffering.

- When program execution halts (and autobuffering is turned off), up to 12 additional autobuffer transfers occur. This causes up to 12 data locations to be overwritten with data received at a non-periodic and slower rate. This may result in invalid data received depending on the type of framing used (internal or external) or data being lost.
- When program execution resumes (autobuffering is turned on), the first 115 autobuffer transfers (at most) do not occur when expected.

You can avoid these additional non-periodic autobuffer transfers by refraining from interrupting program execution while autobuffering is enabled or by disabling autobuffering in your program when you must interrupt program execution.