ECE 320
Debugging and Troubleshooting

Daniel Sachs - February 28, 2002

1 Introduction

The Code Composer environment actually provides a rich debugging environment, which allows you to step through your code, set breakpoints, and examine registers as your code executes. This document provides a brief introduction to some of these debugging features.

2 Debugging Your Code

2.1 Affecting Program Flow

Breakpoints are points in the code where execution is stopped, and control of the DSP is returned to the debugger. Breakpoints can be activated or deactivated by double-clicking on any line of code in the disassembly window. (They can also be set by pressing F9 on a line in the source-file window. However, verify that the breakpoint appears at the corresponding location in the disassembly window if you do this; there have been problems with breakpoints being set inaccurately by this method in the past.)

You may also want to step through your program code. This can be done by choosing the "Step Into" or "Step Over" menu options from the "Debug" pull-down menu. (Unlike "Step Over," "Step Into" traces subroutine calls caused by "call" opcodes.)

The DSP which we are using (like most DSPs) is a pipelined processor which means that there are several stages to the execution of an instruction. Unfortunately, our debugger does not "flush" the pipeline of all current instructions so that when a program halts, the register values shown in the register and memory windows may not actually be the last values written. This is due to pipeline latencies in the processor.

The values shown often correspond to values written several cycles before the current instruction. If it is necessary to know the exact contents of the registers at any particular point in the program flow, simply put three or more "nop" (no operation) instructions after the instruction in question. These will flush the pipeline and allow the actual values in the register at that time to be viewed.

You can choose the "Run Free" option from the "Debug" pull-down menu to allow the your code to run freely, ignoring any breakpoints. The code will continue running until explicity halted with the "Halt" command.

Note that stopping and restarting execution sometimes confuses the A/D and D/A converter on the 6-channel surround-sound board. If this happens, the output will generally go to zero or become completely unrelated to the input signal. This can be fixed by simply resetting the DSP and starting your code from the beginning.

The bar on the left-hand side of the Code Composer Studio window contains shortcuts for many of the commands in the Debug menu.

Do this: Practice setting breakpoints in your program code and single-stepping by setting a breakpoint after the WAITDATA call and tracing through the program flow for several iterations of the FIR filter code. What code does the WAITDATA call correspond to in the disassembly window?

3 Troubleshooting

The DSP boards are sometimes somewhat temperamental. If there's no output, try the following (from less to more drastic):

- ☐ Use the Debug menu to halt and reset the DSP, verify that the PMST is set to 0xFFE0, reload the code, reset the DSP, and restart the code.
- ☐ Hit the "Reset" button on the DSP evaluation board, then use the Code Composer Studio menus to halt, reset the DSP, verify the PMST, reload, reset the DSP again, and restart your code.
- ☐ Close the Code Composer Studio, unplug the DSP board, then plug it back in. Restart the Code Composer Studio. You will need to reset the PMST to 0xFFE0, then reload, reset the DSP, and execute your code.

If you encounter problems that result in your code not loading correctly, close the Code Composer Studio and power cycle the DSP.

If problems persist after unplugging the DSP and plugging it back in, verify that the DSP board is functioning properly by executing known-good code. To do this, load and execute the file `v:\ece320\54x\dsplib\thru.out`; don't forget to set the PMST and to reset the DSP from Code Composer Studio menu. This code sends the input from channel 1 of the A/D to channel 1-3 of the D/A, and from channel 2 of the A/D to channel 4-6 of the D/A. (Since the code you've been given for this lab works properly, this should not be necessary.)

If you try all of these steps and the DSP board is still being balky, contact a TA to ask for assistance.