

LABORATORY VI

BAE 5413

SPRING 2007

TITLE: Compiling and debugging microcontroller code

OBJECTIVE: To survey techniques used in interfacing digital signals to computer parallel interfaces and understand the nature of logic signals.

REFERENCES: Texas Instruments eZ430-F2013 kit and documentation.

PROCEDURE:

Before starting the compiler, create a directory for project files under “My Documents” called IAR and a subdirectory under IAR called “Test1”

Open the IAR compiler and Create new project in the empty workspace using a C program template with a “main” program. Create a project file under the IAR/test1 directory called “test1_project”

Save the workspace as “test1_workspace

Enter the following program into main.c

```
/* Test program for BAE 5413  
This program is intended to allow compilation to be tested.  
*/  
  
int i,j;  
  
int main( void )  
{  
    i = 9;  
    j = 33;  
    j += i;  
  
    i = j;  
  
    return 0;  
}
```

Expand the Workspace tree to show all of the input and output files associated with the program. Highlight the “Test1_project” project file to allow setting options for the project. Right select Options or select project/Options for this project. Under General Options, select the Target microcontroller to be the MSP430F2013. (Do not change

options unless you have read thoroughly regarding the option.) Under Debugger, ascertain that the simulator is selected. Under Debugger/FET Debugger, set the connection to TI USB FET so that you may run this program on the USB device if you wish. (The debugger would need to be changed from Simulator to FET Debugger). There are no options that should need to be set on the simulator. Select “OK” to set these options and then save the project to save the options.

Select “Project Rebuild All” to compile, assemble, and link this program. You should observe any error status in the Messages window at the bottom of the screen.

Select “Project/Debug to begin running the simulator. You should see the “to be executed code highlighted and a disassembly window on the right hand side of the screen. The disassembly lists the machine code and assembly code for the program (machine code in Hexidecimal) for this microcontroller. Note that the machine code listed is column 1, address, column 2, instruction code, and column 3 data if needed. The associated assembly language is listed further to the right. You will need to expand the window slightly to see this.

Select View/Watch from the menu and add “i” and “j” as expressions to the watch window.

Select “Debug/Step Over” to execute the first line of the program. You may alternatively use the mouse to select the step over icon or use function key F10. Step Over executes the first line and if the line contains a function, step over does not trace through the function. Use “step into” to trace through a function.

The watch window should update to display the new value of “i” which will be highlighted in red to indicate it is the variable in the watch that has just changed. Repeat the step for the next line. Note that in the assembly language, the constants are given in hexadecimal. You can use the right-click context menu in the watch window to change the format of the displayed value.

There is no need to trace the “return” statement and you may use the Debug/stop debugging to quit debugging. After stopping the debugger, use the context menu (right click) in the editor to set a breakpoint at the return statement. This will allow the code to run to the breakpoint without single stepping. (very useful for a long loop for example). Then start the debugger and select Debug “go” via icon or menu entry. You should see the same numerical results in the watch window as before.

Change the value that you set “j” to be 32764. Leave off the semicolon after the number and rebuild the program. Fix the program. Set the display format for “i” in the watch window to default and “j” to Hexidecimal. Single-step the program and observe the results. Why does this behavior occur?

Replace the contents of “main” with the following code and “single step” the program until it reaches “return”. What is the maximum value limit for an “int”?

```

for ( j =32764; j > 0; j++)
{
    i = j;
}

return 0;

```

Save All, and Save the Workspace and finally Close the workspace in preparation to loading a new workspace.

Create the following program and use the “step into” debugging function to examine the function of the program. Is the “i” in the main program a different “i” than in the function? How many assembly language instructions are required for an integer multiply?

```

/* Test program for BAE 5413
This program is intended to allow a function to be tested.
*/

int square(i)
{
    return(i*i);
}

int main( void )
{
    int i,j;

    for( j = 0 ; j < 10; j++)
    {
        i = square(j);
    }

    return i;
}

```

Run the “Flashing the LED” example in the “eZ430-F2013 Development Tool – User Guide” (SLAU176B). Create a document that describes exactly what each line of the “Flashing the LED” main program does. Measure (by watch) the frequency at which the LED flashes and modify the program to flash the LED at twice the frequency originally set in the program. Describe the modification to the program.