# **Microprocessors (662-133) NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Lab #3 –Memory, I/O, and Addressing

Description

This lab will introduce the student to the methods used to store and retrieve data from memory. Accessing peripherals through memory mapped I/O will be explained and practiced in the lab.

Learning objectives

1. Create the proper file structure needed by Eclipse.
2. Download an assembly language program into the processor.
3. Use the Nios II debugger to edit and view the program.
4. Use the stepping functions to debug a program.
5. Insert a “break” point to help debug a program.
6. Use the “suspend” function to debug a program.
7. Use the ldw and stw instructions to read and write to memory.
8. Use the ldw and stw instructions to read and write to I/O devices.

Procedure

Logon to the computer.

1. If you have not already created a folder to hold your projects (usually named 'workspace'), do so now, otherwise you may use the same folder. This may be on a USB flash drive for convenience. If you make it on your system's hard drive, remember:
2. Do not use any folders that have spaces in the names, as some of the Altera tools have issues with this. In particular, Program Files, or My Documents
   1. Use the .sopcinfo file that should already be in this folder.
3. Make sure the DE-1 board is turned on. Start the NIOS II EDS Software Build Tools
4. When EDS starts, it will ask you to choose a workspace. Select the folder you created in step 1. If you like, you can set this as the default, otherwise EDS will ask on every start.
5. Create a new Project by going to File->New-> Nios II Application and BSP from Template.
6. Set the SOPC Information File to the one located in your workspace folder. The CPU name will come up automatically to 'cpu'.
7. Now choose a name for the project. In this case use 'lab3'
8. The Default Location will place all your projects in a subfolder called 'Software'. If you don't wish to have all your labs in the same place, uncheck “Use default location” and change it to simply X:\workspace\lab3
9. For a project template, chose 'Blank Project'
10. Click Next
11. Leave all the settings on this page as default, and click Finish
12. The project will build and after a few seconds, you will see 'lab3' and 'lab3\_bsp' in the Project Explorer. If you see any errors pop up during this step, most likely it is because you used a folder name with a space in it, or you don't have write permissions to the folder.
13. Let’s start building the program:

Right-click on the folder 'lab3' in the Project Explorer and select New->Source File

Set Source file: to 'lab3.s' and change the template to <None>

Click Finish

A window may appear, indicating that the make files are being updated, you can check the option to 'do this in the background' from now on, before it closes. A new blank file is created, so paste the contents of the included 'lab3.s' into it.

1. The desired result for this exercise is to generate a Pulse Width Modulated Signal whose output voltage is determined by the slider switches. The signal will be able to be measured on pin 1 on the header strip JP2 with a voltmeter and or oscilloscope. Remember a PWM signal is a square wave whose duty cycle can be changed and thus results in an adjustable dc voltage.

To construct our PWM signal using the microprocessor, look at the PWM module in Figure 1. Notice that it has three addresses, the base address for the clock divider (the number entered here determines the output frequency of the PWM signal), the base address for the duty cycle (the number entered here determines the output voltage of the PWM signal), and the base address for the enable signal (a “0” turns off the signal, a “1” turns it on.

CPU

PWM

Control

Data

Address

Clock Divider

Duty Cycle

Enable

Figure 1

|  |  |  |
| --- | --- | --- |
| Base Address | Address Range | Peripheral |
| 0x10000030 | 0x10000030-0x1000003F | Slider Switches |
| 0x10000300 | 0x10000300-0x10000303 | PWM0 clock divider |
| 0x10000304 | 0x10000304-0x10000307 | PWM0 duty cycle |
| 0x10000308 | 0x10000308-0x1000030B | PWM0 enable |
| 0X10000310 | 0X10000310-0X10000313 | PWM1 clock divider |
| 0X10000314 | 0X10000314-0X10000317 | PWM1 duty cycle |
| 0X10000318 | 0X10000118-0X1000031B | PWM1 enable |

Table 1

Table 1 summarizes the addresses need for generating the PWM signal. Note there are two PWM signals hence the “0” and “1” designations.

Formulas for calculating the PWM signal parameters.

Calculating PWM clock dividers:

For example, Since the master clock runs at 50 MHz, if we put 49,999,999 in the clock divider register, the frequency of the PWM output will be exactly 1Hz. Since 50,000,000/50Mhz is 1. Calculate the value you should use for the clock divider to get a 20KHz frequency for PWM0 and make a note of it.

Calculating duty cycles:

By setting the duty cycle register, we are in effect choosing how many cycles of the master clock the output is 'on', so when it's 0, the output is always off, and when it's equal to the Clock Divider Register, it's always on.

Procedure:

At the beginning of the program, write an .equ statement for the base addresses of the PWM0 peripherals and slider switches. Choose a name you think is helpful to remember.

After the main: label, you will need to enter the value calculated for the clock divider into the clock divider address for the PWM peripheral. Then enable the PWM by writing a 1 to its enable address.

For writing to an I/O register:

1. Load the BASE address into a register using the movia instruction
2. Write to the I/O's memory address using stw, an alternative method would be to write just one .equ statement using the base address of PWM0 clock divider and them apply the appropriate offset of 0,4,8,16,etc...in the stw instruction to store the values into memory locations of the duty cycle and enable.

In the loop portion of the program need you to:

1. Read the slider switches and multiply by 4 (this is a constant for the equation).
2. Write the total value in step 1 above to the duty cycle address.

When setup correctly, the DC voltmeter should display a voltage according to the following equation based on the setting of the switches.

(if connecting directly to the board)

Given a desired output voltage of 2.90v, what should you set the slider switches to? (in decimal and binary) test your result.

When you are finished writing your program,

1. Save the file. Then go to Project → Build All. This may take a minute or two the first time. Any typos or errors you may have will be stated in the Console window, as well as highlighted in your program. Hovering your mouse cursor over the instruction will give a popup describing the error.
2. Run this program on the DE1 hardware. Make sure the board is on and the USB cable is connected.
3. Setup the debugger. Click the small downward arrow next to the debugging icon
4. Choose Debug Configurations
5. Select the Nios II Hardware option in the list on the left, then click on the New Launch Configuration icon just above. You can change the Name at the top to 'Lab3 Debug' or similar
6. On the first Project tab, change the project name to Lab3, the Project ELF file name: should be automatically filled out as something like X:\workspace\lab3\lab3.elf
7. Go to the Target Connection tab and click Refresh Connections once or twice on the far right. The USB-Blaster should show up on both lists. (You may have maximize your window to see this if the USB-blaster still doesn’t show up, close eclipse, turn your board off then on, then restart eclipse.
8. Go to the Debugger tab. You can see the option 'Stop on startup at: main' uncheck that box.
9. Click Apply to save changes, then go ahead and click Debug
10. After a few seconds, the project will be built and downloaded to the board.
11. You may see a warning about this kind of launch begin configured to open the Nios II Debug perspective when it suspends. Check the 'Remember my decision' checkbox and click YES

Observe the output on pin 2 of JP2 (upper right pin- ask instructor for jumper wire to make this connection) directly using an oscilloscope (observe proper safety practices connecting test equipment to a PC), logic analyzer, or multi-meter that can measure frequency. Change the switches and observe the duty cycle produced.

Further work:

* 1. Try multiplying the switches by a value of 1 instead of 4. Why might this be useful for targeting certain duty cycles? How is it less useful?
  2. What is the tradeoff between accuracy of output voltage and frequency when changing the clock divider?
  3. The unit has a second PWM (PWM1) try setting up both at the same time to run at difference frequencies and duty cycles.