ECE263

Lab 2

Spring 2013

Testing Bit Banging
w/ Logic Analyzer

Revised 3 Feb 2013

(intentionallyleft blank)

Object: In this lab you will design and code a simple assembly language program that will use two output ports and then you will analyze the operation of the program in real time usingthe Agilent 16801A Logic Analyzer.

Material: CodeWarrior software package
Dragon 12 Plus Development Board
P&E USB Multilink
CPU12 Reference Guide (CPU12RG)
Agilent 16801A Logic Analyzer

Background: In order to operate and do anything useful, computers must exchange data with external devices by means of I/O ports. Most processors accomplish that function using a technique known as memory mapped I/O. Rather than having a few special purpose ports and special input and output instructions, memory mapped I/O allows the designer to designate any location(s) in the entire address space to be used for I/O and to also use any instruction to send data to or get data from those locations. The thing that distinguishes an I/O location from normal memory is that the hardware connects each bit of these locations to an external pin of the integrated circuit. In the system, these I/O pins then get attached to external hardware so whenever any data is written to an I/O location, the corresponding bits of the I/O port will be driven either high or low. If the processor needs to get information from an external device, the external device would drive the I/O pins to the desired state (H/L) and then the processor would read the I/O location and thereby receive the information.

As an example, let's say you have a cash register that is being controlled by a microcontroller. The cash drawer of this device is opened by briefly energizing a solenoid which will trip the latch holding it closed. The buffer that will provide current to energize the solenoid is attached to an I/O pin of the microcontroller. Whenever the drawer is to be opened, the program running in the microcontroller would send a bit pattern to the I/O location that will activate the specific pin of the device which would in turn activate the buffer and solenoid. After the appropriate activation time, the program would then send another bit pattern to the I/O location to de-activate the pulse. This method of using the program to control I/O pins by writing explicit values to the desired I/O locations is known as "bit banging".

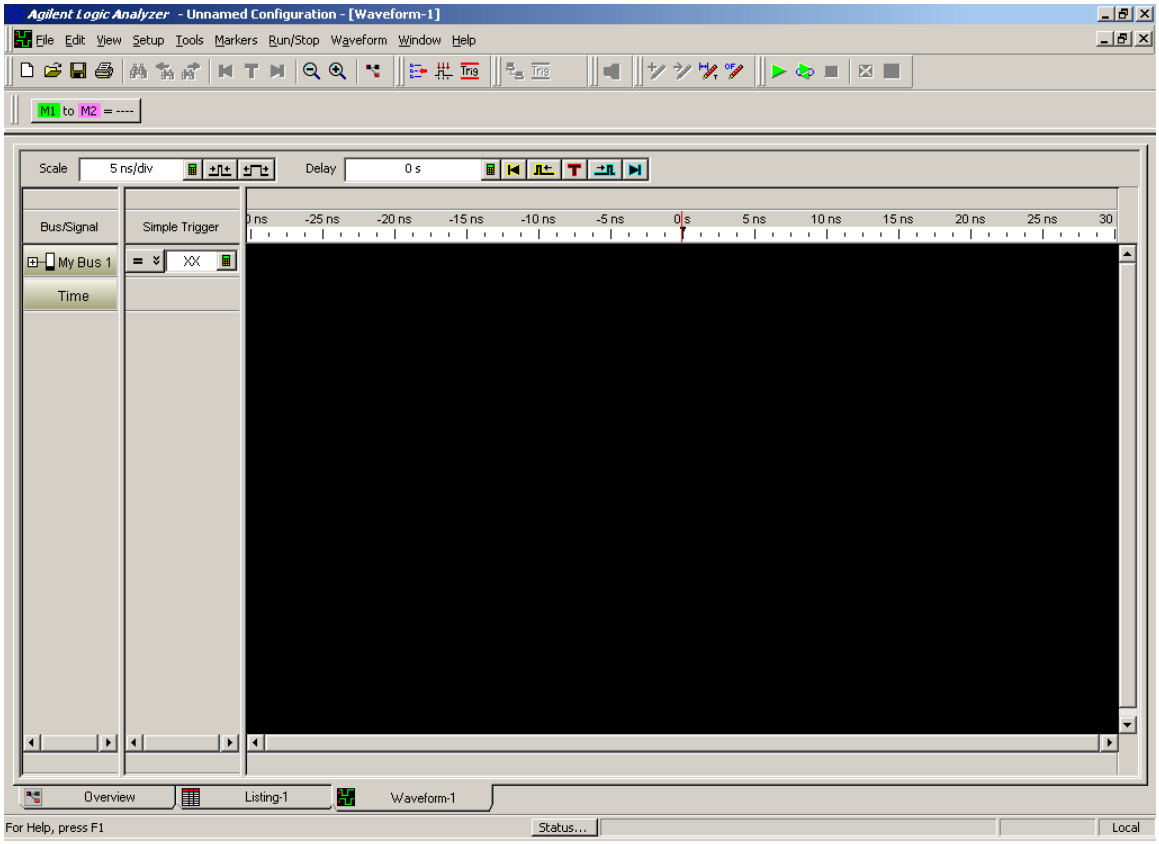
Preparation: This lab will use the program(s) described at the end of this write-up. For the pre-lab, generate a flow chart and/or pseudo code that shows the design of the program and create a text file of the assembly source code for the program.

Procedure: Working with your lab partner, complete the following steps

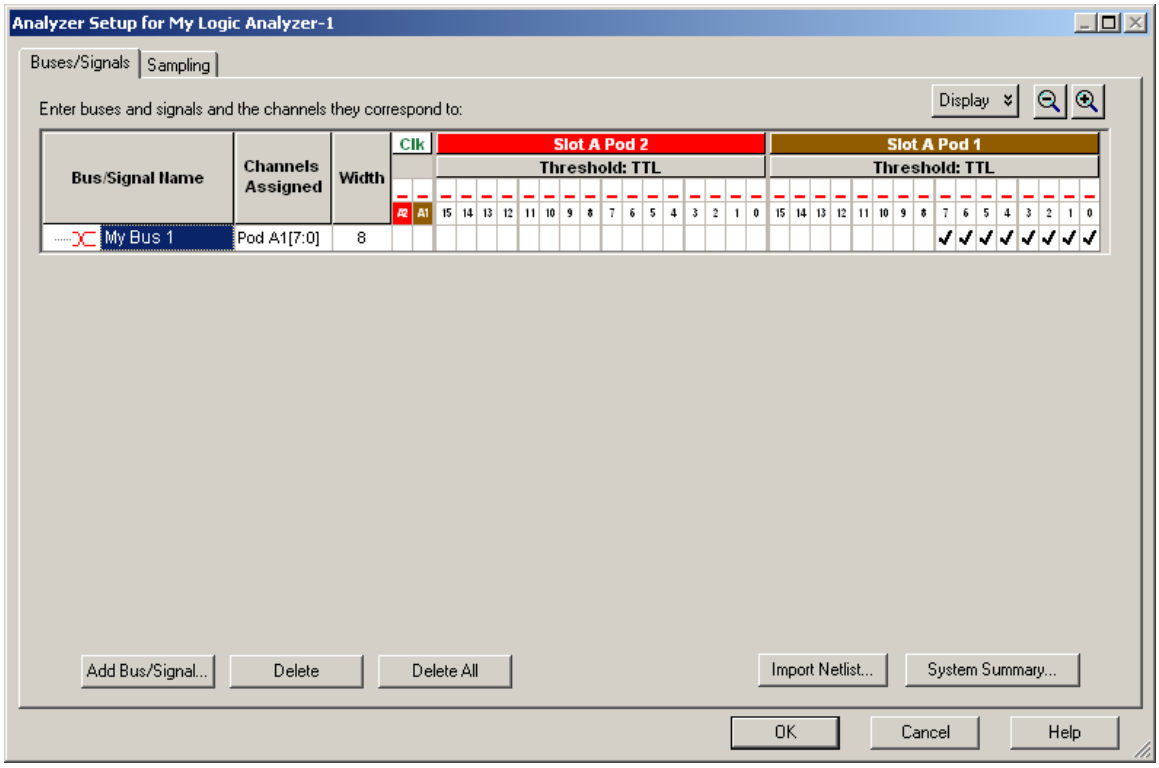
1. Using the procedures you learned in the previous labs, create a new project and enter your program as *main.asm*. Generate the executable object file using the *Make* facility and then download the program on the Dragon 12 Plus.
2. In the memory window, set the address to location $0000.
3. Open a second Memory window and set the address of this Memory window to $1000.
4. Now we are going to set up the Logic Analyzer. Get one "flying lead" pod from the box at the front of the room.

(Note: handle these pods carefully - we have just enough pods for the lab and they cost approx. $300 each to replace)

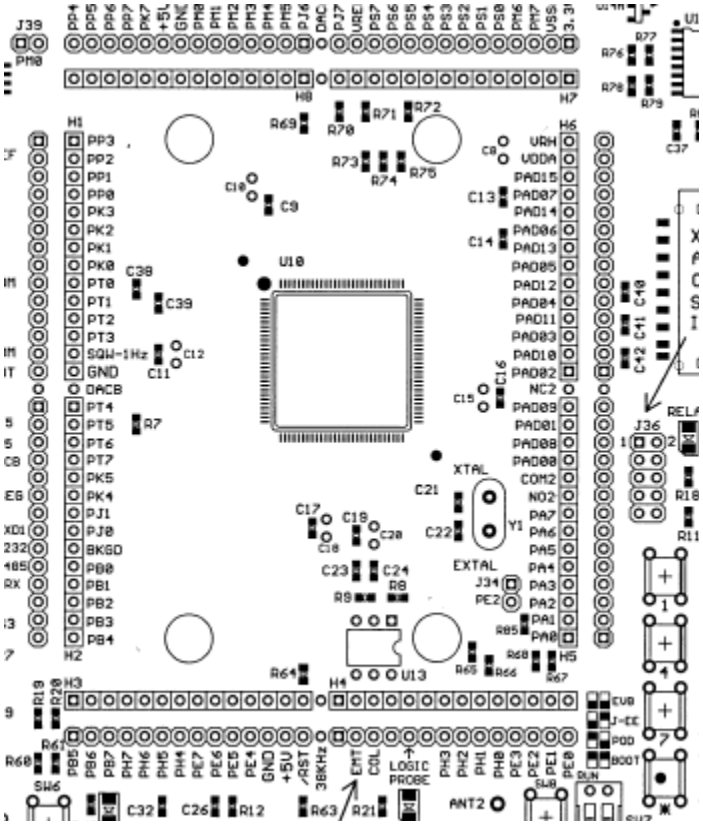
5. On the side of the logic Analyzer there are two high density ribbon cables. Attach the pod to the cable labeled **Pod 1**.
6. On the desktop double click on the Agilent Logic Analyzer icon. The following screen should now appear:



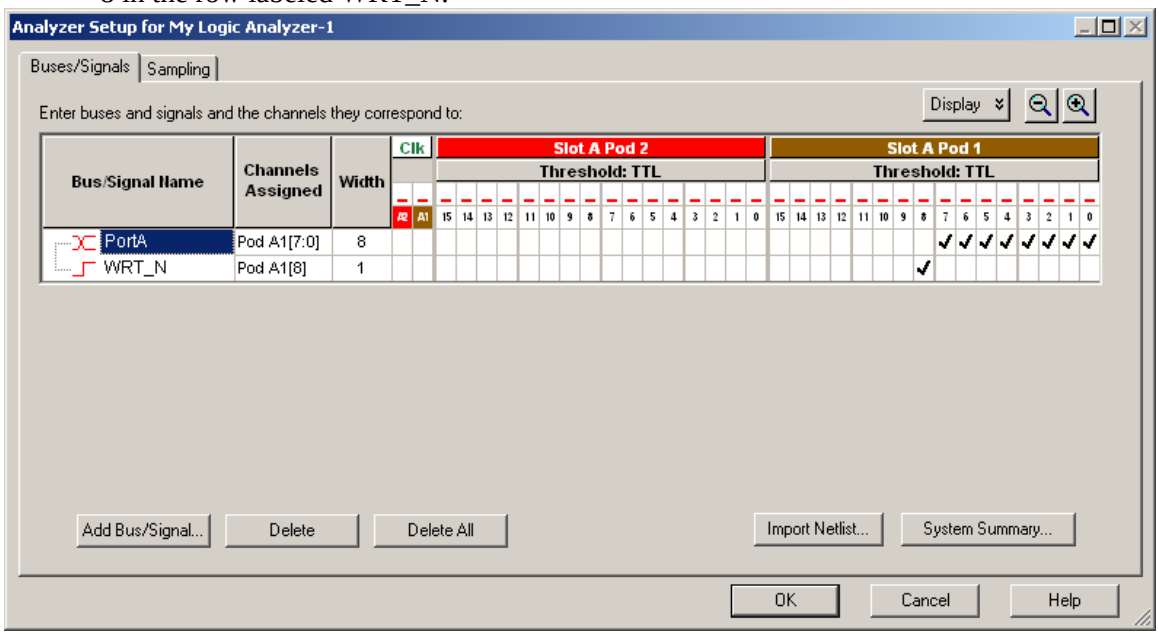
7. Click on the Setup tab at the top of this window. The follow pop-up screen will appear.



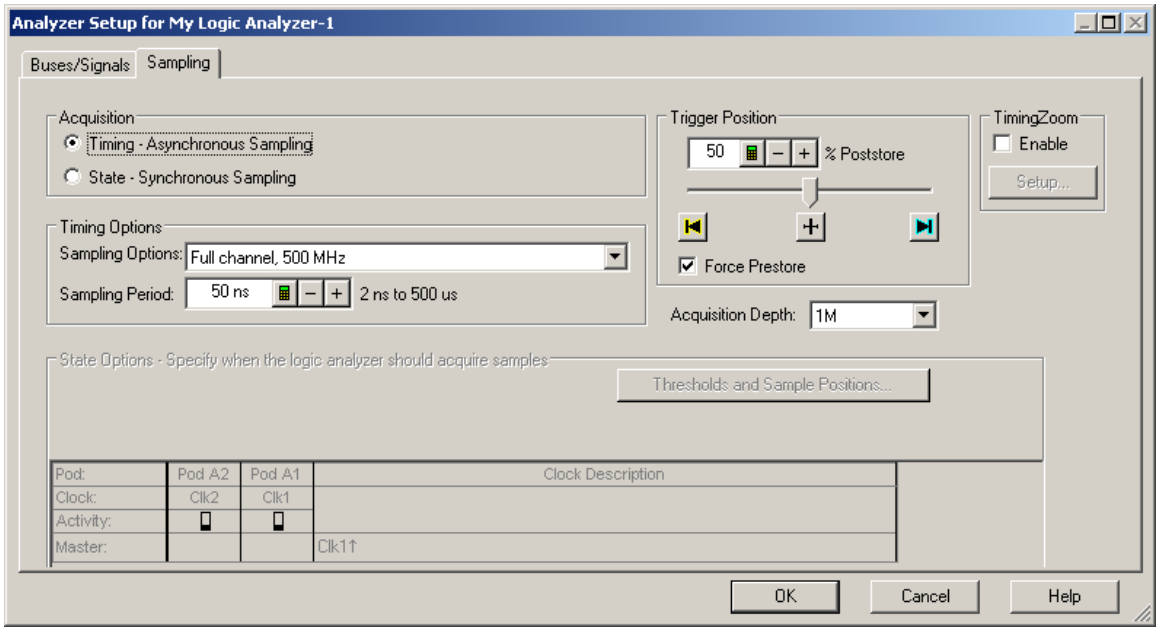
8. In the Bus/Signal Name column replace *MY BUS 1* with *PortA* and leave the check marks in the boxes under *Slot A Pod 1*.
9. Using the leads of the pod, attach the lead labeled 0 (left most as you read the label) to the pin labeled PA0 on the Dragon 12Plus – lower right on diagram below.



10. Continue placing lead 1 on PA1, lead 2 on PA2 through lead 7 on PA7.
11. Now add another signal by clicking on the Add Bus/Signal button on the bottom of the Analyzer Setup window. Enter WRT_N as the signal name and check the box in column 8 in the row labeled WRT_N.



12. Attach lead 8 of the pod to the pin labeled PB0 on the Dragon 12 Plus - lower left in the previous diagram.
13. Now follow the same procedure and add another signal labeled STB_N to pod 1 lead 9 and attach it to PB1.
14. Now click on the Sampling tab at top of the Analyzer Setup window and set it as follows:



15. Click OK to finish the setup.
16. The analyzer is set up so now start the debug process using the *True Time Simulator/Debugger*.

17. Start debugging Program 1 as you normally would by using single step and breakpoints as appropriate. When you think it is working correctly, you will use the logic analyzer to see if the data was sent to Port A correctly.
18. Verify the data in the two memory windows is what you expect.
19. We have to set up the trigger condition on the Logic Analyzer. Click on the + in front of PortA in the signal list to expand the selections. In the column labeled Simple Trigger on the main window, click on the box next to the Bus/Signal labeled PortA0. In the drop-down window select falling edge
20. On the analyzer window start a capture sequence by clicking on the Run/Stop tab at the top of the window and then clicking on RUN, or by clicking on the green arrow on the tool bar. The analyzer is now ready to capture data after it detects the trigger condition.
21. Back on the debugger window reset your program to the beginning and run it at full speed with no breakpoints.
22. If everything is working properly you should see a capture of all of the data you sent to the output port.
23. You should see a repetitive series of values on the trace labeled PORTA and they should be $00, $55, $AA and $FF. This is the data that your program just sent out.
    a. Did you receive them?
    b. Are there $100 sets of values?
    c. Is there any activity on the other two traces?
    d. How long is each value (time)?
    e. How long did it take to output all $100 values?
24. Print out a plot of the captured waveform for inclusion in your lab report.
25. Demonstrate the operation of Program 1 to the instructor or TA

26. Now compile and load Program 2 to the Dragon board.
27. Start debugging Program 2 as you normally would by using single step and breakpoints as appropriate. When you think it is working correctly, you will use the logic analyzer to verify the I/O signals.
28. Verify the data in the two memory windows is what you expect.
29. On the analyzer window, set the trigger condition to STB_N for the falling edge.
30. Back on the debugger window reset your program to the beginning and run it at full speed with no breakpoints.
31. You should see a series of low going pulses on the trace labeled STB_N and you should see a series of values on the bus labeled PORTA. Use the display controls on the analyzer window to zoom in and look at the data on PortA at each time STB_N goes low. The numbers should correspond to the data your program placed in memory (locations $1000 through $10FF).
    a. Do the traces look like the timing diagram described in the Program Spec?
    b. Are the data values correct?
    c. Are there $100 sets of values?
    d. How long are the WRT_N pulses?
    e. How long are the STB_N pulses?
    f. What is the time between successive WRT_N pulses?
    g. How long did it take to output all $100 values?
32. Print out a screen shot of the entire sequence and another screen shot when zoomed into one or two STB_N in the low state. Include these prints in the lab report.
33. Demonstrate the operation of Program 2 to the instructor or TA.

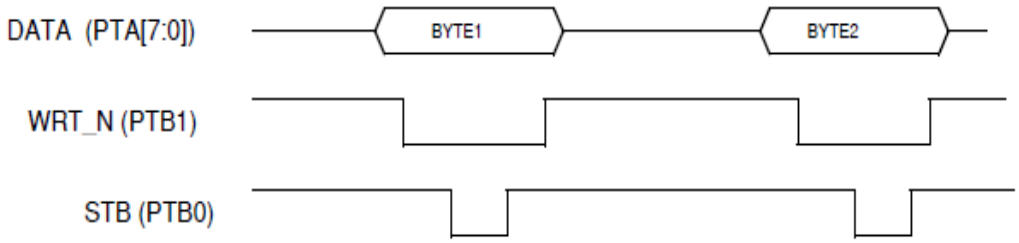
Lab 2 Programming Assignments:

For this lab you will write two simple assembly code programs. For the first program you will send data patterns to Port A (PTA) located at memory address $0000 and then examine the results using the logic analyzer. You will have to write the value $FF to the data direction register of Port A (DDRA) at memory address $0002 in order to designate that the bits of Port A will be used for output. The bit patterns to be written are $00, $55, $AA, and $FF. Write the four bytes exactly $100 times. Use the following template for your program. Your code will be in the section labeled *code*. The last line of your program should be a branch always to itself so that the program does no keep going executing garbage.

```
; add project name
; your name(s), group #, lab number, date


; export symbols
            XDEF Entry, _Startup     ; export 'Entry' symbol
            ABSENTRY Entry           ; for absolute assembly: mark this as entry point

;***************************************************************
; definitions

RAMStart:  EQU  $1000            ; absolute address of the start of RAM
ROMStart:  EQU  $C000            ; absolute address to place code/constant data

;***************************************************************
; variable section

            ORG  RAMStart

;     ADD YOUR VARIABLES HERE (if any)

;***************************************************************
; code section
            ORG   ROMStart
Entry:
_Startup:
                  |
 ADD YOUR PROGAM CODE HERE
                  |

;***************************************************************
;*            Interrupt Vectors                *
;***************************************************************
            ORG      $FFFE
            DC.W     Entry        ; Reset Vector
```

For the second program, you will simulate sending data to a printer. You will get the data to be printed from a block of RAM locations and you will generate the required printer control signals using bit banging. The data for the printer will be sent out using PTA and the printer control signals will be sent using Port B (PTB). There will be two “active low” control signals: WRT_N,

and STB_N attached to PTB1 and PTB0 respectively. The timing diagram below illustrates the relationship between the various signals that you must generate. For example, in order to initialize WRT_N and STB_N to the high state, you would write a %xxxxxx11 to PTB, then to start the low pulse on WRT_N you would write a %xxxxxx01 to make it go low and keep STB_N high.



Your program will first initialize the locations from $1000 to $10FF with a data pattern of your choice (e.g incrementing, decrementing, random). You also need to initialize both DDRA and DDRB to $FF to designate both ports as output. You will then send out the data from $2000 to $20FF as described above.

Memory Map:

Register	Memory Address
PTA	$0000
PTB	$0001
DDRA	$0002
DDRB	$0003
RAM	$1000
RAM_END	$3FFF
FLASH	$C000