

ECE 3803: Microprocessor System Design
Term D, 2007
Laboratory 1 – CPU and EEPROM

Notes: Try to start as soon as the assignment is posted. **Read the entire lab assignment document before starting.** Plan for at least 15 hours of laboratory time. **You should work in groups of two.** Submit one lab report for your group.

Posted: Tuesday, 03/13/2007
Signoff: By 4:50pm Thursday, 03/29/2007. Early signoff earns extra credit.
Report: In class on Monday 04/02/2007. **Do not forget to attach the signoff sheet.**

Reading/reference material

- Textbooks: Wolf 2.1-2.3, 4.2, 4.3, 4.7; Furber Ch. 2, 3, 5, 8.1, 8.6, 8.7
- Component datasheets (see course website)
- Tutorials on programming the EEPROM and using the logic analyzer (see course website)

Concepts

- Microprocessor interface signals
- EEPROM, software image creation/programming
- Read bus cycle timing analysis
- Logic analyzer
- Debugging

In part 1 of the ECE 3803 labs, you will begin constructing a standalone embedded microprocessor system with the most basic setup required to run software. This requires a CPU, and some non-volatile memory to hold the program. You will initially demonstrate the system's proper operation using the logic analyzer and the IAR development environment. Eventually towards the end of the lab you will be demonstrating a very simple output signal which would flash an LED.

Problem statement

Your TA will supply you with an initial lab kit containing the following components:

Quantity	Description
1	Solderless prototyping board
1	AT91M42800A + DIP-adapter (CPU)
1	ARM-JTAG cable
1	Parallel cable
2	28C256 32k×8 EEPROM (Atmel)
1	Crystal, 32.768 KHz, 2-pin
1	Pushbutton
1	LED
1	Resistor, 150kΩ
1	Resistor, 510Ω
1	Resistor, 380Ω (PLL R)
1	Capacitor, 0.022μF (PLL C ₂)
1	Capacitor, 0.22μF (PLL C)
1	Capacitor, 10μF
10	Capacitor, 0.1μF
1	Diode

Note: These parts are relatively expensive and are expected to be returned at the end of the term. **Failure to return all the parts at the end of the term will result in not receiving a lab grade.**

You are to build a basic microprocessor system capable of running software from non-volatile memory. You may use components other than the ones provided at your own expense, but within reason. Proper operation of your system must be initially demonstrated by looking at the bus cycles on the logic analyzer.

Step 1. To get started you will need to install the CPU module on the breadboard. Note that the CPU requires two power supply voltage levels, VDDCORE = +3.3V and VDDIO = +5V, connected to the two corresponding pins on the CPU module. It is very important to make sure that the correct voltage is connected to the correct pin; otherwise you will be running the chance of frying the CPU. Naturally, all the ground pins need to be connected to the common ground of the entire board. The crystal should be connected to pins XIN and XOUT; keep in mind that the crystal has to be directly connected to the pins. The two signals PLLA and PLLB will be initially connected to ground, but this will change in the later steps of the lab. Finally you need to have a reset button; this will be a simple circuit which will later be modified in Lab2. For the purposes of this lab, it should suffice to build the circuit shown in Figure 1.

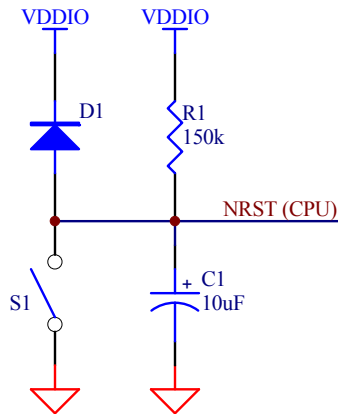


Figure 1. Reset circuit.

Step 2. Now you can verify that the processor is connected properly. You can do this by first observing the clock signal on the pin MCKO using the oscilloscope. You should see a clean 32.768 KHz square wave. In addition, you can verify the operation of the processor by observing the bus. Although nothing is connected to the CPU bus, you should be able to observe the instruction fetch bus cycles immediately after reset on the logic analyzer. Capture digital images of the oscilloscope and the logic analyzer outputs to include in your report.

Step 3. Install your EEPROM. You will need a 0.1 μ F **bypass capacitor** connected between the VCC (+5V) and GND pins of this device (and every other IC in your system). Also, to ensure clean signals in your system, try to make GND a low-inductance path from any point on your board. In a PCB you would use a ground plane, but in your circuit, connect the nearest neighbor GND rails together in several places evenly distributed around your board.

To test the EEPROM, you will need to write a very simple piece of software. The simplest such software will contain an infinite loop located directly at the reset address of the CPU:

```
org      0x00000000
here    B    here ; infinite loop
```

Now, since you still have not used an assembler, you will need to manually assemble this program. This will translate into four bytes of ARM code, which you will have to program to the EEPROM.

Ask the TA to demonstrate to you how to properly program the EEPROM. You should be able to observe the infinite loop running on the logic analyzer. Again, capture digital images for the logic analyzer output.

Step 4. So far you have been running your system at a low clock frequency of 32.768 KHz, but any realistic application requires a higher clock frequency. We will be aiming to use a 10 MHz clock throughout the rest of the labs. To achieve this, you need to use a PLL, which requires an R-C filter. The proper resistor and capacitor values for the **PLL A** filter at 10 MHz have been provided. Refer to the CPU data sheet (page 57) for a PLL filter wiring diagram. After you

connect the capacitors and resistors you will **not** be able to verify the new frequency until you program the PLL in software.

Step 5. Now you can step into the more exciting world of debugging and C programming. You will need to download the project “3803_Lab.zip” from the course website. The project already contains the required code to set the new clock frequency in addition to a number of startup features needed before you can write C code. The project also contains an infinite loop program written in C.

You will need to connect the PC to your CPU module using the parallel port cable and the ARM-JTAG cable. Now, using the development environment, you should be able to download the software onto your board. This marks the end of having to use the EEPROM programmer. Finally, using the oscilloscope, verify that the CPU is outputting the new clock frequency. In addition, using the logic analyzer, verify that the infinite loop program works.

Step 6. If you have done your steps properly you will notice that the clock coming out of the MCKO pin is actually at 11 MHz rather than 10 MHz. You will need to figure out how to change this to 10 MHz, as desired. Refer to the CPU data sheet (chapter 12) and the automatic PLL calculation spreadsheet found on the course website. Modify the startup code to set the correct frequency and verify that the clock is indeed at 10 MHz; again, take digital images.

Step 7. The code we have provided inserts some wait states to match the timing constraints of the EEPROM. In our code we were a little bit too conservative and inserted 5 wait states. Find the lowest number of wait states that would still allow the EEPROM to function properly. Refer to the CPU datasheet (chapter 11). Modify the startup code accordingly and verify the change. As always, digital images are your proof of correctness. Document your bus timing analysis in the lab report.

Step 8. At this point you have all the needed tools to write C code and directly download it into the EEPROM. The final step is to use the digital I/O (PIO) of the CPU to light an LED and toggle it in any pattern you desire. You can use the pin IRQ1 to connect your LED. Keep in mind that you need to connect the LED to the pin through a 510Ω resistor. This is vital; otherwise you might end up burning that particular pin. This pin IRQ1 is shared between the interrupt signal and the parallel I/O signal PA1. See the CPU data sheet (chapter 15) for information on how to assign this pin to PIO, configure it as an input or output, and transfer data to/from this pin. Configure the I/O pin and demonstrate the flashing LED to the TA.

Step 9. As an extra credit step we will be running a clock speed contest. You receive 2 points of extra credit for participating, which amounts to demonstrating the LED flashing software functioning at a CPU clock frequency higher than 20 MHz. To win, you need to achieve one of the three highest clock frequencies in the lab. The winner will get 3 extra points in addition to the 2 points he gets for participating. The next runner up will get 2 extra points, and finally the third runner up will get 1 extra point.

Laboratory Assignment

1. Draw a detailed schematic of your basic ARM-based system, including all pin numbers.
2. Wire, debug, and test each component of your hardware system. Do not attempt to program your EEPROM until the TA has shown you how to do it correctly.
3. Demonstrate the proper operation of your system to the TA, who will fill in and sign the signoff sheet. Try to sign off each step as soon as you get it working. Include the signoff sheet with your report.
4. Document your complete hardware and software design, and testing results in your lab report. Include all source code.

Hints

1. Make sure your wiring is neat - you will find it very difficult to debug your circuit unless you can clearly see the address, data and control bus lines.
2. Keep in mind that mistakes are almost unavoidable when constructing a circuit by hand.
To ease troubleshooting:
 - a. Make your circuit as simple as possible.
 - b. Assemble and test in steps.
 - c. Configure the power supply for a low current limit.
 - d. Discharge any static electricity from your body.
 - e. Double-check your wiring before turning on the power.
3. If you get stuck, do not hesitate to ask the instructor for help.
4. Read the signoff sheet to better understand the lab requirements.
5. Write your report concurrently with your system assembly/testing.

Developed by Ghaith Hammouri (hammouri@wpi.edu) and Gene Bogdanov (gene@wpi.edu),
March 2007.

ECE 3803 D2007 Lab 1
Signoff Sheet
Due date: 03/29/2007

Names

Student 1: _____ **ECE mailbox:** _____

Student 2: _____ **ECE mailbox:** _____

Requirement	Max points	TA's assessment	TA's initials
32.768 KHz clock frequency	10		
CPU resets properly. Demonstrate read bus cycles on the logic analyzer.	10		
Infinite loop program (hand assembled)	10		
Infinite loop using the C code	5		
Adjust the clock to 10 MHz	5		
EEPROM runs at the minimum acceptable number of wait states	5		
Flashing LED	15		
Clock frequency contest (extra credit)	5		
Neatness of wiring, proper use of bypass capacitors and proper grounding	10		
Report	30		
Total points	105		

- Notes: 1. To pass this lab, you need to demonstrate the C infinite loop running on your circuit.
 2. Remember that the labs are cumulative: you need this circuit for lab 2.
 3. Maximum score: 110.25 when taking advantage of maximum early signoff bonus.

TA's signature: _____ **Date:** _____