Microcontroller System Design

ECE121-Lab0

Professor Peterson

James Huang

Collaborated with Aaryan Redkar, Peter Le, Justin

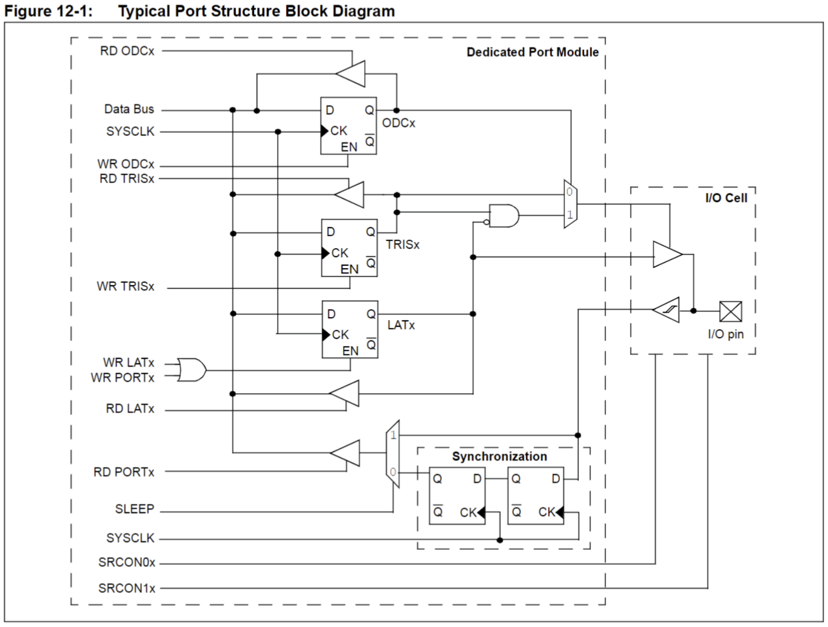
01/19/24

1 Introduction

1.1 Hardware Background

I/O ports come with control registers, which configures the port to certain functions. In our PIC32 I/O ports, there are 9 different types of control registers. For Lab0 and simplicity, 3 registers will be relevant here. First, Registers for Configuring Tri-state Functions (TRISx), this register determines if the I/O port is going to be used for input or output, where 1 is input and 0 is an output. Second, Register for Configuring PORT Functions (PORTx) , this one is used for reading port data, it is connected directly to the I/O port. It can be used for writing as well, however, the write function is directly connected to the corresponding LATx, so writing to PORTx is the same as writing to LATx. Lastly, Register for Configuring Latch Functions (LATx), this register is used for writing to the I/O port, as it is directly connected to the port. Reading with LATx is not the same as reading with PORTx, when reading with LATx, it is not directly connected to the I/O, but reading the LATx register itself. Therefore, it is better to read using PORTx, as it is connected to the port directly.

Figure 12-1: Typical Port Structure Block Diagram



(Figure 1: Diagram with control registers and I/O port.)

1.2 Software Background

Due to the fact that the PIC32 is single threaded CPU, it cannot do simultaneous sequence of instructions. Therefore, when we want to delay something by a certain amount of time, we want to write non-blocking code. When waiting for a condition to be true, if it is not ready, skip it and move on. By using an assembly instruction, no operation (NOP), this delays the processor for 1 clock cycle, and nothing happens. A for loop with NOP, can easily be used to delay a certain amount of time, with the right number of iterations.

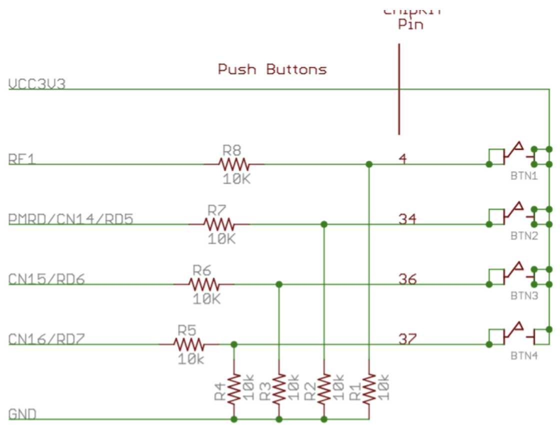
2 Lab0

2.1 Lab Introduction

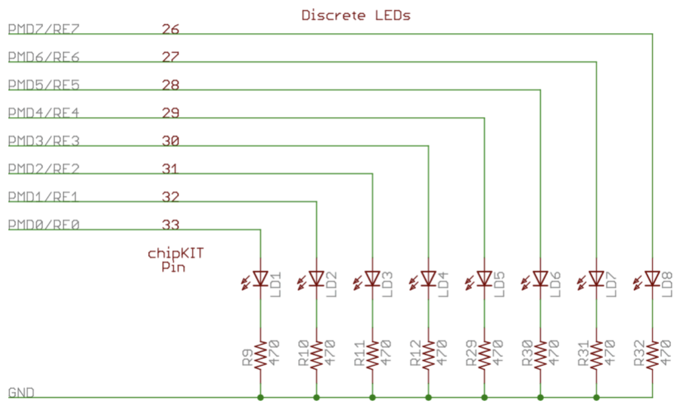
In Lab0, the buttons are used to control the LEDs, with the 4 buttons, it can light up different LEDs corresponding to different buttons. Then a simple LED incrementer, where the LEDs are continuously incrementing (like an 8-bit counter), with 5 ms delay in between. When a button is pressed or when all the LEDs are lit (it has counted to the highest, 11111111), it will be reset.

2.2 Initial Approach

First I have to find the corresponding port to the buttons and LED, in order to do anything with them. In the basic I/O schematic, I was able to find the buttons and LEDs, with their corresponding ports.

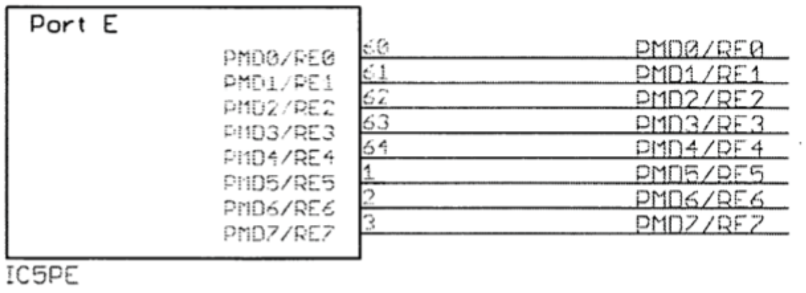


(Figure 2: Buttons and its corresponding ports.)



(Figure 3: LEDs and its corresponding ports.)

Looking at Figure 2 and 3, I see that LEDs use RE0-7, and the buttons use RF1 and RD5-7. In the chipkit u32 schematic, I see that all of RE0-7 is connected to Port E, which can be seen in Figure 4.



(Figure 4: RE0-7 is connected to Port E.)

I now have the specific ports I need for buttons and LEDs.

2.3 Implementation

For part 1 of Lab0, I first declared TRISE to 0, configuring all of Port E to be output ports, so we can see the LEDs as an output. Then I set LATE to 0, which writes 0s to Port E, turning off all LEDs. Then using if-elif statements, I check button activities using the PORTx to read the specific buttons ports. If a button is pressed, LATx is written to the corresponding LEDs and turns them on.

For part 2, similarly with part 1, first set the LEDs, and check the buttons for reset or if it has incremented all the way. The harder part is finding the 5ms delay. The internal clocks run at 80MHz, and I used the stopwatch and debug tool to help me find the number of clock cycles for running my for loop. In the end, I found that about 260,000 for loop runs of NOPs is about 5ms.

3 Conclusion

3.1 Summary

Lab0 taught me a lot about how the hardware directly interacts with the software we wrote. By configuring the control registers, we see how the code changes the I/O ports. I learned how to directly set the hardware using code and seeing its results. It also introduced me to debugger and stopwatch functions, setting breakpoints and counting clock cycles of my function.

3.2 Challenges

One of the challenges I had was configuring the control registers. According to the TA, I am supposed to configure the D and F ports, as they are used by the buttons. However, configuring actually broke the buttons, pressing them did not turn on the LEDs. I showed the TA what happened, she said she doesn't know why, but I was supposed to configure it. Therefore, I did not directly configure the button ports, by leaving it default it would be set as input and my code worked that way. Lastly, the stopwatch and debugger tools were kind of confusing at first, but once I got the hang of it, it was easy to use.