I. Learner Objectives:

At the conclusion of this lab, participants should be able to:

- Read and analyze a set of requirements for a real-world application
- Apply read and write operations to ports
- Discover I/O registers
- Manipulate peripheral modules connected to the Digilent Cerebot MX4cK Board
- Compose a small embedded C language program
- Comment a small embedded C language program

II. Prerequisites:

Before reading this lab, participants should be able to:

- Summarize the material found in the PIC32MX MCU unit
- Diagram and annotate the architecture associated with the PIC32MX460F512L microcontroller

III. Keywords:

Port, I/O register, peripheral module, data direction register, data register, output pin, input pin

IV. Pre-lab (in-lab):

This pre-lab needs to be completed either before lab or you may work on it at the beginning of lab (just for this week!). Please do not hesitate to ask your TA questions about the pre-lab! This pre-lab will not only help you with the understanding of this lab, but the overall illustrious interconnection of ports and microcontrollers.

The first part of the pre-lab requires that you delve into the PIC32MX4XX Data Sheet and PIC32MX4XX I/O Port Reference to understand port operations. The block diagram on page 3 of the I/O Port Reference will give you a quick foray into these ports. Figure 1-1 on page 21 of the data sheet, provides you with a logical connection between the ports and peripheral bus, clocked by SYSCLK. List the names of the ports. Also, describe the number of pins that exist per port!

The second part of the pre-lab requires that you briefly describe the function of a register, and then describe the three port control registers directly associated with each port. You may find information about these control registers on page 4 of the I/O Port Reference.

V. Background:

Simon Says is a classic electronic game that uses tones and lights to help strengthen and test memory skills. The computer plays a series of tones and lights up colored buttons. You must use your memory to recall the same sequence of buttons lit up by the computer. The game progresses with a new tone and button lighting up every

turn. The game is over once the player incorrectly recalls the sequence of buttons. In this lab you will implement a simplified game of Simon Says.

You will use Digilent *peripheral modules* to represent the Simon Says game controls, including the buttons (represented by switches) and lights (LEDs). However, in this lab you will not implement the tone controls. You will interface with the peripherals via input/output (I/O) *ports*.

I/O ports allow the microcontroller to interact with peripheral devices or peripheral modules. Types of peripheral modules that we will use throughout the semester include: LEDs, switches, speakers, H-bridges, sensors, and LCDs. Refer to Digilent's peripheral modules link for a complete list. The I/O port is really the interface between the microcontroller and the outside world. All peripheral modules that want access to the microcontroller and its various resources must go through a port. A port consists of the required drivers and I/O port registers. Port drivers constitute, through software, an interface between a peripheral module and the microcontroller.

Figure 1, below, illustrates the basic parts of a typical port.

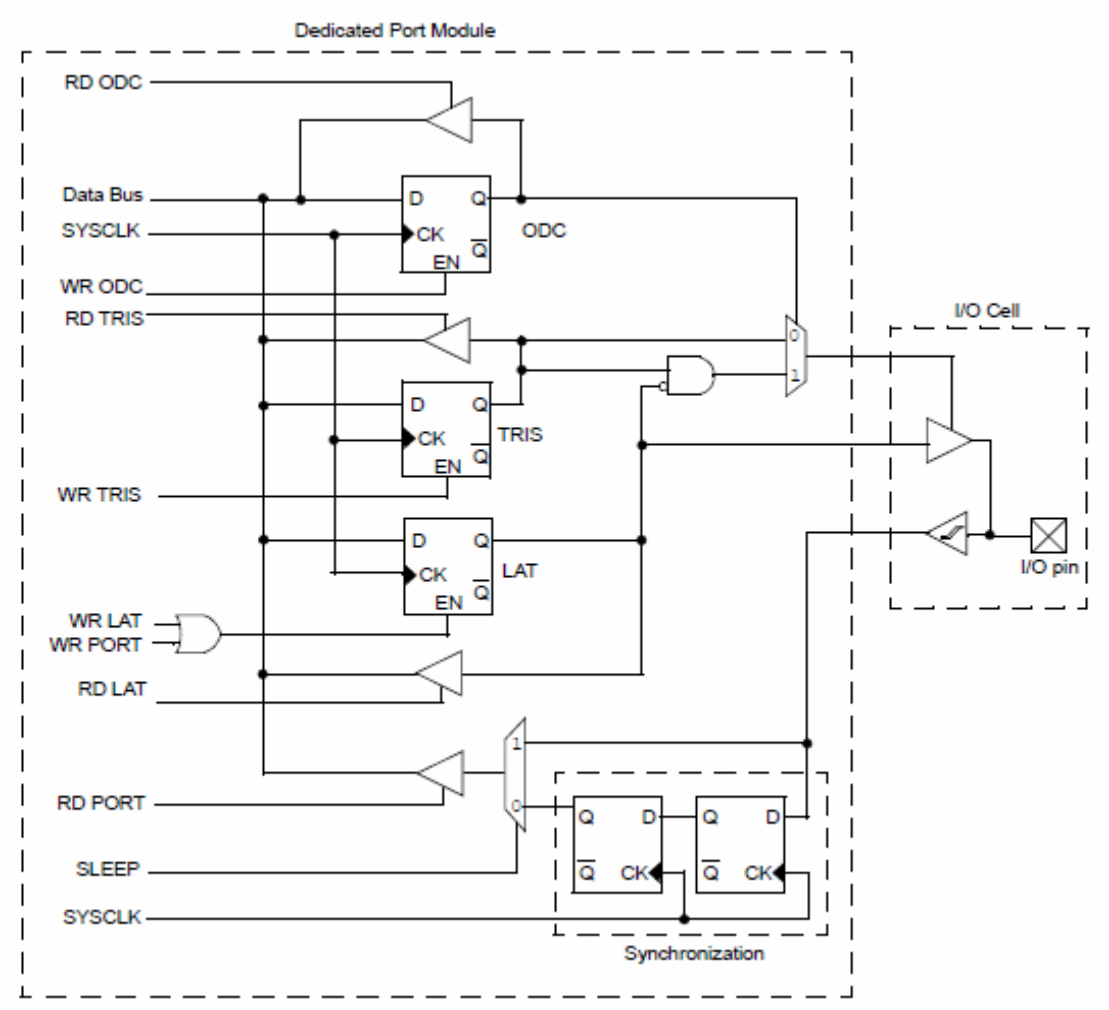

Figure 1: Typical Port Structure (p. 3 in I/O Port Reference[1])

Although you will not be explicitly using the I/O port control registers in your C program, you should start to make a connection between the ports.h library functions and the control registers.

VI. Lab:

What to complete:

Before you begin writing C code for the lab make sure you have one end of the USB cable plugged into the Cerebot MX4cK micro-B "DEBUG" port and the other end into one USB-A port on your PC. Also, be sure to plug in the switch module that came with your Digilent board package into the JK-01:04 connector of the Cerebot MX4cK board. Lastly, plug the peripheral LED module into the JJ-01:04 connector. The top pins correspond to 01:06. **NOTE: If you have one of the newer LED modules (i.e. with 8 LEDs), then you will plug the module into JJ-01:12. However, you will only be concerned with 4 of the LEDs.** Turn on power to the board by flipping the switch next to the "DEBUG" port.

Remember the underlying application for this lab is to implement a simplified version of the Simon Says game. You will construct a C language program that will receive data from a Digilent switch peripheral module (representing the Simon Says button controls) and write data to a Digilent **LED peripheral module** (representing the Simon Says light controls). You may start with the code template C file. The goal of this lab is to write C code that will light up LD_{m-1} on the peripheral LED module when SW_m is on and turn it off when SW_m is off. Note that *m* represents 1 – 4. Table 1 lists the required operation of the peripheral LEDs and switches for this lab.

	LD0 (on)	LD1 (on)	LD2 (on)	LD3 (on)
SW1	On	Off	Off	Off
SW2	Off	On	Off	Off
SW3	Off	Off	On	Off
SW4	Off	Off	Off	On

Table 1: Switch and Peripheral LED Operation

Table 2 summarizes the pin connections between the Digilent 6-pin 4-slide switch peripheral module (plugged into the JK-01:04 connector) and the Microchip PIC32MX460F512L microcontroller. Note that you may find the port and function for the pin in the Cerebot MX4cK Reference Manual document.

[1] I/O Port Reference

Cerebot Pin	MCU Port/bit	Switch	On-board LED
JK-01	RB10	SW1	LD1
JK-02	RB11	SW2	LD2
JK-03	RB12	SW3	LD3
JK-04	RB13	SW4	LD4

Table 2: Port to Switch Mapping

Table 3 summarizes the pin connections between the Digilent 6-pin or 12-pin LED peripheral module (plugged into the JJ connector) and the Microchip PIC32MX460F512L microcontroller. Recall your goal is to map the state (OFF/ON) of a switch to the state (OFF/ON) of a peripheral LED.

Cerebot Pin	Port/bit	Peripheral LED
JJ-01	RB0	LD0
JJ-02	RB1	LD1
JJ-03	RB2	LD2
JJ-04	RB3	LD3

Table 3: Port to Peripheral LED Mapping

Lastly, soak up as much knowledge as possible from this lab!!!

Hints for Getting Started:

Make sure that you have downloaded the C code template. The template itself contains a main ( ) function and an infinite *event* loop!

In the electronic game of Simon Says lights are randomly turned on in a sequence and the player must recall the order of lights and press the buttons to light up the same sequence of lights. You will use Table 1 to determine which switch (Simon Says button) turns on which LED (Simon Says light). Before the game starts you will need to determine a sequence of LEDs to light up, this will be the pattern that the user will try to mimic. For this lab you need to provide only one sequence of LEDs. I recommend the sequence lights up at least 8 LEDs (one at a time). You may hard-code the sequence of LEDs (the easiest solution) and the delay between each LED flash. Once the LED sequence has been played, the user will use the switches to try to re-create the sequence. Your program must determine the player is correct on the sequence.

In this lab you will want to use functions where possible. For example, you will want to create functions to setup the I/O ports. I recommend one function to setup the pins corresponding to the switches as digital inputs, and one function to setup the pins corresponding to the LEDs as digital outputs. These functions should be called outside the event loop. Consider using functions PORTSetPinsDigitalIn ( ) and PORTSetPinsDigitalOut ( ) as helpers to your user-defined functions. The signatures for these functions include:

```
void    PORTSetPinsDigitalIn(IoPortId portId, unsigned int inputs);
void    PORTSetPinsDigitalOut(IoPortId portId, unsigned int outputs);
```

Including <plib.h> allows you access to these port functions. However, you may find the descriptions for these functions in <peripheral/ports.h>.

In order to read the pins corresponding to the switches, you should use PORTRead ( ). To write to the pins corresponding to the LEDs, you should use PORTWrite ( ). The signatures for these functions are listed below:

```
unsigned int    PORTRead(IoPortId portId);
void    PORTWrite(IoPortId portId, unsigned int bits);
```

You may also find descriptions for these functions in <peripheral/ports.h>.

To create a delay between LED flashes, implement nested loops. Continue to nest loops until you are able to achieve a desired delay.

VII. Questions:

Please include the answers to the following questions in your formal lab write-up! NOTE: you do NOT need to formally answer the questions that are throughout the lab. You only need to include the answers to the following questions in your lab write-up.

1. Explore the C:\Program Files\Microchip\MPLAB C32 Suite\pic32-libs\include\ folder (it may be a slightly different path on your machine) and any subdirectories. Describe the purpose of five of the .h files.
2. Analyze the disassembly listing for your program. Explain the function of three assembly instructions listed. You may reference to the MIPS32 Quick Reference Guide for assistance.
3. Any comments you would like to make to spice-up or improve this lab for the future!