**Summary:**

The purpose of the lab was to communicate between a keyboard pad and two SO8 micro-controllers. One being the slave and the other being master. The master communicates to the slave with I2C and from the slave we flash a LED pattern, with four different patterns, static on/off, slide right, arrow, and diamond.

**Pre-liminary Solutions:**

With there being 3 different aspects that needed to be coded for everything to work together correctly we wanted to start with the keyboard interrupt faze first. This means that we made sure that when we pushed a row and column on our keypad we had the correct output going into and out of the master S08. Then from there we wanted to code the LED patterns on the slave. This allows us to then focus on debugging the I2C code communication between the master and slave.

**Solutions:**

First, we wanted to tackle the keypad and master communication. Through code warrior, after we wired up the two devices, we can check specific variables in order to see what the interrupt is getting when we push a button. For the interrupt to work in the first place we need to set up the keyboard interrupt vector in the .prm file. This allows for the keypad, when we push a button, to have the S08 go through its interrupt. This was straight forward and didn’t cause many problems. Some things we ran into were not hooking up all the ports in the micro-controllers. Some pins were floating and causing problems in our code that we haven’t seen before.

Next, we wanted to tackle the LED bar. This was going to be communication between the slave S08 and the LED bar. At first, we created a lookup table and then ran loops specific to each pattern and ran the code like that. This was to test to make sure that the wiring was correct between the slave and LED bar. This would allow us to know that code for the patterns work and if there is any problems, it is coming from the I2C communication.

Finally, we wrote the code for the I2C for the master and slave to communicate. This needs in both files another interrupt vector set, which was vector 7 named as \_Viic. This allows for both micro-controllers to run through their I2C interrupt code when something was stored into its msg. This created some problems and a hassle to debug. For example, there were some instances were we only needed a small delay to make the code work for it was trying to read it too fast and overriding some code, breaking the rest. Also, with so many declarations and pin read outs, if we missed one single bit, all of our LEDs wouldn’t turn on. However, after the communication was all debugged, next was to see what information the master was sending to the slave. The msg that the master was sending would determine the specific pattern the LED would show.

**Summary:**

In this lab there were multiple devices talking to each other and for that to work everything needed to be set up perfectly. The most confusing problem encountered was for the 245 device. For we left two pins floating that we didn’t think needed to be hooked up and it caused problems not even the TA or teacher understood. However, after we drove one pin high and one pin low to determine the direction of the pin layout the code worked just fine.

Communication between Keypad and Master

Initialize all PTBD and PTAD pin for either output or input

Send button pushed to variables

Store keypad putton push

Push Keypad to send interrupt

Set interrupts for Keypad to work

I2C communication for slave and master

Complete through pattern loop

Have slave read msg and pick pattern

Send information from Master to slave

Set interrupts for \_Viic in master and Slave

Initialize all PTBD and PTAD pin for either output or input

\

Communication between slave and LED bar

Run through pattern

Create branches for each pattern

Initialize all PTBD and PTAD pin for either output or input

Compare \_Vicc msg to pattern branch to determine pattern