EGR 322 Microcontrollers

PSoC and Arduino Calculator

Prepared for: Dr. Foist

Christopher Parisi (390281)
Ryan Canty (384185)

College of Engineering
California Baptist University

05/02/12

TABLE OF CONTENTS

Introduction

In this section, you will give a brief description of the purpose of the lab project, and the procedure that was used to accomplish the task. It may also give the reader some information on what was done in the sections following the introduction (for longer reports).

Implementation

Step 1: Log Book

At the beginning of the project, we started a log book in order to keep track of when tasks were completed as well as who they were completed by. This report contains the information logged within our log book.

Step 2: Overview

Before we began working on our project, we gathered information that was relevant and helpful. We looked at example codes from Cypress and Arduino tutorials. With a basic understanding of what the project required and the steps that were necessary to completing the calculator, we began learning the different components.

Step 3: Arduino Introduction

The first component we needed to learn about was the Arduino UNO microcontroller. We started by reading a Wikipedia entry on the Arduino microcontroller. The web address for this entry is http://en.wikipedia.org/wiki/Arduino. We then viewed a tutorial video given by Jeremy Blum and installed the software required for programming the Arduino. We then took Jeremy's tutorial 1 code and installed it on our own Arduino. This program made an LED blink on the microcontroller board. We changed the delay values to be 100 in the code to increase the blinking speed. The Arduino code we used can be found in Apendix A.

Step 4: PSoC SPI

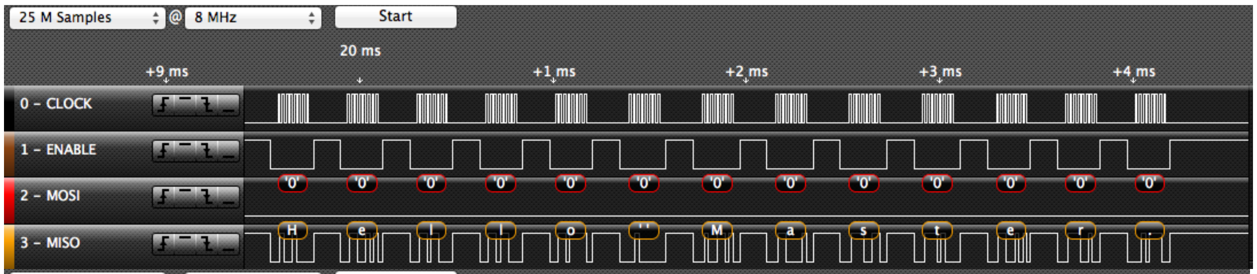
We then had to learn how to run SPI (Serial Peripheral Interface) on the PSoC. We studied the SPI examples and took the information we needed to create our own SPI code that met our requirements. When we wrote our code and debugged it of any errors, we downloaded it to our PSoC Eval board to test it. The LCD showed two messages, "Hello Slave" and "Hello Master".

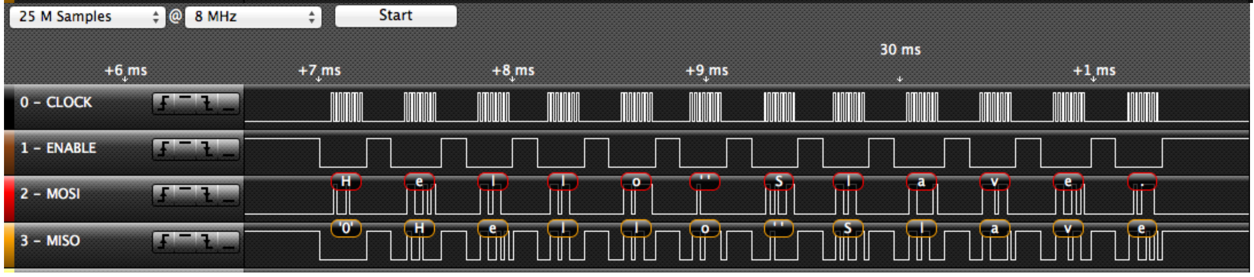
Step 5: Logic Analyzer

Once the PSoC SPI code was completed and downloaded to the PSoC Evall board, we tested the logic of our code by using the Logic Analyzer. We downloaded the Logic 16 software from http://www.saleae.com/downloads and then watched a tutorial video of the program on the same website. Once we understood how the Logic Analyzer worked, we followed the steps given in the development steps. We connected the Logic 16 box to the laptop and a green light appeared as well as a message on the program saying "[connected]". We then connected the cable to Channel A and hooked up the wires to our PSoC board. The connections we used are listed below.

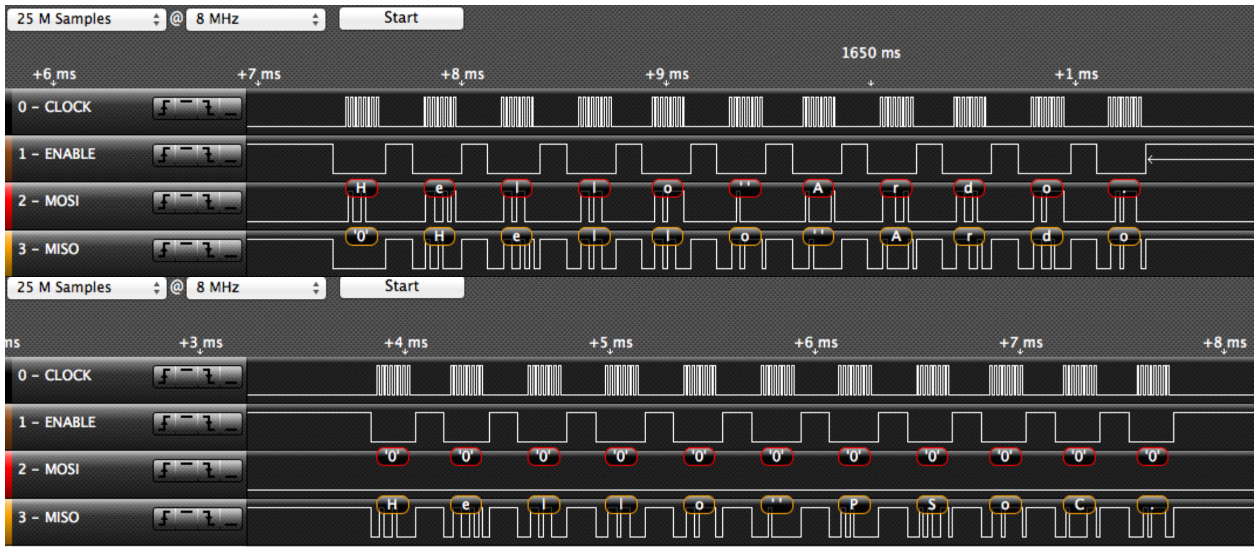
Logic 16 Channel A	PSoC Pin
Gray wire	Ground
Black wire (SClk)	P0[5]
Brown wire (SS_)	P0[7]
Red wire (MOSI)	P0[6]
Orange wire (SClk)	P0[4]

Once the hardware was connected, we changed the parameters on the program to read 25 M Samples at 8 MHz. We added an SPI under *Analyzers* and configured the channels. We started the analyzer and the screen displayed "Hello Master" and "Hello Slave". Screen shots of our results are shown below.



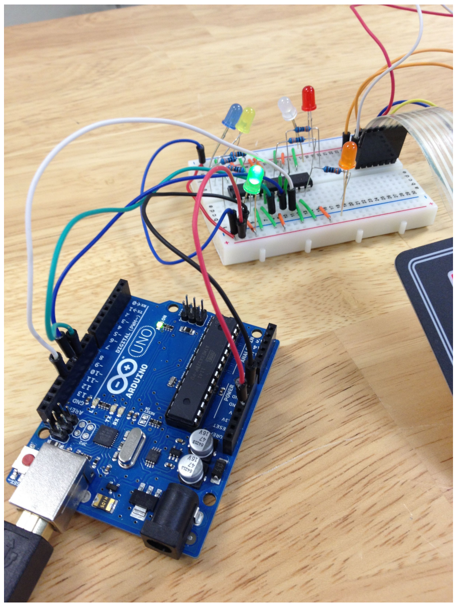


Once we confirmed our program and hardware was functional, we changed the messages to read “Hello PSoC” and “Hello Ardo”. A screenshot of our modified results are shown below.



Step 6: Arduino SPI

Our next task was to control a digital potentiometer using Arduino. We borrowed Dr. Foist’s breadboard that had the AD5206 digital potentiometer circuit constructed. We looked at the AD5205 schematic and figured out which pins to connect to our Arduino. Once the hardware was connected, we found the DigitalPotControl program built into the Arduino program. We uploaded this code to our Arduino and then watched as the digital potentiometer was updated. The LEDs on the breadboard would get brighter and then switch off one at a time. We also changed the delay values to increase the speed at which the LEDs changed. The shorter the delay, the faster the LED lit up and switched. A photo of our hardware is shown below.



Step 7: PSoC SPI Slave Design

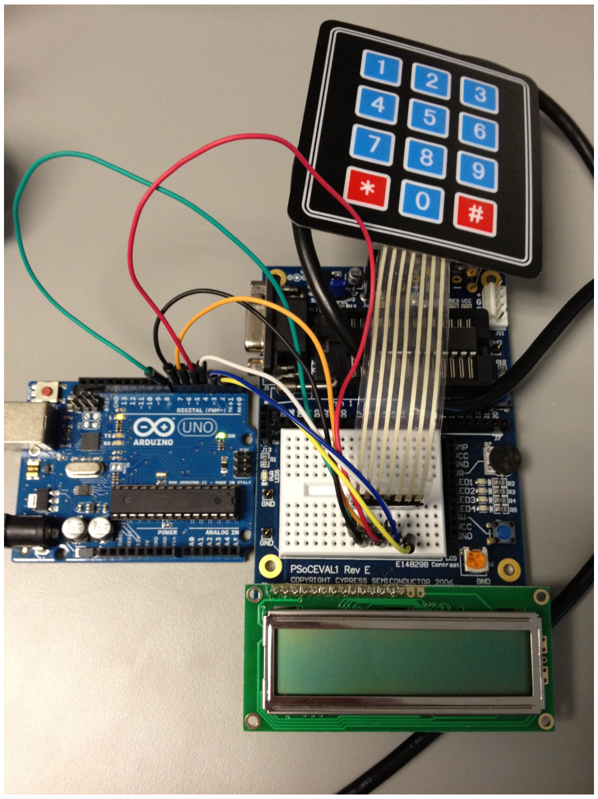
We then had to setup the SPI slave design for our PSoC in order to have the PSoC be able to receive data from the Master, which is our Arduino. We took the PSoC SPI Example 1 project and modified it to contain a SPIS and LCD module. We did this because the Arduino will be sending data through the SPIS and onto the LCD screen.

Step 8: Arduino Master + PSoC Slave

Once the PSoC was set up to receive data as the Slave, we setup the Arduino to send data as the Master. We again took example codes and modified them to fit our requirements. We had the Arduino send over static values of 77 and 99, which appeared on the PSoC's LCD screen.

Step 9: Arduino + Keypad

Next, we learned about how to connect the Keypad to the Arduino. We followed the tutorial given in the project handout and learned how the keypad works. We then downloaded Dr. Foist's keypad code and installed it on our Arduino. In order for the keypad to work, we had to add the keypad library files to the Arduino program files. Once we copied them over, the program compiled and the keypad worked. The LED on the Arduino only toggled up when the "*" and "#" keys were pressed. A screenshot of our hardware connections can be seen below.



Step 10: Arduino Master + PSoC Slave + Keypad

After learning how to use the keypad, we implemented this into our program used in step 8. We added the keypad into the hardware and then modified our code to utilize the keypad. When we pressed a key on the keypad, the ASCII value of the character appeared on the LCD on the PSoC.

Step 11: Top-Level Design: The Complete Calculator

Once we had completed all of the preliminary tasks, we began putting everything we learned together to form a function adder using the Arduino and PSoC. We used the Arduino as the Master to get data from the keypad and then have it sent to the PSoC where the results would be displayed on the LCD. We had to write code for each microcontroller that accomplished this task. The PSoC received the character form the keypad and then figured out the value, then calculated the sum of the two values it was given.

Conclusions

This project has given us given us extensive experience using microcontrollers. Through this project, we have learned more about the PSoC and have learned about another microcontroller called the Arduino UNO. This project has taught us how to implement the Master Slave system design using our two microcontrollers. Data is received from the keypad via the Arduino and then sent over to the PSoC to be added and displayed on the screen. This required us to write code for both the PSoC and Arduino. We have learned so much about using microcontrollers for a practical application and more importantly, we have learned how to do research on our own to accomplish a given problem. This project gave us a real taste of what electrical/computer engineering really is.