EE 337: SPI and ADC Interface to 8051
Lab 6

19^{th} September, 2016

Objectives

- In this lab session, you will learn to program using Embedded C and interface an analog-to-digital converter (ADC) MCP3008 with the 8051 micro-controller.
- You will learn about serial peripheral interface (SPI) protocol, using which ADC (MCP3008) communicates with the micro-controller. SPI is a synchronous, serial communication protocol used for communicating between micro-controllers and peripherals (or between two or more micro-controllers).
- Once the ADC is interfaced, it is used to measure the output of an electronic weighing machine for determining the weight of an object.

Introduction

In this experiment, we are going to measure weights using a weighing machine. The weighing machine output is given to an ADC that is connected to the 8051 micro-controller via the SPI. It uses separate data and clock lines, along with a select line to choose the device you wish to communicate to. SPI is a synchronous serial interface which uses shift registers for converting parallel data to serial. In SPI communication, there is a Master device (typically the uC), and a slave device (typically, a peripheral circuit). In this experiment, we are going to send data from a weighing machine (slave) sensor to the 8051 micro controller (master).

The data in this case is going to be the weight of an object (in gms) kept on the weighing machine sensor. The task is to interface an ADC with the microcontroller and use the ADC to measure the voltage output from the weighing machine and thus determine the weight put on top of it. The weight measured is then to be displayed on the LCD screen.

Homework

1. Go through and understand the reference document on *Embedded C programming using 8051 for Keil* uploaded on Moodle. Have a look at the *programming style sheet* as well. LCD subroutines written in assembly language have been used until now, they are rewritten in embedded C.

   Use the sub-routine `LCD_WriteString` in the given *lcd.c* to display "Hello" on the first line of the LCD screen starting from the first position and "World" on the second line of the LCD starting from mid-position.

2. Read through the document *SPI-Intro* for an introduction to SPI. For the sake of simplicity, the micro-controller is already configured for serial communication with the ADC IC (MCP3008) in the code written in *spi.c*. Read through this code and *ADC Interfacing* to understand the configuration for doing the same.

   Now we need to configure the SPI to take samples from the ADC periodically, for example every 25 ms. To do this, use one of the timers (Timer 0 or 1) to raise an interrupt event every 25 ms. In *spi.c*, modify the following functions to achieve this.

```
void timer0_ISR (void) interrupt 1
{
//Write corresponding statements in C over here
//Initialize TH0
//Initialize TL0
//Increment number of overflows
}


void Timer_Init()
{
//Write corresponding statements in C over here
//Initialize TH0
//Initialize TL0
//Configure timer 0 by setting TMOD appropriately.
//Set ET0 to enable timer 0
//Set TR0 to start timer 0
}
```

   Later in the labwork part, you will use the above code and configuration to interface with a weighing machine.

Lab Work

1. Before we connect weighing machine to the ADC, we first have to test the system configured so far. The output of weighing scale is an analog voltage proportional to the amount of weight kept on it. Hence, it makes sense to connect a variable voltage to the input of ADC and test the system. A potentiometer connected between supply and ground terminals can be used to achieve the same. Display the measured voltage on the LCD in the format,

   ```
   Voltage: xxxx mV
   ```

   Use the *delay_ms* subroutine to provide a delay of 500 msecs between every measurement. Verify that the displayed voltage is correct using a DMM. Use the code written in *spi.c* to perform this part.

2. Once the system is tested for variable input voltage, it is time to calibrate the system for displaying weights in gms. Connect the weighing machine output to the input of ADC chip. The circuit should work when it is connected to the weighing machine. If it does not, it means either weighing machine is not properly connected or it is faulty. Use the timer settings and modified code of *spi.c* written in the Homework to do this part.

   Measure the voltage output from the weighing machine and display the weight on the LCD screen. The weight is to be sampled every 25ms using a hardware timer (interrupt based) and updated on the LCD screen so that a consistent and fast updating display is provided to the user. The program should take 10 ADC samples every 25ms and take an average of these 10 samples, to compensate for the fluctuations in the output voltage. The output provided by the weighing scale is 0.74 mV/gm. The weight measured is then to be displayed on the LCD Screen in the format,

   ```
   Weight: xxxx gm
   ```

   Once you are done with this part, you should be able to answer questions like: What is the maximum and minimum weight that can be measured with the current setup ? or What is the quantization error of the setup ?

Functions in the `lcd.c` and `spi.c` files:

The file contains the following functions

- `SPI_Init( )` : Initializes the SPI Module and configures it to interface with the ADC.

- `LCD_Init( )` : Initializes the LCD screen using the same commands used in the assembly language programs written earlier, clears the LCD and sets the cursor position to Line 1 Position 1.
- `LCD_DataWrite(char dat)` : Writes a character on the LCD screen. e.g.,

  ```
  LCD_DataWrite(0x38);
  ```

- `LCD_CmdWrite(char cmd)` : Writes a command to the LCD. e.g.,

  ```
  LCD_CmdWrite(0xC6);
  ```

- `LCD_WriteString(char * str, unsigned char len)` : Writes a string on the LCD Screen. e.g.,

  ```
  LCD_WriteString(Hello, 5);
  char str[10];
  LCD_WriteString(str[3], 4);
  ```

  Please note that while displaying a string, the number of characters should match the number of characters in the string. Otherwise incorrect values will be displayed.
- `LCD_Ready()` : Checks if the LCD is ready to receive commands
- `sdelay(int delay)` : Produces a small delay (15 μs for a 24MHz clock)