# Lab 1: SPI

ECE/CS

Due by Friday Jan 28,2011 11:59 PM

The goal of this lab is to use the Serial Peripheral Interface Protocol and display some data on the LCD screen of the Experimenter Board. In terms of complexity, this lab is a notch higher than the previous lab. This document is divided into two sections - a tutorial and the problem statement. The tutorial will give you a basic understanding of the SPI Protocol.

## 1 Tutorial: SPI

Open up Chapter 25 of the MSP430x5xx Family User's Guide. Read carefully through sections 25.2 and 25.3. You may skip the Low Power Modes and Interrupts subsections for this lab. Section 25.4 has the description of the configuration registers. Advanced readers may skip this section and proceed to the problem statement.

### 4-wire SPI master

A 4-wire SPI interface has the following signals:-

1. SIMO - Slave In, Master Out
2. SOMI - Slave Out, Master In
3. CLK - Clock
4. STE - Slave Transmit Enable

The STE signal is for enabling SPI communication in a scenario with multiple slaves. In this example, we have only one master and one slave. We shall still persist with a 4-wire implementation for this section. You can view an SPI communication as a couple of register shifts, wherein the master shifts data into the receive buffer of the slave and the slave shifts data into the receive buffer of the master. Data is first loaded into the transmission buffer of the master. Then, the master selects the slave and transmits the associated clock. This is followed by data being transferred.

Simultaneously, the master also receives data. This data which is received by the master is from the transmission buffer of the slave. This data may be used or may be junk depending upon usage and configuration.

### Configuring a 4-wire SPI Master

To begin with, switch off your watchdog timer. Read Chapter 13 to understand what the following line does.

```
WDTCTL = WDTPW + WDTHOLD;
```

Read upon why watchdog timers are used and how they assist in enhancing reliability of embedded systems.

Now select a GPIO which supports SPI communication. Check the MSP430F5438 data sheet[Figure 1] . I picked P10 as this port is also brought out as headers on our board. P10 has UCA3 as the SPI interface. Configure the port to select the functional mode and set the appropriate directions. We are going to use the interface as a Master, therefore decide which of the four SPI signals are to be set as output.

```
P10DIR = 0x19;//Master
P10SEL = 0x39;
```

Now, we initialize UCA3 as a master. The interface is in configuration mode when the reset register is high. Choose SMCLK as the clock for this tutorial. Release the reset after configuring.

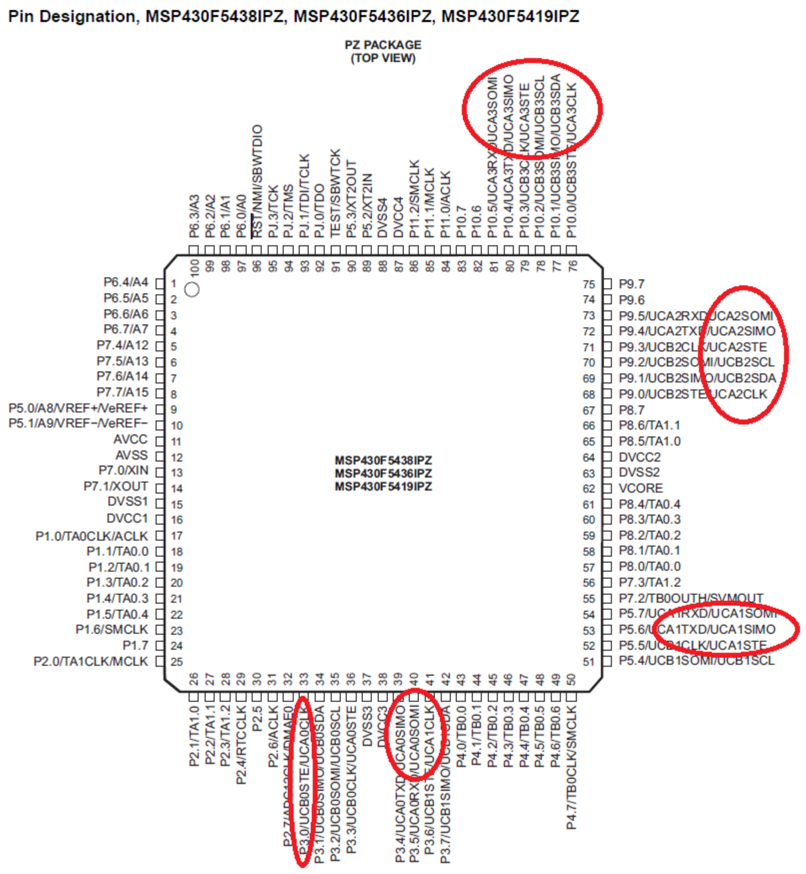
Pin Designation, MSP430F5438IPZ, MSP430F5436IPZ, MSP430F5419IPZ
PZ PACKAGE
(TOP VIEW)
MSP430F5438IPZ
MSP430F5436IPZ
MSP430F5419IPZ

Figure 1: Various GPIOs having SPI functionality

```
UCA3CTL1 = UCSWRST; //Reset while configuring
//Configuring the 8-bit 4-pin SPI Mode,
//Setting Master Mode, MSB First, STE =1
UCA3CTL0 = UCMST + UCSYNC + UCMSB +UCMODE0;
UCA3CTL1 |= UCSSEL1; //SMCLK
UCA3BRW  = 8;
UCA3CTL1 &= ~UCSWRST; // Reset released
```

I used a random value for the UCA3BRW register. You can try out different values and see what happens. Now our interface is ready to be used as an SPI master!

## Data transmission

Two bits in UCA3IFG are reserved for flags UCTXIFG and UCRXIFG. We are going to use this master to send data to a slave. Therefore, we will use the UCTXIFG flag. The transmission buffer flag is set when the buffer is empty. So we have to wait till the processor transfers data into the buffer or until data is ready to be transferred. When, the data is ready, the slave transmit enable has to be set.

```
while(!(UCA3IFG & UCTXIFG));
P10DIR |= 0x40;
P10OUT |= 0x40;
```

Now, when the enable has been sent, the serial data transmission begins. The speed of transfer depends upon the bit rate which was set earlier. The UCA3STAT register shows the interface status. We will use the UCBUSY signal to detect the end of transmission. You may or may not want to use the data sent by the slave. When the transaction is complete, disable the slave signal.

```
UCA3TXBUF = transmit_data;
while(UCA3STAT & UCBUSY);
received_data=UCA3RXBUF;
P10OUT &= 0xBF;
```

This master can now send data to a slave. The slave side configuration is similar to the master side. There is a slight difference in the data handling of a slave. A junk value must be written at the beginning into the transmission buffer of the slave (Think why). We may also use the UCRXIFG flag for control depending upon the scenario for both the master and the slave.

This completes the tutorial for a very simple 4-wire SPI master. For a single master single slave communication, the 3-wire SPI is sufficient.

## 2 Problem Statement

The TI MSP-EXP430F5438 experimenter board has various peripherals on it. One such device is the HD66753, 130 x 110 resolution Hitachi LCD screen. The experimenter documentation describes the interfacing to the LCD as an SPI protocol implementation. Open the HD66753 data-sheet. You may start with the Block Function Description section and continue from there. Concentrate on the configuration registers for the LCD to initialize, read, write etc. Transmit the signals to these configuration registers using SPI protocol. Also keep in mind, the width of each register when you write the driver. [Hint: Look at Table 18 in the HD 66753 data-sheet].

While going through the HD66753 data-sheet, you will notice that the given LCD has *SDA* and *SCL* signals which correspond to the I2C standard. Internally in the experimenter board, the SPI signals SOMI and SIMO are multiplexed and are fed into the SDA input of the LCD. So assume that the LCD given to you has an SPI interface and program. Use the ports of the micro-controller that has been given in the header file *lcd.h*.

As an output, print a character on the LCD screen (example – the letter "E"). The font libraries that you may require are also provided with this handout.

## 3 Submission

Please zip everything before submission and only upload the package. Do not upload every single file separately. The content of the package should be properly organized. Particularly, you should put everything under a directory named *username*_`lab1`, where *username* is your Blackboard system login id. Under the directory, include

1. `lcd.c`: Your implementation of the LCD driver;
2. `main.c`: Where you use the driver to print a character on the LCD.