**Lab 3 - SPI - "I/O"**

**Given code**

* [color.h](http://ece.ninja/382/labs/lab3/color.h) - lists some of the colors at your disposal on the LCD
* [lab3\_given.asm](http://ece.ninja/382/labs/lab3/lab3_given.asm)

**Mega Prelab**

A hard copy of this Mega Prelab is required to be turned in as well as pushed to Bitbucket. Answers should not be handwritten. The timing diagram may be NEATLY drawn by hand with the assistance of a straightedge on engineering paper.

**Delay Subroutine**

In lab3\_given.asm, you have the header for a subroutine (line 569), but there is no code. Write a subroutine that will create a 160ms delay. Show your analysis that proves the delay is indeed very close to 160 ms. Note: the clock rate is set to 8 MHz (see the first two lines of initMSP).

**Delay160ms:**

**push** r5 ; 3 cycles

**push** r6 ; 3 cycles

**mov.w** #0xf9ff, r5 ; 2 cycles

**mov.w** #0x0004, r6 ; 2 cycles

**delay:**

**dec** r5 ; 1 cycle

**inc** r5 ; 1 cycle

**dec** r5 ; 1 cycle

**jnz** delay ; 2 cycles

**dec** r6 ; 1 cycle

**jnz** delay ; 2 cycles

**pop** r6 ; 2 cycles

**pop** r5 ; 2 cycles

**ret** ; 3 cycles

5+3+2+(((1+1+1+2)\*0xf9ff)\*4+(1+2)\*4)+2+2+3 = 1279997

1279997/8000000 = .159999 ms

**ILI9341 LCD BoosterPack**

Look at the schematic for the LCD BoosterPack. Complete the following table. The pin number (1 - 20) should be the pin number that signal connects to on the MSP 430, and the PX.X should be the pin and port it connects to (e.g. P1.0).

| **Name** | **Pin #** | **PX.X** |
| --- | --- | --- |
| S1 | 5 | P1.3 |
| S2 | 8 | P2.0 |
| S3 | 9 | P2.1 |
| S4 | 10 | P2.2 |
| S5 | 11 | P2.3 |
| MOSI | 15 | P1.7 |
| CS | 2 | P1.0 |
| DC | 6 | P1.4 |
| MISO | 14 | P1.6 |

What are the hex values that need to be combined with the below registers for these signals to be properly configured? State whether that hex value needs to be used with the bis or bic instruction with each register to achieve these ends. If the register is not affected for that signal, simply say N/A.

| **Signal** | **PxDIR** | **PxREN** | **PxOUT** | **PxSEL** | **PxSEL2** |
| --- | --- | --- | --- | --- | --- |
| S1 | bic #BIT3 | bis #BIT3 | bis #BIT3 | bic #BIT3 | bic #BIT3 |
| MOSI | N/A | bic #BIT7 | bis #BIT7 | bis #BIT7 | bis #BIT7 |
| CS | bis #BIT0 | N/A | bic #BIT0 | bic #BIT0 | bis #BIT0 |

**Configure the MSP430**

Look at the initMSP subroutine in the lab3\_given.asm file. There are four pins being intialized on port 1: SCLK, CS, MOSI, and DC. What is the pin number (1-20) associated with each of these signals? What function does each signal serve? For example, SCLK is the serial clock.

| **Name** | **Pin #** | **Function** |
| --- | --- | --- |
| SCLK | 1.5 | serial clock |
| CS | 1.0 | Chip Select |
| MOSI | 1.7 | Master Out Slave In |
| DC | 1.4 | Data Collect |

Below the pin configuration code are some lines of code from the lab3\_given.asm file (lines 134 - 141) to properly configure the SPI subsystem. Use this code to answer the next two questions.

1: bis.b #UCSWRST, &UCB0CTL1

2: mov #UCCKPH|UCMSB|UCMST|UCSYNC, &UCB0CTL0

3: bis #UCSSEL\_2, &UCB0CTL1

4: bis #BIT0, &UCB0BR0

5: clr &UCB0BR1

6: bis #LCD\_SCLK\_PIN|LCD\_MOSI\_PIN|LCD\_MISO\_PIN, &P1SEL

7: bis #LCD\_SCLK\_PIN|LCD\_MOSI\_PIN|LCD\_MISO\_PIN, &P1SEL2

8: bic #UCSWRST, &UCB0CTL1

Fill in the chart below with the function that is enabled by the lines 6&7 of the above code. Your device-specific datasheet can help.

| **Pin name** | **Function** |
| --- | --- |
| P1.5 | Serial Clock enabled |
| P1.7 | Master In Slave Out enabled |
| P1.6 | Master Out Slave In enabled |

Next, describe specifically what happens in each of the eight lines of code above. Line 1 and 3 have been done for you as an example.

Line 1: Setting the UCSWRST bit in the CTL1 register resets the subsystem into a known state until it is cleared.   
Line 2: Setting the appropriate bits in the control registers. Sets USYNC for the system to function. Holds the system for changes to be made.  
Line 3: The UCSSEL\_2 setting for the UCB0CTL1 register has been chosen, selecting the SMCLK (sub-main clock) as the bit rate source clock for when the MSP 430 is in master mode.  
Line 4: Sets the proper bit to enable &UCB0BR0 and thus the SMCLK.  
Line 5: Clears &UCB0BR1, thus resetting the SMCLK.  
Line 6: Multiplexes the P1SEL pin to enable SCLK, MOSI, and MISO.  
Line 7: Multiplexes the P1SEL2 pin to enable SCLK, MOSI, and MISO.  
Line 8: Turns off the reset to enable the changes made to the system.

**Communicate with the LCD**

The following code (lines 293 - 333) sends one byte (either data or command) to the TM022HDH26 display using its 8-bit protocol.

;-------------------------------------------------------------------------------

; Name: writeCommand

; Inputs: command in r12

; Outputs: none

; Purpose: send a command to the LCD

; Registers: r12 preserved

;-------------------------------------------------------------------------------

writeCommand:

push r12

bic #LCD\_CS\_PIN, &P1OUT

bic #LCD\_DC\_PIN, &P1OUT

mov.b r12, &UCB0TXBUF

pollC:

bit #UCBUSY, &UCB0STAT ;while UCBxSTAT & UCBUSY

jnz pollC

bis #LCD\_CS\_PIN, &P1OUT

pop r12

ret

;-------------------------------------------------------------------------------

; Name: writeData

; Inputs: data to be written in r12

; Outputs: none

; Purpose: send data to the LCD

; Registers: r12 preserved

;-------------------------------------------------------------------------------

writeData:

push r12

bic #LCD\_CS\_PIN, &P1OUT

bis #LCD\_DC\_PIN, &P1OUT

mov.b r12, &UCBxTXBUF

pollD:

bit #UCBUSY, &UCBxSTAT ;while UCBxSTAT & UCBUSY

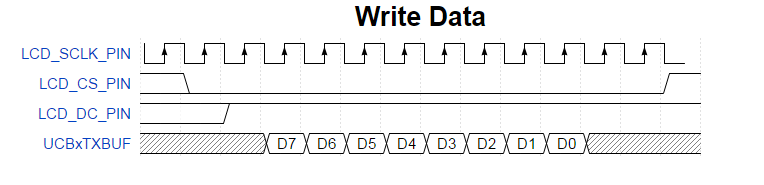
jnz pollD

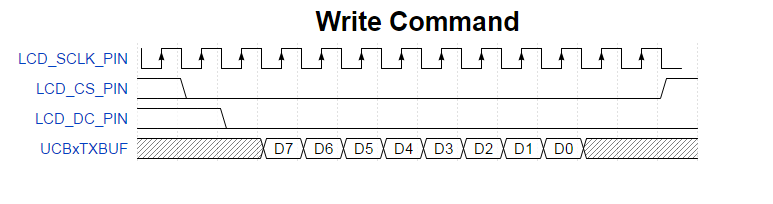
bis #LCD\_CS\_PIN, &P1OUT

pop r12

ret

Use this code to draw two timing diagrams (one for each subroutine) of the expected behavior of LCD\_CS\_PIN, LCD\_DC\_PIN, LCD\_SCLK\_PIN, and UCBxTXBUF from the begining of these subroutines to the end. Make sure that you clearly show the relationship of the edges in the clk and data waveforms.





**Draw a pixel**

The following code (lines 541 - 565) draws a pixel of a predetermined color at the coordinate (R12, R13). However, four subroutines are called to execute this seemingly simple task. Explain the purpose of each of the four subroutine calls:

| **Subroutine** | **Purpose** |
| --- | --- |
| setArea | Defines the area the LCD will write to. |
| splitColor | Splits the color word into color bytes |
| writeData | Send data to the LCD to tell it the color of the pixel |
| writeData | Send data to the LCD to tell it the coordinates of the next pixel to be drawn |

;-------------------------------------------------------------------------------

; Name: drawPixel

; Inputs: x in r12, y in r13, where (x, y) is the pixel coordinate

; Outputs: none

; Purpose: draws a pixel in a particular spot

; Registers: r12, 13, 14, 15 preserved

;-------------------------------------------------------------------------------

drawPixel:

push r12

push r13

push r14

push r15

mov.b r12, r14

mov.b r13, r15

call #setArea

mov #COLOR1, r12

call #splitColor

call #writeData

mov r13, r12

call #writeData

pop r15

pop r14

pop r13

pop r12

ret

**(This marks the end of the Mega Prelab.)**

**Logic Analyzer**

The answers to the logic analyzer section will be posted to Bitbucket in the lab writeup.

**Physical communication**

Connect the LCD Booster Pack to your TI Launch Pad. Make sure that the buttons on the Booster Pack are aligned with the buttons on the MSP430. The pin numbers on the Boosterpack should match the pin numbers of the MSP430.   
Create a project around the lab3\_given.asm file. Be sure to include your Delay160ms subroutine.

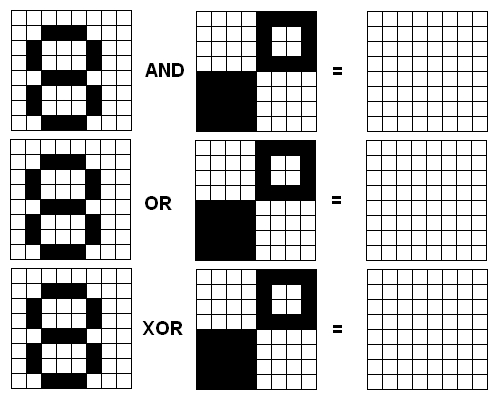
After you insert your subroutine into the code, run the program and observe the output on the LCD every time you press the S1 button.  
  
When S1 is detected as being pressed and released (lines 100 - 102), the drawLine subroutine is called. The MSP430 generates several packets of data that are sent to the LCD, causing a horizontal bar to be drawn. **Find the three calls to writeCommand and eight calls to writeData that generate these packets.** In addition, scan the nearby code to determine the parameters being passed into these subroutines.

Configure the logic analyzer to capture the waveform generated when the S1 button is pressed and released. Decode the data bits of each 8-bit waveform. Explain how the packet contents correspond to what was drawn on the display. Be specific with the relationship between the values sent and what and where the pixels are drawn. Is the packet of information being sent actual data or is it a command? The "Line" column refers to the line of code from which the call to write something to the LCD originated.

| **Packet** | **Line** | **Command/Data** | **8-bit packet** | **Meaning of packet** |
| --- | --- | --- | --- | --- |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |

Be sure to label your signals on the logic analyzer and include screenshots (which can be saved to your external hard drive, for eventual posting in your lab notebook) of each analyzed signal.

**Writing modes**

The native write operation to our LCD will not overwrite any information that is on the display unless it is within the region defined in setArea. However, that may not be the best course of action in your application. The new bits being added to the image may be merged using the AND, OR, XOR operators. To do this treat a black pixel as a logic 1 and a white pixel as a logic 0. The pixel values from the same locations are combined using a logical operator and placed at the corresponding location in the destination imaged. Import the following image into a paint program and show the result of the operation between the two bits maps combined using the logic operator specified. 

**Functionality**

**Required functionality:**

Create a block on the LCD that is 10x10 pixels.

**A functionality:**

Move the 10-pixel block one block (10 pixels) in the direction of the pressed button (up, down, left, right).

**Bonus functionality:**

Write your name or nickname of at least 5 characters to the screen on a solid background when the S1 button is pushed.

**Grading - Lab 3**

[Printable Lab 3 Cutsheet](http://ece.ninja/382/labs/lab3/Lab_3_Cutsheet.pdf)

**Name:**  
  
**Section:**   
  
**Documentation:**

| **Item** | **Grade** | **Points** | **Out of** | **Date** | **Due** |
| --- | --- | --- | --- | --- | --- |
| Mega Prelab | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days |  | 20 |  | EOC L16 |
| Required Logic Analyzer | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days ---- 3Days ---- 4+Days |  | 40 |  | COB L19 |
| Required Functionality | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days ---- 3Days ---- 4+Days |  | 20 |  | COB L19 |
| A Functionality | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days ---- 3Days ---- 4+Days |  | 10 |  | COB L19 |
| Bonus Functionality | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days ---- 3Days ---- 4+Days |  | 5 |  | COB L19 |
| Lab Notebook | **On-Time:** -------------------------------------------------------------------- **Late:** 1Day ---- 2Days ---- 3Days ---- 4+Days |  | 10 |  | COB L19 |
| **Total** |  |  | **100** |  |  |