**10/19/2020**

Joshua Frazer

Lab 5

LCD Display

**Introduction:**

In this lab, we learned how to operate the LCD display. For part 1, we are tasked to display a number of our choosing to the LCD display. In Part 2, we’re given the opportunity to implement a stopwatch. Finally, for part 3, we learned how create a stop and reset function for the stopwatch created in part 2.

**Part 1:**

There was a lot of information to grasp from this part, although straightforward, still required much attention to incorporate it all to CCS. After completing the number array from 0 – 9 – which took most of my time – I had to find a way to display a number on all 6 LCD modules. My solution to this problem was to calculate the modulus a certain number until I hit 0 to get the correct number of digits of a given number. Once this was completed, my program then knew how many digits to display the number and in which order.

// Sample code that prints 430 on the LCD monitor

**#include** <msp430fr6989.h>

**#define** redLED BIT0 // Red at P1.0

**#define** greenLED BIT7 // Green at P9.7

**void** **Initialize\_LCD**();

**void** **display\_num\_lcd**(**unsigned** **int** n);

// The array has the shapes of the digits (0 to 9)

// Complete this array...39

**const** **unsigned** **char** LCD\_Num[10] = {0xFC, 0x60, 0xDB, 0xF3, 0x67, 0xB7, 0xBF, 0xE0, 0xFF, 0xE7};

**int** **main**(**void**) {

**volatile** **unsigned** **int** n;

WDTCTL = WDTPW | WDTHOLD; // Stop WDT

PM5CTL0 &= ~LOCKLPM5; // Enable GPIO pins

P1DIR |= redLED; // Pins as output

P9DIR |= greenLED;

P1OUT |= redLED; // Red on

P9OUT &= ~greenLED; // Green off

// Initializes the LCD\_C module

Initialize\_LCD();

// The line below can be used to clear all the segments

//LCDCMEMCTL = LCDCLRM; // Clears all the segments

// // Display 430 on the rightmost three digits

//

// LCDM19 = LCD\_Num[4];

// LCDM15 = LCD\_Num[3];

// LCDM8 = LCD\_Num[0];

display\_num\_lcd(568);

// Flash the red and green LEDs

**for**(;;) {

**for**(n=0; n<=50000; n++) {} // Delay loop

P1OUT ^= redLED;

P9OUT ^= greenLED;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initializes the LCD\_C module

//\*\*\*Source: Function obtained from MSP430FR6989’s Sample Code\*\*\*

**void** **Initialize\_LCD**() {

PJSEL0 = BIT4 | BIT5; // For LFXT40

// Initialize LCD segments 0 - 21; 26 - 43

LCDCPCTL0 = 0xFFFF;

LCDCPCTL1 = 0xFC3F;

LCDCPCTL2 = 0x0FFF;

// Configure LFXT 32kHz crystal

CSCTL0\_H = CSKEY >> 8; // Unlock CS registers

CSCTL4 &= ~LFXTOFF; // Enable LFXT

**do** {

CSCTL5 &= ~LFXTOFFG; // Clear LFXT fault flag

SFRIFG1 &= ~OFIFG;

}**while** (SFRIFG1 & OFIFG); // Test oscillator fault flag

CSCTL0\_H = 0; // Lock CS registers

// Initialize LCD\_C

// ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX

LCDCCTL0 = LCDDIV\_\_1 | LCDPRE\_\_16 | LCD4MUX | LCDLP;

// VLCD generated internally,

// V2-V4 generated internally, v5 to ground

// Set VLCD voltage to 2.60v

// Enable charge pump and select internal reference for it

LCDCVCTL = VLCD\_1 | VLCDREF\_0 | LCDCPEN;

LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled

LCDCMEMCTL = LCDCLRM; // Clear LCD memory

//Turn LCD on

LCDCCTL0 |= LCDON;

**return**;

}

**void** **display\_num\_lcd**(**unsigned** **int** n)

{

**unsigned** **int** r[6], i = 0;

**while** (n != 0)

{

r[i++] = n % 10;

n /= 10;

}

**if** (i == 1)

{

LCDM8 = LCD\_Num[r[0]];

}

**else** **if** (i == 2)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

}

**else** **if** (i == 3)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

}

**else** **if** (i == 4)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

}

**else** **if** (i == 5)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

}

**else**

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

LCDM10 = LCD\_Num[r[5]];

}

}

**Questions:**

No questions were given.

**Part 2:**

This part is almost identical to Part 1. The only change that was made was adding the configuration of the ACLK to the 32KHz crystal and increment i from 0 once the program starts up.

// Sample code that implements a stopwatch

**#include** <msp430fr6989.h>

**#define** redLED BIT0 // Red at P1.0

**#define** greenLED BIT7 // Green at P9.7

**void** **Initialize\_LCD**();

**void** **increment\_stopwatch**(**unsigned** **int** n);

**void** **config\_ACLK\_to\_32KHz\_crystal**();

// The array has the shapes of the digits (0 to 9)

// Complete this array...39

**const** **unsigned** **char** LCD\_Num[10] = {0xFC, 0x60, 0xDB, 0xF3, 0x67, 0xB7, 0xBF, 0xE0, 0xFF, 0xE7};

**int** **main**(**void**) {

**volatile** **unsigned** **int** n, i=65530;

WDTCTL = WDTPW | WDTHOLD; // Stop WDT

PM5CTL0 &= ~LOCKLPM5; // Enable GPIO pins

P1DIR |= redLED; // Pins as output

P9DIR |= greenLED;

P1OUT &= ~redLED; // Red on

P9OUT &= ~greenLED; // Green off

// Initializes the LCD\_C module

Initialize\_LCD();

// Configure ACLK to the 32 KHz crystal (function call)

config\_ACLK\_to\_32KHz\_crystal();

// The line below can be used to clear all the segments

//LCDCMEMCTL = LCDCLRM; // Clears all the segments

// Configure Timer\_A

// Timer\_A: ACLK, div by 1, up mode, clear TAR

TA0CTL = TASSEL\_1 | ID\_0 | MC\_1 | TACLR;

// Ensure flag is cleared at the start

TA0CTL &= ~TAIFG;

// Set timer period

TA0CCR0 = 32738;

// Infinite loop

**for**(;;) {

// Empty while loop; waits here until TAIFG is raised

**while**(!(TA0CTL & TAIFG)) {}

increment\_stopwatch(i++);

TA0CTL &= ~TAIFG; // Clear the flag

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initializes the LCD\_C module

//\*\*\*Source: Function obtained from MSP430FR6989’s Sample Code\*\*\*

**void** **Initialize\_LCD**() {

PJSEL0 = BIT4 | BIT5; // For LFXT40

// Initialize LCD segments 0 - 21; 26 - 43

LCDCPCTL0 = 0xFFFF;

LCDCPCTL1 = 0xFC3F;

LCDCPCTL2 = 0x0FFF;

// Configure LFXT 32kHz crystal

CSCTL0\_H = CSKEY >> 8; // Unlock CS registers

CSCTL4 &= ~LFXTOFF; // Enable LFXT

**do** {

CSCTL5 &= ~LFXTOFFG; // Clear LFXT fault flag

SFRIFG1 &= ~OFIFG;

}**while** (SFRIFG1 & OFIFG); // Test oscillator fault flag

CSCTL0\_H = 0; // Lock CS registers

// Initialize LCD\_C

// ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX

LCDCCTL0 = LCDDIV\_\_1 | LCDPRE\_\_16 | LCD4MUX | LCDLP;

// VLCD generated internally,

// V2-V4 generated internally, v5 to ground

// Set VLCD voltage to 2.60v

// Enable charge pump and select internal reference for it

LCDCVCTL = VLCD\_1 | VLCDREF\_0 | LCDCPEN;

LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled

LCDCMEMCTL = LCDCLRM; // Clear LCD memory

//Turn LCD on

LCDCCTL0 |= LCDON;

**return**;

}

**void** **increment\_stopwatch**(**unsigned** **int** n)

{

**unsigned** **int** r[6], i = 0;

LCDCMEMCTL = LCDCLRM;

**while** (n != 0)

{

r[i++] = n % 10;

n /= 10;

}

**if** (i == 0)

{

LCDM8 = LCD\_Num[0];

}

**else** **if** (i == 1)

{

LCDM8 = LCD\_Num[r[0]];

}

**else** **if** (i == 2)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

}

**else** **if** (i == 3)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

}

**else** **if** (i == 4)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

}

**else** **if** (i == 5)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

}

**else**

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

LCDM10 = LCD\_Num[r[5]];

}

}

**void** **config\_ACLK\_to\_32KHz\_crystal**() {

// By default, ACLK runs on LFMODCLK at 5MHz/128 = 39 KHz

// Reroute pins to LFXIN/LFXOUT functionality

PJSEL1 &= ~BIT4;PJSEL0 |= BIT4;

// Wait until the oscillator fault flags remain cleared

CSCTL0 = CSKEY; // Unlock CS registers

**do** {

CSCTL5 &= ~LFXTOFFG; // Local fault flag

SFRIFG1 &= ~OFIFG; // Global fault flag

} **while**((CSCTL5 & LFXTOFFG) != 0);

CSCTL0\_H = 0; // Lock CS registers

**return**;

}

**Questions:**

No questions were given.

**Part 3:**

Once again, this code was nearly identical to its previous predecessor. The difference between this one and last was that it took me a little longer to full proof the code. As for code changes, we needed to configure the buttons by assigning one as the pause/resume and one as the reset. Upon doing so, we used the LED’s to confirm state changes.

// Sample code that implements a stopwatch with pause and reset

**#include** <msp430fr6989.h>

**#define** redLED BIT0 // Red at P1.0

**#define** greenLED BIT7 // Green at P9.7

**#define** BUT1 BIT1 // Button S1 at Port 1.1

**#define** BUT2 BIT2 // Button S2 at Port 1.2

**void** **Initialize\_LCD**();

**void** **increment\_stopwatch**(**unsigned** **int** n);

**void** **config\_ACLK\_to\_32KHz\_crystal**();

// The array has the shapes of the digits (0 to 9)

// Complete this array...39

**const** **unsigned** **char** LCD\_Num[10] = {0xFC, 0x60, 0xDB, 0xF3, 0x67, 0xB7, 0xBF, 0xE0, 0xFF, 0xE7};

**int** **main**(**void**) {

**volatile** **unsigned** **int** n, i=0;

WDTCTL = WDTPW | WDTHOLD; // Stop WDT

PM5CTL0 &= ~LOCKLPM5; // Enable GPIO pins

P1DIR |= redLED; // Pins as output

P9DIR |= greenLED;

P1OUT &= ~redLED; // Red on

P9OUT |= greenLED; // Green off

// Configure buttons

P1DIR &= ~(BUT1|BUT2); // input

P1REN |= (BUT1|BUT2); // enable built-in resistors

P1OUT |= (BUT1|BUT2); // Set resistor as pull-up

P1IFG &= ~(BUT1|BUT2); // Clear flags for buttons

// Initializes the LCD\_C module

Initialize\_LCD();

// Configure ACLK to the 32 KHz crystal (function call)

config\_ACLK\_to\_32KHz\_crystal();

// The line below can be used to clear all the segments

//LCDCMEMCTL = LCDCLRM; // Clears all the segments

// Configure Timer\_A

// Timer\_A: ACLK, div by 1, up mode, clear TAR

TA0CTL = TASSEL\_1 | ID\_0 | MC\_1 | TACLR;

// Ensure flag is cleared at the start

TA0CTL &= ~TAIFG;

// Set timer period

TA0CCR0 = 32738;

// Infinite loop

**for**(;;) {

// Empty while loop; waits here until TAIFG is raised

**while**(!(TA0CTL & TAIFG)) {}

**if** ((BUT1 & P1IFG) == BUT1)

{

// Turn on the red LED

P1OUT |= redLED;

// Turn off the green LED

P9OUT &= ~greenLED;

// Clear BUT1 in P1IFG

P1IFG &= ~BUT1;

**for**(;;)

{

**if** ((BUT1 & P1IFG) == BUT1)

**break**;

**if** ((BUT2 & P1IFG) == BUT2)

{

i = 0;

increment\_stopwatch(i);

}

}

// Turn off the red LED

P1OUT &= ~redLED;

// Turn on the green LED

P9OUT |= greenLED;

// Clear BUT1 in P1IFG

P1IFG &= ~BUT1;

}

**if** ((BUT2 & P1IFG) == BUT2)

{

i = 0;

// Clear BUT2 in P1IFG

P1IFG &= ~BUT2;

}

increment\_stopwatch(i++);

TA0CTL &= ~TAIFG; // Clear the flag

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initializes the LCD\_C module

//\*\*\*Source: Function obtained from MSP430FR6989’s Sample Code\*\*\*

**void** **Initialize\_LCD**() {

PJSEL0 = BIT4 | BIT5; // For LFXT40

// Initialize LCD segments 0 - 21; 26 - 43

LCDCPCTL0 = 0xFFFF;

LCDCPCTL1 = 0xFC3F;

LCDCPCTL2 = 0x0FFF;

// Configure LFXT 32kHz crystal

CSCTL0\_H = CSKEY >> 8; // Unlock CS registers

CSCTL4 &= ~LFXTOFF; // Enable LFXT

**do** {

CSCTL5 &= ~LFXTOFFG; // Clear LFXT fault flag

SFRIFG1 &= ~OFIFG;

}**while** (SFRIFG1 & OFIFG); // Test oscillator fault flag

CSCTL0\_H = 0; // Lock CS registers

// Initialize LCD\_C

// ACLK, Divider = 1, Pre-divider = 16; 4-pin MUX

LCDCCTL0 = LCDDIV\_\_1 | LCDPRE\_\_16 | LCD4MUX | LCDLP;

// VLCD generated internally,

// V2-V4 generated internally, v5 to ground

// Set VLCD voltage to 2.60v

// Enable charge pump and select internal reference for it

LCDCVCTL = VLCD\_1 | VLCDREF\_0 | LCDCPEN;

LCDCCPCTL = LCDCPCLKSYNC; // Clock synchronization enabled

LCDCMEMCTL = LCDCLRM; // Clear LCD memory

//Turn LCD on

LCDCCTL0 |= LCDON;

**return**;

}

**void** **increment\_stopwatch**(**unsigned** **int** n)

{

**unsigned** **int** r[6], i = 0;

LCDCMEMCTL = LCDCLRM;

**while** (n != 0)

{

r[i++] = n % 10;

n /= 10;

}

**if** (i == 0)

{

LCDM8 = LCD\_Num[0];

}

**else** **if** (i == 1)

{

LCDM8 = LCD\_Num[r[0]];

}

**else** **if** (i == 2)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

}

**else** **if** (i == 3)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

}

**else** **if** (i == 4)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

}

**else** **if** (i == 5)

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

}

**else**

{

LCDM8 = LCD\_Num[r[0]];

LCDM15 = LCD\_Num[r[1]];

LCDM19 = LCD\_Num[r[2]];

LCDM4 = LCD\_Num[r[3]];

LCDM6 = LCD\_Num[r[4]];

LCDM10 = LCD\_Num[r[5]];

}

}

**void** **config\_ACLK\_to\_32KHz\_crystal**() {

// By default, ACLK runs on LFMODCLK at 5MHz/128 = 39 KHz

// Reroute pins to LFXIN/LFXOUT functionality

PJSEL1 &= ~BIT4;PJSEL0 |= BIT4;

// Wait until the oscillator fault flags remain cleared

CSCTL0 = CSKEY; // Unlock CS registers

**do** {

CSCTL5 &= ~LFXTOFFG; // Local fault flag

SFRIFG1 &= ~OFIFG; // Global fault flag

} **while**((CSCTL5 & LFXTOFFG) != 0);

CSCTL0\_H = 0; // Lock CS registers

**return**;

}

**Questions:**

No questions were given.

**Student Q&A:**

1. Explain whether this statement is true or false. If false, explain the correct operation. “an LCD segment is given 1 to turn it on and 0 to turn it off, just like the colored LEDs”. Explain whether it’
   1. **The LCD segment is not given a 1 to turn it on neither does it give a 0 to turn it off. To ‘turn on’ an LCD segment you need to give the right hexadecimal code to configure the correct digit or letter you wish to display.**
2. What is the name of the LCD controller the LaunchPad uses to interface the LCD display? Is the LCD controller located on the display module or in the microcontroller?
   1. **The name of the LCD controller the LaunchPad uses to interface the LCD display is the LCD\_C module. It is located inside the MCU.**
3. In what multiplexing configuration is the LCD module wired (2-way, 4-way, etc)? What does this mean regarding the number of pins used at the microcontroller?
   1. **The LCD module is wired to 4-way multiplexing. This means that for the 108 pins there are 31 lines of wiring to control the LCD segments.**

**Conclusion:**

This lab was extremely informational. Part 1 of the lab carries the most significant source of information. It basically showed us what the LCD display is capable of and gave us much of the specifications for the LCD in which is required to operate the module. Part 2 required more knowledge from previous labs. The information was not handfed to you, which required you to derive up your own solution to the task. Part 3 was a recap of buttons and once again was a task where it required you to think for yourself. I am starting to see that as we progress, we are on our own with all the knowledge we have been given from previous labs.