**EGR 226: Microcontroller Programming and Applications**

**Winter 2021**

Instructor: Professor Trevor Ekins

Lab 7: Liquid Crystal Display

Sam Wieneke

March 17, 2021

**Contents**

1. **Objectives**
2. **Equipment**
3. **Introduction**
4. **Procedure**
5. **Results/ Discussion**
6. **Conclusion and Future Work**

**Appendices**

1. **Source Code: Part 1**

**Source Code: Part 2**

**Source Code: Part 3**

1. **Objectives**

The objectives of this experiment were to interface with an LCD with an MCU using I/O signals for data and control and to employ the MSP432 SysTick timer to create precise timing intervals.

1. **Equipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Description | Model | Measured Value | Notes |
| Code Composer Studios | Texas Instruments programming environment | Version 9.3.0 | N/A | N/A |
| GitLab | Remote Repository for code maintenance | N/A | N/A | Makes collaboration on team projects and code very convenient. |
|  |  |  |  |  |

1. **Introduction**

**Part 1: Blinking the cursor on the LCD**

Part 1 involved using SysTick to generate the required delays to initialize the LCD. When the initialization process was done correctly, the LCD cursor was blinking at the top left of the screen.

**Part 2: Displaying name on the LCD**

Part 2 involved displaying words on the LCD. There was a word on each line (first name, last name, “EGR”, “226”). C strings were required to get full credit for this section. The words had to be centered on the LCD and in the order specified. If this section was done correctly it would look like a nametag for this class.

**Part 3: Scrolling Marquee**

Part 3 involved modifying the program from part 2 to only display a scrolling message on the top of the screen. It was supposed to display “LABORATORY OVER” and would start with the entire message being displayed on the top row and then would shift over 1 character every second (or shorter if desired). Once the entire message was shifted over so no characters were left, it scrolled back in from the right side of the screen 1 character at a time and will continue forever.

1. **Procedure**

**Part 1**

Part 1 was created by initializing all of the pins that are connected to the LCD (P4.0, 4.2, 4.4-7). This was done by setting each pin as an output by using |=. The LCD was then initialized using commands given in the lab directions but changing them to fit the names given in the code. This was the heart of the code because in the main only 3 functions were called: LCD\_pins, SYSTick\_Init, and LCD\_init. The commandWrite function was used to get the LCD to perform certain functions. CommandWrite(3) was part of the reset sequence, (2) was setting the LCD to 4 bit mode, (0x28) was setting it to 4 lines in a 5x7 format, (0x0F) set the display on with the cursor on and blinking, (0x01) cleared the display, and (0x06) incremented the cursor.

**Part 2**

Part 2 involved copying part 1 and editing it. 4 char’s had to be initialized with each char being what the user wanted on each line of the LCD. After this a for loop was created to display each character on the line.

commandWrite(0x84); // Setting cursor to 5th spot on line 1

**for**(i = 0; i < **strlen**(firstName) ; i++)

{

dataWrite(firstName[i]);

}

This code shows how the top line was written on the LCD. Basically it incremented though all the letters and started where the cursor was placed so the word would be roughly centered in the LCD. After this section of code was completed almost completely identical 3 other for loops were created for the other 3 lines on the LCD.

**Part 3**

Part 3 was a step up from part 2. It only had one line on the LCD for a string but the string was on a constant loop that shifted it over 1 character to the left at a time until they were all gone and then re-shifted them in from the right. This code was similar to part 2 because it started off displaying the line of code on the LCD but then it had a while loop with a for loop inside of it that shifted the characters over. The commandWrite(0x18) function was used to shift the entire line of code over and a delay was placed after it so it would move 1 character over every half a second.

**for**(j=0; j<**strlen**(firstline); j++)

{

dataWrite(firstline[j]);

delay\_milli(500); //half second delay between shifting letters

commandWrite(0x18);

delay\_milli(4);

}

This code was the heart of the function with a few extra touches to make sure it didn’t carry over onto line 3. This code prints the string to the LCD then delays for a half second and then moves everything over 1 and continues in this matter until all the letters are gone. It was inside a while loop that ran forever so this code will continue for as long as needed.

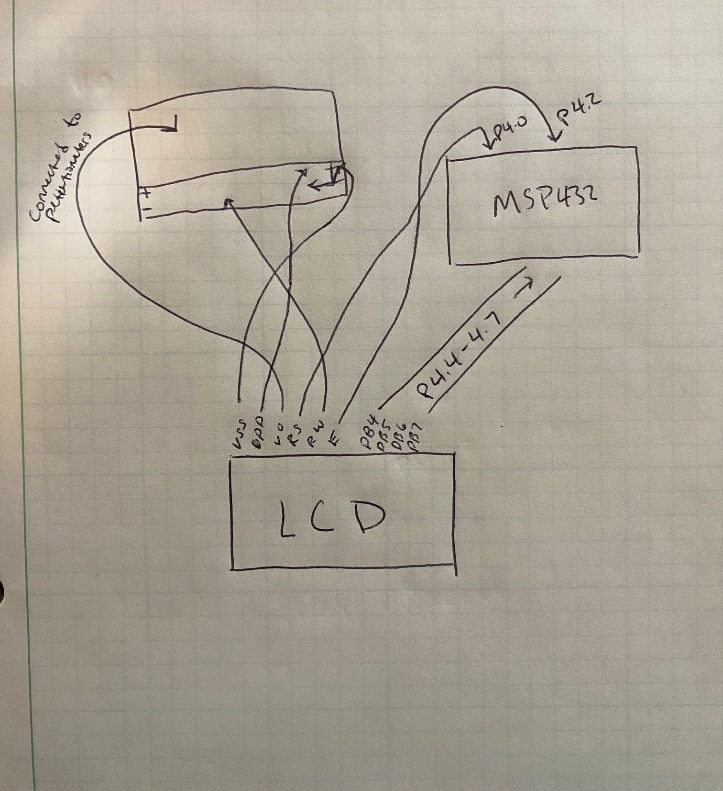
1. **Results/ Discussion**

**Lab Results**

There were no questions in the lab handout and the code was competed correctly and the demonstrations went smoothly. Everything worked as planned and there was no error checking that was needed for this lab as it was designed to only print certain things and not ask for any prompt.

**Prelab:**

The prelab involved drawing a circuit diagram of how the LCD would be setup and writing the program to carry out the initialization steps. A character is displayed on the LCD using the function dataWrite() and this sends the character as hex and in pieces with the upper nibble and the lower nibble.



1. **Conclusion/ Future Work**

This lab was the first use of the LCD. Code was written do display different messages to the screen and the messages perform different functions. Some challenges that were faced in the lab was figuring out a way for part 3 to move all the characters over 1 space and then restarting them when all the characters were moved over. Am improvement to the code might have have moved the “LABORATORY OVER” from the top line to a line lower, and would continue in that fashion instead of only repeating on the top line but this would have been more advanced but this concept could be used for the final project if applicable.

**Appendix A**

Part 1 Source Code:

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

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 7

\* File:

\* Description: This program will display various characters to the LCD

\*

\*

\*

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

**#include** "msp.h"

**#include** <stdio.h>

**void** **LCD\_pins**(**void**);

**void** **SysTick\_Init** (**void**);

**void** **SysTick\_delay** (uint16\_t delay);

//uint8\_t Read\_Keypad(void);

**void** **LCD\_init** (**void**);

**void** **delay\_micro**(uint32\_t microsecond);

**void** **delay\_milli**(uint32\_t millisecond);

**void** **PulseEnablePin** (**void**);

**void** **pushNibble** (uint8\_t nibble);

**void** **pushByte** (uint8\_t byte);

**void** **commandWrite**(uint8\_t command);

**void** **dataWrite**(uint8\_t data);

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

LCD\_pins();

SysTick\_Init();

LCD\_init();

**while**(1)

{

}

}

/\*

uint8\_t Read\_Keypad(void)

{

uint8\_t col, row;

for ( col = 0; col < 3; col++ )

{

P4->DIR = 0x00; // Set Columns to inputs

P4->DIR |= BIT ( 4+col ); // Set column 3 to output

P4->OUT &= ~BIT ( 4+col ); // Set column 3 to LOW

SysTick\_delay (10); // Delay the while loop

row = P4->IN & 0x0F; // read all rows

while ( !(P4IN & BIT0) | !(P4IN & BIT1) | !( P4IN & BIT2) | !( P4IN & BIT3) );

if (row != 0x0F) break; // if one of the input is low, some key is pressed.

}

P4->DIR = 0x00; // Set Columns to inputs

if ( col == 3)

return 0;

if (row == 0x0E) num = col + 1; // key in row 0

if (row == 0x0D) num = 3 + col +1; // key in row 1

if (row == 0x0B) num = 6 + col +1; // key in row 2

if (row == 0x07) num = 9 + col+1; // key in row 3

return 1;

}

\*/

**void** **LCD\_pins**(**void**)

{

//RS

P4->SEL1 &= ~BIT0; // configure P4.0 as simple I/O

P4->SEL0 &= ~BIT0;

P4->DIR |= BIT0; // P4.0 set as output pin \*/

//E

P4->SEL1 &= ~BIT2; // configure P2.4 as simple I/O

P4->SEL0 &= ~BIT2;

P4->DIR |= BIT2; // P4.2 set as output pin \*/

//DB4

P4->SEL1 &= ~BIT4; // configure P4.4 as simple I/O

P4->SEL0 &= ~BIT4;

P4->DIR |= BIT4; // P4.4 set as output pin \*/

//DB5

P4->SEL1 &= ~BIT5; // configure P4.5 as simple I/O

P4->SEL0 &= ~BIT5;

P4->DIR |= BIT5; // P4.5 set as output pin \*/

//DB6

P4->SEL1 &= ~BIT6; // configure P4.6 as simple I/O

P4->SEL0 &= ~BIT6;

P4->DIR |= BIT6; // P4.6 set as output pin \*/

//DB7

P4->SEL1 &= ~BIT7; // configure P4.7 as simple I/O

P4->SEL0 &= ~BIT7;

P4->DIR |= BIT7; // P4.7 set as output pin \*/

}

**void** **LCD\_init** (**void**)

{

commandWrite(3);

delay\_milli(20);

commandWrite(3);

delay\_micro(200);

commandWrite(3);

delay\_milli(20);

commandWrite(2);

delay\_micro(100);

commandWrite(0x28);

delay\_milli(100);

delay\_micro(10);

commandWrite(0x0F);

delay\_micro(100);

commandWrite(0x01);

delay\_micro(120);

commandWrite(0x06);

delay\_micro(100);

}

**void** **delay\_milli**(uint32\_t millisecond)

{

SysTick -> LOAD = (millisecond\*3000 - 1); // delay\*3000

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}

**void** **delay\_micro**(uint32\_t microsecond)

{

SysTick -> LOAD = (microsecond\*3 - 1); // delay\*3

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}

**void** **PulseEnablePin** (**void**)

{

//E is P4.2

P4OUT &=~BIT2; // make sure pulse starts out at 0V

delay\_micro(10);

P4OUT |=BIT2;

delay\_micro(10);

P4OUT &=~BIT2;

delay\_micro(10);

}

**void** **pushNibble** (uint8\_t nibble)

{

P4OUT &= ~0xF0; // clear P4.4-P4.7

P4OUT |= (nibble & 0x0F) << 4; // port pins P4.4 - P4.7 wired to D4 - D7

PulseEnablePin();

}

**void** **pushByte** (uint8\_t byte)

{

uint8\_t nibble;

nibble = (byte & 0xF0) >> 4;

pushNibble(nibble);

nibble = byte & 0x0F;

pushNibble(nibble);

delay\_micro(100);

}

**void** **commandWrite**(uint8\_t command)

{

P4OUT &= ~BIT0;

pushByte(command & 0xF0);

pushByte(command & 0x0F);

delay\_milli(4);

}

**void** **dataWrite**(uint8\_t data)

{

P4OUT |= ~BIT0;

pushByte(data & 0xF0);

pushByte(data & 0x0F);

delay\_milli(4);

}

**void** **SysTick\_Init** (**void**)

{ //initialization of systic timer

SysTick->CTRL = 0; // disable SysTick During step

SysTick->LOAD = 0x00FFFFFF; // max reload value

SysTick->VAL = 0; // any write to current clears it

SysTick->CTRL = 0x00000005; // enable systic, 3MHz, No Interrupts

}

**void** **SysTick\_delay** (uint16\_t delay)

{ // Systick delay function

SysTick->LOAD = ((delay \* 3000) - 1); //delay for 1 msecond per delay value

SysTick->VAL = 0; // any write to CVR clears it

**while** ( (SysTick->CTRL & 0x00010000) == 0); // wait for flag to be SET

}

Part 2 Source Code:

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

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 7

\* File:

\* Description: This program will display various characters to the LCD

\*

\*

\*

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

**#include** "msp.h"

**#include** <stdio.h>

**#include** <string.h>

**void** **SysTick\_Init**(**void**);

**void** **LCD\_init**(**void**);

**void** **SysTick\_delay**(uint16\_t delay);

**void** **commandWrite**(uint8\_t command);

**void** **pushNibble**(uint8\_t data, uint8\_t control);

**void** **dataWrite**(uint8\_t data);

**void** **delay\_micro**(uint32\_t microsecond);

**void** **delay\_milli**(uint32\_t millisecond);

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

{

SysTick\_Init();

LCD\_init();

**char** firstName[6]={'S', 'A', 'M', 'U', 'E', 'L'};

**char** lastName[7]={'W', 'I', 'E', 'N', 'E', 'K', 'E'};

**char** line3[3]={'E', 'G', 'R'};

**char** line4[3]={'2', '2', '6'};

**int** i;

commandWrite(1); // Clear display

commandWrite(0x84); // Setting cursor to 5th spot on line 1

**for**(i = 0; i < **strlen**(firstName) ; i++)

{

dataWrite(firstName[i]);

}

commandWrite(0xC4); // Setting cursor to 5th spot on line 2

**for**(i = 0; i < **strlen**(lastName); i++)

{

dataWrite(lastName[i]);

}

commandWrite(0x96); // Setting cursor to 7th spot on line 3

**for**(i = 0; i < **strlen**(line3); i++)

{

dataWrite(line3[i]);

}

commandWrite(0xD6); // Setting cursor to 7th spot on line 4

**for**(i = 0; i < **strlen**(line4); i++)

{

dataWrite(line4[i]);

}

**while**(1)

{

delay\_milli(500);

}

}

/\*\*\*\*| SysTick\_Init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the SysTick timer

\* param:

\* (void) data: N/A

\* return:

\* N/A

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

**void** **SysTick\_Init**(**void**)

{

SysTick -> CTRL = 0; // Disable SysTick

SysTick -> LOAD = 0x00FFFFFF; // Max reload value

SysTick -> VAL = 0; // Any write to current clears it

SysTick -> CTRL = 0x00000005; // Enable systick, 3MHz, no interrupts

}

/\*\*\*\*| LCD\_init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the LCD

\* param:

\* (void) data: N/A

\* return:

\* N/A

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

**void** **LCD\_init**(**void**)

{

// Setting pins 4.0, 4.2, 4.4-4.7 to output

P4->DIR |= 0xF5; //0b 1111 0101

commandWrite(3); //reset sequence

delay\_milli(20);

commandWrite(3);

delay\_micro(200);

commandWrite(3);

delay\_milli(20);

commandWrite(2); //setting 4 bit mode

delay\_micro(100);

commandWrite(0x28); //4 limes 5x7 format

delay\_milli(100);

delay\_micro(10);

commandWrite(0x0F); //display on, cursor on, blinking

delay\_micro(100);

commandWrite(0x01); //clear display

delay\_micro(120);

commandWrite(0x06); //increment cursor

delay\_micro(100);

}

/\*\*\*\*| SysTick\_delay | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function sets the SysTick timer to be used as a

\* delay when called

\* param:

\* (uint16\_t) data: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **SysTick\_delay**(uint16\_t delay)

{

SysTick->LOAD = ((delay \* 3000) - 1); // Delay 1 msecond per delay value

SysTick->VAL = 0; // Any write to CVR clears it

**while** ((SysTick->CTRL & 0x00010000) == 0); // Wait for flag to be SET

}

/\*\*\*\*| pushNibble | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function pushes 1 nibble onto the data pins and

\* pulses the enable pin

\* param:

\* (uint8\_t) data & control: two integers worth of data used to assign bits

\* to data pins, tutor helped with this and edited to fit code in class because code given did not work for whatever reason.

\* return:

\* N/A

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

**void** **pushNibble**(uint8\_t data, uint8\_t control)

{

P4->OUT &= ~0x10; // Set RS = 0

data &= 0xF0; // Assign most significant 4 bits to data pins

control &= 0x0F; // Assign least significant 4 bits to data pins

P4->OUT = data | control | 4; // Pulse enable pin

SysTick\_delay(1);

P4->OUT = data; // Clear enable pin

P4->OUT = 0;

}

/\*\*\*\*| commandWrite | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function writes one byte of command by calling

\* the pushNibble() function twice with the command parameter

\* param:

\* (uint8\_t) data: one integer worth of data that gives a

\* command to the LCD

\* return:

\* N/A

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

**void** **commandWrite**(uint8\_t command)

{

pushNibble(command & 0xF0, 0); // Sending upper nibble

pushNibble(command << 4, 0); // Sending lower nibble

delay\_milli(4);

}

/\*\*\*\*| dataWrite | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function writes one byte of data by calling

\* the pushNibble() function twice with the data parameter

\* param:

\* (void) data: one unsigned char worth of data to be sent

\* to the LCD

\* return:

\* N/A

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

**void** **dataWrite**(**unsigned** **char** data)

{

pushNibble(data & 0xF0, 1); // Sending upper nibble

pushNibble(data << 4, 1); // Sending lower nibble

SysTick\_delay(1);

}

/\*\*\*\*| delay\_milli | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function uses Systick to get a millisecond delay

\*

\* param:

\* (uint32\_t) millisecond: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **delay\_milli**(uint32\_t millisecond)

{

SysTick -> LOAD = (millisecond\*3000 - 1); // delay\*3000

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}

/\*\*\*\*| delay\_micro | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function uses Systick to get a microsecond delay

\*

\* param:

\* (uint32\_t) microsecond: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **delay\_micro**(uint32\_t microsecond)

{

SysTick -> LOAD = (microsecond\*3 - 1); // delay\*3

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}

Part 3 Source Code:

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

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 7

\* File:

\* Description: This program will display various characters to the LCD

\*

\*

\*

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

**#include** "msp.h"

**#include** <stdio.h>

**#include** <string.h>

**void** **SysTick\_Init**(**void**);

**void** **LCD\_init**(**void**);

**void** **SysTick\_delay**(uint16\_t delay);

**void** **commandWrite**(uint8\_t command);

**void** **pushNibble**(uint8\_t data, uint8\_t control);

**void** **dataWrite**(uint8\_t data);

**void** **delay\_micro**(uint32\_t microsecond);

**void** **delay\_milli**(uint32\_t millisecond);

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

{

SysTick\_Init();

LCD\_init();

**char** firstline[16]={'L', 'A', 'B', 'O', 'R', 'A', 'T', 'O', 'R', 'Y', ' ', 'O', 'V', 'E', 'R'};

**int** i,j;

commandWrite(1); // Clear display

commandWrite(0x80); // Setting cursor to 1st spot on line 1

delay\_milli(4);

**for**(i = 0; i < **strlen**(firstline) ; i++)

{

dataWrite(firstline[i]);

delay\_milli(4);

}

commandWrite(0x0C); //turns off cursor

delay\_milli(500); //delay for half second

**while**(1)

{

**for** (i=0; i<**strlen**(firstline); i++) //shifts the string left

{

commandWrite(0x18);

delay\_milli(500); //half second delay between each shift

**if**(i==7) //there is 7 spaces needed so it doesn't carry over onto line 3

{

commandWrite(0x80);

delay\_milli(10);

dataWrite(' ');

dataWrite(' ');

dataWrite(' ');

dataWrite(' ');

dataWrite(' ');

dataWrite(' ');

dataWrite(' ');

delay\_milli(4);

}

}

commandWrite(0x01); //clears LCD

delay\_milli(100);

**for**(j=0; j<**strlen**(firstline); j++)

{

commandWrite(0x1C);

delay\_milli(4);

}

**for**(j=0; j<**strlen**(firstline); j++)

{

dataWrite(firstline[j]);

delay\_milli(500); //half second delay between shifting letters

commandWrite(0x18);

delay\_milli(4);

}

delay\_milli(500);

}

}

/\*\*\*\*| SysTick\_Init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the SysTick timer

\* param:

\* (void) data: N/A

\* return:

\* N/A

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

**void** **SysTick\_Init**(**void**)

{

SysTick -> CTRL = 0; // Disable SysTick

SysTick -> LOAD = 0x00FFFFFF; // Max reload value

SysTick -> VAL = 0; // Any write to current clears it

SysTick -> CTRL = 0x00000005; // Enable systick, 3MHz, no interrupts

}

/\*\*\*\*| LCD\_init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the LCD

\* param:

\* (void) data: N/A

\* return:

\* N/A

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

**void** **LCD\_init**(**void**)

{

// Setting pins 4.0, 4.2, 4.4-4.7 to output

P4->DIR |= 0xF5; //0b 1111 0101

commandWrite(3); //reset sequence

delay\_milli(20);

commandWrite(3);

delay\_micro(200);

commandWrite(3);

delay\_milli(20);

commandWrite(2); //setting 4 bit mode

delay\_micro(100);

commandWrite(0x28); //4 limes 5x7 format

delay\_milli(100);

delay\_micro(10);

commandWrite(0x0F); //display on, cursor on, blinking

delay\_micro(100);

commandWrite(0x01); //clear display

delay\_micro(120);

commandWrite(0x06); //increment cursor

delay\_micro(100);

}

/\*\*\*\*| SysTick\_delay | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function sets the SysTick timer to be used as a

\* delay when called

\* param:

\* (uint16\_t) data: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **SysTick\_delay**(uint16\_t delay)

{

SysTick->LOAD = ((delay \* 3000) - 1); // Delay 1 msecond per delay value

SysTick->VAL = 0; // Any write to CVR clears it

**while** ((SysTick->CTRL & 0x00010000) == 0); // Wait for flag to be SET

}

/\*\*\*\*| pushNibble | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function pushes 1 nibble onto the data pins and

\* pulses the enable pin

\* param:

\* (uint8\_t) data & control: two integers worth of data used to assign bits

\* to data pins, tutor helped with this and edited to fit code in class because code given did not work for whatever reason.

\* return:

\* N/A

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

**void** **pushNibble**(uint8\_t data, uint8\_t control)

{

P4->OUT &= ~0x10; // Set RS = 0

data &= 0xF0; // Assign most significant 4 bits to data pins

control &= 0x0F; // Assign least significant 4 bits to data pins

P4->OUT = data | control | 4; // Pulse enable pin

SysTick\_delay(1);

P4->OUT = data; // Clear enable pin

P4->OUT = 0;

}

/\*\*\*\*| commandWrite | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function writes one byte of command by calling

\* the pushNibble() function twice with the command parameter

\* param:

\* (uint8\_t) data: one integer worth of data that gives a

\* command to the LCD

\* return:

\* N/A

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

**void** **commandWrite**(uint8\_t command)

{

pushNibble(command & 0xF0, 0); // Sending upper nibble

pushNibble(command << 4, 0); // Sending lower nibble

delay\_milli(4);

}

/\*\*\*\*| dataWrite | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function writes one byte of data by calling

\* the pushNibble() function twice with the data parameter

\* param:

\* (void) data: one unsigned char worth of data to be sent

\* to the LCD

\* return:

\* N/A

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

**void** **dataWrite**(**unsigned** **char** data)

{

pushNibble(data & 0xF0, 1); // Sending upper nibble

pushNibble(data << 4, 1); // Sending lower nibble

SysTick\_delay(1);

}

/\*\*\*\*| delay\_milli | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function uses Systick to get a millisecond delay

\*

\* param:

\* (uint32\_t) millisecond: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **delay\_milli**(uint32\_t millisecond)

{

SysTick -> LOAD = (millisecond\*3000 - 1); // delay\*3000

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}

/\*\*\*\*| delay\_micro | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function uses Systick to get a microsecond delay

\*

\* param:

\* (uint32\_t) microsecond: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

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

**void** **delay\_micro**(uint32\_t microsecond)

{

SysTick -> LOAD = (microsecond\*3 - 1); // delay\*3

SysTick -> VAL = 0; //clears counter

**while**((SysTick -> CTRL & 0x00010000) == 0);

}