TA Andrew Gerber 10/1/2023

Lab 3 Report

Benjamin Meads
A02323437
Nick Templeton
A02268123
Richard Snider
A02298443

Objectives

The purpose of this lab is to familiarize with two main things. We are writing a microcontroller program in C code. This means we need to edit our GPIO pins, the clock, and the logic using C code. The other thing we needed to familarize ourselves with was the LCD screen. Being able to set it up, and configure the LCD screen to the 4-pin mode. We needed to set our clock down to 16 MHz. Then initialize our LCD to 4-bit, 2 lines, and get the 5x8 matrix. We needed to get the cursor to respond correctly. Then to get our different functions set up. This is all done in the initialization function. A write function to change the settings of the LCD screen. A write communications function to actually display content onto the screen. Then a DisplayString function to configure how the data is shown on the screen.

Procedure

This lab was actually a very smooth process. We were able to very quickly see how the lab was supposed to be laid out. We were able to very quickly power through getting the clocks initialized using what was given in class slides. We were quickly able to transform the assembly code we've used in the past into the c code. The transformations we made worked very well for the MODER and PUPDR. We were able to look through the other files given to us to quickly find which lines were applicable. Then we were able to set up. In this process, we found that setting and editing the bit masking using a dummy variable. Then storing that information back in one instruction is a much better method. This way we don't have any time in an intermediate state where the data is not correct.

Setting up our functions in our initialization stage was pretty straightforward for us. We were able to quickly find the parts of the technical document to show us what our commands needed to be. Then we were able to call our write command functions to actually send those commands to our LCD. Setting up the WriteCOM and WriteData functions were almost identical. We first split up the function input into 2 different variables. One of them concerning the higher bits, and the other for the lower bits. We then set our E variable to be high while we edit our information. Then When we're ready to send our command/data, we set E to low to actually execute the command/data. These functions went by without any real issues. The last function was our LCD_DisplayString. We did have an issue with this function. Our for loop where we actually write the data. We initially had our condition to continue for the input string to be 16 bits long which is the width of the screen. The other condition is a null string. This didn't end well, because when we got our null character, it kept printing out nonsense characters because it was after the null character. But after fixing it, the screen displayed exactly what we wanted.

We were able to get this lab written out and have everything ready to go at the end of our first lab session. We were very close to actually being able to pass off. However, for some reason, our screen wouldn't display what we were expecting. We went over our code in the debugger several times with no success. When I was going through the code with a friend, he recommended we plug in his LCD screen. Which very quickly displayed exactly what we were looking for. So all the time debugging was just because my solder job wasn't good and my LCD didn't display anything.

Figures

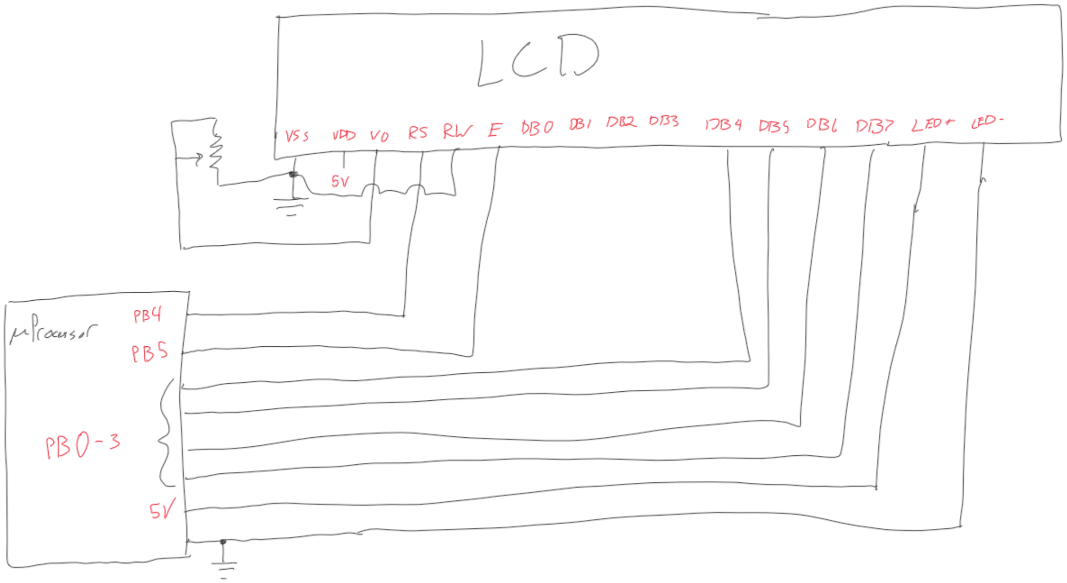

Fig 1: Schematic for LCD screen set up

Results

This lab went by much smoother than previous labs. We had 3 lab partners and we were able to really come together and make immediate progress. This lab went by without too many complications. However, the one problem with the LCD screen really set us back. But once we plugged in a separate LCD screen, we immediately got the result we were looking for.

Conclusion

Having 3 lab partners was amazing. We were all able to bring something to the table. We all had something that others in the group didn’t. This made this lab a quick and seamless process. This lab was really good for fully understanding how to use C code to manipulate the clock, GPIO pins, and everything else with the microcontroller. We will be able to use this setup and program for our next lab.

Program

```
#include "LCD.h"
#include "stm32l476xx.h"

void delay_ms(unsigned int ms) { //Delays "ms" milliseconds
	volatile unsigned int i,j;
	for(i=0; i < ms ;i++)
	{
		for(j=0; j < 800; j++);
	}
}

/*
Enable GPIO clock and configures GPIO Pins to control the 4-bit bus, E, RW, and RS lines.
Then sends the following commands to LCD
1. 4-bit bus, 2-line display mode, 5x8 dot matrix
	2. display on, cursor off, cursor blink off
	3. cursor moves left -> right, no display scrolling
	4. clear dislay
	5. Set up CGRAM address to start at 0
*/
void LCD_Init(void){
	//Setting the clock to HSI 16MHz
	RCC->CR |= RCC_CR_HSION;
	while ((RCC->CR & RCC_CR_HSIRDY) == 0);
	//Enable clock to GPIO B
	RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;
```

```
        //Sets PB0-5 to be outputs
        //writing to dummy variable
        uint32_t new_mode = GPIOB->MODER;
        //changing dummy variable
        new_mode &= (0XFFFFF000);
        new_mode |= (0X555);
        //writing dummy variable back to MODER
        GPIOB->MODER = new_mode;

        uint32_t new_pupdr = GPIOB->PUPDR;
        //changing dummy variable
        new_pupdr &= (0XFFFFF000);
        new_pupdr |= (0X555);
        //writing dummy variable back to PUPDR
        GPIOB->PUPDR = new_pupdr;



        //setting 4 bit mode
        uint32_t temp_GPIOB = GPIOB->ODR;
        //                                                                                   Setting
upper bits of command
        //Clear lower 16 bits
        temp_GPIOB &= (0xFFFF0000);
        //Set E = 1, RS = 0, and lower 4 bits equal to upper_com
        temp_GPIOB |= (0x00000022);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //
Execute
        //Set E = 0
        temp_GPIOB &= (0xFFFFFFDF);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //

        LCD_WriteCom(0x28); //2-line display mode, and 5x8 dot matrix
        LCD_WriteCom(0x0C);         //Display on, cursor off, cursor blink off
        LCD_WriteCom(0x14); //cursor moves to the right, no display scrolling
        LCD_WriteCom(0x01); //Clear out the display
        LCD_WriteCom(0x40); //Setting CGRAM address to start at 0
}

//writes command in com to the LCD. See timing diagram & reference code in LCD datasheet
void LCD_WriteCom(unsigned char com) {
        //Our commands will upper_com
        uint8_t upper_com, lower_com;
        upper_com = com&0xf0;
        lower_com = (com<<4)&0xf0;
        //Set up dummy variable
        uint32_t temp_GPIOB = GPIOB->ODR;
        //                                                                           Setting
upper bits of command
        //Clear lower 16 bits
        temp_GPIOB &= (0xFFFF0000);
        //Set E = 1, RS = 0, and lower 4 bits equal to upper_com
        temp_GPIOB |= (0x00000020) | (upper_com>>4);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //
Execute
        //Set E = 0
        temp_GPIOB &= (0xFFFFFFDF);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);

        //                                                      Setting lower bits of
command
        // clear out bottom 4 command bits
        temp_GPIOB &= (0xFFFFFFF0);
        //Set E = 1, RS = 0, and lower 4 bits equal to lower_com
        temp_GPIOB |= (0x00000020) | (lower_com>>4);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //                                                                    Execute
        //Set E = 0
        temp_GPIOB &= (0xFFFFFFDF);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
}

//writes data in "dat" to LCD. See timing diagram & reference code in LCD datasheet
void LCD_WriteData(unsigned char dat) {
        uint8_t upper_dat, lower_dat;
        upper_dat = dat&0xf0;
        lower_dat = (dat<<4)&0xf0;
        //Set up dummy variable
        uint32_t temp_GPIOB = GPIOB->ODR;
        //                                                                    Setting
upper bits of command
        //Clear lower 16 bits
        temp_GPIOB &= (0xFFFF0000);
        //Set E = 1, RS = 1, and lower 4 bits equal to upper_dat
        temp_GPIOB |= (0x00000030) | (upper_dat>>4);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //
Execute
        //Set E = 0
        temp_GPIOB &= (0xFFFFFFDF);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);

        //                                                    Setting lower bits of
command
        // clear out bottom 4 data bits
        temp_GPIOB &= (0xFFFFFFF0);
        //Set E = 1, RS = 1, and lower 4 bits equal to lower_dat
        temp_GPIOB |= (0x00000030) | (lower_dat>>4);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
        //                                                              Execute
        //Set E = 0
        temp_GPIOB &= (0xFFFFFFDF);
        //Write dummy variable back to GPIOB ODR
        GPIOB->ODR = temp_GPIOB;

        delay_ms(4);
}

//clears the LCD
void LCD_Clear(void){
  LCD_WriteCom(0x01); //Clear out the display
}

//Displays text on the LCD, where "line" is the line number (0 or 1) and "*ptr" is a pointer to a
C-tyle string (i.e. null character terminated)
void LCD_DisplayString(unsigned int line, unsigned char *ptr) {
        uint8_t i;
        //If line = 0, Display on first line
        if(line==0){
                LCD_WriteCom(0x80);
        }
        //If line = 1, Display on second line
        else{
                LCD_WriteCom(0xC0);
        }
        //step through array and print until 16 bits or null character
        for(i=0;ptr[i]!='\0';i++)
        {
                LCD_WriteData(ptr[i]);
        }

}
```