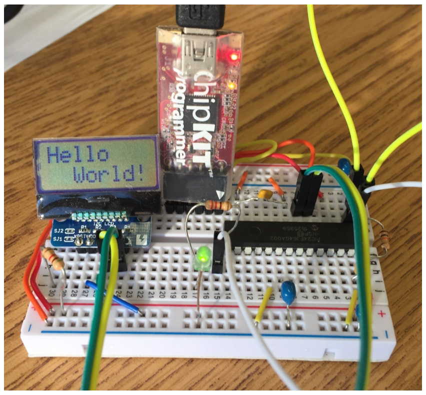
Laboratory #5



I2C - Interfacing with a Display

Summary

In this lab you will connect your PIC to an LCD and display the message "Hello World!". You will then implement a software library (API) to setup and communicate with your Serial LCD display quickly and easily. If all goes well the result will be a very similar interface to the Grove LCD available in EE1301 Group Kit.

Purpose

Serial interfaces are very common in industry today. They allow for low pin count communication between integrated circuits (ICs), like the PIC24 and an LCD. Without a serial interface, LCDs typical require 10+ pins to operate. In this lab you will develop the code to utilize the serial interface called "IIC", "I^{2}C", "I2C", or Inter-IC communication serial interface (from now on referred to as I2C).

Supplemental Resources

This lab is structured such that you have most, but not all of the information you need available in the Lab 5 background material. There are considerably more external resources and specifications to read for this lab. It is highly encouraged for you to complete the reading assignments, as this will be the last lab to provide you with exact guidance on what to read and where it is located.

Critical Resources

Introduction to I2C - External website -

http://www.embedded.com/electronics-blogs/beginner-s-corner/4023816/Introduction-to-I2C

Lab 5 - I2C LCD - Background

Required Microchip and Supplier Resources

PIC24 Family Reference Manual - Section 24. Inter-Integrated Circuit (I2C™)

PIC24FJ64GAxxx Family Datasheet - Inter-Integrated Circuit (I^{2}C™) - Section 16

AQM0802A-RN-GBW - LCD display+chip datasheet -
https://drive.google.com/file/d/0B5wD_vPJ2ttWSHRLYmdJVUZOZW8/view?usp=sharing
ST7032 - Controller chip datasheet -
https://drive.google.com/file/d/0B5wD_vPJ2ttWTUZIM18teWhjc2s/view?usp=sharing
I2C small LCD - Breakout board schematic (this is 5V variant, lab kit has 3.3V variant) -
http://doc.switch-science.com/schematic/I2C_LCD_breakout/I2C_LCD_breakout_5V.pdf

Additional Resources (informational)

Official Specification for I2C by NXP - http://www.nxp.com/documents/user_manual/UM10204.pdf
I2C Tutorial on which this lab is based - http://www.engscope.com/pic24-tutorial/10-2-i2c-basic-functions/
I2C small LCD - Manufacturer website - https://international.switch-science.com/catalog/1407/

Pre-Lab "Procedure"

As you transition from walk-through heavy labs (1,2,3) into the moderate difficulty labs (4,5) and into the final lab (6) and project, it is important to begin to understand how to dig information out of the reference documentation in order to implement a requested feature.

This *is* a two week lab, but it is required that you read the entire Pre-Lab Reading before week one. We know this is a lot of information to absorb, it is not expect that you will get all the details on the first pass. However, it will orient you to the location of various topics, so you can find them in the future. Throughout these two weeks you will refer back to the Pre-Lab Reading and the reference docs many times before all the details begin to fit together.

If after the first lab session you do not have your first demo for the teaching assistant complete, the pre-lab assignment for the second week is to complete the core functions.

Week 1 Pre-Lab Checklist
- ❏ Read "Introduction to I2C"
- ❏ Read Lab 5 Background
- ❏ New project setup with an attempt at **`lcd_cmd();`**
- ❏ Take Moodle Quiz before Lab

Week 2 Pre-Lab Checklist
- ❏ Complete the functions
  - ❏ `lcd_cmd()`
  - ❏ `lcd_init()`
  - ❏ `lcd_setCursor()`
  - ❏ `lcd_printChar()`
- ❏ Have a working single character display ready for your TA to view

Required Components

Standard Components to use and support the PIC24FJ64GA002 (pull-up, caps, etc.)
Switch Science I2C LCD Display (8x2 character)
220 Ohm Resistor

Red LED T1 3/4 Diffused

Lab Procedure

In this lab you objective is to create a fully functional library for the 3.3V Small LCD display by SwitchScience. This library should consist of an initialization function, cursor location set function, and a string writing function. Together you will use these functions to create and animate a message on the LCD display. You will need to package these up into a stand alone library and provide appropriate documentation.

Week 1 Procedure

1.) Setup a new project, boilerplate, and include your helper library (djolib.h, djolibASM.s, and djolibC.c for example).

2.) Setup the Baud Rate Generator, PIC24 I/O Registers, and wire up your project.

Wire up your LCD to VDD (3.3V), GND, SDA2, and SCL2. XRST can be left open.

It can be very helpful to have a “heart beat” LED on this project to determine if the PIC24 is hung. Setup RA0 as an output and connect it to and LED via a reasonable resistor.

Create the standard program structure of main() calling a setup() once and a loop() repeatedly.

In order to avoid signal issues and easily debug the system we will run the I2C interface at the minimum frequency in the table, 100 kHz. Our PIC24 should be running at its maximum frequency (FCY = 16 MHz). Find (in the documentation) and set the appropriate value of the I2C2BRG register.

Hints: During initialization it is good practice to disable the I2C2 peripheral before changing the Baud Rate Generator Settings, wrap your I2C2BRG update with I2CEN=0 and I2CEN=1. You will also want to clear the interrupt flag (MI2C2IF), just to be safe.

3.) Create an lcd_cmd function

```
void lcd_cmd(char command);
```

This function should take a single byte/char command and write it out the I2C bus. The complete packet should consist of a START bit, address byte with R/nW, control byte, command/data byte, and STOP bit. It is probably a good idea to use blocking code to implement this at first, but you are encouraged to use polling or interrupts in your final library.

4.) Create an lcd_init function

```
void lcd_init(void);
```

This function should implement the complete sequence of lcd_cmd() writes and delays needed to initialize the LCD.

Make sure the contrast is adjustable (by a global variable, argument, or compiler definition).

5.) Create an lcd_set_cursor(x,y) function

```
void lcd_setCursor(char x, char y)
```

This function should consist of a single lcd_cmd() statement with a masked and calculated command to set the Cursor to row y, column x.

6.) Create an lcd_printChar(myChar) function

```
void lcd_printChar(char myChar)
```

This is really only a test function to verify the LCD operation. It should consist of single I2C packet (START, Address, control, data, STOP). Unlike the lcd_cmd() function it will have different values for RS and CO bit values.

7.) Print a single character to the display. Show it to your TA.
This is step is potentially far harder than it seems. As I2C is a hardware/hardware interface, it is very difficult to debug in simulation. This step is where you must verify both the software and the hardware simultaneously.

If your heartbeat LED is not blinking (and your code is stuck on a polling wait for flag operation), it is just as likely a hardware problem as a software problem. If the PIC24 doesn't see an ACK signal from the ST7032 (because it is not connected) it will wait forever.

Common mistakes:

- SDA or SCL have poor connection
- SDA and SCL are swapped
- SDA and SCL are plugged into SDA1 and SCL1 rather than SDA2 and SCL2
- The I2C packet got interrupted mid-packet - Remove VDD from the LCD for 3 seconds, Replace VDD to the LCD. Then reset your PIC24 (by pulling the nMCLR line LOW). This is a problem common to all I2C interfaces, see the EXTEND section below for more info.
- A missing STOP bit in a function
- Incorrect calculation of Slave Address (use your micro or compiler to generate the address+R/Wn combination. Don't do it yourself, you will make mistakes!)

Week 2 Procedure

8.) Create an lcd_print(myString) function

`void lcd_printStr(const char *s)`

This function should take an arbitrary length string and send it to the LCD. This will require multiple command/data byte pairs with the RS and CO bits controlling when the packet ends. Don't forget to finish the packet with a STOP bit.

9.) Verify your printStr() function works in hardware

10.) Package up and migrate your code into a library (for example x500_LCD.h)

11.) Create an animated demo (for example, scrolling the words "Hello World!" across your display)

Post-Lab Checklist

- ❏ Complete the functions
  - ❏ `lcd_cmd()`
  - ❏ `lcd_init()`
  - ❏ `lcd_setCursor()`
  - ❏ `lcd_printChar()`
  - ❏ `lcd_printStr()`
- ❏ Demo your LCD animation to for your TA

Lab Report

Put together full documentation for the library created in this lab. The documentation should be in a commonly accessible format (gDoc, Word, HTML, PDF, etc.) It should include the following:

- Description of the library
- Download link for your library (.c, and .h files zipped)
- State the supported microprocessor and LCD
- Describe any other libraries required (dependencies)
- Document the four main functions required in this lab:
  - **`void lcd_init(void);`**
  - **`void lcd_setCursor(char x, char y)`**
  - **`void lcd_printChar(char myChar)`**
  - **`void lcd_printStr(const char *s)`**
  - List all functions, all arguments, describe what the functions do, provide an example of how they are used together

The Lab TA must see a demo of your animation. Make sure they check off your work before you leave the lab (or, setup a time outside the lab to demo if you don't have time to finish).

Going Further

- Identify and implement the software fix in Quick Lesson - I2C Bus Hang Fixes
- Create a custom 8x5 pixel character and display it on your display
- Make a function to set contrast on the fly. Clean up your library to make contrast an input to the init() function.

```
        void lcd_setContrast(char c);
        void lcd_init(char myContrast);
```

- Create the above library but utilize states with interrupts and/or non-blocking polling to control I2C communications
- Make a 1 line display with twice the height
- Can you add a cursor display (blinking)