Lab 3 - User Input and LCD



Alexis Adie and Timonty Lum

ELC 343-L2: Microcomputer Systems

Submitted:

October 11th, 2017

Lab 3 - User Input and LCD

Alexis Adie and Timonty Lum

Department of Electrical and Computer Engineering

The College of New Jersey

2000 Pennington Road, Ewing, NJ 08618, USA

(adiea1, lumt1)@tcnj.edu

1. INTRODUCTION

In this lab, students will develop a project in PSoC Creator that writes and displays messages to the Liquid Crystal Display (LCD). Activation of a pushbutton will change the current state of the assigned LED and the LCD will be updated according to the user’s pushbutton usage. The task will be achieved through wiring inputs/outputs within the design schematics as well as generation of C code.

II. METHODOLOGY

1. *Initial Code and Debugging Notes*

First, the team was tasked with creating an LCD-based project which presented information to a user in an efficient and elegant way.

Since blocks of text were given, it was decided that the code would implement a scrolling text design which would allow the user to easily read the text.

Once constructed, the group debugged the program. At first, the display was only presenting the first and last parts of the text and the delay of 1000ms for each sentence was too short. After a few attempts, the team devised a way to make the text scroll. This method was to present the text on the LCD, delay for 1000 ms, and then delete the first two letters of the previous sentence. The program then repeated until all of the text was shown to the user.

The board was programmed, but the screen once again was displaying the sentences’ beginnings and ends. Code that cleared the screen was then added after each delay. After programming the PSoC board once more, the scrolling effect was achieved.

While the first sentence simply appeared and began to scroll to the left, the team wanted the second one to scroll in as well as out. More code was added, which put the first two letters of the sentence onto the LCD and added two more letters.

For the final instruction, the group simply printed the first part of the sentence, noticing that if spaces were added after, the rest of it would print to the second line. This part of the code did not have a scrolling effect so that the user could view this instruction indefinitely.

Initially, the program waited for a button to be pushed and it would be completed. In order to meet this requirement, an input pin was added to the project and connected to one of the pushbuttons. Then, the code was modified to look for whether the pushbutton was toggled.

*B. LED Toggling*

Once the code was completed, the PSoC board was programmed to light both LED’s on startup and wait for the user to select a pushbutton. If the first one was pushed, the LCD screen would say “LED 1 Toggled” and if the second one was pushed, the display would read “LED 2 Toggled.”

The operation of the PSoC board can be seen in Video 2.1 and Video 2.2.

**Video 2.1: LCD Initial Message**

https://drive.google.com/file/d/0Bw5KhT-n\_1lScld1RzBkZGduS2s/view?usp=sharing

**Video 2.2: Pushbutton Operation**

https://drive.google.com/file/d/0Bw5KhT-n\_1lSTC1XZ3RPTWs4Mk0/view?usp=sharing

III. DISCUSSION

One aspect of the pushbutton to be considered is bouncing. A pushbutton is a switch that activates when two metal parts come together and connect. When the two metal components come together, they actually make multiple contacts. So in reality, they separate and reconnect multiple times during the process. It bounces between being in contact and not being in contact which results in the hardware thinking the switch is being pressed several times. As a result, this must be taken account in the code by using delays in reading the pushbutton. Constantly changing between being in contact and not being in contact means the switch is constantly turning on and off. Instead, we want to read the pushbutton when it’s only in one constant state. Knowing that the bouncing cannot occur for more than 100ms with this specific pushbutton, a delay of 100ms was set in the C code before reading in the state of the pushbutton.

# 

**Figure 3.1: Oscilloscope Capture of Switch Debounce**

IV. CONCLUSION

The objectives of this lab were to wire LEDs to pushbuttons, change LED states when pushbuttons were activated, and update the states of each LEDs on a Liquid Crystal Display. On the designer, the LEDs were mapped to the pushbuttons and the states of each pushbutton and LCD was dealt with in the C code. Using delays to take into account the bouncing of the switches, the C code allowed the LEDs to change states with the usage of the pushbuttons. Finally, the Liquid Crystal Displays were able to output the toggle status of each LED. The results of the code and design were as expected and the overall objective of the lab was met with success.

# V. Code

#include <project.h>

#include <stdint.h>

int main(void)

{

CyGlobalIntEnable; /\* Enable global interrupts. \*/

//Lights both LEDS

LED\_1\_Write(1);

LED\_2\_Write(1);

LCD\_Start();

LCD\_ClearDisplay();

//Initializes LCD And Begins to show the first part of the text

//delaying to allow the user to read it, clearing the display, and

//printing the same sentence with the first two letters deleted.

//This allows for a scrolling effect

LCD\_PrintString("My First Microcontroller I/O");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" First Microcontroller I/O");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("irst Microcontroller I/O Console");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("st Microcontroller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("icrocontroller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("rocontroller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("controller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ntroller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("roller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ller I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("er I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" I/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("/O Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Console Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("onsole Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("sole Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("le Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Program");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("rogram");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("gram");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("am");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Mi");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Micr");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microc");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcon");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontr");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontrol");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontrolle");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller ");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller Pr");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller Prog");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller Progra");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString(" Microcontroller Programm");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("Microcontroller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("crocontroller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ocontroller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ontroller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("troller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("oller Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ler Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("r Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("Programmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("ogrammer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("rammer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("mmer");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("er");

CyDelay(500);

LCD\_ClearDisplay();

LCD\_PrintString("Tap a switch to");

LCD\_PrintString(" turn off its LED");

for(;;)

{

// LED\_1\_Write(InputPin\_1\_Read());

// LED\_2\_Write(InputPin\_2\_Read());

if(LED\_1\_Read()==0)

{

if(InputPin\_1\_Read()==0)

{

LCD\_ClearDisplay();

LED\_1\_Write(1);

CyDelay(100);

LCD\_PrintString("LED 1 Toggled");

CyDelay(500);

}

}

if(LED\_2\_Read()==0)

{

if(InputPin\_2\_Read()==0)

{

LCD\_ClearDisplay();

LED\_2\_Write(1);

CyDelay(100);

LCD\_PrintString("LED 2 Toggled");

CyDelay(500);

}

}

while(InputPin\_1\_Read()==0)

{

LCD\_ClearDisplay();

LED\_1\_Write(0);

CyDelay(100);

LCD\_PrintString("LED 1 Toggled");

CyDelay(500);

}

while(InputPin\_2\_Read()==0 )

{

LCD\_ClearDisplay();

LED\_2\_Write(0);

CyDelay(100);

LCD\_PrintString("LED 2 Toggled");

CyDelay(500);

}

}

}