

Project 1

hello world electronic lock

Kyle Rosenthal | CPE 329-05-Gerfen | 4/17/2017

# Purpose

The project enabled a basic overview of the General Purpose I/O functionalities. This includes interfacing with an LCD screen and a keypad. The project also demonstrates core functionalities of the logical abilities of a microcontroller by keeping track of the entered keys and checking if it matches the supplied password.

## Link to Video

<https://www.youtube.com/watch?v=nd1DUSVy7XI>

# System Requirements

On startup, bad pin, and after the star is pressed, the LCD will display the message “Locked Enter Key”. While the user enters a key it will be displayed on the lower row. When the correct pin combo is entered, the message “Hello world” will be displayed on the first row. When the star key is pressed the pad will reset to the startup state.

# System Specifications

* Lock intialializes to “Locked Enter Key”
* Lock resets to “Locked Enter Key” after Star key
* Lock resets to “Locked Enter Key” after bad pin entered
* LCD displays “Hello world!” when correct pin entered
* LCD displays pin as entered on lower row

# System Architecture

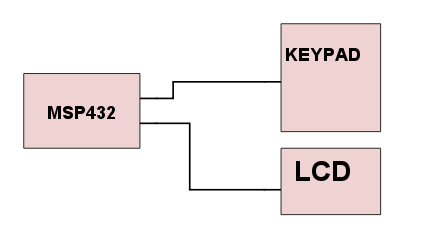


Figure 1 – System Architecture Diagram

# Component Design

All setting are default settings of MSP432P401R Launchpad running at 3 MHz.

## Schematic

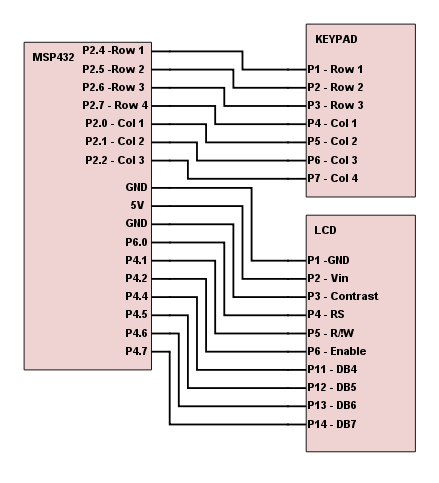


Figure 2 – Schematic Diagram

# Bill of Materials

* MSP432P401R Launchpad
* LCM-S01602DTR with KS0066U LCD Driver
* Simple 7 pin – 12 button keypad
* 17 jumper cables – male to female

# System Integration

The system was built by first learning basic operations of the GPIO pins. Then the interface with the LCD Driver was connected up and debugged. Finally the keypad was hooked up. Significant bugs with bad pins resulted in around four lost man hours of development time and in the future a tool, program, or procedure to test all pins beforehand will be necessary. After reformatting the code, it resulted in less pretty code, but was necessary to accommodate all the bad pins.

# Answers to Questions

No question for this project.

# Conclusion

This project stressed the importance of functional hardware. *It is imperative when first debugging to make sure pins are each doing what they were told to do. In addition, meaning hardware devices are very sensitive to timing. Furthermore, unlike bit on a computer, charge build up in hardware does not simply disappear and must be accounted for carefully.* The design could be improved with added functionality for the ‘\*’ and ‘#’ key, such

# Appendices

## Referances

* CPE 329 - Project 1 - Hello World v0.02 - S2017
* LCDDataSheet - Composite Brief v2 File
* MSP432 - Technical Reference Manual File
* Schematic created with: http://www.schematics.com/

## Code

### main.c

/\* www.MicroDigitalEd.com

\* p3\_3.c: Initialize and display "hello" on the LCD using 4-bit data mode.

\* Data and control pins share Port 4.

\* This program does not poll the status of the LCD.

\* It uses delay to wait out the time LCD controller is busy.

\* Timing is more relax than the HD44780 datasheet to accommodate the

\* variations among the LCD modules.

\* You may want to adjust the amount of delay for your LCD controller.

/\*

\* Project 1: main.c

\*/

**#include** "msp.h"

**#include** "proj1.h"

**static** ul curr\_freq = FREQ\_3\_MHz;

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

LCD\_init();

Key\_init();

Key key = *NONE*;

Key entered[4];

**for**(;;) {

key = *NONE*;

Clear\_LCD(); /\* clear display \*/

LCD\_string(" Locked ",8);

LCD\_string("Enter Key",9);

LCD\_command(0xC0);

delayMs(500);

**while** (key == *NONE* || key == *K\_POUND*) key = Get\_key();

**if** (key == *K\_STAR*) **continue**;

entered[0] = key;

LCD\_key(key);

delayMs(10);

**while** (key != *NONE*) key = Get\_key();

**while** (key == *NONE* || key == *K\_POUND*) key = Get\_key();

**if** (key == *K\_STAR*) **continue**;

entered[1] = key;

LCD\_key(key);

delayMs(10);

**while** (key != *NONE*) key = Get\_key();

**while** (key == *NONE* || key == *K\_POUND*) key = Get\_key();

**if** (key == *K\_STAR*) **continue**;

entered[2] = key;

LCD\_key(key);

delayMs(10);

**while** (key != *NONE*) key = Get\_key();

**while** (key == *NONE* || key == *K\_POUND*) key = Get\_key();

**if** (key == *K\_STAR*) **continue**;

entered[3] = key;

LCD\_key(key);

delayMs(10);

**while** (key != *NONE*) key = Get\_key();

Home\_LCD();

**if** (entered[0] == password[0] && entered[1] == password[1] &&

entered[2] == password[2] && entered[3] == password[3]) {

LCD\_string("Hello World! ", 16);

delayMs(1000);

} **else** {

LCD\_string("Bad password :( ", 16);

delayMs(1000);

}

}

}

**void** **LCD\_string**(**char** \* str, **int** len)

{

**int** i;

**for** (i = 0; i < len; i++)

{

LCD\_data(str[i]);

}

}

**void** **LCD\_key**(Key retMe) {

**switch** (retMe) {

**case** *K1*:

LCD\_data('1');

**break**;

**case** *K2*:

LCD\_data('2');

**break**;

**case** *K3*:

LCD\_data('3');

**break**;

**case** *K4*:

LCD\_data('4');

**break**;

**case** *K5*:

LCD\_data('5');

**break**;

**case** *K6*:

LCD\_data('6');

**break**;

**case** *K7*:

LCD\_data('7');

**break**;

**case** *K8*:

LCD\_data('8');

**break**;

**case** *K9*:

LCD\_data('9');

**break**;

**case** *K\_STAR*:

LCD\_data('\*');

**break**;

**case** *K0*:

LCD\_data('0');

**break**;

**case** *K\_POUND*:

LCD\_data('#');

**break**;

}

}

**delay**(){}

Key **Get\_key**(**void**) {

Key retMe = *NONE*;

P2->OUT = 0xF7;

//row 1

P2->DIR = 0x00;

P2->DIR |= 0xF0;

P2->OUT &= ~0x10;

delay();

**unsigned** **char** in = P2->IN & 0x07;

P2->OUT |=0x10;

**if** (in == 0x6)

retMe = *K3*;

**if** (in == 0x5)

retMe = *K2*;

**if** (in == 0x3)

retMe = *K1*;

**if** (retMe != *NONE*) **return** retMe;

// row 2

//P2->DIR = 0x00;

//P2->DIR |= 0x20;

P2->OUT &= ~0x20;

delay();

in = P2->IN & 0x07;

P2->OUT |=0x20;

**if** (in == 0x6)

retMe = *K6*;

**if** (in == 0x5)

retMe = *K5*;

**if** (in == 0x3)

retMe = *K4*;

//row 3

P1->DIR = 0x00;

P1->DIR |= 0x40;

P1->OUT &= ~0x40;

delay();

in = P2->IN & 0x07;

P1->OUT |=0x40;

**if** (in == 0x6)

retMe = *K9*;

**if** (in == 0x5)

retMe = *K8*;

**if** (in == 0x3)

retMe = *K7*;

//row 4

P1->DIR = 0x00;

P1->DIR |= 0x80;

P1->OUT &= ~0x80;

delay();

in = P2->IN & 0x07;

P1->OUT |=0x80;

**if** (in == 0x6)

retMe = *K\_POUND*;

**if** (in == 0x5)

retMe = *K0*;

**if** (in == 0x3)

retMe = *K\_STAR*;

P2->OUT = 0xF7;

P2->DIR = 0x00;

delayMs(1);

**return** retMe;

}

/\* P2(0:2) is cols / input; P2(4:7) is rows/output \*/

**void** **Key\_init**(**void**) {

P2->DIR = 0x00;

P2->REN = 0x07;

P2->OUT = 0x07;

}

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

P4->DIR = 0xFF; /\* make P4 pins output for data and controls \*/

P6->DIR = 0x01;

delayMs(30); /\* initialization sequence \*/

LCD\_nibble\_write(0x30, 0);

delayMs(10);

LCD\_nibble\_write(0x30, 0);

delayMs(1);

LCD\_nibble\_write(0x30, 0);

delayMs(1);

LCD\_nibble\_write(0x20, 0); /\* use 4-bit data mode \*/

delayMs(1);

LCD\_command(0x28); /\* set 4-bit data, 2-line, 5x7 font \*/

LCD\_command(0x06); /\* move cursor right after each char \*/

LCD\_command(0x01); /\* clear screen, move cursor to home \*/

LCD\_command(0x0F); /\* turn on display, cursor blinking \*/

}

/\* With 4-bit mode, each command or data is sent twice with upper

\* nibble first then lower nibble.

\*/

**void** **LCD\_nibble\_write**(**unsigned** **char** data, **unsigned** **char** control) {

data &= 0xF0; /\* clear lower nibble for control \*/

control &= 0x0F; /\* clear upper nibble for data \*/

P4->OUT = data | control; /\* RS = 0, R/W = 0 \*/

P6->OUT = control;

P4->OUT = data | control | EN; /\* pulse E \*/

delayMs(0);

P4->OUT = data; /\* clear E \*/

P4->OUT = 0;

P6->OUT = 0;

}

**void** **LCD\_command**(**unsigned** **char** command) {

LCD\_nibble\_write(command & 0xF0, 0); /\* upper nibble first \*/

LCD\_nibble\_write(command << 4, 0); /\* then lower nibble \*/

**if** (command < 4)

delayMs(4); /\* commands 1 and 2 need up to 1.64ms \*/

**else**

delayMs(1); /\* all others 40 us \*/

}

**void** **LCD\_data**(**unsigned** **char** data) {

LCD\_nibble\_write(data & 0xF0, RS); /\* upper nibble first \*/

LCD\_nibble\_write(data << 4, RS); /\* then lower nibble \*/

delayMs(1);

}

**void** **Clear\_LCD**(**void**) {

// send 0x01 in two nibbles

// Write upper

P6->OUT &= ~BIT0;

P4->OUT = 0;

P4->OUT |= EN;

delayMs(0);

P4->OUT = 0;

//write lower

P4->OUT = 0x10;

P4->OUT |= EN;

delayMs(0);

P4->OUT &= ~EN;

P4->OUT = 0;

}

**void** **Home\_LCD**(**void**) {

// send 0x02 in two nibbles

// Write upper

P6->OUT &= ~BIT0;

P4->OUT = 0;

P4->OUT |= EN;

delayMs(0);

P4->OUT = 0;

//write lower

P4->OUT = 0x20;

P4->OUT |= EN;

delayMs(0);

P4->OUT &= ~EN;

P4->OUT = 0;

}

/\* delay milliseconds when system clock is at 3 MHz \*/

**void** **delayMs**(**int** n) {

**int** i, j;

**for** (j = 0; j < n; j++)

**for** (i = 750; i > 0; i--); /\* Delay \*/

}

/\* delay milliseconds \*/

**void** **delay\_ms**(ul ms) {

ul i;

ul num = ms \* (curr\_freq / 1000);

ul itr = num / CYCLES\_PER\_LOOP;

itr -= SETUP\_CYCLES;

**for** (i = 0; i < itr; i++); /\* Delay 1 loop\*/

}

**void** **delay\_ns**(ul ns) {

ul i;

//ul num = ns \* (freq / 100000) / 10000;

ul itr = ns \* (curr\_freq / 100000) / 10000 / CYCLES\_PER\_LOOP;

itr -= SETUP\_CYCLES;

**for** (i = 0; i < itr; i++); /\* Delay 1 loop\*/

}

### proj1.h

/\*

\* proj1.h

\*

\* Created on: Apr 17, 2017

\* Author: kmrosent

\*/

**#ifndef** PROJ1\_H\_

**#define** PROJ1\_H\_

**#define** RS 1 /\* P4.0 mask \*/

**#define** RW 2 /\* P4.1 mask \*/

**#define** EN 4 /\* P4.2 mask \*/

**#define** CYCLES\_PER\_LOOP 11

**#define** SETUP\_CYCLES 5

**#define** FREQ 32\_KHz 32768

**#define** FREQ\_1\_5\_MHz 1500000

**#define** FREQ\_3\_MHz 3000000

**#define** FREQ\_6\_MHz 6000000

**#define** FREQ\_12\_MHz 12000000

**#define** FREQ\_24\_MHz 24000000

**#define** FREQ\_48\_MHz 48000000

**typedef** **unsigned** **long** ul;

**void** **delayMs**(**int** n);

**void** **delay\_ms**(ul ms);

**void** **delay\_ns**(ul ns);

**void** **LCD\_nibble\_write**(**unsigned** **char** data, **unsigned** **char** control);

**void** **LCD\_command**(**unsigned** **char** command);

**void** **LCD\_data**(**unsigned** **char** data);

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

**void** **Key\_init**(**void**);

**void** **Clear\_LCD**(**void**);

**void** **Home\_LCD**(**void**);

**void** **LCD\_string**(**char** \* str, **int** len);

**typedef** **enum** { *K1* = 0, *K2*, *K3*, *K4*, *K5*, *K6*, *K7*, *K8*, *K9*, *K\_STAR*, *K0*, *K\_POUND*, *NONE* } Key;

Key **Get\_key**(**void**);

**void** **LCD\_key**(Key retMe);

Key password[4] = {*K4*, *K5*, *K4*, *K1*};

**#endif** /\* PROJ1\_H\_ \*/