## ECEN260 – Final Project

## Score Counter

Dale Andreoli

Gonzalo Martinez

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Program Specifications**

This project shows an understanding of the following concepts:

ADC (potentiometer)

I/O (switches)

Display (GLCD)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Program Description**

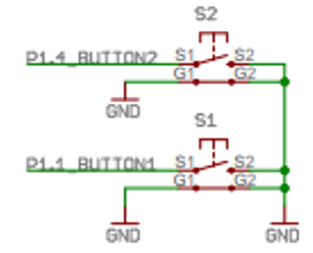
This project is designed to count in various increments. It can be used to keep score in games or any application where a count is needed. A potentiometer is used to select the amount by which to increment to total. Switches are used to increment or decrement the total by the selected amount. An LCG screen displays both the total and the currently selected increment value.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

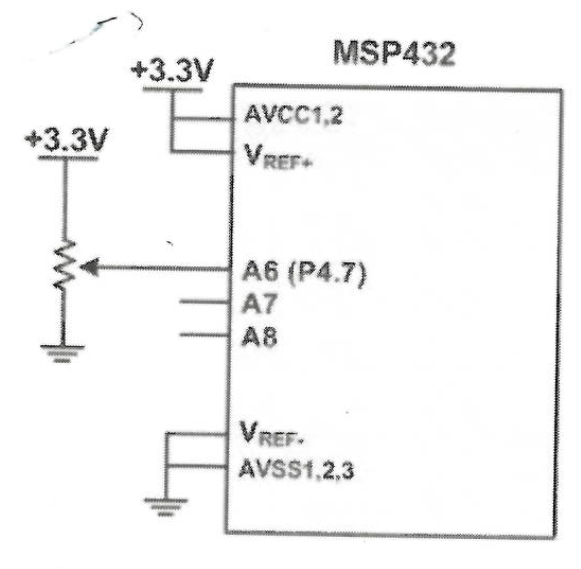
**Connections (Wiring Schematic)**

SW1 is connected to P1.1

SW2 is connected to P1.4

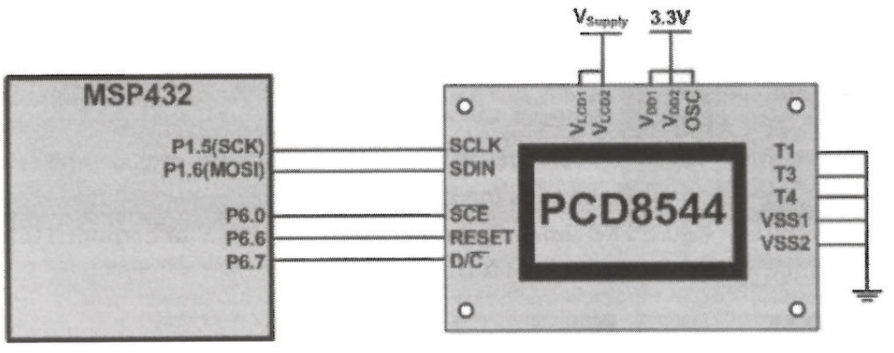


Potentiometer connected to P4.7



The PCD8544 GLCD is connected as shown below.

NOTE: There is only one VCC and only one GND connection on the display. All of the shown voltages and grounds are connected internally on the board.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Plan and Results**

To test our program, we will make sure our potentiometer is properly converted from an analog value to one of our three increment options. We will test our switches to make sure the Switch 1 increment the total by the selected value while Switch 2 decrements by the selected value. We will test our display by making sure all elements are properly displayed.

Our program successfully converts the value of our potentiometer into the appropriate increment value. Our switches properly increment and decrement the total with accurate debouncing and input delays. Switch 1 increments the total while Switch 2 decrements the total. Our display neatly outputs the current total and the currently selected increment value with aesthetic white space to make it more readable

The biggest struggle for this project was properly converting our potentiometer position into the proper increment value.

Attached is a video demonstrating all the features of our score counter accompanied by an oral explanation of the project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code**

**#include** "msp.h"

**#include** "stdbool.h"

**#define** CE 0x01 /\* P6.0 chip select \*/

**#define** RESET 0x40 /\* P6.6 reset \*/

**#define** DC 0x80 /\* P6.7 register select \*/

/\* define the pixel size of display \*/

**#define** GLCD\_WIDTH 84

**#define** GLCD\_HEIGHT 48

**#define** S1 BIT1

**#define** S2 BIT4

**#define** DEBOUNCE 300

**#define** DELAY 150000 // used for SW switch debouncer using 300 clock cycles

**void** **GLCD\_setCursor**(**unsigned** **char** x, **unsigned** **char** y);

**void** **GLCD\_clear**(**void**);

**void** **GLCD\_init**(**void**);

**void** **GLCD\_data\_write**(**unsigned** **char** data);

**void** **GLCD\_command\_write**(**unsigned** **char** data);

**void** **GLCD\_putchar**(**int** c);

**void** **SPI\_init**(**void**);

**void** **SPI\_write**(**unsigned** **char** data);

/\*font table \*/

**const** **char** font\_table[][6] = {

{0x3e, 0x41, 0x41, 0x41, 0x3e, 0x00}, /\* 0 \*/

{0x04, 0x02, 0x7f, 0x00, 0x00, 0x00}, /\* 1 \*/

{0x42, 0x61, 0x51, 0x49, 0x46, 0x00}, /\* 2 \*/

{0x22, 0x41, 0x49, 0x49, 0x36, 0x00}, /\* 3 \*/

{0x10, 0x18, 0x14, 0x12, 0x7f, 0x00}, /\* 4 \*/

{0x4f, 0x49, 0x49, 0x49, 0x31, 0x00}, /\* 5 \*/

{0x3e, 0x49, 0x49, 0x49, 0x31, 0x00}, /\* 6 \*/

{0x41, 0x21, 0x11, 0x09, 0x07, 0x00}, /\* 7 \*/

{0x36, 0x49, 0x49, 0x49, 0x36, 0x00}, /\* 8 \*/

{0x46, 0x49, 0x49, 0x49, 0x3e, 0x00}, /\* 9 \*/

{0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, // \_ 10

{0x7e, 0x11, 0x11, 0x11, 0x7e, 0x00}, //A 11

{0x7f, 0x49, 0x49, 0x49, 0x36, 0x00}, //B 12

{0x3e, 0x41, 0x41, 0x41, 0x22, 0x00}, //C 13

{0x7f, 0x41, 0x41, 0x41, 0x3e, 0x00}, //D 14

{0x7f, 0x49, 0x49, 0x49, 0x41, 0x00}, //E 15

{0x7f, 0x09, 0x09, 0x09, 0x01, 0x00}, //F 16

{0x3e, 0x41, 0x49, 0x49, 0x7a, 0x00}, //G 17

{0x7f, 0x08, 0x08, 0x08, 0x7f, 0x00}, //H 18

{0x41, 0x41, 0x7f, 0x41, 0x41, 0x00}, //I 19

{0x20, 0x40, 0x40, 0x40, 0x3f, 0x00}, //J 20

{0x7f, 0x08, 0x14, 0x22, 0x41, 0x00}, //K 21

{0x7f, 0x40, 0x40, 0x40, 0x40, 0x00}, //L 22

{0x7f, 0x02, 0x0c, 0x02, 0x7f, 0x00}, //M 23

{0x7f, 0x04, 0x08, 0x10, 0x7f, 0x00}, //N 24

{0x3e, 0x41, 0x41, 0x41, 0x3e, 0x00}, //O 25

{0x7f, 0x09, 0x09, 0x09, 0x06, 0x00}, //P 26

{0x3e, 0x41, 0x51, 0x60, 0x7e, 0x00}, //Q 27

{0x7f, 0x09, 0x19, 0x29, 0x46, 0x00}, //R 28

{0x26, 0x49, 0x49, 0x49, 0x32, 0x00}, //S 29

{0x01, 0x01, 0x7f, 0x01, 0x01, 0x00}, //T 30

{0x3f, 0x40, 0x40, 0x40, 0x3f, 0x00}, //U 31

{0x1f, 0x20, 0x40, 0x20, 0x1f, 0x00}, //V 32

{0x3f, 0x40, 0x38, 0x40, 0x3f, 0x00}, //W 33

{0x63, 0x14, 0x08, 0x14, 0x63, 0x00}, //X 34

{0x03, 0x04, 0x78, 0x04, 0x03, 0x00}, //Y 35

{0x61, 0x51, 0x49, 0x45, 0x43, 0x00}, //Z 36

{0x00, 0x00, 0x5f, 0x00, 0x00, 0x00}, //! 37

{0x00, 0x14, 0x20, 0x20, 0x14, 0x00}, // :) 38

{0x00, 0x00, 0x7e, 0x81, 0xb5, 0xa1}, // lefftSmile 39

{0xa1, 0xb5, 0x81, 0x7e, 0x00, 0x00}, // right Smile 40

{0x00, 0x24, 0x00, 0x00, 0x00, 0x00} }; // : 41

/\*\*

\* main.c

\*/

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

{

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

P1->DIR &= ~BIT1; // set P1.1 as input

P1->REN |= BIT1; // turn on P1.1 pull resistor

P1->OUT |= BIT1; // configure P1.1 resistor as pull-up

P1->DIR &= ~BIT4; // set P1.4 as input

P1->REN |= BIT4; // turn on P1.4 pull resistor

P1->OUT |= BIT4; // configure P1.4 resistor as pull-up

// configure ADC14

ADC14->CTL0 = 0x00000010;

ADC14->CTL0 |= 0x04080300; // configure

ADC14->CTL1 = 0x00000020;

ADC14->MCTL[5] = 0x06;

P4->SEL1 |= 0x80;

P4->SEL0 |= 0x80;

ADC14->CTL1 |= 0x00050000;

ADC14->CTL0 |= 0x02;

GLCD\_init(); /\* initialize the GLCD controller \*/

GLCD\_clear(); /\* clear display and home the cursor \*/

**int** count = 0; // total

**int** i = 0;

**int** result;

**int** increment; // increment

**int** digiResult;

bool pressed = false;

**while**(1)

{

// delay for switch debouncing

**for** (i = 0; i < DEBOUNCE; i++){}

//ADC configuration

ADC14->CTL0 |=1;

**while** (!ADC14->IFGR0);

result = ADC14->MEM[5];

digiResult = (result >> 8) % 30;

//increment selector

**if** ( digiResult >= 12)

increment = 1;

**else** **if** (digiResult <= 2)

increment = 10;

**else**

increment = 5;

// check S1

**if** ((P1->IN & S1) == 0x00)

{

count += increment; // increment total

pressed = true;

}

// check S2

**if** ((P1->IN & S2) == 0x00)

{

count -= increment; //decrement total

pressed = true;

}

// display

GLCD\_clear(); /\* clear display and home the cursor \*/

//display spaces

**for** ( i = 0; i < 4; i++)

GLCD\_putchar(10);

GLCD\_putchar(30); //T

GLCD\_putchar(25); //o

GLCD\_putchar(30); //t

GLCD\_putchar(11); //a

GLCD\_putchar(22); //l

GLCD\_putchar(41); //:

GLCD\_putchar((count / 100)); //digit 1

GLCD\_putchar((count % 100) / 10); //digit 2

GLCD\_putchar((count % 10)); //digit 3

//display spaces

**for** ( i = 0; i <= 14; i++)

GLCD\_putchar(10);

GLCD\_putchar(19); // display INCREMENT:

GLCD\_putchar(24);

GLCD\_putchar(13);

GLCD\_putchar(28);

GLCD\_putchar(15);

GLCD\_putchar(23);

GLCD\_putchar(15);

GLCD\_putchar(24);

GLCD\_putchar(30);

GLCD\_putchar(41);

GLCD\_putchar(10); //space

GLCD\_putchar(increment / 10); //digit 1

GLCD\_putchar(increment % 10); //digit 2

//input delay this loop cycle if a switch was pressed

**if** (pressed == true)

{

**for** (i = 0; i < DELAY; i++){}

pressed = false;

}

}

}

**void** **GLCD\_putchar**(**int** c)

{

**int** i;

**for**(i = 0; i < 6; i++)

GLCD\_data\_write(font\_table[c][i]);

}

**void** **GLCD\_setCursor**(**unsigned** **char** x, **unsigned** **char** y)

{

GLCD\_command\_write(0x80 | x); /\* column \*/

GLCD\_command\_write(0x40 | y); /\* bank (8 rows per bank) \*/

}

/\* clears the GLCD by writing zeros to the entire screen \*/

**void** **GLCD\_clear**(**void**)

{

int32\_t index;

**for**(index = 0; index < (GLCD\_WIDTH \* GLCD\_HEIGHT / 8); index++)

GLCD\_data\_write(0x00);

GLCD\_setCursor(0, 0); /\* return to the home position \*/

}

/\* send the initialization commands to PCD8544 GLCD controller \*/

**void** **GLCD\_init**(**void**)

{

SPI\_init();

/\* hardware reset of GLCD controller \*/

P6->OUT |= RESET; /\* deasssert reset \*/

GLCD\_command\_write(0x21); /\* set extended command mode \*/

GLCD\_command\_write(0xB8); /\* set LCD Vop for contrast \*/

GLCD\_command\_write(0x04); /\* set temp coefficient \*/

GLCD\_command\_write(0x14); /\* set LCD bias mode 1:48 \*/

GLCD\_command\_write(0x20); /\* set normal command mode \*/

GLCD\_command\_write(0x0C); /\* set display normal mode \*/

}

/\* write to GLCD controller data register \*/

**void** **GLCD\_data\_write**(**unsigned** **char** data)

{

P6->OUT |= DC; /\* select data register \*/

SPI\_write(data); /\* send data via SPI \*/

}

/\* write to GLCD controller command register \*/

**void** **GLCD\_command\_write**(**unsigned** **char** data)

{

P6->OUT &= ~DC; /\* select command register \*/

SPI\_write(data); /\* send data via SPI \*/

}

**void** **SPI\_init**(**void**)

{

EUSCI\_B0->CTLW0 = 0x0001; /\* put UCB0 in reset mode \*/

EUSCI\_B0->CTLW0 = 0x69C1; /\* PH=0, PL=1, MSB first, Master, SPI, SMCLK \*/

EUSCI\_B0->BRW = 3; /\* 3 MHz / 3 = 1MHz \*/

EUSCI\_B0->CTLW0 &= ~0x001; /\* enable UCB0 after config \*/

P1->SEL0 |= 0x60; /\* P1.5, P1.6 for UCB0 \*/

P1->SEL1 &= ~0x60;

P6->DIR |= (CE | RESET | DC); /\* P6.7, P6.6, P6.0 set as output \*/

P6->OUT |= CE; /\* CE idle high \*/

P6->OUT &= ~RESET; /\* assert reset \*/

}

**void** **SPI\_write**(**unsigned** **char** data)

{

P6->OUT &= ~CE; /\* assert /CE \*/

EUSCI\_B0->TXBUF = data; /\* write data \*/

**while**(EUSCI\_B0->STATW & 0x01);/\* wait for transmit done \*/

P6->OUT |= CE; /\* deassert /CE \*/

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_