Xuecong Fan A01972388
Robert McKay A02080393
November 8, 2017

Lab 5

Introduction:

For this lab mainly we are used breadboard and LCD/touch screen for our lab equipment. The purposed of this lab is to show three button on the LCD/touch screen and use the button to control different color of LED light.

Lab Detail:

First in this lab we interfaced with an LCD. We wrote C functions to send "CMD" and "DAT" to GPIO to code the button we want. For button, we need three different button to show on the touch screen. For each button we used square shape to show them, they are color are red,green and yellow. Then we are interfaced the touchscreen to calculate and found out the "x" and "y" coordinate on the screen. Because the touch screen we are used is resistive touchscreen therefore we can find "X" and "Y", from Figure 1 and Figure 2 we can see how they are work on touch screen.

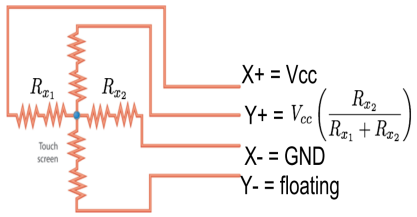

Figure 1:"X"

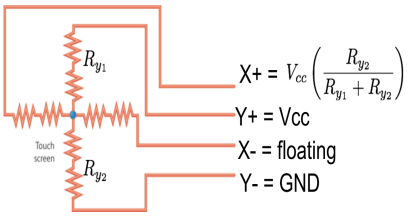

Figure 2:"Y"

After we found "X" and "Y" coordinate, now we could specifically used them to found our buttons range position on the touchscreen. Next we used the button "X" and "Y" coordinate we found before to modify each button range we would touch. Then selected the range we want to changed color when we touched the button, the reason to do it is to show touch is worked on the touchscreen. At last, we used red, green and yellow buttons on the touch screen to control red, green and yellow LED light.

Software Design:

```
#include "lcd.h"
#include "lcd_configuration_test.h" //You can remove this line after you've re

unsigned int* ssi0 = (unsigned int*) 0x40008000;
unsigned char* systemClock = (unsigned char*) 0x400FE000;
unsigned int* portA = (unsigned int*) 0x40004000;
unsigned int* portB = (unsigned int*) 0x40005000;
unsigned int* portE = (unsigned int*) 0x40024000;
unsigned int* corePeripheral = (unsigned int*) 0xE000E000;
unsigned int TxFEmask = 0;
unsigned int RxFNmask = 0;
unsigned int finalx;
unsigned int finaly;
unsigned int redcount = 0;
unsigned int greencount = 0;
unsigned int yellowcount = 0;
unsigned int i = 0;

void redswitch(void) {
    redcount++;
    if (redcount % 2 == 1) {
        write_cmd(0x2A);
        write_dat2(87);
        write_dat2(153);
        write_cmd(0x2B);
        write_dat2(225);
        write_dat2(291);
        write_cmd(0x2C);
        for(i = 0; i < 66*66; i++) {
            write_dat2(black);
        }
        portE[0x3FC/4] &= 0x6;
    }
    else {
        write_cmd(0x2A);
        write_dat2(84);
        write_dat2(156);
```

Figure 3: code-1

```
            write_cmd(0x2B);
            write_dat2(222);
            write_dat2(294);
            write_cmd(0x2C);
            for(i = 0; i < 72*72; i++) {
                write_dat2(red);
            }
            portE[0x3FC/4] |= 0x8;
        }
    }
void greenswitch(void) { ... }
void yellowswitch(void) { ... }
int getx(void) {
    unsigned int x;
    portB[0x3FC/4] = 0;
    while ((ssi0[0xC/4] & 0x4) == 0x4) {
        x = ssi0[0x8/4];
    }
    ssi0[0x8/4] = 0xD0;
    ssi0[0x8/4] = 0x0;
    ssi0[0x8/4] = 0x0;
    while ((ssi0[0xC/4] & 0x4) == 0x0) {
        ;
    }
    x = ssi0[0x8/4];
    while ((ssi0[0xC/4] & 0x4) == 0x0) {
        ;
    }
    x = (ssi0[0x8/4] << 5);
    while ((ssi0[0xC/4] & 0x4) == 0x0) {
        ;
    }
    x |= (ssi0[0x8/4] >> 3);
    portB[0x3FC/4] = 1;
    return x;
}
```

Figure 4: code-2

```
int gety(void){ ... }
void GPIOA_Handler(void) {
    portA[0x41C/4] = 0x80; // acknowledge interrupt
    i = 0;
    while (portA[0x3FC/4] >> 7 == 0) {
        i++;
        finalx += getx();
        finaly += gety();
//      finalx = getx();
//      finaly = gety();
    }
    finalx = finalx / i;
    finaly = finaly / i;
    if (finalx >= 0x580 && finalx <= 0x980 && finaly >= 0xA40 && finaly <= 0xD20) {
        redswitch();
    }
    else if (finalx >= 0x5A0 && finaly >= 0x611 && finalx <= 0x9A0 && finaly <= 0x940) {
        greenswitch();
    }
    else if (finalx >= 0x540 && finaly >= 0x240 && finalx <= 0x980 && finaly <= 0x540) {
        yellowswitch();
    }
}

int main(void)
{
    systemClock[0x608] = 0x13; // Enable GPIOA, E, and B in clock
    systemClock[0x61C] = 0x1; // Enable ssiModuleBaseAddress in clock
    portA[0x400/4] = 0x6C; // 0110 1100, input for PA4 (ssiModuleBaseAddressRx) and PA7, output for PA5 (ssiModuleBaseAddressTx), PA6, and PA2
    portB[0x400/4] = 0x1; //PB0 is the chip select for the touchscreen, needs to be output
    portE[0x400/4] = 0xE; //PE1-PE3 output
    portA[0x420/4] = 0x34; // 0011 0100, enable alternate function ssiModuleBaseAddress on PA2-PA5
    portB[0x420/4] = 0x0; //disable alternate functions on port B
    portE[0x420/4] = 0x0;
    portA[0x51C/4] |= 0xFC; // 1111 1100, Digital enable for all ssiModuleBaseAddress and PA6 (D/CX); note: PA2 (ssiModuleBaseAddressClk) and PA3 (ssiModuleBaseAddressFss)
    portB[0x51C/4] = 0x1; //Digital enable for PB0
    portE[0x514/4] = 0xE; //pull down resistors for PE1-PE3
```

Figure 5: code-3

```
portE[0x51C/4] = 0xE; //Digital enable for PE1-PE3
//portA[0x52C/4] = 0x220200; // Control register used in conjunction with AFSEL
ssi0[0x4/4] = 0x0; //SSICR1 set to master configuration
ssi0[0x10/4] = 0x8; //Set ssiModuleBaseAddress bit transfer to 16M/2 = 8MHz
ssi0[0xFC8/4] = 0x0; //Configure SSICC to system clock
ssi0[0x0/4] = 0xC7; //Configure SSICR0. This enables SPH and SPO, and also sets 8 bit data transfer and other options low.
ssi0[0x4/4] |= 0x2; //Enable ssiModuleBaseAddress
portB[0x3FC/4] = 1;
portA[0x3FC/4] |= 0x8;
lcd_init();
write_cmd(0x2A); //columns
write_dat2(0x0);
write_dat2(240);
write_cmd(0x2B); //pages
write_dat2(0x0);
write_dat2(320);
write_cmd(0x2C);
for(i = 0; i < 320*240; i++) {
    write_dat2(black);
}
write_cmd(0x2A);
write_dat2(84);
write_dat2(156);
write_cmd(0x2B);
write_dat2(26);
write_dat2(98);
write_cmd(0x2C);
for(i = 0; i < 72*72; i++) {
    write_dat2(yellow);
}
write_cmd(0x2A);
write_dat2(84);
write_dat2(156);
write_cmd(0x2B);
write_dat2(124);
write_dat2(196);
write_cmd(0x2C);
```

Figure 6: code-4

```
for(i = 0; i < 72*72; i++) {
    write_dat2(green);
}
write_cmd(0x2A);
write_dat2(84);
write_dat2(156);

write_cmd(0x2B);
write_dat2(222);
write_dat2(294);
write_cmd(0x2C);
for(i = 0; i < 72*72; i++) {
    write_dat2(red);
}
portE[0x3FC/4] = 0xE;
corePeripheral[0x100/4] = 1; //enable interrupt for GPIOA
portA[0x404/4] = 0; //edge sensitive
portA[0x408/4] = 0; //no double edge
portA[0x40C/4] = 0; //falling edge triggered
portA[0x410/4] = 0x80; //Enables interrupt on PA7, for the touchscreen

//After you have made sure that your ssi configuration and your gpio configuration are correct,
// you can delete the call to test_configuration and write your own code to draw the buttons.
// It is recommended that you write your LCD screen interfacing code in lcd.c and provide
// corresponding function declarations in lcd.h. That will allow you to copy your lcd.c and lcd.h
// files into any other project that requires the LCD screen.
while(1);
}
```

Figure 7: code-5

Data transmit:
In this lab we would has data transmit. Figure 8 shows that it transmitting data between breadboard and LCD/touch screen.

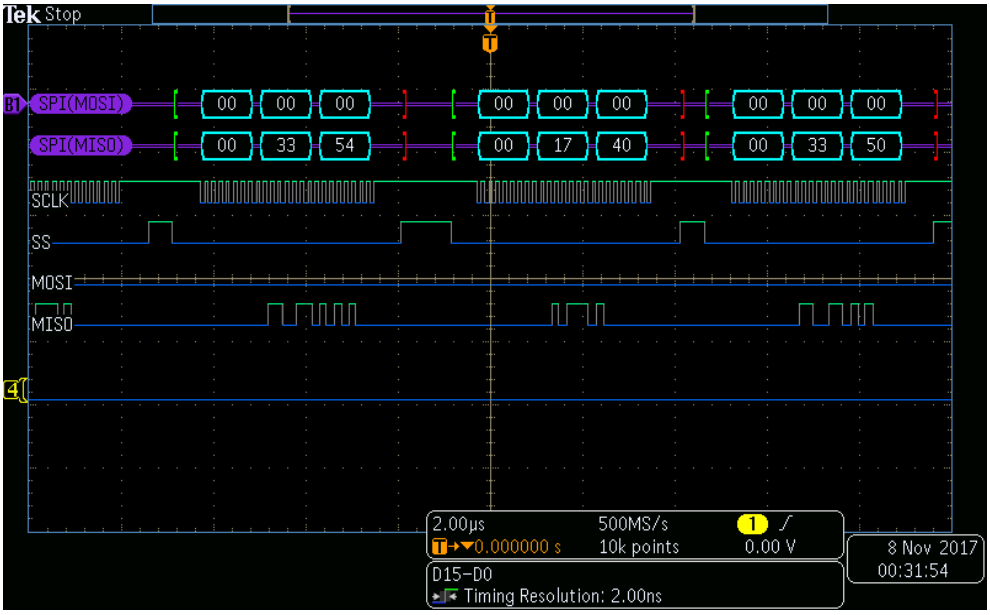

Figure 8: signal shot

Issue:
In this lab the biggest issue is put wrong pins in the LCD/touch screen, because if you put wrong pins in the LCD/touch screen it would burn it. Other than that the hardest part

for code is to calculate the coordinate on the touch screen, this part is the longest time we spend in this lab.

Conclusions:
From this lab we studied how to set up touch screen and how to show LCD on it. We also learned how to show different color and size, how to find coordinate on the touch screen and make touch button on the touch screen work.