1. Introduction

This document describes the design for an LCD touch screen prototype. An LCD touch screen consists of a grid of pixels and a grid of resistors for emitting a colored light and tracking touch position respectively. Practical application of this technology is frequently utilized across the entire demographic spectrum of consumers, businesses, schools, and government institutions. Most often an LCD touch screen is utilized in mobile phones and tablets.

2. Overview

This document discusses, in general, the electrical and software design for the LCD touch screen prototype. It includes the requirements, design details and commentary. A hardware diagram, hardware key, and code debugging screenshots are included to reinforce the results and conclusions of the lab. The complete code is located in the Appendix.

3. Requirements

The following are the given requirements for the hardware key logger prototype:

1. Three buttons must be drawn on the LCD. One red, one green, and one yellow. The buttons will be blank colored outlines initially (like the red and yellow button in figure 1).
2. When a button is touched the button will be filled (like the green button in figure 1). When that button is touched again the button should be return to a blank outline.
3. The red, green, and yellow LEDs should be wired to three available ports for output. The LEDs must be lit up with the corresponding LCD button. If the LCD button is solid, the LED should be on and vice versa.
4. The screen refresh rate must be acceptably fast. What is the maximum clock rate that the ILI9341 controller can handle? PLL is one method to speed up the clock rate, which will result in faster refresh rates.
5. The program only needs to be able to capture and display lowercase alphanumeric characters and space.

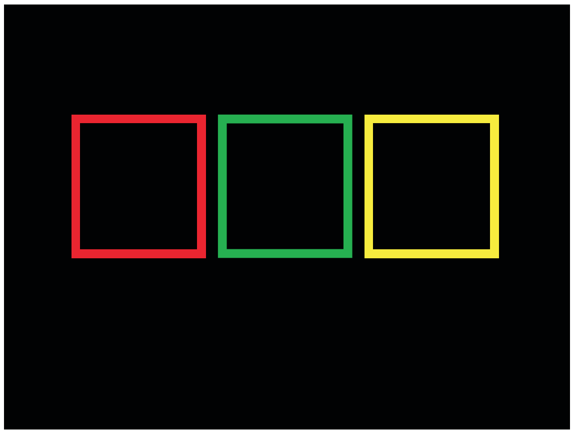
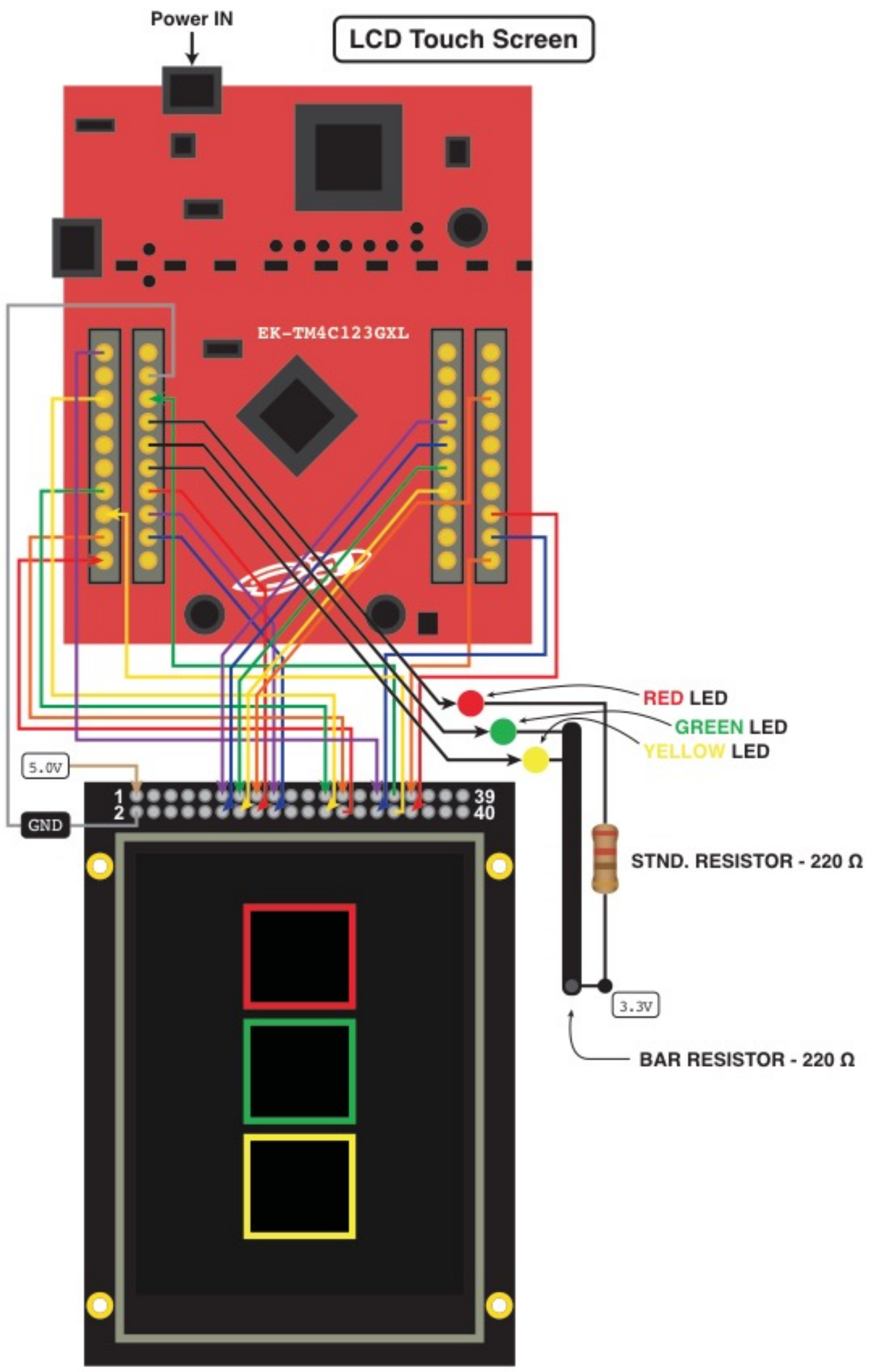


Figure 1: LCD Button Example

4. Design Details

4.1 Hardware Diagram



4.2 Hardware Key

LCD Pin No.	uController Pin	Symbol	Description
1	GND	VSS	Ground
2	5V	VDD	Power Supply
11-14	PC4-PC7	DB8-DB11	16-Bit Parallel Bi-Directional Bus
15-18	PE0-PE3	DB12-DB15	16-Bit Parallel Bi-Directional Bus
23	PB4	LCD_/CS	LCD Chip Select Input Pin (“Low” enable)
24	PB1	D/C(SCL)	Command = 0, Data = 1
25	PA6	/WR	Write Signal
26	PA7	/RD	Read Signal
29	+3.3V	BL	Backlight Control Input (“High” enable)
30	PB6	TP_/CS	Touch Screen Chip Select Input Pin (“Low” enable)
31	PD0	/TP PEN	Touch Screen Pen Interrupt (“Low” enable)
32	PA5	SDO	Serial Data Output
33	PA2	SCL	Serial Clock Signal Input
34	PA4	SDI	Serial Data Input

5. Commentary

5.1 Testing

Behavior of the code has been exhaustively observed via the debugger mode of Keil uVision. The LCD screen draws the colored squares exactly according to the lab requirements. Touching only within or on the borders of the colored boxes appropriately removes the inner black square on top of the colored square touched. Only upon release of the finger from the touch screen are changes activated. However, the touch screen interrupt does suddenly and expectedly fail. It is proposed that the sudden failure is due to overheating.

5.2 Conclusions

Lab 5 demonstrates the function of interrupts on the Tiva C microcontroller and the SSI module such that the microcontroller is able to interface with the LCD touch screen to capture the coordinates of the user’s touch. The interrupts were configured dependent upon touch of the LCD touch screen.

6. Appendix

6.1 Program Code

```
// This function draws three empty boxes. The user touches the box
// to fill the box and turn on a corresponding external light.

#include "LCD.H"
unsigned char *PA = (unsigned char *) 0x40004000;
unsigned char *PB = (unsigned char *) 0x40005000;
unsigned char *PC = (unsigned char *) 0x40006000;
unsigned char *PE = (unsigned char *) 0x40024000;
unsigned char *SYSCTL = (unsigned char *) 0x400FE000;
volatile unsigned int x=0;
volatile unsigned int y=0;
volatile unsigned char *gstatus = (unsigned char*)0x20002000;
volatile unsigned char *ystatus = (unsigned char*)0x20002200;
volatile unsigned char *rstatus = (unsigned char*)0x20002400;
volatile unsigned char *doStuff = (unsigned char*)0x20002600;


unsigned int *PA_int = (unsigned int *) 0x40004000;
unsigned int *PB_int = (unsigned int *) 0x40005000;
unsigned int *PC_int = (unsigned int *) 0x40006000;
unsigned int *PE_int = (unsigned int *) 0x40024000;
unsigned int *SYSCTL_int = (unsigned int *) 0x400FE000;

//set-up of Interrupt Port D
volatile int PD_IS_R __attribute__((at(0x40007404)));
volatile int PD_IEV_R __attribute__((at(0x4000740C)));
volatile int PD_IBE_R __attribute__((at(0x40007408)));
volatile int PD_IM_R __attribute__((at(0x40007410)));
volatile int PD_ICR_R __attribute__((at(0x4000741C)));
volatile int SYS_NVIC_R __attribute__((at(0xE000E100)));

//set-up of GPIO Port D
volatile int PD_DATA_R __attribute__((at(0x400073FC)));
volatile int PD_DIR_R __attribute__((at(0x40007400)));
volatile int PD_AF_R __attribute__((at(0x40007420)));
volatile int PD_DEN_R __attribute__((at(0x4000751C)));
volatile int PD_ULK_R __attribute__((at(0x40007520)));
volatile int PD_CR_R __attribute__((at(0x40007524)));
volatile int PD_PUL_R __attribute__((at(0x40007510)));

// System Control Base Address : SYSCTL (pg. 229 - MDS)
// Core Peripherals Base Address : M4CP (pg. 132 - MDS)
unsigned char *M4CP = (unsigned char *) 0xE000E000;

// SSI0 <> GPIO Port A (Clk: PA2, Fss: PA3, RX: PA4, TX: PA5) (pg. 892 - MDS)
// GPIO Port A (PA) Base Address (pg. 656 - MDS)
// SSI 0 Base Address (pg. 962 - MDS)
unsigned char *SSI0 = (unsigned char *) 0x40008000;

void SSIInit()        // (pg. 961 - MDS)
{
// 1. Enable Clock for GPIO & SSI
SYSCTL[0x61C] |= 0x1; // SSI0
```

```
// 2. Enable Alt. Function on GPIO
PA[0x420] |= 0x3C;    // PA:5.2
PA_int[0x52C/4] |=0x222200;//PCTL

// 3. Initialize SSI
SSI0[0x004] = 0x00;   // Initialize SSI (Master) CR1

// 4. Set SSI Clock Source
SSI0[0xFC8] = 0x0;    // System Clock

// 5. Set SSI Clock Prescale Divisor
SSI0[0x010] = 0x32;   // SSI Clock

// 6. Set SSI Configuration
SSI0[0x00]=0xCF;              // FRF = Microwave, DSS = 8-bit data

// 7. Enable SSI
SSI0[0x004] = 0x02;   // Enable SSI
}
void GPIOConfig(void)
{
              SYSCTL[0x608] |= 0x3F;              //Enable clock on GPIOA, GPIOB, GPIOC, GPIOD, GPIOE,
GPIOF

              //GPIOA
              PA[0x420] = 0x0;                    //Disable alternative functions
              PA[0x400] |= 0xEC;          //Output: pins 2-7 & 6 output
              PA[0x510] |= 0xEC;          //Pull up resistors: pins 2-7& 6 output
              PA[0x51C] |= 0xEC;          //Digital Enable: pins 2-7& 6 output

              //GPIOB
              PB[0x420] = 0x0;                    //Disable alternative functions
              PB[0x400] |= 0xFF;          //Output: pins 0-7
              PB[0x510] |= 0xFF;          //Pull up resistors: pins 0-7
              PB[0x51C] |= 0xFF;          //Digital Enable: pins 0-7

              //GPIOC
              PC[0x420] = 0x0;                    //Disable alternative functions
              PC[0x400] |= 0xF0;          //Output: pins 4-7
              PC[0x510] |= 0xF0;          //Pull up resistors: pins 4-7
              PC[0x51C] |= 0xF0;          //Digital Enable: pins 4-7


              //GPIOE
              PE[0x420] = 0x0;                    //Disable alternative functions
              PE[0x400] |= 0x3F;          //Output: pins 0-5
              PE[0x510] |= 0x3F;          //Pull up resistors: pins 0-5
              PE[0x51C] |= 0x3F;          //Digital Enable: pins 0-5

              // >> GPIO Port D     input from the keyboard
  // 1. Unlock Pins / Disable Alt. Function
  PD_ULK_R=0x4C4F434B;
                      // Unlock Code (pg. 681 - MDS)
  PD_CR_R=0x01;                                     // Unlock Pins - PF0 (BUTTON1) (0x01=0b0000.0001) GPIO
Commit : GPIOCR (pg. 682 - MDS)
  // 2. Set Pins as Input or Output
  PD_DIR_R=0xFE;                                    // Enable as Input - PF0 (BUTTON1) (0xFE=0b1111.1110)
GPIO Direction : GPIODIR (pg. 660 - MDS)
  // 3. Set for Pull-Up or Pull-Down
  PD_PUL_R=0x01;                                    // Set for Pull-Up Resistor (0x01 = 0b0000.0001) GPIO
```

```
Pull-Up Select : GPIOPUR (pg. 674 - MDS)
  // 4. Enable Pins
  PD_DEN_R=0x01;                                    //Enable Pins - PF0 (BUTTON1) (0x01 = 0b0000.0001)
GPIO Digital Enable : GPIODEN (pg. 679 - MDS)
}
void setInt(void)
{
       //setting up Interrupt for PORT D0
       PD_IS_R=0x0;//set to trigger on edge 0=edge 1=level
       PD_IEV_R=0x0;//setting edge to 0 which equals rising edge 0=falling 1=rising
       PD_IBE_R=0x0;//set to 0 which takes the input from the IEV0=from IEV 1=both rising and falling
edge
       PD_IM_R=0x1;//set to 1 for it to recognize as an Interrupt 0=masked/not seen 1=enabled for
Interrupt
       PD_DEN_R=0x1;
       SYS_NVIC_R=0x8;

}
void write_Dat(unsigned char dat)
{
       PA_int[0x3FC/4]=0x80;
       PB_int[0x3FC/4]=0x1;
       PC_int[0x3FC/4]=(dat<<4);//PC 4-7
       PE_int[0x3FC/4]=(dat>>4);//PE 0-3
  __nop();
  __nop();
  __nop();
       PA_int[0x3FC/4]=0xC0;
  __nop();
  __nop();
  __nop();

}

void write_Dat2(unsigned short dat)
{

       write_Dat(dat >> 8);
       write_Dat(dat);
       __nop();
  __nop();
  __nop();
}

void write_Cmd(unsigned char cmd)
{
       PA_int[0x3FC/4]=0x80;
       PB_int[0x3FC/4]=0x0;

       PC_int[0x3FC/4]=(cmd<<4);
       PE_int[0x3FC/4]=(cmd>>4);
       PA_int[0x3FC/4]=0xC0;
  __nop();
  __nop();
  __nop();
       PA_int[0x3FC/4]=0xC0;
  __nop();
  __nop();
  __nop();
```

```
}
void set_Area(unsigned short x1, unsigned short x2, unsigned short y1, unsigned short y2)
{
		//set column
	write_Cmd(0x2A);
	write_Dat2(x1);
	write_Dat2(x2);
	//set row
	write_Cmd(0x2B);
	write_Dat2(y1);
	write_Dat2(y2);

}
void write_Color(unsigned short color)
{
	write_Cmd(0x2C);
	write_Dat2(color);
}

void set_Square(unsigned short x1, unsigned short x2, unsigned short y1, unsigned short y2,unsigned short
color)
{
	int i=0;
	set_Area(x1,x2,y1,y2);
	write_Cmd(0x2C);
	for(;i<(x2-x1)*(y2-y1);i++)
	{
		write_Dat2(color);
	}

}
void getX()
{
	SSI0[0x8]=0xD0;//send TX data to SSI
	__nop();
	__nop();
	x=SSI0[0x8];
	//0xD0
}
void getY()
{
	SSI0[0x8]=0x90;//send TX data to SSI
	__nop();
	__nop();
	y=SSI0[0x8];
	//0x90
}

void pstat(char rpstatus,char gpstatus,char ypstatus)
{
	if(rpstatus==0)
	{
		if(80<x>160 && 20<y>100)
		{
			set_Square(80, 160, 20, 100,red); //red
	//		PB[0x3FC]|=0x2;//turn on port B pin 1
		}
				rpstatus=1;
	}
		if(gpstatus==0)
		{
			if(80<x>160 && 120<y>200)
			{
				set_Square(80, 160, 120, 200,green); //green
	//				PB[0x3FC]|=0x4;//turn on port B pin 2
			}
					gpstatus=1;
		}
			if(ypstatus==0)
		{
			if(80<x>160 && 220<y>300)
			{
				set_Square(80, 160, 220, 300,yellow); //yellow
	//				PB[0x3FC]|=0x8;//turn on port B pin 3
			}
			ypstatus=1;
		}

	if(rpstatus==1)
	{
		if(80<x>160 && 20<y>100)
		{
			set_Square(90, 150, 30, 90,black); //red
	//			PB[0x3FC]&=0xFD;//turn on port B pin 1
		}
				rpstatus=0;
	}
		if(gpstatus==1)
		{
			if(80<x>160 && 120<y>200)
			{
				set_Square(90, 150, 130, 190,black);//green
		//			PB[0x3FC]&=0xFB;//turn on port B pin 2
			}
			gpstatus=0;
		}
			if(ypstatus==1)
		{
			if(80<x>160 && 220<y>300)
			{
				set_Square(90, 150, 230, 290,black); //yellow
		//			PB[0x3FC]&=0xF7;//turn on port B pin 3
			}
			ypstatus=0;
		}
	*rstatus=rpstatus;
	*gstatus=gpstatus;
	*ystatus=ypstatus;
}

void lights()
{
	char rpstatus=0;
	char gpstatus=0;
	char ypstatus=0;

	rpstatus=*rstatus;
	gpstatus=*gstatus;
	ypstatus=*ystatus;
	getX();//get x coordinates
	getY();//get y coordinates

	set_Square(80, 160, 20, 100,red);
	set_Square(90, 150, 30, 90,black);

	set_Square(80, 160, 120, 200,green);
	set_Square(90, 150, 130, 190,black);

	set_Square(80, 160, 220, 300,yellow);
	set_Square(90, 150, 230, 290,black);

	pstat(rpstatus,gpstatus,ypstatus);
	*doStuff = 0;
}

void GPIOPortD_Handler(void)
{
	char DOS = 1;
	*doStuff = DOS;

	PD_ICR_R=0x1;//1=Interrupt recognizes that it has been taken care of. 0=nothing happens
}

int main(void)
{
	*doStuff = 0;
	*rstatus=0;//red on or off
	*gstatus=0;//green on or off
	*ystatus=0;//yellow on or off
	//setup_data_command_pin(); // - PB0 D/C, PB6 TB_CS
	//setup_wr_rd_pins();                    // PA6 and PA7 as /WR and /RD
	//setup_DB8_11_pins();                   // PC4-7
	//setup_DB12_15_pins();                  // PE0-3
	GPIOConfig();
	setInt();
	SSIInit();
	LCD_Init();
	flash_screen(black);

  while (1)
  {
		if(*doStuff == 1)
			lights();
	}


}
```