SPI Color Display with 2D Vector Graphics

Archana Ramalingam
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 95192
archana.ramalingam@sjsu.edu

***Abstract* – Interfacing an LCD display with the microcontrollers is required in order to visualize the output of an embedded program. This paper describes the hardware and software design for the interfacing of LPC 1769 and Color TFT (Thin Film Transistor) LCD display. The goal is to display colorful patterns that can be displayed as screensavers. The description given here mostly concentrates on implementation, design and testing of the hardware and software components. LPCXpresso is the development platform used here. The pin connections between LPC and LCD modules are discussed. Also the data transfer methodology is explained in detail.**

***Keywords* – LPC 1769, LCD display, SPI, LPCXpresso, screensaver**

1. INTRODUCTION

The LPC 1769 is an ARM Cortex-M3 microcontroller, which is often used for applications that require high level of integration but low power dissipation. The rendering of a 2D or a 3D graphics is primary importance in computer graphics. This is accomplished by using a 1.8" TFT color LCD display, with a built in controller. The data transfer between LCD and LPC modules occur via SPI interface. The aim of this work is to design, implement and track data transfer through SPI interface. LPC 1769, LCD TFT display, LM 7805 voltage regulator and LPCXpresso are also discussed briefly.

2. METHODOLOGY

2.1 Objectives and technical challenges

As discussed earlier, the main objective of this work is to display screensavers and understand the communication between the LPC module and LCD module. The implementation of SPI interface has the following objectives:

1. Familiarization of LPC 1769, LCD TFT color display, 7805 IC.
2. Design and implementation of a power circuit for the LPC module.
3. Practical exposure and experience with LPCXpresso IDE.
4. Inclusion of a graphical engine in the prototype board.
5. Learning 2D Vector graphics and the corresponding C code to implement this work.
6. Designing, implementing, testing and debugging CPU, LCD and other components.

The challenges faced are:

1. Understanding Pin structure of LPC and LCD modules.
2. Understanding transformation and rotation concepts of vector graphics
3. Calculating the random coordinates for the squares

2.2 Problem formulation and design

The overall system layout should be prototyped first. Our prototype board layout is shown in the figure 1 below. Data is transferred from the host to the LPC module, which in turn transfers data to the LCD module. The SPI communication follows a master-slave model of control.

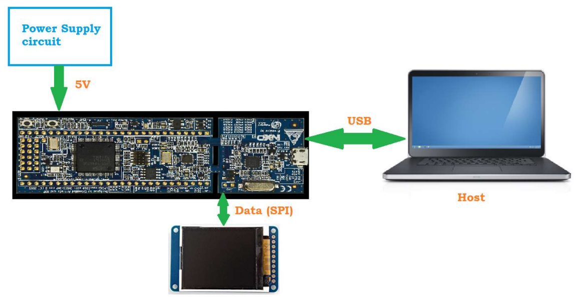


Fig.1. System layout

As for the software design, the aim is to draw square within a square and perform this operation in a random fashion. In order to do this, we first start with a line drawing function. We then use the following equation to get the vertices of the second level of squares. The value of lambda is kept as 0.8, while it can be changed to 0.2 for user defined value.

P(x,y) = P_1(x_1,y_1) + lamda * (P_2(x_2,y_2) - P_1(x_1,y_1)) → 1

Using rand(), we are randomizing the starting vertex of each consecutive square. The color for each set of squares are fixed.

3. IMPLEMENTATON

The design methodology describes the layout of the prototype system, hardware and software design. After successful design of the power circuit, the LPC and LCD modules are soldered to the circuit board.

3.1 Hardware Design

The hardware design can be divided into the following sections.

1. Power circuit

In order to provide power for the entire circuit, which includes LPC and LCD module, a power circuit is needed. An adaptor that outputs 9V DC is connected to the 1st pin of LM 7805 IC voltage regulator. A capacitor of 10 µf and a switch is present in parallel between the adaptor and the regulator. The 3rd pin (output) of 7805 is connected in parallel to another capacitor of 10 µf, a resistor of 1kΩ, to protect LED from transients and a LED, which indicates if the power circuit is ON or OFF. The 2nd pin of 7805 is grounded.

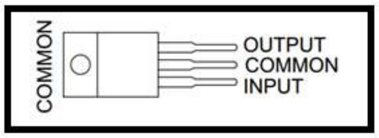


Fig.2. LM 7805 IC pin description

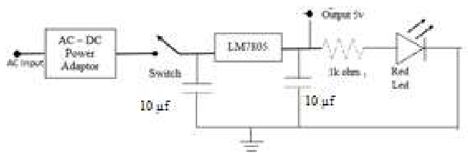


Fig.3. Power circuit diagram

TABLE I. POWER UNIT SPECIFICATIONS

Items	Description
Power Socket	Connector to external power
AC Adapter	External Adapter 110AC/5VDC @1500 mA
Switch	SPST Switch to control ON/OFF
LM7805	Voltage regulator, 5V
Capacitorsx2	10µF
LED	Power Indicator (8mA,1.8 VDC)
Resistors	1 KΩ

2. LPC and LCD interfacing

The aim of this subsystem is to implement the master-slave SPI interface between the mentioned modules. LPC module has four important signals: SCK (Serial Clock - output from master), MOSI (Master Output Slave Input - output from master), MISO (Master Input Slave Output - output from slave) and CS (Chip Select - active low - output from master). It has up to 512 Kb flash memory and up to 64 kB data memory. Low power consumption and wider temperature range is another reason to choose this module. The LCD module is 128x160 pixel display with a TFT driver ST7735R, which can display full 18-bit color. Due to the thin film transistor technology, the display has very good resolution.

The LPC module acts as master and its inputs are in the form of MISO, whereas the LCD acts as a slave and its inputs are in the form of MOSI. Serial clock signal and chip select are taken as inputs from the master. The connection of signals between two modules is shown in the fig. 4 below.

The connection between LPC and LCD requires the knowledge of Pin configuration of both the modules. The LPC 1769 pin layout is shown in figure 5. The pins that are required for the interface alone are shown here. The LCD module pin layout is very much restricted, which is shown in figure 6.

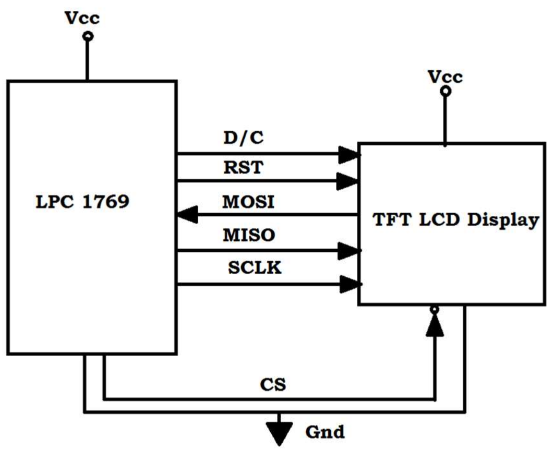


Fig.4. LPC to LCD signals

The corresponding pins to be connected between the 2 modules is shown in the form of a table below. It is to be noted that only the pins that are required for the SPI interface are shown here.

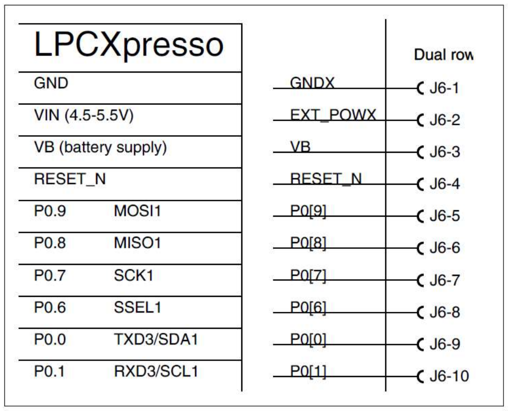


Fig.5. LPC pin layout

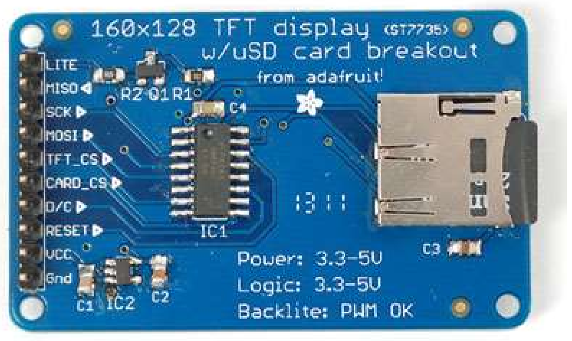


Fig.6. LCD TFT color display pin layout

TABLE II. PIN CONNECTIONS

LPC Pin			LCD Pin	
Description	Port.Pin	Pin No.	Description	Pin No.
Vcc	P0.28	J6-28	LITE	1
MOSI	P0.9	J6-5	MOSI	4
SCK	P0.7	J6-8	SCK	3
SSEL	P0.6	J6-8	TFT_CS	5

The D/C of LCD is connected to the pin 23 and RESET to pin 24. Reset needs to be 1, so it is activated suing logic 0. 0x11 is sent to awake the LCD. 0x29 provides display. The LCD is made a slave.

TABLE III. THE COMPONENT SPECIFICATIONS

UNIT	**DESCRIPTION**	**NOTES**
CPU Model NXP LPC 1769	120 MHz	Clock Rate
	32/16kB	Cache Memory
	3.3 V (2.4 V to 3.6 V)	Power Consumption
	ARM Cortex M3	Architecture
	512 kB	Flash Memory
	64kB	Data Memory
I/O Peripheral Controllers	SPI Protocol used	
Power Supply	5V DC @1500 mA	Description in Table I
External Flash	AT45D011	14 pins

TABLE IV. BILL OF MATERIALS

Description	**Quantity**
Wire Wrapping Board	1
Wire Wrapping Tool Kit	1
Wire for Wire Wrapping	1
DC Power Supply 6V 10A	1
1/4 inch stands for board	4
Red LED	1
LM7805 5V Regulator	1
10μF Capacitor	2
1kΩ resistor	3
SPST Switch	2
AT45D011	1
LPC 1769	1
Header Pins for Wire Wrapping	3
1.8” TFT Color LCD display	1

3.2 Software Design

The software component of this project involves C program run on an IDE named LPCXpresso. This is provided by NXP to run, build, test and debug programs for LPC 1769. The IDE is low cost, user friendly and can manage multiple workspaces and multiple projects simultaneously. The LCD display demands specific opcodes cast for specific operations, setting registers, initializations, etc. Datasheets of both modules have been immensely helpful in getting the opcodes for the program. Also, the relevant header files and functions to be included in the program is taken care of as well.

3.2.1 Algorithm for Software implementation

The algorithm explains the basic steps followed to setup and run the software components.

Step 1: Start

Step 2: Initialize SPI

- Set PCONP’s 21st bit to enable SSP0
- SSP_CLK selected as PCLK/4, by writing PCLKSEL1 as 0
- Set J6 11-14 pins functionality as SSP 0
- Set SSEL0 as GPIO out
- SSP0 data width is set to 8 bit
- SCR register value to 7
- Pre scale CLK value set to 2

Step 3: Initialize LCD

- Set SSEL0 as 0, to make LCD slave
- D/C connected to J6-23
- RESET connected to J6-24
- Set both pins as output
- P0.24 pin value is set as logic 1
- Provide delay of 500 ms
- P0.24 pin value is set as logic 0
- P0.24 pin value is set as logic 1
- Provide delay of 500 ms
- P0.24 pin value is set as logic 0
- Initialize SSP buffer to 0
- Send 0x11 to SSP 0 to wake LCD from sleep
- Give a delay to LCD
- Send 0x29 to SSP0 to display
- Give a delay to LCD

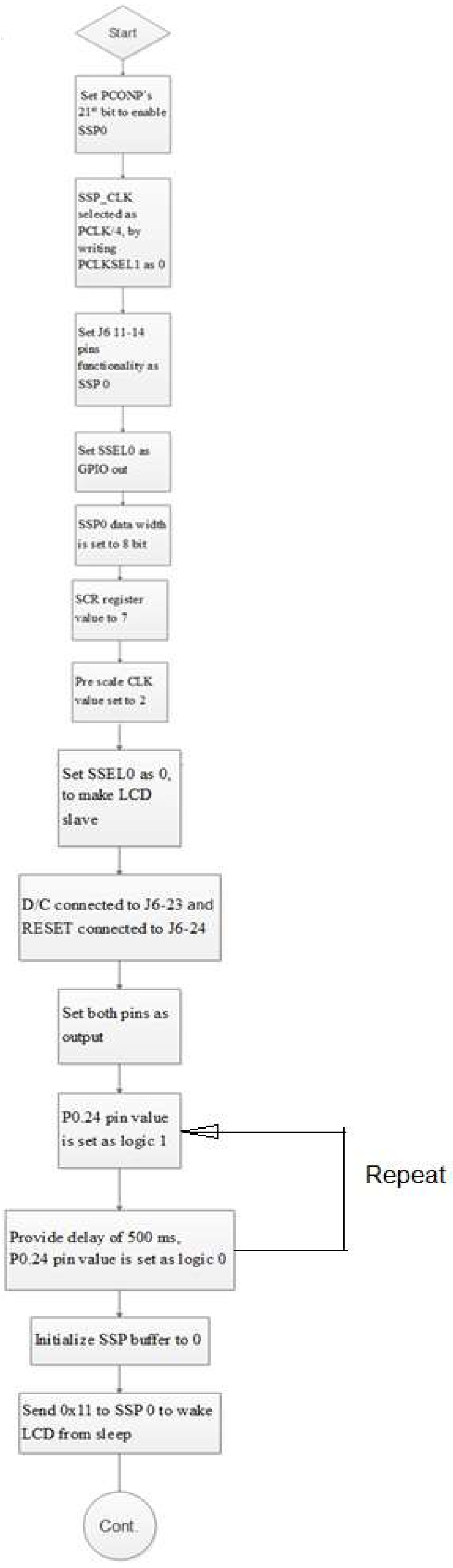
Step 4: Draw square block screensaver

- Fill the entire LCD display with white using fill rectangle method
- Using draw square function draw squares at random positions
- Display more than 7 levels in each square block

Step 5: Stop

3.2.2 Flowchart

The following flowchart describes the steps of implementation used in pseudo code.


Start
Set PCONP's 21st bit to enable SSP0
SSP_CLK selected as PCLK/4, by writing PCLKSEL1 as 0
Set J6 11-14 pins functionality as SSP 0
Set SSEL0 as GPIO out
SSP0 data width is set to 8 bit
SCR register value to 7
Pre scale CLK value set to 2
Set SSEL0 as 0, to make LCD slave
D/C connected to J6-23 and RESET connected to J6-24
Set both pins as output
P0.24 pin value is set as logic 1
Repeat
Provide delay of 500 ms, P0.24 pin value is set as logic 0
Initialize SSP buffer to 0
Send 0x11 to SSP 0 to wake LCD from sleep
Cont.

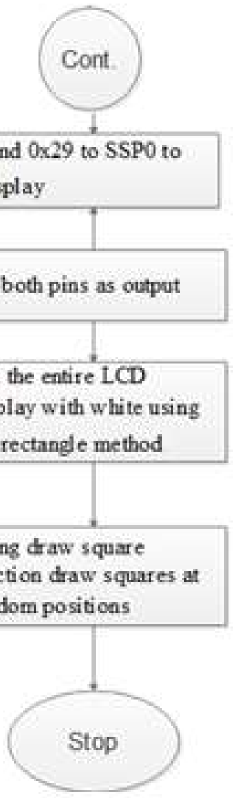

Cont.
Send 0x29 to SSP0 to display
Set both pins as output
Fill the entire LCD display with white using fill rectangle method
Using draw square function draw squares at random positions
Stop

Fig.7. Flowchart of software implementation

3.2.3 Pseudo code

A. Software implementation

```
int main (void)
{
  uint32_t i, portnum = PORT_NUM;
  portnum = 1 ; // For use 1 (LCD)

   if ( portnum == 0 )
    SSP0Init();

// Initialize SSP port To
  else if ( portnum == 1 )
    SSP1Init();
  for ( i = 0; i < SSP_BUFSIZE; i++
)
  {
    src_addr[i] = (uint8_t)i;
    dest_addr[i] = 0;
  }

  //Initialize LCD
      lcd_init();

      int16_t x;
      int16_t y;
while(1)
{
      fillrect(0, 0, 220, 220,
BLACK); // For the background

      x=random_between(0,127);
      y=random_between(0,159);
      drawsquare(x,y,WHITE);

      x=random_between(0,127);
      y=random_between(0,159);
      drawtriangle(x,y,WHITE);

      x=random_between(0,127);
      y=random_between(0,159);
      drawsquare(x,y,RED);

      x=random_between(0,127);
      y=random_between(0,159);
      drawtriangle(x,y,RED);

      x=random_between(0,127);
      y=random_between(0,159);
      drawsquare(x,y,YELLOW);

      x=random_between(0,127);
      y=random_between(0,159);
      drawtriangle(x,y,YELLOW);

      x=random_between(0,127);
      y=random_between(0,159);
      drawsquare(x,y,BLUE);

      x=random_between(0,127);
      y=random_between(0,159);
      drawtriangle(x,y,BLUE);

      x=random_between(0,127);
      y=random_between(0,159);
      drawsquare(x,y,GREEN);

      x=random_between(0,127);
      y=random_between(0,159);
      drawtriangle(x,y,GREEN);

}
  return 0;
}
```

B. Code execution on IDE

In order to run this code on the LPC modue, the program is run on the IDE on the host laptop. Then it is copied to LPC board and run.

1. Create LPCXpresso C project in the IDE
2. Import draw a line folder into IDE
3. Open LCD_test.c file
4. Connect LPC to laptop using USB
5. Build the project
6. Run the code after debugging.

4. Testing and verification

This section explains the testing and verification required to be done for this circuit. Once the lights on the microcontroller light up, we can import the project into IDE including ssp.c, sp.h and SSP_Test.c files. If the project has no errors, it will run successfully with a connection to CPU module. The data will be transferred in the form of string to CPU via USB cable. This will create the screensaver display on the LCD screen. Thus communication between the LPC and LCD via the SPI interface has been tested and verified successfully.

5. Conclusion

Design and implementation of the power circuit and SPI interface communication has been implemented and tested successfully. The data was transferred via the SPI interface from the CPU module to the LCD module and the screensaver was displayed. This project required to study the functions and characteristics of LPC 1769, LM 7805, LCD TFT Color Display, LPCXpresso, 2D vector graphics and soldering techniques.

6. Acknowledgement

I express deep gratitude to Professor Li for providing the motivation for the implementation of this project. He also taught about the functional component specifications, basics of pin layout for various modules, designing of power circuit, 2D vector graphics.

7. References

[1] NXP, "UM10360 user manu.pdf", UM10360 LPC176x/5x User manual.

[2] H. Li, Lecture Notes of CMPE 240, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.

[3] LM7805 5V Regulator Datasheet https://www.sparkfun.com/datasheets/Components/LM7805.pdf

[4]LPCXpresso 1769 Datasheet http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf

8. Appendix

```
========================================
 Name       : Screensaver.c
 Author     : Archana Ramalingam
 Version    :
 Copyright  :
 Description : main definition
========================================
#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "longhorn_sunset.h"


// Variable used to store CRP value. Placed automatically by linker when "Enable Code Read Protect" selected.
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;
#include "LPC17xx.h"  // LPC17xx definitions
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define PORT_NUM            1
#define LOCATION_NUM        0
#define pgm_read_byte(addr) (*(const unsigned char *)(addr))
uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];
int colstart = 0;
int rowstart = 0;
/*****************************************
*********************************
** Function name:       LCD_TEST
**
** Descriptions:        Draw line function
** parameters:           None
** Returned value:       None
******************************************
*********************************/
//LCD
#define ST7735_TFTWIDTH  127
#define ST7735_TFTHEIGHT 159
#define ST7735_CASET   0x2A
#define ST7735_RASET   0x2B
#define ST7735_RAMWR   0x2C
#define swap(x, y) { x = x + y; y = x - y; x = x - y; }
// delay
void delay(int ms)
{
        int count = 24000;
        int i;
        for ( i = count*ms; i--; i > 0);
}
//Colours
#define GREEN 0x00FF00
#define BLACK 0x000000
#define RED 0xFF0000
#define BLUE 0x0000FF
#define WHITE 0xFFFFFF
#define PINK 0xFFC0CB
#define PURPLE 0x800080
#define YELLOW 0xFFFF00
#define LIME 0x00FF00
#define MAGENTA 0xFF00FF
#define CYAN 0x00FFFF
#define SILVER 0xC0C0C0
```

```
#define GREY 0x808080
#define ORANGE 0xFFA500
#define BROWN 0xA52A2A
#define MAROON 0x800000
//Axes
int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;
int cursor_x = 0, cursor_y = 0;
//for writing data into the SPI
void spiwrite(uint8_t c)
{
   int portnum = 1;
   src_addr[0] = c;
   SSP_SSELToggle( portnum, 0 );
   SSPSend( portnum, (uint8_t *)src_addr, 1 );
   SSP_SSELToggle( portnum, 1 );
}
//writing commands into SPI
void writecommand(uint8_t c) {
   LPC_GPIO0->FIOCLR |= (0x1<<21);
   spiwrite(c);
}
//making LCD ready to write data
void writedata(uint8_t c) {
   LPC_GPIO0->FIOSET |= (0x1<<21);
   spiwrite(c);
}
//writing data to the LCD
void writeword(uint16_t c) {
   uint8_t d;
   d = c >> 8;
   writedata(d);
   d = c & 0xFF;
   writedata(d);
}
//write colour
void write888(uint32_t color, uint32_t repeat) {
   uint8_t red, green, blue;
   int i;
   red = (color >> 16);
   green = (color >> 8) & 0xFF;
   blue = color & 0xFF;
   for (i = 0; i< repeat; i++) {
           writedata(red);
      writedata(green);
      writedata(blue);
   }
}
void setAddrWindow(uint16_t x0, uint16_t y0,
uint16_t x1, uint16_t y1) {
    writecommand(ST7735_CASET);
    writeword(x0);
    //delay(10);
    writeword(x1);
    writecommand(ST7735_RASET);
    writeword(y0);
    //delay(10);
    writeword(y1);
}
void drawPixel(int16_t x, int16_t y, uint32_t color) {
   if((x < 0) ||(x >= _width) || (y < 0) || (y >= _height))
return;
   setAddrWindow(x,y,x+1,y+1);
   writecommand(ST7735_RAMWR);
```

```
    write888(color, 1);

}

void HLine(int16_t x0,int16_t x1,int16_t y,uint16_t color){

    width = x1-x0+1;

    setAddrWindow(x0,y,x1,y);

    writecommand(ST7735_RAMWR);

    write888(color,width);

}

void VLine(int16_t x,int16_t y0,int16_t y1,uint16_t color){

    width = y1-y0+1;

    setAddrWindow(x,y0,x,y1);

    writecommand(ST7735_RAMWR);

    write888(color,width);

}

//Initialize LCD

void lcd_init()

{

/*

 * portnum     = 0;

 * cs         = p0.16 / p0.6

 * rs        = p0.21

 * rst        = p0.22

 */

    uint32_t portnum = 1;

    int i;

    printf("LCD initialized\n");

    /* Notice the hack, for portnum 0 p0.16 is used */

    if ( portnum == 0 )

      {

        LPC_GPIO0->FIODIR |= (0x1<<16); /* SSP1, P0.16 defined as Outputs */

      }

      else

      {

        LPC_GPIO0->FIODIR |= (0x1<<6); /* SSP0 P0.6 defined as Outputs */

      }

    /* Set rs(dc) and rst as outputs */

    LPC_GPIO0->FIODIR |= (0x1<<21); /*rs/dc P0.22 defined as Outputs */

    LPC_GPIO0->FIODIR |= (0x1<<22); /* rst P0.21 defined as Outputs */

    /* Reset sequence */

    LPC_GPIO0->FIOSET |= (0x1<<22);

 delay(500);                       /*delay 500 ms */

    LPC_GPIO0->FIOCLR |= (0x1<<22);

 delay(500);                       /* delay 500 ms */

    LPC_GPIO0->FIOSET |= (0x1<<22);

 delay(500);                       /* delay 500 ms */

     for ( i = 0; i < SSP_BUFSIZE; i++ )     /* Init RD and WR buffer */

       {

           src_addr[i] = 0;

           dest_addr[i] = 0;

       }

     /* Sleep out */

     SSP_SSELToggle( portnum, 0 );

     src_addr[0] = 0x11;    /* Sleep out */

     SSPSend( portnum, (uint8_t *)src_addr, 1 );

     SSP_SSELToggle( portnum, 1 );

delay(200);

    /* delay 200 ms */

    /* Display on */

     SSP_SSELToggle( portnum, 0 );
    src_addr[0] = 0x29;    /* Display On */
    SSPSend( portnum, (uint8_t *)src_addr, 1 );
    SSP_SSELToggle( portnum, 1 );
   /* delay 200 ms */
delay(200);
}
void fillrect(int16_t x0, int16_t y0, int16_t x1, int16_t
y1, uint32_t color)
{
            int16_t i;
            int16_t width, height;
            width = x1-x0+1;
            height = y1-y0+1;
            setAddrWindow(x0,y0,x1,y1);
            writecommand(ST7735_RAMWR);
            write888(color,width*height);
}
//Draw line function
void  drawline(int16_t  x0,  int16_t  y0,  int16_t  x1,
int16_t y1,uint32_t color) {
            int16_t slope = abs(y1 - y0) > abs(x1 - x0);
            if (slope) {
            swap(x0, y0);
            swap(x1, y1);
            }
            if (x0 > x1) {
            swap(x0, x1);
            swap(y0, y1);
            }
            int16_t dx, dy;
            dx = x1 - x0;
            dy = abs(y1 - y0);
            int16_t err = dx / 2;
            int16_t ystep;
            if (y0 < y1) {
            ystep = 1;
            } else {
            ystep = -1;
            }
            for (; x0<=x1; x0++) {
            if (slope) {
            drawPixel(y0, x0, color);
            } else {
            drawPixel(x0, y0, color);
            }
            err -= dy;
            if (err < 0) {
            y0 += ystep;
            err += dx;
            }
            }
}
void drawsquare(int16_t x, int16_t y, uint32_t color)
{
            int16_t a0,a1,a2,a3;
            int16_t b0,b1,b2,b3;
            uint32_t j;
            int16_t x0,x1,x2,x3;
            int16_t y0,y1,y2,y3;
            x0=x;
            y0=y;
            x1=x+40;
            y1=y;
            x2=x+40;
```

```
y2=y+40;
x3=x;
y3=y+40;
            drawline(x0,y0,x1,y1,color);
            delay(10);
            drawline(x1,y1,x2,y2,color);
            delay(10);
            drawline(x2,y2,x3,y3,color);
            delay(10);
            drawline(x3,y3,x0,y0,color);
for(j=0;j<10;j++)
{
            a0=(0.8*(x1-x0))+x0;
            b0=(0.8*(y1-y0))+y0;

            a1=(0.8*(x2-x1))+x1;
            b1=(0.8*(y2-y1))+y1;

            a2=(0.8*(x3-x2))+x2;
            b2=(0.8*(y3-y2))+y2;

            a3=(0.8*(x0-x3))+x3;
            b3=(0.8*(y0-y3))+y3;
            drawline(a0,b0,a1,b1,color);
            delay(10);
            drawline(a1,b1,a2,b2,color);
            delay(10);
            drawline(a2,b2,a3,b3,color);
            delay(10);
            drawline(a3,b3,a0,b0,color);
                        x0=a0;
                        x1=a1;
                        x2=a2;
                        x3=a3;
                        y0=b0;
                        y1=b1;
                        y2=b2;
                        y3=b3;
            }
}
void drawtriangle(int16_t x, int16_t y, uint32_t color)
{
            int16_t a0,a1,a2;
            int16_t b0,b1,b2;
            uint32_t j;
            int16_t x0,x1,x2;
            int16_t y0,y1,y2;
            x0=x;
            y0=y;
            x1=x+30;
            y1=y;
            x2=x+30;
            y2=y+30;

                        drawline(x0,y0,x1,y1,color);
                        delay(10);
                        drawline(x1,y1,x2,y2,color);
                        delay(10);
                        drawline(x2,y2,x0,y0,color);
                        delay(10);


            for(j=0;j<10;j++)
```

```
		{
				a0=(0.8*(x1-x0))+x0;
				b0=(0.8*(y1-y0))+y0;

				a1=(0.8*(x2-x1))+x1;
				b1=(0.8*(y2-y1))+y1;

				a2=(0.8*(x0-x2))+x2;
				b2=(0.8*(y0-y2))+y2;
				drawline(a0,b0,a1,b1,color);
				delay(10);
				drawline(a1,b1,a2,b2,color);
				delay(10);
				drawline(a2,b2,a0,b0,color);
				x0=a0;
				x1=a1;
				x2=a2;
				y0=b0;
				y1=b1;
				y2=b2;
		}
}
int random_between(int min, int max) {
    return rand() % (max - min + 1) + min;
}
// Main function
int main (void)
{
  //EINTInit();
  uint32_t i, portnum = PORT_NUM;
  portnum = 1 ; /* For LCD use 1 */
  /*      SystemClockUpdate()      updates      the
SystemFrequency variable */
//  SystemClockUpdate();
   if ( portnum == 0 )
    SSP0Init();           /* initialize SSP port */
  else if ( portnum == 1 )
    SSP1Init();
  for ( i = 0; i < SSP_BUFSIZE; i++ )
  {
    src_addr[i] = (uint8_t)i;
    dest_addr[i] = 0;
  }
  //initialize LCD
		lcd_init();
		int16_t x;
		int16_t y;
		 /*drawPixel(0, 0, WHITE);
		 drawPixel(1, 0, WHITE);
		 drawPixel(0, 1, WHITE);
		 drawPixel(1, 1, WHITE);
		 drawPixel(2, 0, WHITE);
		 drawPixel(3, 0, WHITE);
		 drawPixel(4, 0, WHITE);
		drawPixel(128, 0, WHITE);
		drawPixel(0, 160, WHITE);
		 drawPixel(127, 0, WHITE);
		 drawPixel(126, 0, WHITE);
		 drawPixel(0, 159, WHITE);
		 drawPixel(0, 158, WHITE);*/
while(1)
{
		fillrect(0, 0, 220, 220, BLACK);
x=random_between(0,127);
y=random_between(0,159);
drawsquare(x,y,WHITE);

x=random_between(0,127);
y=random_between(0,159);
drawtriangle(x,y,WHITE);

x=random_between(0,127);
y=random_between(0,159);
drawsquare(x,y,YELLOW);

x=random_between(0,127);
y=random_between(0,159);
drawtriangle(x,y,YELLOW);

x=random_between(0,127);
y=random_between(0,159);
drawsquare(x,y,RED);

x=random_between(0,127);
y=random_between(0,159);
drawtriangle(x,y,RED);

x=random_between(0,127);
y=random_between(0,159);
drawsquare(x,y,GREEN);

x=random_between(0,127);
y=random_between(0,159);
drawtriangle(x,y,GREEN);

x=random_between(0,127);
y=random_between(0,159);
drawsquare(x,y,PINK);

x=random_between(0,127);
y=random_between(0,159);
drawtriangle(x,y,PINK);
}
  return 0;
}
//End//
```