# CMPE 240 - Advanced Computer Design

**Assignment:** 2D – Graphics Design

**Name:** Tirumala Saiteja Goruganthu

**SJSU ID:** 016707210

**Team:** Student_Team1 (Harish Marepalli, Debashish Panigrahi, Shahnawaz Idariya are my team members)

**Professor:** Harry Li

**Date:** 10/15/2022

**Target Board:** LPC1769

**Purpose:**

This assignment focuses on collecting data from a CPU Module to implement 2D visual designs on an LCD Display. The CPU Module utilized in this project is the LPC 1769, which is based on the ARM Cortex M0 Core. For the implementation of our task, we use the MCUXpresso IDE. The main goal of this assignment is to learn how to use the SPI Communication Protocol between an LCD (ST7735) and a CPU Module (LPC1769) to show several 2D screen savers. Following is the description of the tasks given to achieve the goal.

**Given Tasks:**

Design and prototype LPC1769 micro-processor system board and enable a SPI LCD.

1. Display and generate 2D screen saver of rotating squares based on vector graphics.
    a. To use P(x,y) = P_1(x_1,y_1) + lamda * ( P_2(x_2,y_2) - P_1(x_1,y_1)) with lamda = 0.8 by default, and lamda = 0.2 when prompted for user selected input.
    b. Create two dimensional rotating patterns with data set of "parent" square.
    c. Randomized location by using rand() function.
    d. Randomized reduction of the parent square.
    e. Choose one color for each set of rotation patterns and rotates at least 10 levels or higher.
    f. Continue to display each set of patterns without erasing the patterns.

2. Generate 2D trees with its branches level no less than 10 or higher based on vector graphics formula discussed in the class.
    a. To use P(x,y) = P_1(x_1,y_1) + lamda * ( P_2(x_2,y_2) - P_1(x_1,y_1)) with lamda = 0.8 by default for tree branch reduction.
    b. Create patch of forest by modifying one parent tree.
    c. Randomized location of the new trees by using rand() function.
    d. Randomized reduction of the parent tree trunks and branches.
    e. Randomized angles for the branches.
    f. Continue to display trees without erasing till the keyboard input detected.

**Components Used:**

1. Prototype Board
2. LPC1769 inserted into female header pins and soldered to the board.
3. LCD TFT ST7735 Display Device.
4. USB probe to dump the program into on-chip flash memory.

**Theory and Formulation:**

The LPC1769 and 1.8" TFT LCD Display are soldered together on a prototype board and connected to the power circuit to work as a standalone module. In SPI Communication, we have a Master – Slave architecture where one-part acts as a Master of the communication and the other acts as a Slave. This type of architecture is implemented by connecting LCD module as a slave and LPC CPU Module as a master. There are four main SPI logic signals used. They are 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). Data Transfer take place from host computer to CPU module through the USB link and then the CPU transfers the data to the LCD Display module.

a. To implement a 2D screen saver of rotating squares we select a set of points along the sides the outer square which make the vertices of the inside square. Vector equation followed in implementation of rotating square screen saver is as follows, $P = P_i + \lambda * d$, where $d = P_{i+1} - P_i$. To draw the rotating squares, we by default set $\lambda = 0.8$ and then let the user select $\lambda = 0.2$. The same process is repeated, and squares are drawn inside the main square. These rotating squares are then iterated over with randomized data sets and colors to form multiple rotating squares all over the LCD.

b. To implement the second screen saver, we must draw multiple trees to create an image of a forest. The algorithm used is as below and is also discussed during the class.

c. First step is to draw the tree trunk and then generate the child nodes i.e., the center, the left and the right nodes.

   The center branch is generated using the equation,
   $P_c = P_{end\_pt} + \lambda * d$ where $d = P_{end\_pt} - P_{start\_pt}$ and $\lambda = 0.8$.

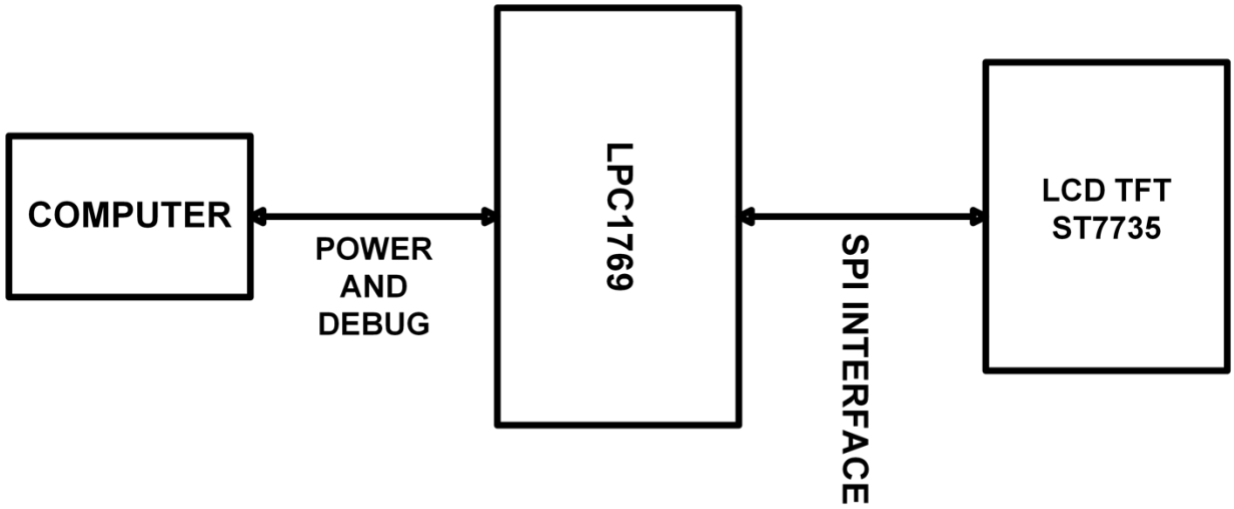
   To draw left and right branches we need to rotate the center branch with reference to the trunk end point.

   Translate the reference point to origin and rotate the vector along the new reference point.

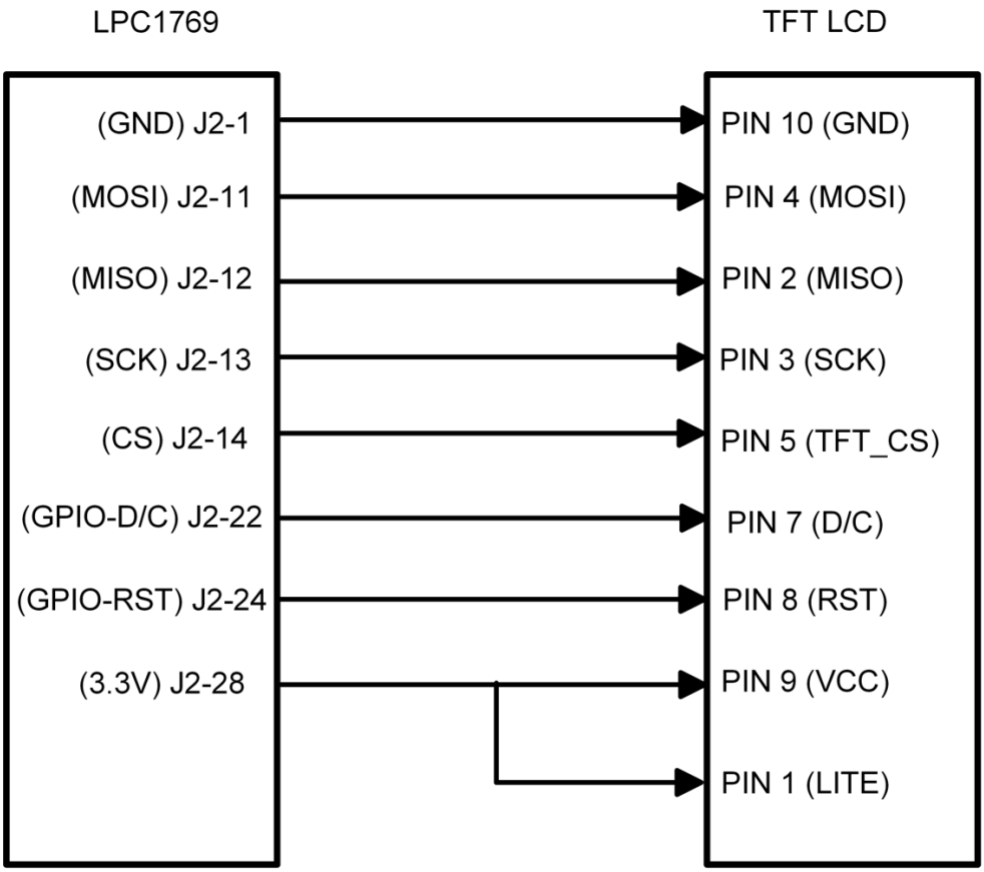
   Finally, translate back to the reference point.

**Rubrics - Block Diagrams:**

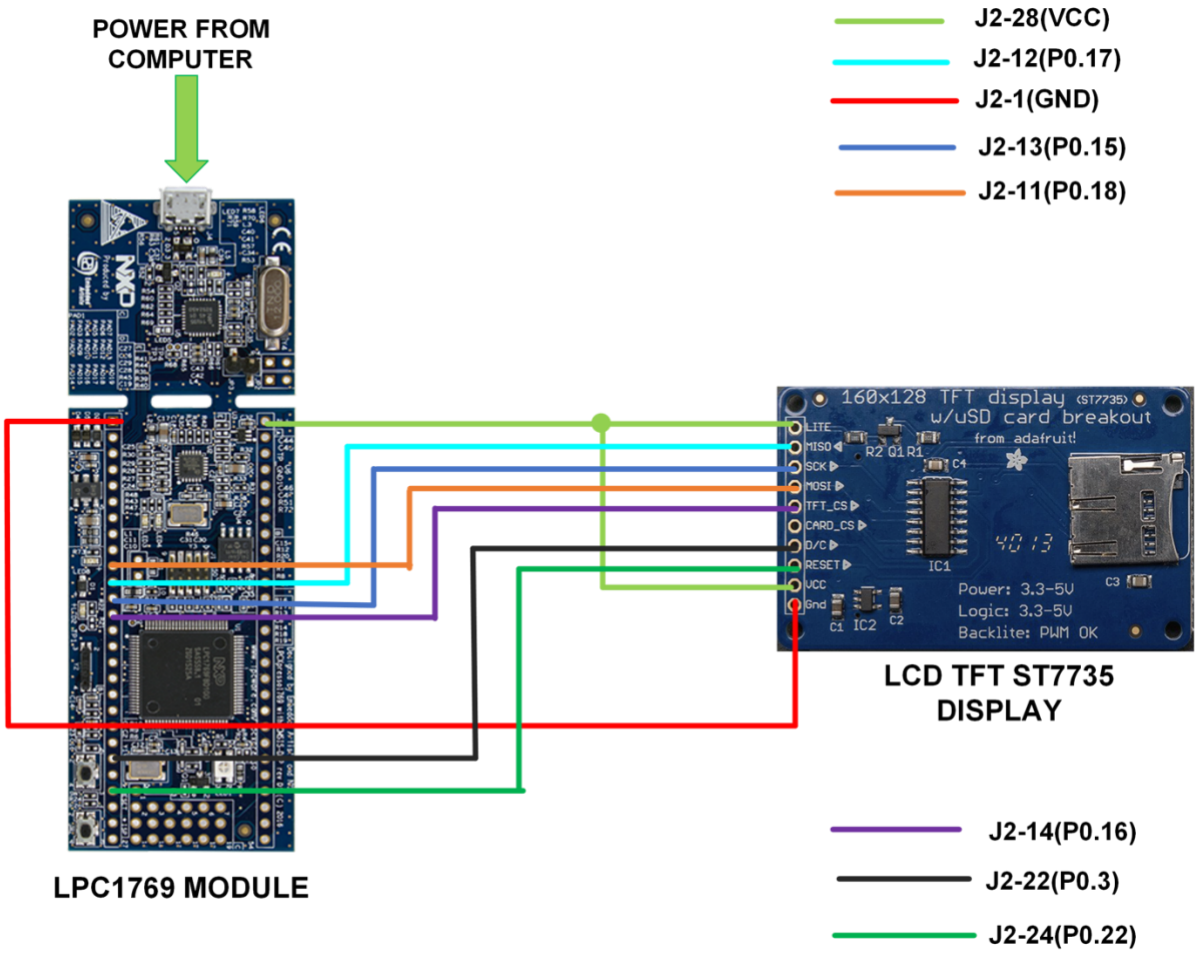
1. The block diagram of the entire system setup including the computer block (Done using Visio tool):




2. The system block diagram of the SPI color LCD interface (Done using Visio tool):



3. The schematic diagram of the interface connections between the LPC1769 interface to LCD color display panel (Done using Visio tool):

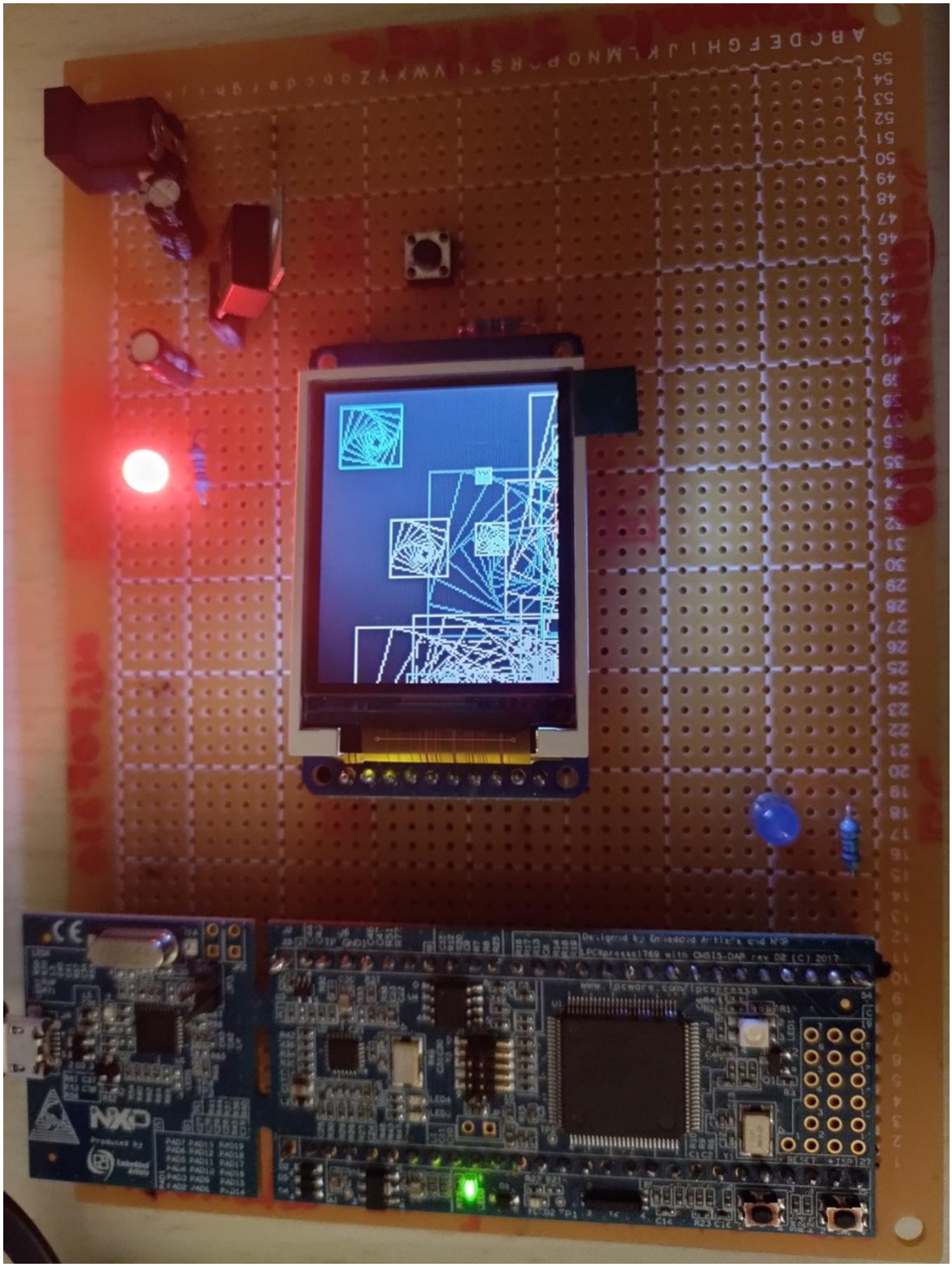



4. The pin-connectivity table for the interface is as follows:

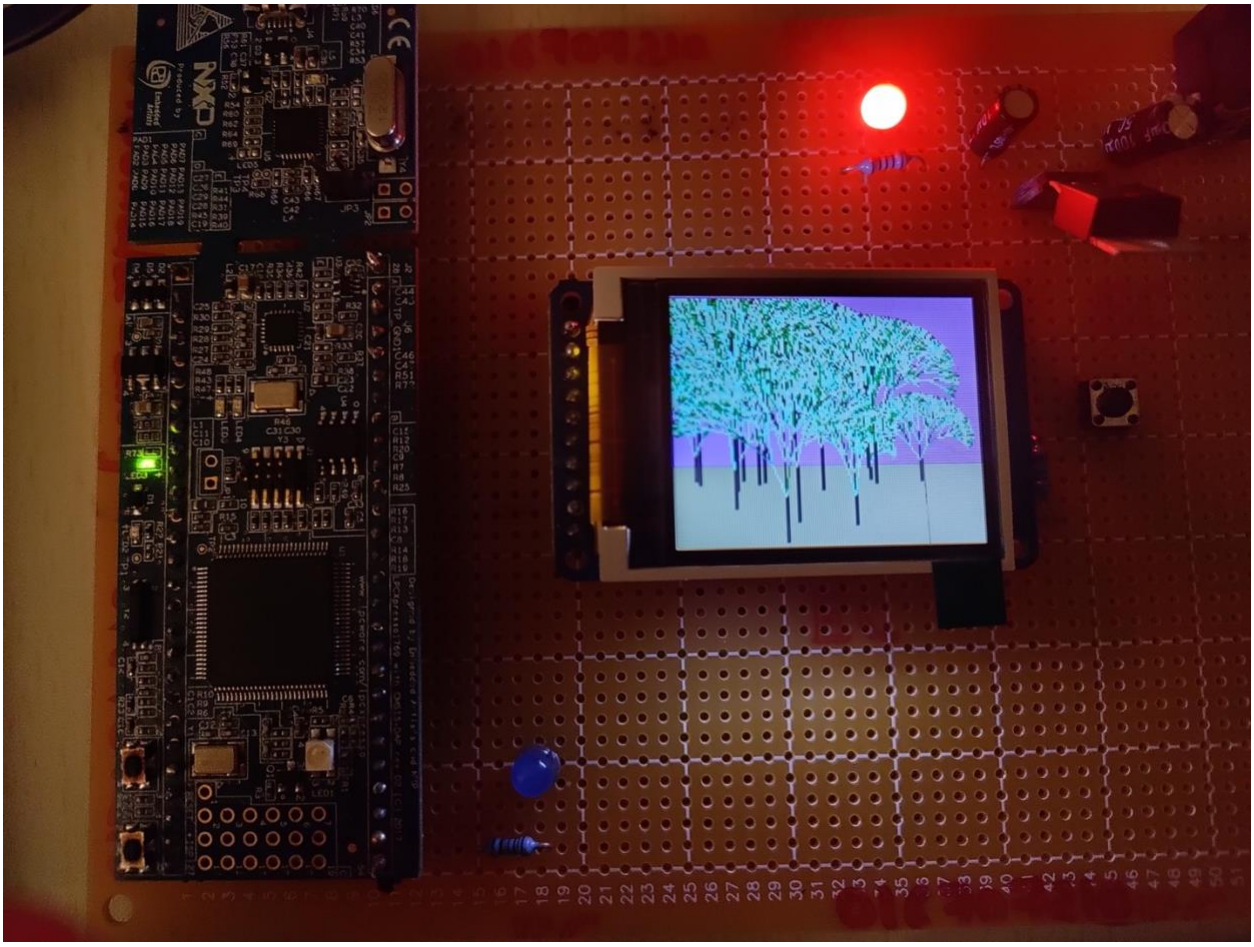
| LPC1769 PINS (JTAG PINS) | LCD TFT ST7735 |
|---|---|
| VCC (J2-28) | LITE |
| P0.17 (J2-12) | MISO |
| P0.15 (J2-13) | SCK |
| P0.18 (J2-11) | MOSI |
| P0.16 (J2-14) | TFT_CS |
| Not Connected | CARD_CS |
| P0.3 (J2-22) | D/C |
| P0.22 (J2-24) | RESET |
| VCC (J2-28) | VCC |
| GND (J2-1) | GND |

**Rubrics – Snippets:**

1. The snippet of the implemented screen saver I (Rotating Squares) is as follows:



2. The snippet of the implemented screen saver II (trees) is as follows:



**CONCLUSION:**

Upon completion of the experiment, we understand how to implement desired graphics using vector concepts and communicate it with LCD for displaying purpose. The experiment related source code has been included in the appendix.

# APPENDIX

## Source Code:

```
/*
===============================================================================
 Name        : 2DScreensavers.c
 Author      : Tirumala Saiteja Goruganthu
 Version     :
 Copyright   : $(copyright)
 Description : This assignment focuses on collecting data from a CPU Module to implement 2D visual designs on an LCD Display. The CPU Module utilized in this project is the LPC 1769, which is based on the ARM Cortex M0 Core. For the implementation of our task, we use the MCUXpresso IDE. The main goal of this assignment is to learn how to use the SPI Communication Protocol between an LCD (ST7735) and a CPU Module (LPC1769) to show several 2D screen savers.
===============================================================================
*/

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "LPC17xx.h"                        /* LPC17xx definitions */
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/* Be careful with the port number and location number, because

some of the location may not exist in that port. */

#define PORT_NUM            0


uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];
```

```
#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29



#define swap(x, y) {x = x + y; y = x - y; x = x - y ;}

/* defining color values */

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF

/* Self Defined Colors */
#define BROWN 0xA52A2A
#define YELLOW 0xFFFE00

/* Self Defined GREEN Shades */
#define GREEN1 0x38761D
#define GREEN2 0x002200
#define GREEN3 0x00FC7C
#define GREEN4 0x32CD32
#define GREEN5 0x228B22
#define GREEN6 0x006400
```

```
/* Self Defined RED Shades */
#define RED1 0xE61C1C
#define RED2 0xEE3B3B
#define RED3 0xEF4D4D
#define RED4 0xE88080

int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;

#define pi 3.1416

/* Define a structure for a Point object */
typedef struct Point
{
    float x;
    float y;
}Point;

/* Write to the target device using SSP protocol */
void spiwrite(uint8_t c)

{

 int pnum = 0;

 src_addr[0] = c;

 SSP_SSELToggle( pnum, 0 );

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

 SSP_SSELToggle( pnum, 1 );

}


/* Write a command to the LCD using SSP Protocol by clearing the D/C pin */
```

```
void writecommand(uint8_t c)

{

 LPC_GPIO0->FIOCLR |= (0x1<<3);

 spiwrite(c);

}


/* Write data to the LCD using SSP Protocol by asserting the D/C pin */
void writedata(uint8_t c)

{

 LPC_GPIO0->FIOSET |= (0x1<<3);

 spiwrite(c);

}


/* Write word to the LCD using SSP Protocol by asserting the D/C pin */
void writeword(uint16_t c)

{

 uint8_t d;

 d = c >> 8;

 writedata(d);

 d = c & 0xFF;

 writedata(d);
```

```
}

/* Initialize the color panels in LCD using SSP Protocol */
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);

 }

}

/* Initialize the Row and Column addresses of the LCD device using SSP protocol */
void setAddrWindow(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1)

{
```

```
 writecommand(ST7735_CASET);

 writeword(x0);

 writeword(x1);

 writecommand(ST7735_RASET);

 writeword(y0);

 writeword(y1);

}

/* Initialize all the pixels */
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);

}


/* Insert LCD Delay */
```

```
void lcddelay(int ms)

{

 int count = 24000;

 int i;

 for ( i = count*ms; i > 0; i--);

}


/* Initialize the LCD Device */
void lcd_init()

{

 int i;
 printf("Welcome to my CMPE240 Assignment - 2D Screensavers\n");
 // Set pins P0.16, P0.3, P0.22 as output
 LPC_GPIO0->FIODIR |= (0x1<<6);

 LPC_GPIO0->FIODIR |= (0x1<<3);

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

 // Hardware Reset Sequence
 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOCLR |= (0x1<<22);
 lcddelay(500);

 LPC_GPIO0->FIOSET |= (0x1<<22);
 lcddelay(500);
```

```
 // initialize buffers
 for ( i = 0; i < SSP_BUFSIZE; i++ )
 {

   src_addr[i] = 0;
   dest_addr[i] = 0;
 }

 // Take LCD display out of sleep mode
 writecommand(ST7735_SLPOUT);
 lcddelay(200);

 // Turn LCD display on
 writecommand(ST7735_DISPON);
 lcddelay(200);

}

/* Draw a Pixel */
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);

}

/**************************************************************************
```

```
** Descriptions:        Draw line function

**

** parameters:          Starting point (x0,y0), Ending point(x1,y1) and color

** Returned value:       None

**

*********************************************************************/

/* Draw a line */
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;

  }

 }

}

/* Design a square by calling and manipulating multiple drawLine methods */
void designSquare(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, int16_t x3, int16_t y3, uint32_t
color, uint8_t level, float lambda_value)
{
    int16_t new_x0, new_y0, new_x1, new_y1, new_x2, new_y2, new_x3, new_y3;

    if(level == 0)
        return;

    drawLine(x0, y0, x1, y1, color);
    drawLine(x1, y1, x2, y2, color);
    drawLine(x2, y2, x3, y3, color);
    drawLine(x3, y3, x0, y0, color);

    new_x0 = x0 + lambda_value * (x1 - x0);
    new_y0 = y0 + lambda_value* (y1 - y0);
    new_x1 = x1 + lambda_value * (x2 - x1);
    new_y1 = y1 + lambda_value * (y2 - y1);
    new_x2 = x2 + lambda_value * (x3 - x2);
    new_y2 = y2 + lambda_value * (y3 - y2);
    new_x3 = x3 + lambda_value * (x0 - x3);
    new_y3 = y3 + lambda_value * (y0 - y3);

    designSquare(new_x0, new_y0, new_x1, new_y1, new_x2, new_y2, new_x3, new_y3, color, level - 1,
lambda_value);
}
```

```
/* Rotate point p with respect to o and angle <angle> */
Point rotate_point(Point p, Point o, float angle)
{
    Point rt, t, new;

    float s = sin(angle);
    float c = cos(angle);

    //translate point to origin
    t.x = p.x - o.x;
    t.y = p.y - o.y;

    rt.x = t.x * c - t.y * s;
    rt.y = t.x * s + t.y * c;

    //translate point back
    new.x = rt.x + o.x;
    new.y = rt.y + o.y;

    return new;
}

/* Design a Tree using drawline method and rotatepoint method */
void designTree(Point start, Point end, int level ,float lambda_value)
{
    Point c, rtl, rtr;
    if(level == 0)
        return;
    int color[] = {GREEN1, GREEN2, GREEN3, GREEN4, GREEN5, GREEN6};

    int angles = 5;

    // alpha degrees =5,10,15,20,30
    float alpha[] = {pi/36, pi/18, pi/12, pi/9, pi/6};

    c.x = end.x + lambda_value * (end.x - start.x);
```

```
    c.y = end.y + lambda_value * (end.y - start.y);


    drawLine(c.x, c.y, end.x, end.y, BLACK);
    designTree(end, c, level - 1, lambda_value);

    rtl = rotate_point(c, end, alpha[rand()%angles]);
    drawLine(rtl.x, rtl.y, end.x, end.y, color[rand()%5]);
    designTree(end, rtl, level - 1, lambda_value);

    rtr = rotate_point(c, end, -alpha[rand()%angles]);
    drawLine(rtr.x, rtr.y, end.x, end.y, color[rand()%5]);
    designTree(end, rtr, level - 1, lambda_value);
}

/* Design the branch of a tree */
void designTreebranch(Point start, Point end, uint16_t color, uint8_t thickness)
{
    int i;
    for(i = 0; i < thickness; i++)
        drawLine(start.x, start.y + i, end.x, end.y + i, color);
}

/* Loop the designSquare method to form multiple rotating squares */
void designSquareLoop(int color[], float lambda_value)
{
    for(int i = 0; i < 20; i++)
        {
            int x0,y0, x1, y1, x2, y2, x3, y3, len, cindex;
            x0 = rand() % 120;
            y0 = rand() % 155;
            len = rand() % 100;
            cindex = rand() % 8;

            x1 = x0;
            y1 = y0 - len;
            x2 = x0 - len;
```

```
            y2 = y1;
            x3 = x2;
            y3 = y0;

            designSquare(x0, y0, x1, y1, x2, y2, x3, y3, color[cindex], 10, lambda_value);
            lcddelay(5);

        }

}

/* Loop the designTree method to form a forest */
void designTreeLoop(Point start, Point end, float lambda_value)
{
    //int len[] = {4, 6, 8, 10, 12};
    //int lenIdx;
    int len;
    for(int i = 0; i < 15; i++)
    {
        //lenIdx = rand() % 5;
        start.x = rand() % 40;
        start.y = rand() % 150;
        len = rand() % 30;
        if(len == 0)
            len = 20;
        //end.x = start.x + len[lenIdx];
        end.x = start.x + len;
        end.y = start.y;
        designTreebranch(start, end, BLACK, 2);
        designTree(start, end, 7, lambda_value);
    }
}

/* Main function to initialize the whole program */
int main (void)
{
    Point start, end;
```

```
char userInput;

uint32_t pnum = PORT_NUM, width = ST7735_TFTWIDTH / 5, len;
pnum = 0 ;

srand(time(NULL));
if ( pnum == 0 )
    SSP0Init();
else
    puts("Port number is not correct");

lcd_init();

// User input
float lambda_value;

// Set Background and colors
fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);
int color[] = {LIGHTBLUE, GREEN1, RED2, BLACK, BLUE, RED, RED4, PURPLE};

int choice;

do
{
    printf("\nEnter the 2D Screensaver of your choice:\n 1. Squares. \n 2. Trees.\n 3. Exit.  \n Your choice is: ");
    scanf("%d",&choice);

    fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);

   switch(choice) {

      case 1  :
         printf("\nGreat! You selected Squares as your Screensaver...");
         printf("\nNow, Do you want to enter a lambda value (Y/N): ");
         scanf(" %c",&userInput);
         if(userInput == 'Y')
         {
```

```
            printf("\nEnter the value of lambda (0 < lambda < 1): ");
            scanf(" %f",&lambda_value);
            if(0 < lambda_value && lambda_value < 1)
            printf("\nUsing the value of lambda = %f to generate Squares...", lambda_value);
            else
            {
                printf("\nInvalid Input Entered!! Using the Default value of lambda = 0.8 to generate Squares...");
                lambda_value = 0.8;
            }
        }
        else if(userInput == 'N')
        {
            printf("\nUsing the Default value of lambda = 0.8 to generate Squares...");
            lambda_value = 0.8;
        }
        else
        {
            printf("\nInvalid UserInput");
            break;
        }
        fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, BLACK);
        designSquareLoop(color, lambda_value);
       break;

    case 2 :
        lambda_value = 0.8;
        printf("\nGreat! You selected Trees as your Screensaver...");
        printf("\nUsing the Default value of lambda = 0.8 to generate Trees...");
        for(int i = 40; i < ST7735_TFTWIDTH; i++)
        {
            fillrect(i, 0, i+1, ST7735_TFTHEIGHT, 1638655-2*(i-40));
        }
        fillrect(0, 0, 40, ST7735_TFTHEIGHT, 0x4A290A);
        designTreeLoop(start, end, lambda_value);
       break;

    case 3  :
```

```
                break;

        default : printf("Please select a valid option\n");
      }

    }while(choice!= 3);

  lcddelay(1000);
  fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);

  return 0;
}
```