SHADING MODELS

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

***Abstract* – The LPC 176x/5x does not have a built-in graphics engine. This paper describes the design, implementation and testing for the graphics engine prototype for LPC 1769 using a 128x160 LCD TFT display. The goal is to display a 3D world coordinate system and then three 3D cubes with the forest pattern on one surface, Square screensaver pattern on the second surface and the initial of the user on the third surface, where each cube is at a different position, of different size and rotated at a different angle. The description given here mostly concentrates on implementation, design and testing of the hardware and software components required to interface LPC 1769 with the LCD through serial communication. 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

LPC 1769 used here is an ARM Cortex-M3 microcontroller, used often for applications that require high level of integration but low power dissipation.

2D or a 3D graphics rendering is primary importance in computer graphics. The 1.8” TFT color LCD display accomplishes this with a built in controller. The data transfer between LCD and LPC modules occur via a serial communication, established by 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

The main objective of this work is to display tree pattern and a 3D cube and understand the communication between the LPC module and LCD module. The implementation of SPI interface has the following objectives:

1. Understanding of LPC 1769, LCD TFT color display, 7805 IC.
2. Design and implementation of a power circuit for the LPC module.
3. Practical exposure with LPCXpresso IDE.
4. Inclusion of a graphical engine in the prototype board.
5. Learning 3D 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 trees
4. Finding a way to display the patterns on the cube surface continuously.

2.2 Problem formulation and design

The system layout had been 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.

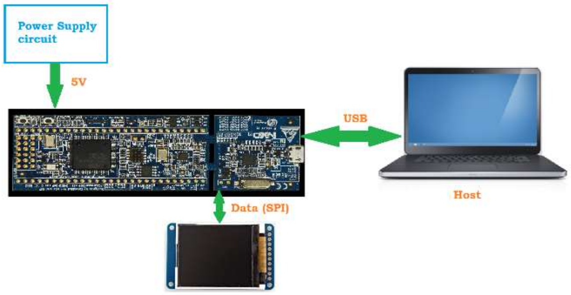


Fig.1. System layout

As for the software design, the goal is to draw three 3D cubes. In order to draw tree pattern and square pattern, we first start with a line drawing function. We then use the following equation to get the vertices of the second level of tree branches. The value of lambda is kept as 0.8.

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 tree. The color for each tree is fixed to green.

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

For the provision of power for the entire circuit, which includes LPC and LCD module, a power circuit is needed. We design and implement a power circuit in this paper. 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.

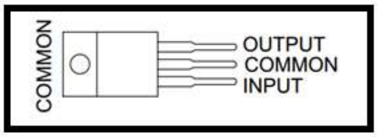


Fig.2. LM 7805 IC pin description

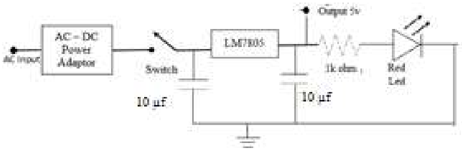


Fig.3. Power circuit diagram

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

This subsystem implements 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 the master and its inputs are in the form of MISO, whereas the LCD acts as the 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.

In order to connect the LPC and LCD, it is important to know the pin configuration of both the modules. The LPC 1769 pin layout is shown in figure 5. The pins that are required for the LCD interface to be accomplished are alone shown here. The LCD module pin layout is very much limited, which is shown in figure 6.

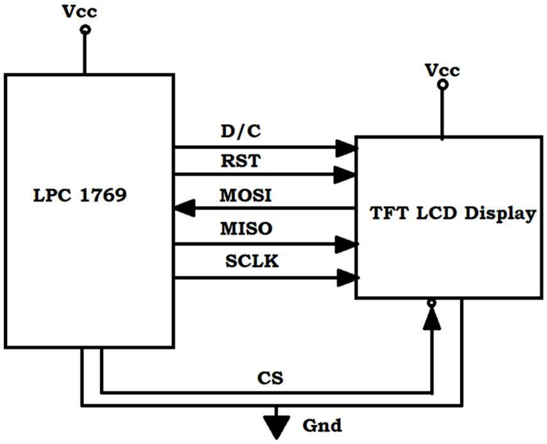


Fig.4. LPC to LCD signals

The corresponding pins to be connected between the LPC 1769 and the TFT LCD are shown in the form of a table (Table II). It is to be noted that only the pins that are required for the SPI interface are shown here. There are however additional pins for other functions in both LPC 1769 and the LCD, which we do not discuss in this paper, as it is out of scope of this paper.

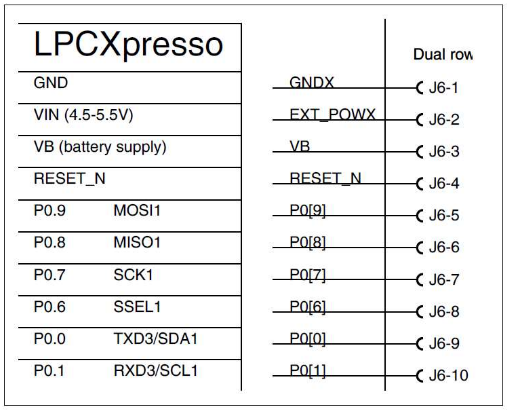


Fig.5. LPC pin layout

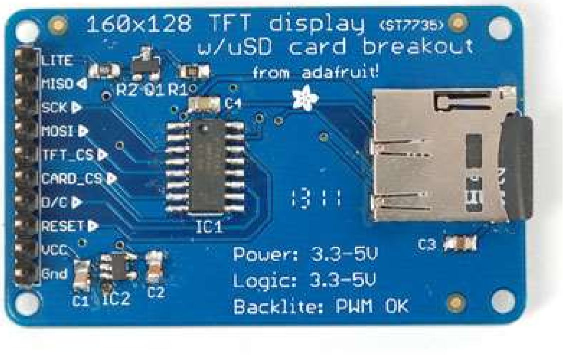


Fig.6. LCD TFT color display pin layout

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.

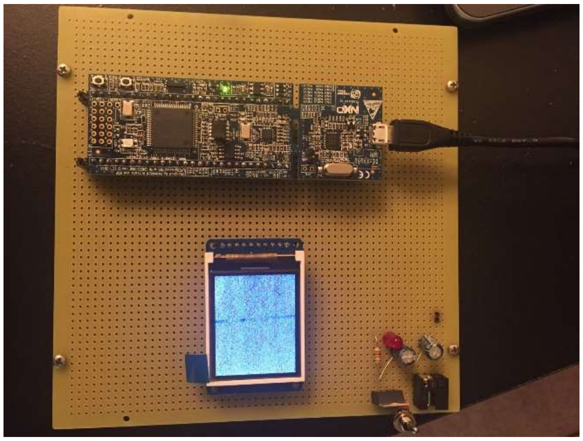


Fig.7. Hardware system setup

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

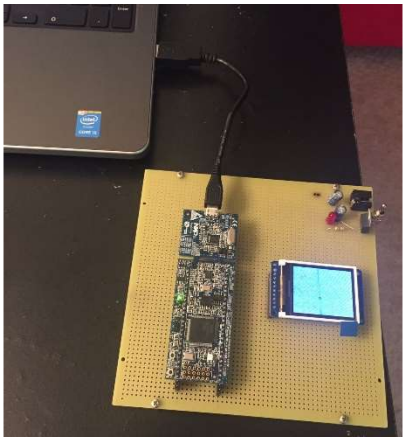


Fig.8. System layout 1

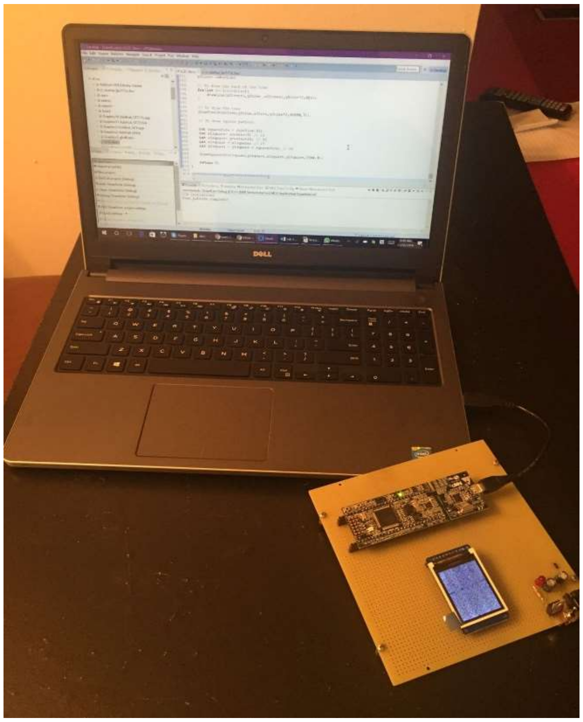


Fig.9. System layout 2

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

LPCXpresso is the IDE used to implement the software part of the project. 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 requires unique opcodes for specific operations, setting registers, initializations, etc. The datasheets of both the 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 following 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 the 3D world coordinates

- Fill the entire LCD display with black using fill rectangle method
- Draw the x,y,z axes in red, green and blue respectively
- Move to the next Step when user enters a new line character ('\n')

Step 5: Draw the 3D cube

- Draw the 3D cube by transposing the points in 3D
- Fill the 3 visible surfaces with different colors

Step 6: Decorate Surface S1 with square pattern screensaver

Step 7: Decorate Surface S2 with Trees

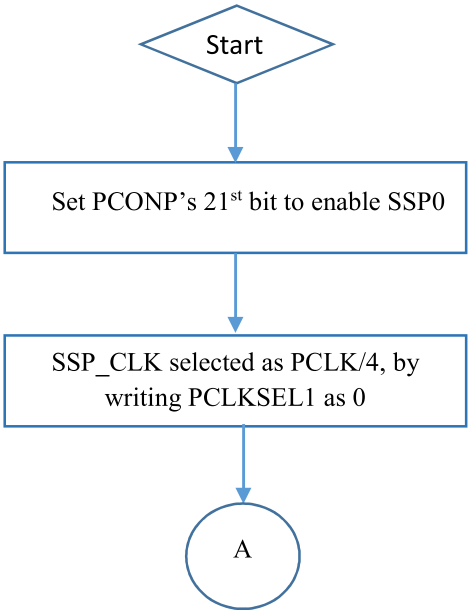
Step 8: Draw the initial font ('A' in this case) on surface S3

Step 9: Repeat the steps 5 to 8 twice

Step 10: 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
A

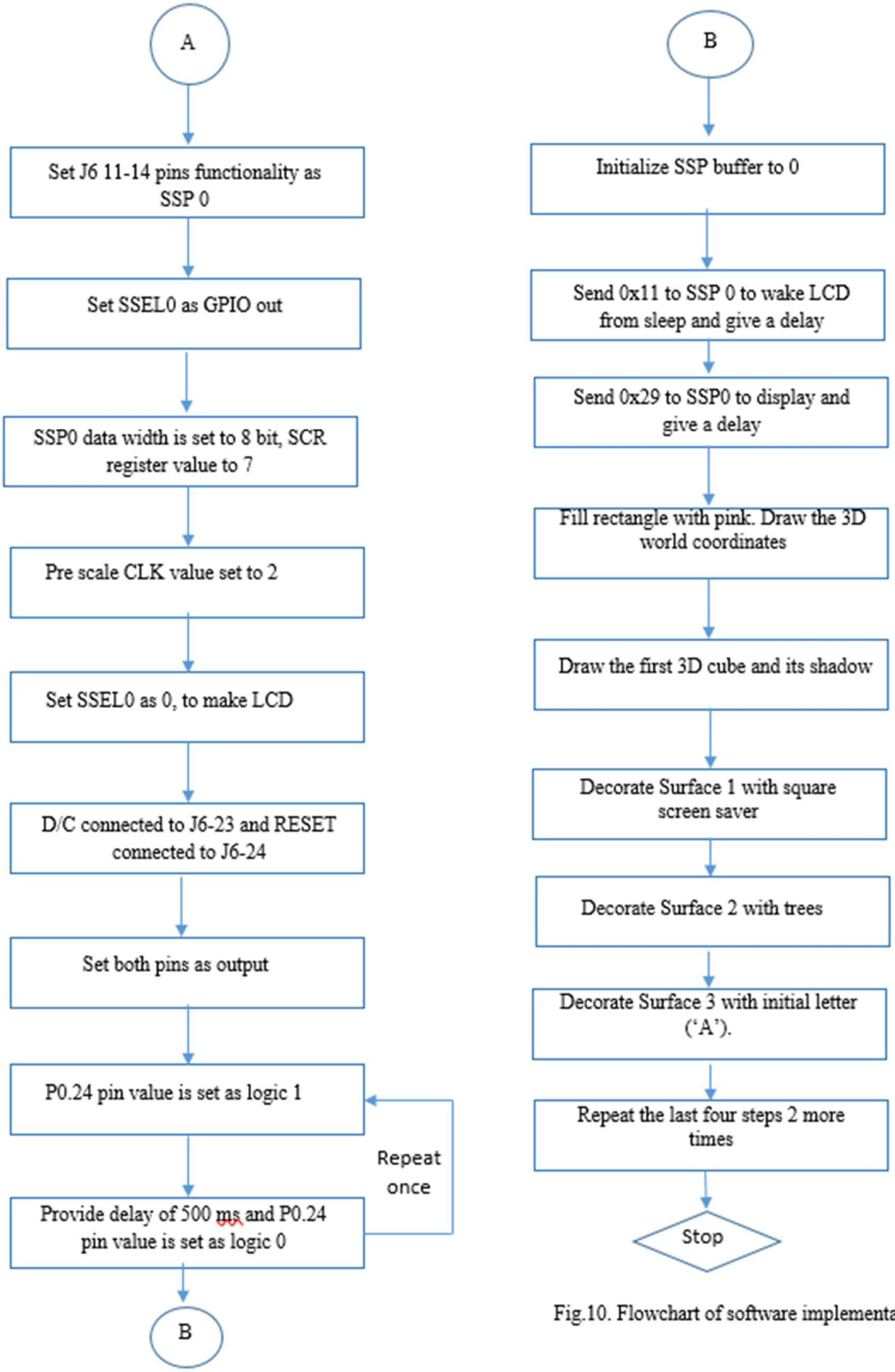

A
Set J6 11-14 pins functionality as SSP 0
Set SSEL0 as GPIO out
SSP0 data width is set to 8 bit, SCR register value to 7
Pre scale CLK value set to 2
Set SSEL0 as 0, to make LCD
D/C connected to J6-23 and RESET connected to J6-24
Set both pins as output
P0.24 pin value is set as logic 1
Provide delay of 500 ms and P0.24 pin value is set as logic 0
Repeat once
B
B
Initialize SSP buffer to 0
Send 0x11 to SSP 0 to wake LCD from sleep and give a delay
Send 0x29 to SSP0 to display and give a delay
Fill rectangle with pink. Draw the 3D world coordinates
Draw the first 3D cube and its shadow
Decorate Surface 1 with square screen saver
Decorate Surface 2 with trees
Decorate Surface 3 with initial letter ('A').
Repeat the last four steps 2 more times
Stop

Fig.10. Flowchart of software implementation

3.2.3 Pseudo code

A. Software implementation

```
int main (void)
{
 uint32_t i, portnum = PORT_NUM;
  uint32_t j =0;
  //uint32_t c;
  int16_t x;
  int16_t y;
  int16_t z;

  portnum = 1 ; /* For LCD use 1 */
   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();
  fillLcd(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, WHITE); // clear screen, fill
it with white

  //Display text info
    lcddelay(500);

//=========================== draw axis
========================
    background(0, 0,_width,_height,PINK);
    draw_XYZ_axis ();
    lcddelay(1000);
//========================3D
CUBE=============================

    int xs = 120, ys = 40, zs = 120;
    pointoflight(xs, ys, zs);
    lcddelay(100);
    int size1 = 50;
    int point = 0;
    cube1shadow(point, size1,xs,ys,zs);
    drawcube1(point,size1);
    lcddelay(500);

    int adj = 80;
    int size2= 30;
    cube2shadow(point,size2, adj,xs,ys,zs);
    drawcube2(point,size2, adj);
    lcddelay(500);
    int ay = 70;
    int ax = 60;
    int size3= 25;
    cube3shadow(point,size3,ax,ay,xs,ys,zs);
    drawcube3(point,size3, ax, ay);

    return 0;
}
```

B. Code execution on IDE

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 throws light on the testing and verification required to be done for this circuit.

- Once the lights on the LPC microcontroller light up, we can import the project into IDE including ssp.c, sp.h and SSP_Test.c files.
- Given the project has no errors, it will run successfully with a connection to CPU module.
- The data is transferred in the form of string to CPU through the USB cable. This will create the tree pattern and the 3D cube on the LCD screen.
- Thus communication between the LPC and LCD via the SPI interface has been tested and verified successfully.

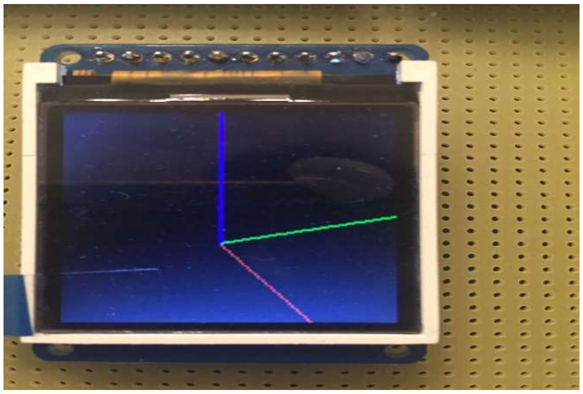


Fig.11. 3D world coordinate axes (x (red),y (green),z (blue))

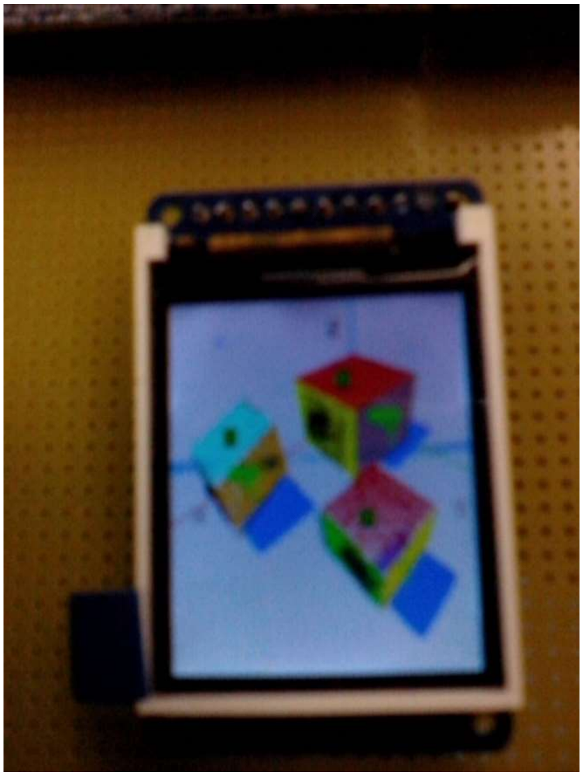


Fig.12. 3D world coordinate axes and three 3D cubes with surfaces S1, S2 and S3 decorated

5. Conclusion

The 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 3D cube were displayed with 3D world coordinate axes. 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, 3D 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        : SSP_Test_lcd.c
 Author      : Archana Ramalingam
 Version     :
 Copyright   : $(copyright)
 Description : This program allows us, using the 1.8"
Color TFT LCD via ssp ports,
                                    to be able to display 3D
world coordinates and three 3D cubes on the screen
==========================================
==========================================
=====
*/

#include <NXP/crp.h>
// Variable to store CRP value. Placed automatically
// by the linker when "Enable Code Read Protect"
selected.
// See crp.h header for more information
__CRP const unsigned int CRP_WORD =
CRP_NO_CRP ;

#include "LPC17xx.h"
#include "ssp.h"
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "font.h"

// Port number is 1 for the circuit used here

#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;

struct coordinates{
```

```
                int x;
                int y;
        };
struct coordinate{
                int x;
                int y;
                int z;
        };

int xcamera = 150,ycamera = 150,zcamera = 150;

/****************************************
*********************************
** 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; }

// 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;

uint16_t colortext = GREEN, colorbg= YELLOW;
float textsize = 2;
```

```
int wrap = 1;

// To draw Vertical line
void drawFastVLine(int16_t x, int16_t y,int16_t h,
uint32_t color) {
        drawline(x, y, x, y+h-1, color);
}

// To draw Horizontal line
void drawFastHLine(int16_t x, int16_t y, int16_t w,
uint16_t color) {
        drawline(x, y, x+w-1, y, color);
}

// To fill the background with given color and
dimensions
void background(int16_t x, int16_t y, int16_t w,
int16_t h,uint32_t color) {
        int16_t i;
  for (i=x; i<x+w; i++) {
        drawFastVLine(i, y, h, color);
 }
}

// To write 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 );
}

// To write commands into the SPI
void writecommand(uint8_t c) {
  LPC_GPIO0->FIOCLR |= (0x1<<21);
  spiwrite(c);
}

// To make LCD ready to write data
void writedata(uint8_t c) {

  LPC_GPIO0->FIOSET |= (0x1<<21);
  spiwrite(c);
}

// To write data to the LCD
void writeword(uint16_t c) {

  uint8_t d;
  d = c >> 8;
  writedata(d);
  d = c & 0xFF;
  writedata(d);
}
```

```
// To write color
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);
    writeword(x1);
    writecommand(ST7735_RASET);
    writeword(y0);
    writeword(y1);
}

// To 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);
}

// To provide delay to LCD
void lcddelay(int ms)
{
        int count = 24000;
        int i;
        for ( i = count*ms; i--; i > 0);
}

// Initialize the LCD
void lcd_init()
{

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


  /* Reset sequence */
  LPC_GPIO0->FIOSET |= (0x1<<22);

  lcddelay(500);                    /*delay 500 ms */
  LPC_GPIO0->FIOCLR |= (0x1<<22);
  lcddelay(500);                    /* delay 500 ms */
  LPC_GPIO0->FIOSET |= (0x1<<22);
  lcddelay(500);                    /* delay 500 ms */

        for ( i = 0; i < SSP_BUFSIZE; i++ )    /* Init
RD and WR buffer */
     {
       src_addr[i] = 0;
       dest_addr[i] = 0;
     }

   /* do we need Sw reset (cmd 0x01) ? */

   /* Sleep out */
   SSP_SSELToggle( portnum, 0 );
   src_addr[0] = 0x11;    /* Sleep out */
   SSPSend( portnum, (uint8_t *)src_addr, 1 );
   SSP_SSELToggle( portnum, 1 );

   lcddelay(200);
  /* delay 200 ms */
  /* Disp on */
   SSP_SSELToggle( portnum, 0 );
   src_addr[0] = 0x29;    /* Disp On */
   SSPSend( portnum, (uint8_t *)src_addr, 1 );
   SSP_SSELToggle( portnum, 1 );
  /* delay 200 ms */
   lcddelay(200);
}

// To fill a rectangle, with given parameters, with a
given color
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);
}

// To 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;
        }
        }
 }

// To write a character
void writeChar(int16_t x, int16_t y, unsigned char c,
uint16_t color, uint16_t bg, uint8_t size)
{
   int8_t i, j;
   if((x >= 128)           || // Clip right
     (y >= 280)          || // Clip bottom
     ((x + 6 * size - 1) < 0) || // Clip left
     ((y + 8 * size - 1) < 0))   // Clip top
      return;

   for (i=0; i<6; i++ ) {
     uint8_t line;
     if (i == 5)
       line = 0x0;
     else
       line = pgm_read_byte(font+(c*5)+i);
     for (j = 0; j<8; j++) {
       if (line & 0x1) {
         if (size == 1) // default size
           drawPixel(x+i, y+j, color);
         else {  // big size
           fillrect(x+(i*size), y+(j*size),
                size + x+(i*size), size + y+(j*size),
color);
         }
       } else if (bg != color) {
         if (size == 1) // default size
           drawPixel(x+i, y+j, bg);
         else {  // big size
           fillrect(x+i*size, y+j*size, (size +
x+i*size), (size + y+j*size), bg);
         }
       }
       line >>= 1;
     }
   }
}

// Helper function to write a character on display
void writeHelp(uint8_t c) {

   if (c == '\n') {
     cursor_y += textsize*8;
     cursor_x  = 0;
   } else if (c == '\r') {

   } else {
     writeChar(cursor_x, cursor_y, c,
colortext,colorbg, textsize);
     cursor_x += textsize*6;
     if (wrap && (cursor_x > (_width - textsize*6)))
{
       cursor_y += textsize*8;
       cursor_x = 0;
     }
   }
}

// To write text on the display
void writeText(char *str,int x,int y,float size, uint32_t
color)
```

```
  {
    char c;
    cursor_x=x;
    cursor_y=y;
    colortext = color;
    colorbg= 0;
    textsize=size;
    while(*str != NULL)
    {
                    c = *str++;
                    writeHelp(c);
    }
}

// To generate square screensaver
void squarePattern(int x1, int y1, int x2,int y2,int
x3,int y3,int x4,int y4)
{
        int x1_1=0, x2_1=0, x3_1=0, x4_1=0,
y1_1=0,y2_1=0, y3_1=0, y4_1=0;
        float lambda=0.8;

        // Draw 10 levels of squares in each pattern
        for(int i=1;i<=10;i++){
                lcddelay(100);
                // Use the equation given in class to
calculate the 4 vertices' coordinates of the recursive
squares
                x1_1 = (x2+(lambda*(x1-x2)));
                y1_1 = (y2+(lambda*(y1-y2)));
                x2_1 = (x3+(lambda*(x2-x3)));
                y2_1 = (y3+(lambda*(y2-y3)));
                x3_1 = (x4+(lambda*(x3-x4)));
                y3_1 = (y4+(lambda*(y3-y4)));
                x4_1 = (x1+(lambda*(x4-x1)));
                y4_1 = (y1+(lambda*(y4-y1)));

                // Draw a square (4 drawline() for 4
sides)

        drawline(x1_1,y1_1,x2_1,y2_1,ORANGE);

        drawline(x2_1,y2_1,x3_1,y3_1,ORANGE);

        drawline(x4_1,y4_1,x3_1,y3_1,ORANGE);

        drawline(x1_1,y1_1,x4_1,y4_1,ORANGE);

                // Initiate the original vertices'
values with the new calculated vertices' values
                x1 = x1_1;
                x2 = x2_1;
                x3 = x3_1;
                x4 = x4_1;

                y1 = y1_1;
                y2 = y2_1;
                y3 = y3_1;
                y4 = y4_1;
        }
}

// To convert world to viewer coordinates
struct coordinates Transformation_pipeline (int xw,
int yw, int zw)
{
        int xdoubleprime, ydoubleprime, D=100,
length1=80, length2=50;
        double xPrime, yPrime, zPrime, theta, phi,
rho;
        struct coordinates projection;

        theta =
acos(xcamera/sqrt(pow(xcamera,2)+pow(ycamera,2))
);
        phi =
acos(zcamera/sqrt(pow(xcamera,2)+pow(ycamera,2)
+pow(zcamera,2)));
        rho=
sqrt((pow(xcamera,2))+(pow(ycamera,2))+(pow(zca
mera,2)));

        xPrime = (yw*cos(theta))-(xw*sin(theta));
        yPrime = (zw*sin(phi))-
(xw*cos(theta)*cos(phi))-(yw*cos(phi)*sin(theta));
        zPrime = rho-(yw*sin(phi)*cos(theta))-
(xw*sin(phi)*cos(theta))-(zw*cos(phi));

        xdoubleprime = xPrime*D/zPrime;
        ydoubleprime = yPrime*D/zPrime;
        xdoubleprime = length1+xdoubleprime;
        ydoubleprime = length2-ydoubleprime;

        projection.x = xdoubleprime;
        projection.y = ydoubleprime;
        return projection;
}

// To draw the world 3D coordinate system - x,y,z
void drawCoordinate()
{
        struct coordinates axis;
        int x1,y1,x2,y2,x3,y3,x4,y4;

                axis = Transformation_pipeline
(0,0,0);
                x1=axis.x;
                y1=axis.y;

                axis = Transformation_pipeline
(180,0,0);
                x2=axis.x;
                y2=axis.y;

                axis = Transformation_pipeline
(0,180,0);
                x3=axis.x;
                y3=axis.y;

                axis = Transformation_pipeline
(0,0,180);
                x4=axis.x;
                y4=axis.y;

                drawline(x1,y1,x2,y2,RED);
                  //x axis  Red
                drawline(x1,y1,x3,y3,GREEN);
            //y axis  Green
                drawline(x1,y1,x4,y4,BLUE);
             //z axis  Blue
}

// To draw the tree
void treePattern(int x, int y, float angle, int length, int
level, int color){

        int x1,y1,len;
        float ang;

        if(level>0){
                // To calculate the x,y vertices for
the branch after rotation
                x1 = x+length*cos(angle);
            y1 = y+length*sin(angle);

            // To draw the tree branch
            drawline(x,y,x1,y1,color);

            // Add 30 degree to angle to rotate it to
right
            ang = angle + 0.52;
            // To calculate 80% of the line length
            len = 0.8 * length;

            // Call drawTree2d function recursively to
draw tree pattern
            treePattern(x1,y1,ang,len,level-1,color);

            // Subtract 30 degree from the angle to
rotate it to right
            ang = angle - 0.52;
            len = 0.8 * length;

            treePattern(x1,y1,ang,len,level-1,color);

            // Draw the next level
            ang = angle;
            len = 0.8 * length;
```

```
            treePattern(x1,y1,ang,len,level-1,color);
        }
}

// To fill the shadow
void fillTriangle(int16_t x0, int16_t y0,int16_t x1,
int16_t y1,int16_t x2, int16_t y2, uint16_t color) {

        int16_t a, b, y, last;

        // Sort the coordinates by the Y order (y2 >=
y1 >= y0)
        if (y0 > y1) {
        swap(y0, y1); swap(x0, x1);
        }
        if (y1 > y2) {
        swap(y2, y1); swap(x2, x1);
        }
        if (y0 > y1) {
        swap(y0, y1); swap(x0, x1);
        }
        if(y0 == y2) {
        a = b = x0;
        if(x1 < a) a = x1;
        else if(x1 > b) b = x1;
        if(x2 < a) a = x2;
        else if(x2 > b) b = x2;
        drawFastHLine(a, y0, b-a+1, color);
        return;
        }

        int16_t dx01 = x1 - x0, dy01 = y1 - y0, dx02
= x2 - x0, dy02 = y2 - y0, dx12 = x2 - x1, dy12 = y2 -
y1;
        int32_t sa = 0, sb = 0;

        // For upper part of triangle:
        // find scanline crossings for segments 0-1
and 0-2.
        // If y1=y2 (flat-bottomed triangle), the
scanline y1
        // is included here (and second loop will be
skipped, avoiding a /0
        // error there), otherwise scanline y1 is
skipped here and handled
        // in the second loop, which also avoids a /0
error here if y0=y1 (flat-topped triangle).

        if(y1 == y2) last = y1; // y1 scanline
        else last = y1-1;
        for(y=y0; y<=last; y++) {
        a = x0 + sa / dy01;
        b = x0 + sb / dy02;
        sa += dx01;
        sb += dx02;
```

```
    if(a > b) swap(a,b);
    drawFastHLine(a, y, b-a+1, color);
    }

    // For lower part of triangle:
    // find scanline crossings for segments 0-2 and 1-2. This loop is skipped if y1=y2.

    sa = dx12 * (y - y1);
    sb = dx02 * (y - y0);
    for(; y<=y2; y++) {
    a = x1 + sa / dy12;
    b = x0 + sb / dy02;
    sa += dx12;
    sb += dx02;

    if(a > b) swap(a,b);
    drawFastHLine(a, y, b-a+1, color);
    }
}

// To draw a 3D cube
void draw3dcube1(int point, int cube_size)
{
    struct coordinates cube;
    int x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,i;
    xcamera = 110;
    ycamera = 110;
    zcamera = 110;
    cube = Transformation_pipeline(point,point,(cube_size+point));
    x1=cube.x;
    y1=cube.y;
    cube = Transformation_pipeline((cube_size+point),point,(cube_size+point));
    x2=cube.x;
    y2=cube.y;
    cube = Transformation_pipeline((cube_size+point),(cube_size+point),(cube_size+point));
    x3=cube.x;
    y3=cube.y;
    cube = Transformation_pipeline(point,(cube_size+point),(cube_size+point));
    x4=cube.x;
    y4=cube.y;
    cube = Transformation_pipeline((cube_size+point),point,point);
    x5=cube.x;
    y5=cube.y;
    cube = Transformation_pipeline ((cube_size+point),(cube_size+point),point);
    x6=cube.x;
    y6=cube.y;
    cube = Transformation_pipeline (point,(cube_size+point),point);
    x7=cube.x;
    y7=cube.y;
    drawline(x1, y1, x2, y2,BLACK);
    lcddelay(500);
    drawline(x2, y2, x3, y3,BLACK);
    lcddelay(500);
    drawline(x3, y3, x4, y4,BLACK);
    lcddelay(500);
    drawline(x4, y4, x1, y1,BLACK);
    lcddelay(500);
    drawline(x2, y2, x5, y5,BLACK);
    lcddelay(500);
    drawline(x5, y5, x6, y6,BLACK);
    lcddelay(500);
    drawline(x6, y6, x3, y3,BLACK);
    lcddelay(500);
    drawline(x6, y6, x7, y7,BLACK);
    lcddelay(500);
    drawline(x7, y7, x4, y4,BLACK);

    // decorate cube
    decorate3dcube1(point, cube_size);
    writeText("A",x1-7,y1+7,1.0, YELLOW); // Writes initials
    squarePattern(x2+2, y2+5, x3-5, y3-5, x6-2, y6-2, x5+2, y5+5); // Draws square pattern on
    // To draw tree pattern
    drawline(x3+15,y3+10,x6+15,y6-15,GREEN); // draws tree trunk
    treePattern(x3+15,y3+10,5.23,4,4,GREEN); // draws with right branch (angle = 5.23 rad/300 deg)
    treePattern(x3+15,y3+10,4.18,4,4,GREEN); // draws with left branch (angle = 4.18 rad/240 deg)
    treePattern(x3+15,y3+10,4.71,4,4,GREEN); // draws with center branch (angle = 4.71 rad/0 deg)
}

// To decorate a 3D cube
void decorate3dcube1(int point,int cube_size)
{   struct coordinates filler;
    int i,j,a[cube_size][cube_size];
    cube_size=cube_size+point;
    for(i=0;i<cube_size;i++)
    {
        for(j=0;j<cube_size;j++)
        {

            filler=Transformation_pipeline(j,i,cube_size);    //S2
```

```
		drawPixel(filler.x,filler.y,RED);

		filler=Transformation_pipeline(i,cube_size,j
);		//S1

		drawPixel(filler.x,filler.y,BLUE);

		filler=Transformation_pipeline(cube_size,j,i
);		//S3

		drawPixel(filler.x,filler.y,YELLOW);
			}
		}
}

// To draw shadow for cube 1
void cube1shadow(double point, double cube_size,
double xs, double ys, double zs)
{
		int xp[8]={0}, yp[8]={0},
zp[8]={0},k=0,j=0,i=0;
		struct coordinates s1,s2,s3,s4;
		struct coordinates filler;
		double x[8] =
{point,(point+cube_size),(point+cube_size),point,poi
nt,(point+cube_size),(point+cube_size),point};
		double y[8] = {point, point,
point+cube_size, point+cube_size, point, point,
(point+cube_size), (point+cube_size) };
		double z[8] = {point, point, point, point,
(point+cube_size), (point+cube_size),
(point+cube_size), (point+cube_size)};

		for(i=0; i<8; i++){
		xp[i]=x[i]-((z[i]/(zs-z[i]))*(xs-x[i]));
		yp[i]=y[i]-((z[i]/(zs-z[i]))*(ys-y[i]));
		zp[i]=z[i]-((z[i]/(zs-z[i]))*(zs-z[i]));
		}
		s1 = Transformation_pipeline
(xp[4],yp[4],zp[4]- cube_size);
		s2 = Transformation_pipeline
(xp[5],yp[5],zp[5] - cube_size);
		s3 = Transformation_pipeline
(xp[6],yp[6],zp[6] - cube_size);
		s4 = Transformation_pipeline
(xp[7],yp[7],zp[7] - cube_size);

		drawline(s1.x,s1.y,s2.x,s2.y,BLACK);
		drawline(s2.x,s2.y,s3.x,s3.y,BLACK);
		drawline(s3.x,s3.y,s4.x,s4.y,BLACK);
		drawline(s4.x,s4.y,s1.x,s1.y,BLACK);
		fillTriangle( s1.x,s1.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
		fillTriangle( s3.x,s3.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
```

```
}

// To draw cube 2
void draw3dcube2(int point, int cube_size,int zn)
{
		struct coordinates cube;
		int
x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,i;
		xcamera = 110;
		ycamera = 110;
		zcamera = 110;
		cube =
Transformation_pipeline((point+zn),point,(cube_size
+point+20));
		x1=cube.x;
		y1=cube.y;
		cube =
Transformation_pipeline((cube_size+point+zn),point,
(cube_size+point));
		x2=cube.x;
		y2=cube.y;
		cube =
Transformation_pipeline((cube_size+point+zn),(cube
_size+point),(cube_size+point));
		x3=cube.x;
		y3=cube.y;
		cube =
Transformation_pipeline((point+zn),(cube_size+poin
t),(cube_size+point+20));
		x4=cube.x;
		y4=cube.y;
		cube =
Transformation_pipeline((cube_size+point+zn),point,
point);
		x5=cube.x;
		y5=cube.y;
		cube = Transformation_pipeline
((cube_size+point+zn),(cube_size+point),point);
		x6=cube.x;
		y6=cube.y;
		cube = Transformation_pipeline
((point+zn),(cube_size+point),point+20);
		x7=cube.x;
		y7=cube.y;
		cube = Transformation_pipeline
((point+zn),point,point+20);
		x8=cube.x;
		y8=cube.y;
		drawline(x1, y1, x2, y2,BLACK);
		lcddelay(500);
		drawline(x2, y2, x3, y3,BLACK);
		lcddelay(500);
		drawline(x3, y3, x4, y4,BLACK);
		lcddelay(500);
		drawline(x4, y4, x1, y1,BLACK);
		lcddelay(500);
```

```
        drawline(x2, y2, x5, y5,BLACK);
        lcddelay(500);
        drawline(x5, y5, x6, y6,BLACK);
        lcddelay(500);
        drawline(x6, y6, x3, y3,BLACK);
        lcddelay(500);
        drawline(x6, y6, x7, y7,BLACK);
        lcddelay(500);
        drawline(x7, y7, x4, y4,BLACK);
        lcddelay(500);
        drawline(x8, y8, x1, y1,BLACK);
        lcddelay(500);
        drawline(x8, y8, x7, y7,BLACK);
        lcddelay(500);
        drawline(x8, y8, x5, y5,BLACK);

        decorate3dcube2(point, cube_size, zn);
        writeText("A",x1-9,y1+13,1.0, YELLOW);
// Writes initials
        squarePattern(x2+1, y2+2, x3-2, y3-2, x6-2,
y6-2, x5+2, y5+2); // draws square pattern
        drawline(x3+15,y3-2,x6+10,y6-
15,GREEN); // draws tree trunk
        treePattern(x3+15,y3-2,5.23,4,4,GREEN); //
draws the right branch (angle = 5.23 rad/300 deg)
        treePattern(x3+15,y3-2,4.18,4,4,GREEN); //
draws the left branch (angle = 4.18 rad/240 deg)
        treePattern(x3+15,y3-2,4.71,4,4,GREEN); //
draws the center branch (angle = 4.71 rad/0 deg)
}

// To decorate cube 2
void decorate3dcube2(int point,int cube_size, int zn)
{
        struct coordinates filler;
        int i,j,a[cube_size][cube_size];
        cube_size=cube_size+point;
        for(i=0;i<cube_size;i++)
        {
                for(j=0;j<cube_size;j++)
                {

        filler=Transformation_pipeline(zn+j,i,(cube
_size+20)-(j/2+0.4));        //S2

        drawPixel(filler.x,filler.y,CYAN);

        filler=Transformation_pipeline(zn+i,cube_si
ze,(20+j)-(i/2+0.8));        //S1

        drawPixel(filler.x,filler.y,PURPLE);

        filler=Transformation_pipeline(cube_size+z
n,j,i);     //S3

        drawPixel(filler.x,filler.y,BROWN);
```

```
                }
        }
}

// To draw shadow for cube 2
void cube2shadow(double point, double cube_size,int
zn, double xs, double ys, double zs)
{
        int xp[8]={0}, yp[8]={0},
zp[8]={0},k=0,j=0,i=0;
        struct coordinates s1,s2,s3,s4;
        double x[8] =
{(point+zn),(point+cube_size+zn),(point+cube_size+
zn),(point+zn),(point+zn),(point+cube_size+zn),(poin
t+cube_size+zn),(point+zn)};
        double y[8] = {point, point,
point+cube_size, point+cube_size, point, point,
(point+cube_size), (point+cube_size) };
        double z[8] = {point, point, point, point,
(point+cube_size), (point+cube_size),
(point+cube_size), (point+cube_size)};

        for(i=0; i<8; i++){
        xp[i]=x[i]-((z[i]/(zs-z[i]))*(xs-x[i]));
        yp[i]=y[i]-((z[i]/(zs-z[i]))*(ys-y[i]));
        zp[i]=z[i]-((z[i]/(zs-z[i]))*(zs-z[i]));
        }
        s1 = Transformation_pipeline
(xp[4],yp[4],zp[4] - cube_size);
        s2 = Transformation_pipeline
(xp[5],yp[5],zp[5] - cube_size);
        s3 = Transformation_pipeline
(xp[6],yp[6],zp[6] - cube_size);
        s4 = Transformation_pipeline
(xp[7],yp[7],zp[7] - cube_size);

        drawline(s1.x,s1.y,s2.x,s2.y,BLACK);
        drawline(s2.x,s2.y,s3.x,s3.y,BLACK);
        drawline(s3.x,s3.y,s4.x,s4.y,BLACK);
        drawline(s4.x,s4.y,s1.x,s1.y,BLACK);
        fillTriangle( s1.x,s1.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
        fillTriangle( s3.x,s3.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
}

// To draw cube number 3
void draw3dcube3(int point, int cube_size, int zx, int
zy)
{
        struct coordinates cube;
        int
x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,i;
        xcamera = 110;
        ycamera = 110;
        zcamera = 110;
```

```
	cube = Transformation_pipeline((point+zx),(point+zy),(cube_size+point));
	x1=cube.x;
	y1=cube.y;
	cube = Transformation_pipeline((cube_size+point+zx),(point+zy),(cube_size+point));
	x2=cube.x;
	y2=cube.y;
	cube = Transformation_pipeline((cube_size+point+zx),(cube_size+point+zy),(cube_size+point));
	x3=cube.x;
	y3=cube.y;
	cube = Transformation_pipeline((point+zx),(cube_size+point+zy),(cube_size+point));
	x4=cube.x;
	y4=cube.y;
	cube = Transformation_pipeline((cube_size+point+zx),(point+zy),point);
	x5=cube.x;
	y5=cube.y;
	cube = Transformation_pipeline ((cube_size+point+zx),(cube_size+point+zy),point);
	x6=cube.x;
	y6=cube.y;
	cube = Transformation_pipeline ((point+zx),(cube_size+point+zy),point);
	x7=cube.x;
	y7=cube.y;
	cube = Transformation_pipeline ((point+zx),(point+zy),point);
	x8=cube.x;
	y8=cube.y;
	drawline(x1-5, y1-5, x2-5, y2-5,BLACK);
	lcddelay(500);
	drawline(x2-5, y2-5, x3, y3,BLACK);
	lcddelay(500);
	drawline(x3, y3, x4-5, y4-5,BLACK);
	lcddelay(500);
	drawline(x4-5, y4-5, x1-5, y1-5,BLACK);
	lcddelay(500);
	drawline(x2-5, y2-5, x5-5, y5-5,BLACK);
	lcddelay(500);
	drawline(x5-5, y5-5, x6, y6,BLACK);
	lcddelay(500);
	drawline(x6, y6, x3, y3,BLACK);
	lcddelay(500);
	drawline(x6, y6, x7-5, y7-5,BLACK);
	lcddelay(500);
	drawline(x7-5, y7-5, x4-5, y4-5,BLACK);
	lcddelay(500);
	drawline(x8-5, y8-5, x1-5, y1-5,BLACK);
	lcddelay(500);
	drawline(x8-5, y8-5, x7-5, y7-5,BLACK);
	lcddelay(500);
	drawline(x8-5, y8-5, x5-5, y5-5,BLACK);

	decorate3dcube3(point, cube_size, zx-5, zy-5);
	writeText("A",x1-5,y1+10,1.0, YELLOW); // writes initials
	squarePattern(x2+4, y2+2, x3-1, y3-1, x6-1, y6-1, x5+1, y5+1); // draws square pattern
	drawline(x3+10,y3-1,x6+10,y6-20,GREEN); // draws tree trunk
	treePattern(x6+10,y6-20,5.23,1,1,GREEN); // draws the right branch (angle = 5.23 rad/300 deg)
	treePattern(x6+10,y6-20,4.18,1,1,GREEN); // draws the left branch (angle = 4.18 rad/240 deg)
	treePattern(x6+10,y6-20,4.71,1,1,GREEN); // draws the center branch (angle = 4.71 rad/0 deg)
}

// To decorate cube 3
void decorate3dcube3(int point,int cube_size, int zx, int zy)
{
	struct coordinates filler;
	int i,j,a[cube_size][cube_size];
	cube_size=cube_size+point;
	for(i=0;i<cube_size+5;i++)
	{
		for(j=0;j<cube_size+5;j++)
		{

	filler=Transformation_pipeline(zx+j,zy+i,cube_size+5);	//S2

	drawPixel(filler.x,filler.y,BROWN);

	filler=Transformation_pipeline(zx+i,zy+cube_size+5,j);	//S1

	drawPixel(filler.x,filler.y,YELLOW);

	filler=Transformation_pipeline(cube_size+zx+5,zy+j,i);	//S3

	drawPixel(filler.x,filler.y,GREEN);
		}
	}
}

void pointoflight(int xs, int ys, int zs){
				struct coordinates filler;

	filler=Transformation_pipeline(xs,ys,zs);
	//S2
```

```
            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys,zs+1);
            //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys+1,zs);
            //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys+1,zs+
1);         //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs+1,ys,zs);
            //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs+1,ys,zs+
1);         //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs+1,ys+1,z
s);         //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs+1,ys+1,z
s+1);       //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys,zs+2);
            //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys+2,zs);
            //S2

            drawPixel(filler.x,filler.y,BLUE);

            filler=Transformation_pipeline(xs,ys+2,zs+
2);         //S2

            drawPixel(filler.x,filler.y,BLUE);
}

// To draw shadow for cube 3
```

```
void cube3shadow(double point, double cube_size,int
zx, int zy, double xs, double ys, double zs)
{
            int xp[8]={0}, yp[8]={0},
zp[8]={0},k=0,j=0,i=0;
            struct coordinates s1,s2,s3,s4;
            double x[8] =
{(point+zx),(point+cube_size+zx),(point+cube_size+
zx),(point+zx),(point+zx),(point+cube_size+zx),(poin
t+cube_size+zx),(point+zx)};
            double y[8] = {(point+zy), (point+zy),
(point+cube_size+zy), (point+cube_size+zy),
point+zy, point+zy, (point+cube_size+zy),
(point+cube_size+zy) };
            double z[8] = {point, point, point, point,
(point+cube_size), (point+cube_size),
(point+cube_size), (point+cube_size)};

            for(i=0; i<8; i++){
            xp[i]=x[i]-((z[i]/(zs-z[i]))*(xs-x[i]));
            yp[i]=y[i]-((z[i]/(zs-z[i]))*(ys-y[i]));
            zp[i]=z[i]-((z[i]/(zs-z[i]))*(zs-z[i]));
            }
            s1 = Transformation_pipeline
(xp[4],yp[4],zp[4]- cube_size);
            s2 = Transformation_pipeline
(xp[5],yp[5],zp[5] - cube_size);
            s3 = Transformation_pipeline
(xp[6],yp[6],zp[6] - cube_size);
            s4 = Transformation_pipeline
(xp[7],yp[7],zp[7] - cube_size);

            drawline(s1.x,s1.y,s2.x,s2.y,BLACK);
            drawline(s2.x,s2.y,s3.x,s3.y,BLACK);
            drawline(s3.x,s3.y,s4.x,s4.y,BLACK);
            drawline(s4.x,s4.y,s1.x,s1.y,BLACK);
            fillTriangle( s1.x,s1.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
            fillTriangle( s3.x,s3.y,s2.x,s2.y,s4.x,s4.y,
BLACK);
}

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

   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;
 }

        // To initialize the LCD
        lcd_init();

        // To draw the 3D world coordinate system
  background(0, 0,_width,_height,PINK);
  drawCoordinate();
  lcddelay(1000);

  // To draw the three 3D cubes
  int xs = 120, ys = 40, zs = 120, point=0;
  pointoflight(xs, ys, zs);
  lcddelay(100);

  int cube1size = 40;
  cube1shadow(point,cube1size,xs,ys,zs);
  draw3dcube1(point,cube1size);
  lcddelay(500);

  int zn= 80;
  int cube2size= 30;
  cube2shadow(point,cube2size,zn,xs,ys,zs);
  draw3dcube2(point,cube2size,zn);
  lcddelay(500);

  int zy = 70;
  int zx = 60;
  int cube3size= 25;
  cube3shadow(point,cube3size,zx,zy,xs,ys,zs);
  draw3dcube3(point,cube3size,zx,zy);

  return 0;
}
```