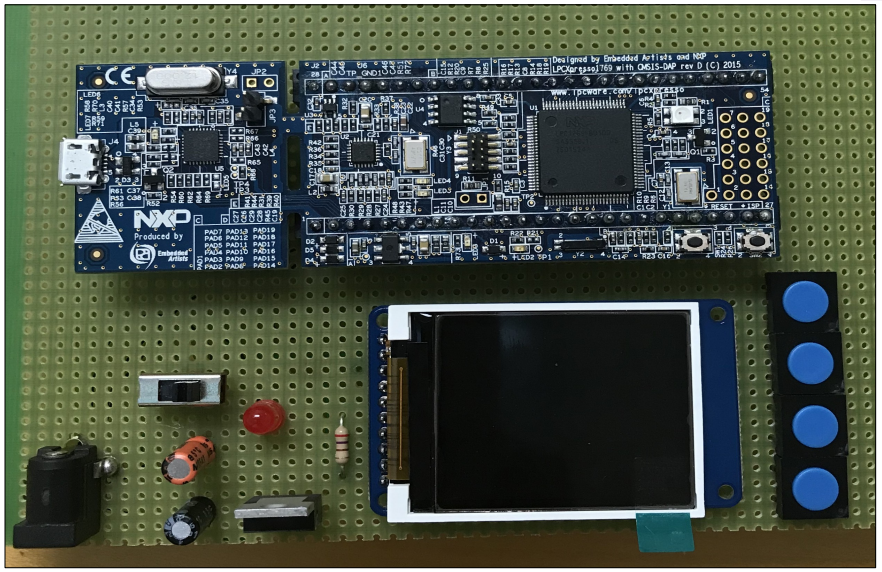
LPC1769 DISPLAY UNIT

- LPC 1769
- Color TFT LCD display

Resolution : 128x160,
Pixel Depth: 18-bit (262,144) colors
Controller: ST7735
Interface: SPI interface



LCD Pins	LPC1769 Pins
Gnd	Gnd
VCC	VCC (3.3V)
RST	GPIO output (P0.22)
RS/DC	GPIO output (P0.21)
CARD_CS	X
TFT_CS	SSEL0 (P0.16)
MOSI	SSP0 MOSI (P0.18)
SCK	SCK0 (P0.15)
MISO	SSP0 MISO (P0.17)
LITE	VCC (3.3V)

Note from Harry Li: if for CMPE127 new SPI LCD module, use this pin connection

Note from Harry Li: for CMPE127 class, ignore 4 button switch, just connect SPI LCD.

LPC-1769		LCD
Label	Pin	
MOSI	P 0.18	SDA
MISO	P0.17	MISO
SCK	P0.15	SCL
CS	P0.16	TFT_CS
GPIO-DC (Data / command)	P0.21	AO
GPIO-Reset	P0.22	RST
3.3V		LED+
GND		LED-

Reference: CTI One Corporation

SSP0 CONFIGURATION

- Set PCSSP0 interface power/clock bit to one in the PCONP register
- Set Peripheral clock selections PCLK_SSP0 bits in PCLKSEL1 register
- Select Pins P0.15 – P0.18 through PINSEL registers and PINMODE
- Select data size, frame format, CPOL,CPHA and SCR in the control register SSP0CR0
- SSP0CPSR, clock prescale register is set
- Enable interrupt in NVIC using SSP0_IRQn register
- Enable master mode on the SSP by setting SSE bit in SSP0CR1 register to one
- Set RORIM and RTIM bits to 1 in SSP0IMSC register to enable error related interrupts

Reference: CTI One Corporation

Main Code

```
int main (void)
{
	uint32_t pnum = PORT_NUM;
	pnum = 0 ;
	if ( pnum == 0 )
		SSP0Init();
	else
		puts("Port number is not correct");
	lcd_init();

	fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);
	int x0,x1,y0,y1;

	x0 = 20;
	x1 = 80;
	y0 = 60;
	y1 = 140;

	drawLine(x0,y0,x1,y1,PURPLE);
	return 0;
}
```

Reference: CTI One Corporation

DEFINITIONS

```
#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 RAJAH 0xffb266

#define BLACK 0x000000

#define BLUE 0x0007FF

#define RED 0xFF0000

#define MAGENTA 0x00F81F

#define WHITE 0xFFFFFF

#define PURPLE 0xCC33FF
```

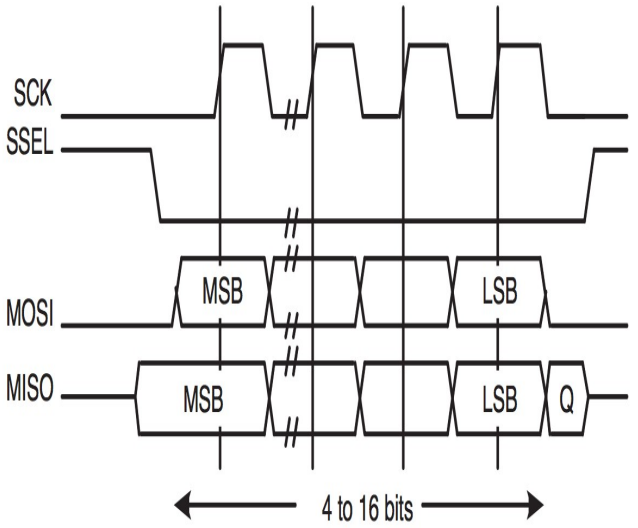
- ST7735_CASET – column address set, 4 arguments, no delay
- ST7735_RASET – row address set, 4 arguments, no delay
- ST7735_RAMWR – memory write to write data on display
- ST7735_SLPOUT – wake controller out of sleep mode, no arguments, delay required
- ST7735_DISPON – turn on main screen, no arguments, delay required

Reference: CTI One Corporation

LCD DRIVERS

Send command/data to the LCD controller using SSP0 port

```
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 );

}
```



- To start data transmission, SSEL0 master signal is driven to low.
- Data is captured at each clock cycle.
- SSEL0 master signal is changed to high to indicate end of data transmission

LCD DRIVERS

Send command/data to the LCD controller using SSP0 port

```
void writecommand(uint8_t c)
{
 LPC_GPIO0->FIOCLR |= (0x1<<21);
 spiwrite(c);
}
```

```
void writedata(uint8_t c)
{
 LPC_GPIO0->FIOSET |= (0x1<<21);
 spiwrite(c);
}
```

```
void writeword(uint16_t c)
{
 uint8_t d;
 d = c >> 8;
 writedata(d);
 d = c & 0xFF;
 writedata(d);
}
```

- Serial information sent to LCD can be either be data or command
- For Command, the LCD's RS/DC input (Pin 0.21 on LPC1769) must be 0
- For Data, the LCD's RS/DC input (Pin 0.21 on LPC1769) must be 1
- Information is sent on the SSP0 port after setting or clearing Pin 0.21
- writeword function transfers 2 bytes (16 bits) of data

LCD DRIVERS

Write color data on the LCD display

```
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);
 }
}
```

- Pixel colors are a combination of Red, Green and Blue colors.
- Each sub color is represented by 8 bits.

Color	(R,G,B)	Hex
Red	(255,0,0)	#FF0000
Green	(0,255,0)	#00FF00
Blue	(0,0,255)	#0000FF
Black	(0,0,0)	#000000
White	(255,255, 255)	#FFFFFF

LCD INITIALIZATION

```
void lcd_init()

{

 int i;
 // Set pins P0.16, P0.21, P0.22 as output
 LPC_GPIO0->FIODIR |= (0x1<<16);

 LPC_GPIO0->FIODIR |= (0x1<<21);

 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);

}
```

- Set pins SSEL0 (P0.16), RS/DC(P0.21) and RST (P0.22) as outputs.
- To do a hardware reset on the LCD controller, turn RST line briefly off, wait enough time and then turn on. This reset initializes the controller's registers to their default values
- After reset, controller enters a low power sleep mode. ST7735_SLPOUT command wakes up the controller and its driver circuits
- Finally turn on LCD display by ST7735_DISPON command

LCD DRIVERS

Set display region on the LCD display

```
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);

}
```

- setAddrWindow function is used to set the display region where new data is written on the display screen
- This is a rectangular region with (x0, y0) and (x1,y1) coordinates
- Column Address Set command (ST7735_CASET) sets the column boundaries or the x coordinates (x0,x1)
- Row Address Set command (ST7735_RASET) sets the row boundaries or the y coordinates (y0,y1)

LCD DRIVERS

Draw a pixel on the LCD Display

```
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);

}
```

- The drawPixel function writes a specified color on a pixel of the LCD screen at the specified location
- setAddrWindow function sets the x and y coordinates for the pixel
- ST7735_RAMWR command is issued before writing data to the controller's RAM
- write888 functions write one dot of specified color the LCD screen

LCD DRIVERS

```
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;
  }
 }
}
```

- Input two end points of the line (x0, y0) & (x1, y1)
- Based on the Digital Differential Analyzer algorithm, find the nearest point of the current point and repeat this step each time.
- Use for loop and drawPixel function to draw every single point between the two points (x0, y0) and (x1, y1)

TEST LCD DISPLAY

- Build Project Lcd_Test and check for any errors
- Debug Lcd_Test and run the program
- Fractal tree is drawn on the LCD display