Key Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD Controller</td>
<td>ILI9325</td>
</tr>
<tr>
<td>Touch Screen Controller</td>
<td>XPT2046</td>
</tr>
<tr>
<td>LCD Type</td>
<td>TFT</td>
</tr>
<tr>
<td>LCD Interface</td>
<td>16-bit parallel</td>
</tr>
<tr>
<td>Touch Screen Interface</td>
<td>SPI</td>
</tr>
<tr>
<td>Backlight</td>
<td>LED</td>
</tr>
<tr>
<td>Colors</td>
<td>65536</td>
</tr>
<tr>
<td>Resolution</td>
<td>320*240 DOTS</td>
</tr>
</tbody>
</table>

Contents

3.2inch 320x240 Touch LCD (C) User Manual ................................................................. 1
Key Parameters ........................................................................................................... 1
1. Hardware Resources ............................................................................................... 2
   1.1 ILI9325 ........................................................................................................ 2
   1.2 XPT2046 ........................................................................................................ 4
2. LCD Pin Description ................................................................................................. 5
3. Example Analysis ..................................................................................................... 6
1. Hardware Resources

1.1 ILI9325

- ILI9325C is a 262,144-color one-chip SoC driver for a-TFT liquid crystal display with resolution of 240RGBx320 dots, comprising a 720-channel source driver, a 320-channel gate driver, 172,800 bytes RAM for graphic data of 240RGBx320 dots, and power supply circuit.

- ILI9325C has five kinds of system interfaces which are i80-system MPU interface (8-/9-/16-/18-bit bus width), VSYNC interface (system interface + VSYNC, internal clock, DB[17:0]), serial data transfer interface (SPI), RGB 6-/16-/18-bit interface (DOTCLK, VSYNC, HSYNC, ENABLE, DB[17:0]).

The following figure shows correspondence between 18-bit RGB Assignment and 16-Bit GRAM.

![18-bit RGB Assignment and 16-Bit GRAM](image)

You can see from figure, the useful data bus interfaces of ILI9325 under 16-bit mode are: D17~D10 and D8~D1. D9 and D0 are unused. Actually, D9 and D0 of ILI9341 are not welded in this LCD module. D17~D10 and D8~D1 of ILI9325 correspond to D15~D0 of the MCU. The lower five bits of the 16-bit MCU data indicate blue. The middle six bits indicate green. The higher five bits indicate red. When the value bigger, the color deeper.

Important Register Introduction

Please see ILI9325 datasheet for more details about ILI9325. Here are just some important register introduction.

**Entry Mode (R03h)**

| R/W | RW | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|-----|----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|
| W   | 1  | TR  | DF  | M   | BGR | 0   | 0   | 0  | 0  | 0  | 0  | ORG| 0  | 0  | 1  | 1  | 0  |
|     |    | TRI | DFM | 0   | BGR | 0   | 0   | 0  | 0  | 0  | 0  | ORG| 0  | 1  | 1  | 0  | 0  |

**AM** Control the GRAM update direction.

- When AM = “0”, the address is updated in horizontal writing direction.
- When AM = “1”, the address is updated in vertical writing direction.

**I/D[1:0]** Control the address counter (AC) to automatically increase or decrease by 1 when update one pixel display data. Refer to the following figure for the details.
ORG Moves the origin address according to the ID setting when a window address area is made. This function is enabled when writing data with the window address area using high-speed RAM write.

**ORG = “0”:** The origin address is not moved. In this case, specify the address to start write operation according to the GRAM address map within the window address area.

**ORG = “1”:** The original address “00000h” moves according to the I/D[1:0] setting.

**BGR** Swap the R and B order of written data.

- **BGR="0":** Follow the RGB order to write the pixel data.
- **BGR="1":** Swap the RGB data to BGR in writing into GRAM.

<table>
<thead>
<tr>
<th>AM</th>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>AD7</td>
<td>AD6</td>
<td>AD5</td>
<td>AD4</td>
<td>AD3</td>
<td>AD2</td>
<td>AD1</td>
<td>AD0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>AD16</td>
<td>AD15</td>
<td>AD14</td>
<td>AD13</td>
<td>AD12</td>
<td>AD11</td>
<td>AD10</td>
<td>AD9</td>
<td>AD8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AD[16:0]** Set the initial value of address counter (AC).

- The address counter (AC) is automatically updated in accordance to the setting of the AM, I/D bits as data is written to the internal GRAM. The address counter is not automatically updated when read data from the internal GRAM.

<table>
<thead>
<tr>
<th>AD[16:0]</th>
<th>GRAM Data Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>17'h00000 ~ 17'h000EF</td>
<td>1st line GRAM Data</td>
</tr>
<tr>
<td>17'h00010 ~ 17'h001EF</td>
<td>2nd line GRAM Data</td>
</tr>
<tr>
<td>17'h00020 ~ 17'h002EF</td>
<td>3rd line GRAM Data</td>
</tr>
<tr>
<td>17'h00030 ~ 17'h003EF</td>
<td>4th line GRAM Data</td>
</tr>
<tr>
<td>17'h13D00 ~ 17' h13DEF</td>
<td>318th line GRAM Data</td>
</tr>
<tr>
<td>17'h13E00 ~ 17' h13EEF</td>
<td>319th line GRAM Data</td>
</tr>
<tr>
<td>17'h13F00 ~ 17' h13FEF</td>
<td>320th line GRAM Data</td>
</tr>
</tbody>
</table>

**Write Data to GRAM (R22h)**
This register is the GRAM access port. When update the display data through this register, the address counter (AC) is increased/decreased automatically.

**GRAM Address Map & Read/Write**
ILI9325C has an internal graphics RAM (GRAM) of 172,800 bytes to store the display data and one pixel is constructed of 18 bits. The GRAM can be accessed through the I2C system, SPI and RGB interfaces.

**i2C 18-/16-bit System Bus Interface Timing**

(a) Write to GRAM

<table>
<thead>
<tr>
<th>nCS</th>
<th>RS</th>
<th>nRD</th>
<th>nWR</th>
<th>DB[17:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write WD to index register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write GRAM data (Nth pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write GRAM data (N+1th pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write GRAM data (N+2th pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write GRAM data (N+3th pixel)</td>
</tr>
</tbody>
</table>

(b) Read from GRAM

<table>
<thead>
<tr>
<th>nCS</th>
<th>RS</th>
<th>nRD</th>
<th>nWR</th>
<th>DB[17:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write WD to index register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dummy Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write WD to index register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dummy Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write WD to index register</td>
</tr>
</tbody>
</table>

**1.2 XPT2046**

- The XPT2046 is a 4-wire resistive touch screen controller that incorporates a 12-bit 125 kHz sampling SAR type A/D converter.
- The XPT2046 supports digital I/O interface voltage from 1.5V to VCC in order to connect low voltage uP.
- The XPT2046 can detect the pressed screen location by performing two A/D conversions. In addition to location, the XPT2046 also measures touch screen pressure. On-chip VREF can be utilized for analog auxiliary input, temperature measurement and battery monitoring with the ability to measure voltage from 0V to 5V.
- The XPT2046 also has an on-chip temperature sensor.
- The XPT2046 is available in 16pin QFN thin package (0.75mm in height) and has the operating temperature range of -40°C to +85°C

2. LCD Pin Description

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5V</td>
<td>5V power supply</td>
<td>When powered from 5V supply, Pin 1 &amp; Pin 2 as power input, Pin 33 &amp; Pin 34 provide 3.3V output.</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>Ground</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>D0</td>
<td>Data pin</td>
<td>D0-D15</td>
</tr>
<tr>
<td>4</td>
<td>D1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>D3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>D5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>D6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>D8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>D9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>D11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>D12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>D13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>D14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>D15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>CS</td>
<td>LCD chip select</td>
<td>Low active</td>
</tr>
<tr>
<td>20</td>
<td>RS</td>
<td>Instruction/Data</td>
<td>RS = 1 : Data Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>register selection</td>
<td>RS = 0 : Instruction Register</td>
</tr>
<tr>
<td>21</td>
<td>WR</td>
<td>Write</td>
<td>WR = 0, RD = 1</td>
</tr>
<tr>
<td>22</td>
<td>RD</td>
<td>Read</td>
<td>WR = 1, RD = 0</td>
</tr>
<tr>
<td>23</td>
<td>RESET</td>
<td>Reset the controller</td>
<td>Low active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>chip</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>NC</td>
<td>Not connect</td>
<td>Not connect</td>
</tr>
<tr>
<td>25</td>
<td>BLVCC</td>
<td>5V or 3.3V</td>
<td>Backlight VCC</td>
</tr>
<tr>
<td>26</td>
<td>BLGND</td>
<td>Ground</td>
<td>Backlight GND</td>
</tr>
<tr>
<td>27</td>
<td>BLCNT</td>
<td>Backlight brightness</td>
<td>Control the backlight brightness via PWM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>adjustment</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>TP_IRQ</td>
<td>Touch screen interrupt</td>
<td>Low level while the touch screen detects pressing</td>
</tr>
</tbody>
</table>
### Table 3.2inch 320x240 Touch LCD (C) User Manual

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>TP_CS</td>
<td>Touch screen chip select</td>
</tr>
<tr>
<td>30</td>
<td>TP_SCK</td>
<td>Touch screen SPI clock</td>
</tr>
<tr>
<td>31</td>
<td>TP_SI</td>
<td>Touch screen data input</td>
</tr>
<tr>
<td>32</td>
<td>TP_SO</td>
<td>Touch screen data output</td>
</tr>
<tr>
<td>33</td>
<td>3.3V</td>
<td>3.3V power supply</td>
</tr>
<tr>
<td>34</td>
<td>GND</td>
<td>Ground</td>
</tr>
</tbody>
</table>

### 3. Example Analysis

We use STM32 development board (MCU STM32F103RCT6 onboard) to describe how to use the LCD. You can use the LCD with other similar products.

The following figure is a schematic of the LCD Port of the development board.

![LCD Port Schematic](image-url)
LCD Demo Procedure

1. Initialize the IO of LCD controller
2. Reset LCD and initialize the register of the LCD
3. Initialize the SPI of touch panel
4. Calibrate touch panel
5. If touched:
   - No
   - Yes
     - Get touch-coordinate
     - Converted to display-coordinate
Source Code Analysis

/* The following macro defines image rotation. */
#else define DISP_ORIENTATION 0
#define DISP_ORIENTATION 90
#define DISP_ORIENTATION 180
#define DISP_ORIENTATION 270
#define Set_Cs GPIO_setBits(GPIOC, GPIO_Pin_6); //CS=1:
#define Clr_Cs GPIO_ResetBits(GPIOC, GPIO_Pin_6); //CS=0:
#define Set_Rs GPIO_setBits(GPIOC, GPIO_Pin_7); //RS=1:
#define Clr_Rs GPIO_ResetBits(GPIOC, GPIO_Pin_7); //RS=0:
#define Set_nWr GPIO_setBits(GPIOC, GPIO_Pin_1); //WR=1:
#define Clr_nWr GPIO_ResetBits(GPIOC, GPIO_Pin_1); //WR=0:
#define Set_nRd GPIO_setBits(GPIOC, GPIO_Pin_2); //RD=1:
#define Clr_nRd GPIO_ResetBits(GPIOC, GPIO_Pin_2); //RD=0:
/* Command writing function */
inline void LCD_WriteIndex(uint16_t index)
{
    Clr_Rs; //RS=0
    Set_nRd; //RD=0
    LCD_Delay(0); //Delay
    GPIOB->ODR = index; /* index is an command’s address*/
    LCD_Delay(0); //Delay
    Clr_nWr; //WR=0
    Set_nWr; //WR=1
}
/* Data writing function */
inline void LCD_WriteData(uint16_t data)
{
    Set Rs; //RS=1
    LCD_Delay(0); //Delay
    GPIOB->ODR = data; /* Data writing address*/
    LCD_Delay(0); //Delay
    Clr_nWr; //WR=0
    Set_nWr; //WR=1
}
/* Data reading function */
inline uint16_t LCD_ReadData(void)
{
    uint16_t value;
    Set Rs;
}
Set_nWr;
Clr_nRd;
GPIOB->CRH = 0x44444444;   //Set PB0-PB15 as input pin
GPIOB->CRL = 0x44444444;
value = GPIOB->IDR;       //Reads data
GPIOB->CRH = 0x33333333;   //Set PB0-PB15 as output pin
GPIOB->CRL = 0x33333333;
Set_nRd;
return value;
}

/****************************************************************
**************
Write data to a specified address, LCD_Reg indicates register address while LCD_RegValue indicates register value.
*****************************************************************/
__inline void LCD_WriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValue)
{
Clr_Cs;
LCD_WriteIndex(LCD_Reg);        //Writing command, LCD_Reg is an address to be written in.
LCD_WriteData(LCD_RegValue);    //Writes data.
Set_Cs;
}

/****************************************************************
**************
Read data from a specified address, LCD_Reg indicates register address. This function will return a value from the address.
*****************************************************************/
__inline uint16_t LCD_ReadReg(uint16_t LCD_Reg)
{
uint16_t LCD_RAM;
Clr_Cs;
LCD_WriteIndex(LCD_Reg);        //Writing command, LCD_Reg is an address to be read from.
LCD_RAM = LCD_ReadData();      //Reads data
Set_Cs;
return LCD_RAM;
}

//That’s the basic read-and-write functions by IO emulation. If you want use FSMC from STM32 to control the LCD, you can read another demo LCD + TouchPanel(8080 FSMC)
/*****************************************************************************/
**************
This is LCD initialization function. The initialization value of the LCD is provided by the factory. So usually you can copy them directly to initialize LCD. Please refer to ILI9325 datasheets for more details.
*****************************************************************************/

void LCD_Initializtion(void)
{
    uint16_t DeviceCode;
    LCD_Configuration();                  //LCD Initialization
    GPIO_ResetBits(GPIOC, GPIO_Pin_0);   /* LCD reset*/
    delay_ms(100);
    GPIO_ResetBits(GPIOC, GPIO_Pin_0);
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);     /* Enable back light */
    DeviceCode = LCD_ReadReg(0x0000);    /* Reads ID */
    if( DeviceCode == 0x9325 || DeviceCode == 0x9328 )
    {
        LCD_WriteReg(0x00e7,0x0010);
        LCD_WriteReg(0x0000,0x0001);
        LCD_WriteReg(0x0001,(0<<10)|(1<<8));
        LCD_WriteReg(0x0002,0x0700);
        #if (DISP_ORIENTATION == 0)
            LCD_WriteReg(0x0003,(1<<12)|(1<<5)|(1<<4)|(0<<3));
        #elif (DISP_ORIENTATION == 90)
            LCD_WriteReg(0x0003,(1<<12)|(0<<5)|(1<<4)|(1<<3));
        #elif (DISP_ORIENTATION == 180)
            LCD_WriteReg(0x0003,(1<<12)|(0<<5)|(0<<4)|(0<<3));
        #elif (DISP_ORIENTATION == 270)
            LCD_WriteReg(0x0003,(1<<12)|(1<<5)|(0<<4)|(1<<3));
        #endif
        LCD_WriteReg(0x0004,0x0000);
        LCD_WriteReg(0x0008,0x0207);
        LCD_WriteReg(0x0009,0x0000);
        LCD_WriteReg(0x000a,0x0000);
        LCD_WriteReg(0x000c,0x0001);
        LCD_WriteReg(0x000d,0x0000);
        LCD_WriteReg(0x000f,0x0000);
        /* Power On sequence */
        LCD_WriteReg(0x0010,0x0000);
        LCD_WriteReg(0x0011,0x0000);
        LCD_WriteReg(0x0012,0x0000);
        LCD_WriteReg(0x0013,0x0000);
        delay_ms(50); /* delay 50 ms */
```c
LCD_WriteReg(0x0010, 0x1590);
LCD_WriteReg(0x0011, 0x0227);
delay_ms(50); /* delay 50 ms */
LCD_WriteReg(0x0012, 0x009c);
delay_ms(50); /* delay 50 ms */
LCD_WriteReg(0x0013, 0x1900);
LCD_WriteReg(0x0029, 0x0023);
LCD_WriteReg(0x002b, 0x000e);
delay_ms(50); /* delay 50 ms */
delay_ms(50); /* delay 50 ms */
LCD_WriteReg(0x0030, 0x0007);
LCD_WriteReg(0x0031, 0x0707);
LCD_WriteReg(0x0032, 0x0006);
LCD_WriteReg(0x0035, 0x0704);
LCD_WriteReg(0x0036, 0x1f04);
LCD_WriteReg(0x0037, 0x0004);
LCD_WriteReg(0x0038, 0x0000);
LCD_WriteReg(0x0039, 0x0706);
LCD_WriteReg(0x003a, 0x0000);
LCD_WriteReg(0x003c, 0x0701);
LCD_WriteReg(0x003d, 0x000f);
delay_ms(50); /* delay 50 ms */
LCD_WriteReg(0x0050, 0x0000);
LCD_WriteReg(0x0051, 0x00ef);
LCD_WriteReg(0x0052, 0x0000);
LCD_WriteReg(0x0053, 0x013f);
LCD_WriteReg(0x0060, 0xa700);
LCD_WriteReg(0x0061, 0x0001);
LCD_WriteReg(0x0080, 0x0000);
LCD_WriteReg(0x0081, 0x0000);
LCD_WriteReg(0x0082, 0x0000);
LCD_WriteReg(0x0083, 0x0000);
LCD_WriteReg(0x0084, 0x0000);
LCD_WriteReg(0x0085, 0x0000);
LCD_WriteReg(0x0086, 0x0000);
LCD_WriteReg(0x0087, 0x0000);
LCD_WriteReg(0x0088, 0x0000);
LCD_WriteReg(0x0089, 0x0000);
LCD_WriteReg(0x008a, 0x0000);
LCD_WriteReg(0x008b, 0x0000);
LCD_WriteReg(0x008c, 0x0000);
LCD_WriteReg(0x008d, 0x0000);
LCD_WriteReg(0x008e, 0x0000);
LCD_WriteReg(0x008f, 0x0000);
LCD_WriteReg(0x0090, 0x0010);
LCD_WriteReg(0x0091, 0x0000);
LCD_WriteReg(0x0092, 0x0000);
LCD_WriteReg(0x0093, 0x0000);
LCD_WriteReg(0x0094, 0x0000);
LCD_WriteReg(0x0095, 0x0000);
LCD_WriteReg(0x0096, 0x0000);
LCD_WriteReg(0x0097, 0x0000);
LCD_WriteReg(0x0098, 0x0000);
*/ display on sequence */
LCD_WriteReg(0x0007, 0x0133);
} 

delay_ms(50);
```
Set window coordinate.

```
static void LCD_SetCursor( uint16_t Xpos, uint16_t Ypos )
{
    uint16_t temp;
    #if (DISP_ORIENTATION == 0)
    #elif (DISP_ORIENTATION == 90)
        temp = Xpos;
        Xpos = Ypos;
        Ypos = MAX_X - 1 - temp;
    #elif (DISP_ORIENTATION == 180)
        Xpos = MAX_X - 1 - Xpos;
        Ypos = MAX_Y - 1 - Ypos;
    #elif (DISP_ORIENTATION == 270)
        temp = Ypos;
        Ypos = Xpos;
        Xpos = MAX_Y - 1 - temp;
    #endif
    LCD_WriteReg(0x0020, Xpos); // Sets the horizontal position X
    LCD_WriteReg(0x0021, Ypos); // Sets the vertical position Y
}
```

Clear the screen and fill it with one color.

```
void LCD_Clear(uint16_t Color)
{
    uint32_t index=0;
    LCD_SetCursor(0,0); //Set cursor coordinate X, Y
    Clr_Cs;
    LCD_WriteIndex(0x0022); //Start to write data into GRAM
    for( index = 0; index < MAX_X * MAX_Y; index++ )
    {
        LCD_WriteData(Color);
    }
    Set_Cs;
}
int main(void)
{
    //Delay and initialize your system
    LCD_Initialization(); //LCD initialization
    LCD_Clear(Red);      //Clear the LCD with filled with red.
    //You can fill functions to calibrate touch screen.
    /* Infinite loop */
    while (1)
    {
        //You can fill functions to show touch coordinate on the LCD.
    }
}