1.1 ILI9325

ILI9325 是一個 240x320（RGB）解析度、262144 色的 TFT 液晶顯示幕的驅動晶片；172820 (240 x 320x 18/8) 位元組的 RAM。每個圖元點深度可以達到 18 位。

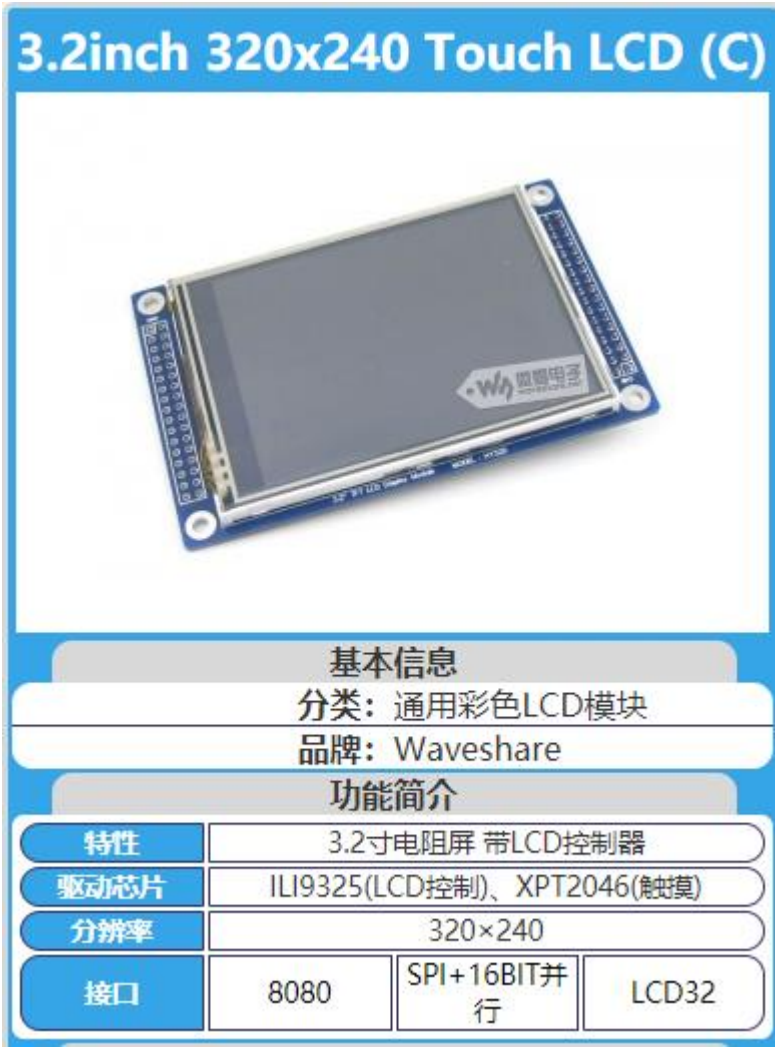
ILI9325 有以下幾種資料介面模式：

1) i80-system MPU 介面(8-/9-/16-/18-bit bus width)

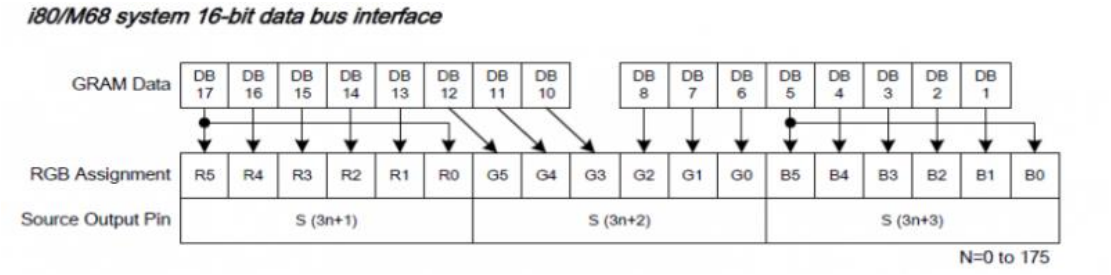
2) VSYNC 介面(system interface + VSYNC, internal clock, DB[17:0])

3) serial data transfer 介面(SPI)

4) RGB 6-/16-/18-bit 介面(DOTCLK, VSYNC, HSYNC, ENABLE, DB[17:0]).



此屏的 ILI9325 的 18 位 RGB 賦值與 LCD GRAM 的對應關係如圖所示：



從圖中可以看出，ILI9325 在 16 位元模式下面，GRAM Data 有用的是：D17~D10 和 D8~D1， D9 和 D0 沒有用到，實際上在我們 LCD 模組裡面， ILI9341 的 D9 和 D0 沒有引出，ILI9325 的 D17~D10 和 D8~D1 對應 MCU 的 D15~D0。MCU 的 16 位元數據，最低 5 位代表藍色，中間 6 位為綠色，最高 5 位為紅色；數值越大，表示該顏色越深。

重要寄存器介紹

寄存器詳細介紹請參閱 ILI9325 的 datasheet。這裡只介紹一些重要的寄存器設置：

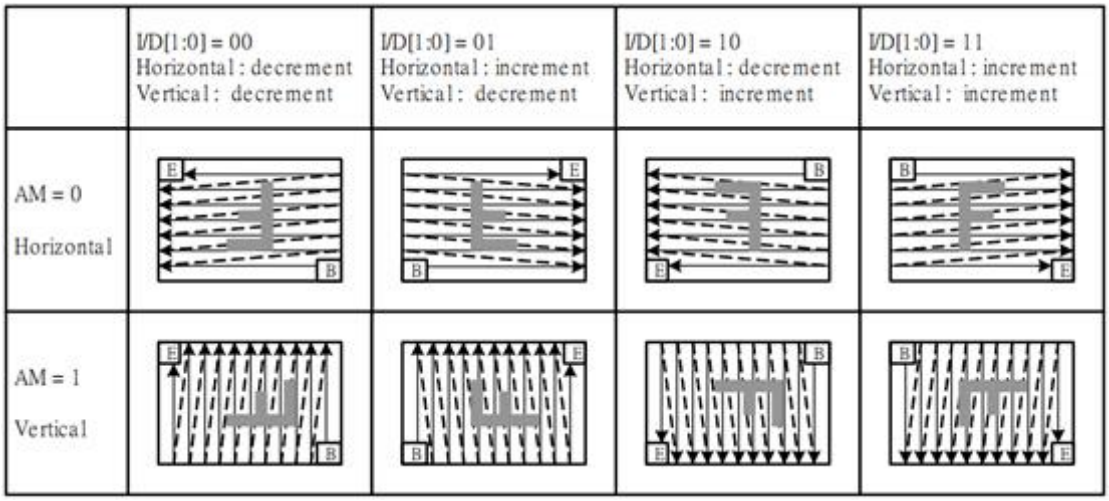
輸入設置（R03h）：

R/W	RS	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W	1	TRI	DFM	0	BGR	0	0	0	0	ORG	0	I/D1	I/D0	AM	0	0	0
Default		0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

AM：控制 GRAM 的更新方向

- 當 AM=0 位址在水準寫入方向得以更新
- 當 AM=1 位址在垂直寫入方向得以更新

I/D[1:0]： 當更新一個圖元資料時， I/D[1:0]位控制位址計數器（AC）自動增加或者減少 1。詳細請見下圖：



ORG ：當視窗位址區域產生後，原點位址的移動根據 ID 的設定。當使用高速寫 RAM 模式寫資料到視窗位址區域這個功能被使能。

- ORG=0 ：原點位址不會移動，在這種情況下，在視窗位址區域根據 GRAM 的位址映射指定一個位址開始寫操作。
- ORG=1 ：原始位址為“00000h” 根據 ID[1 :0]的 設定來移動。

BGR：根據被寫入的資料交換 R 和 B 的順序

- BGR=0 ：根據 RGB 的順序寫入圖元資料
- BGR=1 ：交換 RGB 資料為 BGR 寫到 GRAM

GRAM 水準垂直變址設置(R20h, R21h) 水準位置寄存器變址 0x20 垂直變址寄存器變址 0x21 GRAM Horizontal/Vertical Address Set (R20h, R21h)

R/W	RS
W	1
W	1
Default	

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
0	0	0	0	0	0	0	AD16	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AD[16:0]：用於設置變址計數器（AC）的初始化數值。當資料寫入內部的 GARM 中時，變址計數器（AC）會根據 AM 和 I/D 位元的設置來自動的更新其數值。當從內部 GRAM 中讀數據時變址計數器不會自動更
新。

GRAM 内部数据映射图

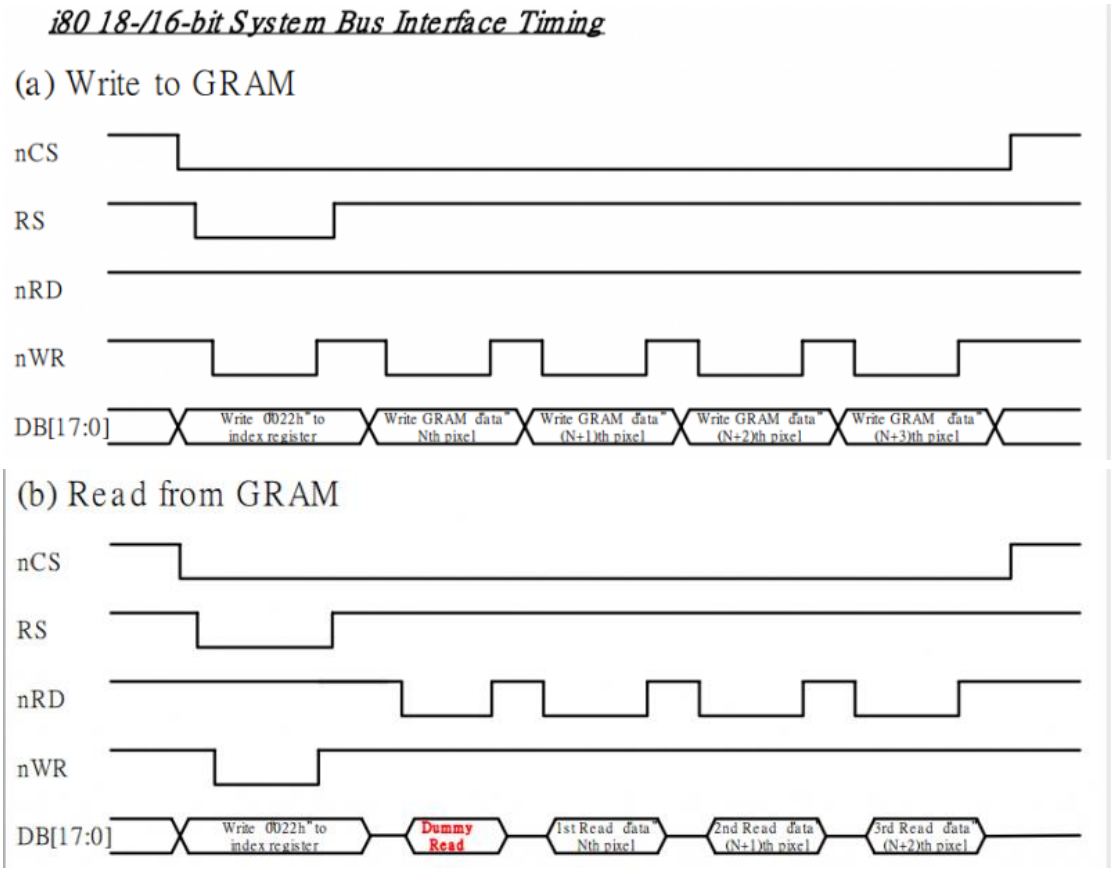
AD[16:0]	GRAM Data Map
0x00000--0x000EF	第 1 行 GRAM Data
0x00100--0x001EF	第 2 行 GRAM Data
0x00200--0x002EF	第 3 行 GRAM Data
0x00300--0x003EF	第 4 行 GRAM Data
……	……
0x13D00--0x13DEF	第 318 行 GRAM Data
0x13E00--0x13EEF	第 319 行 GRAM Data
0x13F00--0x13FEF	第 320 行 GRAM Data

寫數據到 GRAM
(R22h)

R/W	RS	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W	1	RAM write data (WD[17:0], the DB[17:0] pin assignment differs for each interface.																	

該寄存器是 GRAM 的訪問埠，當通過這個寄存器更新顯示資料的時候，位址計數器會自動增加或者減少。

GRAM 位址映射和讀/寫 ILI9325 有一個容量為 172800 bytes 的內部圖片 RAM (GRAM)，用來存儲顯示資料。一圖元由十八位元資料構成， GRAM 可以通過 i80 系統介面，SPI 或者是 RGB 介面來訪問，以下是 GRAM 在 i80 系統介面下讀寫的時序“



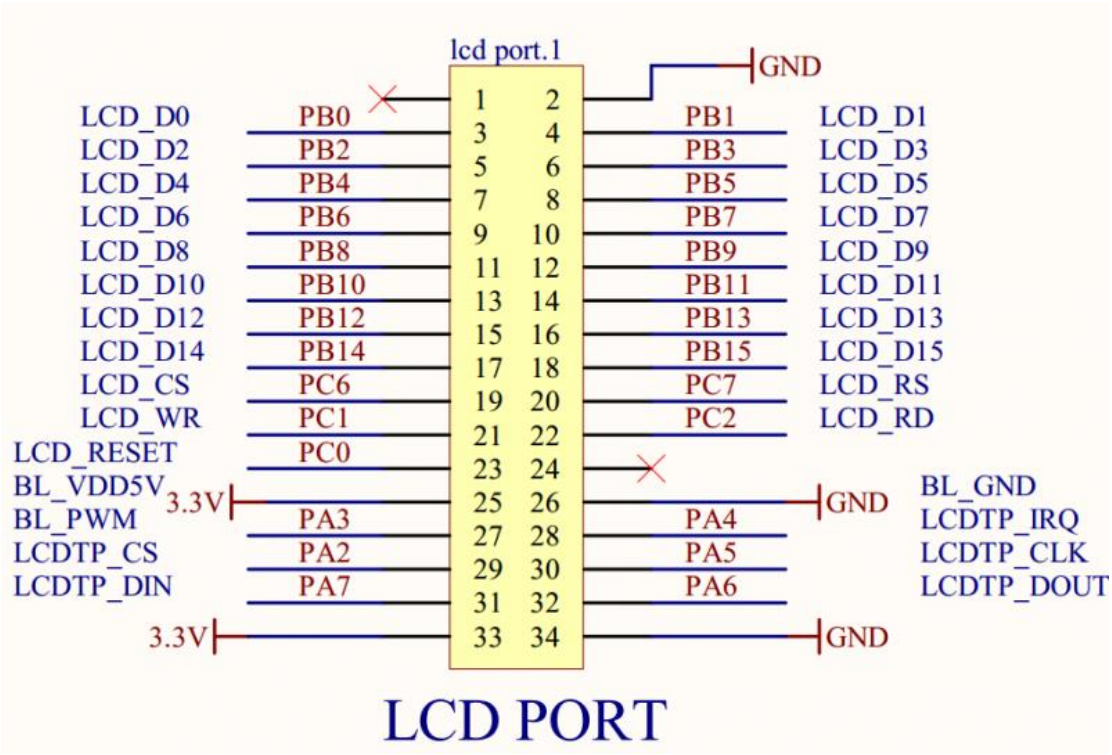
1.2 XPT2046

- XPT2046 是一款 4 線制電阻式觸控式螢幕控制器，內含 12 位解析度 125KHz 轉換速率逐步逼近型 A/D 轉換器。
- XPT2046 支援從 1.5V 到 5.25V 的低電壓 I/O 介面。
- XPT2046 能通過執行兩次 A/D 轉換查出被按的螢幕位置，除此之外，還可以測量加在觸控式螢幕上的壓力。內部自帶 2.5V 參考電壓，可以作為輔助輸入、溫度測量和電池監測之用，電池監測的電壓範圍可以從 0V 到 5V。
- XPT2046 片內集成有一個溫度感測器。在 2.7V 的典型工作狀態下，關閉參考電壓，功耗可小於 0.75mW。XPT2046 採用微小的封裝形式：TSSOP-16,QFN-16 和 VFBGA－48。 工作溫度範圍為-40℃～+85℃。與 ADS7846、TSC2046、AK4182A 完全相容。

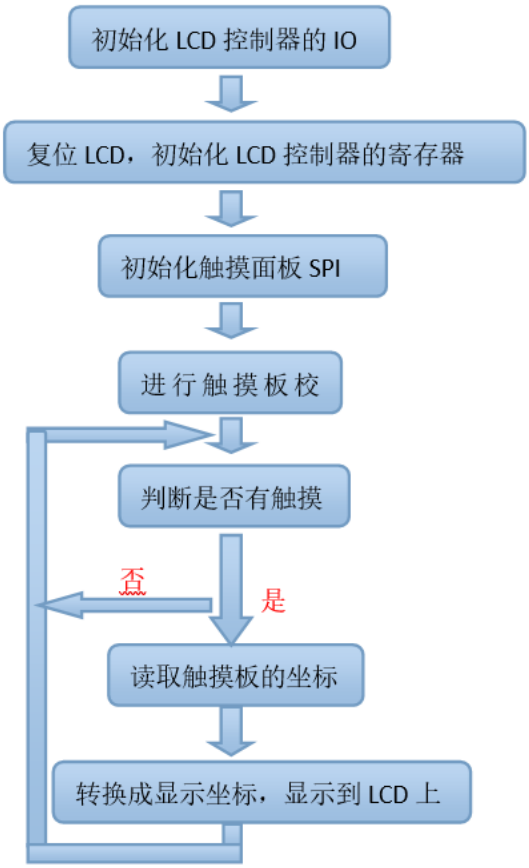
引腳號	標識	描述	功能
1	5V	5V 電源	當 5V 供電時（1，2 腳接 5V 電源），3.3V 端（33，34 腳輸出 3.3V 電壓）
2	GND	接地	GND
3	D0	數據線	D0-D15
4	D1		
5	D2		
6	D3		
7	D4		
8	D5		
9	D6		
10	D7		
11	D8		
12	D9		
13	D10		
14	D11		
15	D12		
16	D13		
17	D14		
18	D15		
19	CS	LCD 片選信號	低電平選擇 LCD
20	RS	指令/資料 寄存器選擇	RS = 0：指令寄存器 RS = 1：資料寄存器
21	WR	寫動作	WR = 0，RD = 1
22	RD	讀動作	WR = 1，RD = 0
23	RESET	晶片重啟	低電平重啟晶片
24	NC		
25	BLVCC	5V 或 3.3V	背光燈 VCC
26	BLGND	接地	背光燈 GND
27	BLCNT	背光燈亮度調節	可以使用 PWM 來控制背光燈亮度
28	TP_IRQ	觸摸面板中斷	檢測到觸摸面板有按下則為低電平
29	TP_CS	觸摸面板片選信號	低電平選擇觸摸面板
30	TP_SCK	觸摸面板 SPI 時鐘信號	連接到 SPI 的 SCK
31	TP_SI	觸摸面板 SPI 資料登錄	連接到 SPI 的 MOSI
32	TP_SO	觸摸面板 SPI 資料輸出	連接到 SPI 的 MISO
33	3.3V	+3.3 電源	當 3.3V 供電時（33，34 腳輸入 3.3V） 1,2 腳懸空
34	GND	接地	

本手冊使用主控晶片 STM32F103RCT6 的開發板說明本款 LCD 的基本使用方法。用戶也可以採用其他類似的開發板進行開發。

3.2inch 320x240 Touch LCD (C)和 STM32F103RCT6 連接介面圖：



程式流程：



原始程式碼解析：

```
1. /*下面巨集定義的是圖像的旋轉角度*/<br />
2. //#define DISP_ORIENTATION
       0
3. //#define DISP_ORIENTATION
       90
4. //#define DISP_ORIENTATION
       180
5. #define DISP_ORIENTATION
       270
6. #define Set_Cs       GPIO_SetBits(GPIOC, GPIO_Pin_6);    //CS=
   1；
7. #define Clr_Cs       GPIO_ResetBits(GPIOC, GPIO_Pin_6);  //CS=
   0；
8.
```

```
#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;
/* 寫命令函數 */
__inline void LCD_WriteIndex(uint16_t index)
{
Clr_Rs;             //RS=0
Set_nRd;            //RD=0
LCD_Delay(0);        //延時
GPIOB->ODR = index; /*寫命令 */
LCD_Delay(0);        //延時
Clr_nWr;            //WR=0
Set_nWr;            //WR=1
}
/* 寫資料函數 */
__inline void LCD_WriteData(uint16_t data)
{
Set_Rs;              //RS=1
LCD_Delay(0);        //延時
GPIOB->ODR = data;   /*寫數據*/
LCD_Delay(0);        //延時
Clr_nWr;            //WR=0
Set_nWr;            //WR=1
}
/* 讀數據函數 */
__inline uint16_t LCD_ReadData(void)
```

```
{
uint16_t value;
Set_Rs;
Set_nWr;
Clr_nRd;
GPIOB->CRH = 0x44444444;   //設置 PB0-PB15 為輸入
GPIOB->CRL = 0x44444444;
value = GPIOB->IDR;         //讀取數據
    GPIOB->CRH = 0x33333333;   //設置 PB0-PB15 為輸出
    GPIOB->CRL = 0x33333333;
    Set_nRd;
    return value;
}
/*************************************************************************
**************

指定的位址寫入資料，LCD_Reg 是位址，LCD_RegValue 是寫入的值。

*************************************************************************
*************/
__inline void LCD_WriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValu
e)
{
Clr_Cs;
LCD_WriteIndex(LCD_Reg);        //寫指令；即要寫入資料的位址；
LCD_WriteData(LCD_RegValue);    //資料寫入；
Set_Cs;
}

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

從指定的位址讀取資料，LCD_Reg 是位址，函數返回讀取出來的值。
<pre>
*************************************************************************
*************/
__inline uint16_t LCD_ReadReg(uint16_t LCD_Reg)
```

```
{
uint16_t LCD_RAM;
Clr_Cs;
LCD_WriteIndex(LCD_Reg);     //寫指令；即要讀出資料的位址；
LCD_RAM = LCD_ReadData();  //數據讀出；
Set_Cs;
return LCD_RAM;
}
//以上是最基本的讀寫函數；IO 模擬操作，如果想 STM32 的 FSMC 來控制的，參考另外一個常式 LCD + TouchPanel(8080 FSMC)
/*****************************************************************
LCD 寄存器的初始化，以下寄存器的初始化值由 LCD 原廠家提供，按照如下配置就可以正常顯示，寄存器請參考晶片手冊。
*****************************************************************/
void LCD_Initializtion(void)
{
uint16_t DeviceCode;
LCD_Configuration();                 //管腳初始化
GPIO_ResetBits(GPIOC, GPIO_Pin_0);   /* LCD 復位*/
delay_ms(100);
GPIO_SetBits(GPIOC, GPIO_Pin_0);
GPIO_SetBits(GPIOA, GPIO_Pin_3);     /*使能背光 */
DeviceCode = LCD_ReadReg(0x0000);    /* 讀取 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(0x0011,0x0007);
    LCD_WriteReg(0x0012,0x0000);

    LCD_WriteReg(0x0013,0x0000);
    delay_ms(50);  /* delay 50 ms */
    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(0x003c,0x0701);
```

```
140.     LCD_WriteReg(0x003d,0x000f);
141.     delay_ms(50);  /* delay 50 ms */
142.     LCD_WriteReg(0x0050,0x0000);
143.     LCD_WriteReg(0x0051,0x00ef);
144.     LCD_WriteReg(0x0052,0x0000);
145.     LCD_WriteReg(0x0053,0x013f);
146.     LCD_WriteReg(0x0060,0xa700);
147.     LCD_WriteReg(0x0061,0x0001);
148.     LCD_WriteReg(0x006a,0x0000);
149.     LCD_WriteReg(0x0080,0x0000);
150.     LCD_WriteReg(0x0081,0x0000);
151.     LCD_WriteReg(0x0082,0x0000);
152.     LCD_WriteReg(0x0083,0x0000);
153.     LCD_WriteReg(0x0084,0x0000);
154.     LCD_WriteReg(0x0085,0x0000);
155.     LCD_WriteReg(0x0090,0x0010);
156.     LCD_WriteReg(0x0092,0x0600);
157.     LCD_WriteReg(0x0093,0x0003);
158.     LCD_WriteReg(0x0095,0x0110);
159.     LCD_WriteReg(0x0097,0x0000);
160.     LCD_WriteReg(0x0098,0x0000);
161.     /* display on sequence */
162.     LCD_WriteReg(0x0007,0x0133);
163.     }
164.     delay_ms(50);
165.     }
166.     /*************************************************************
    **********************
167.     設置顯示視窗的位置 X、Y；
168.     *************************************************************
    ******************/
169.     static void LCD_SetCursor( uint16_t Xpos, uint16_t Ypos )
170.     {
171.     uint16_t temp;
172.     #if (DISP_ORIENTATION == 0)
173.     #elif (DISP_ORIENTATION == 90)
```

```
174.     temp = Xpos;
175.     Xpos =Ypos;
176.     Ypos = MAX_X - 1 - temp;
177.     #elif (DISP_ORIENTATION == 180)
178.     Xpos = MAX_X - 1 - Xpos;
179.     Ypos = MAX_Y - 1 - Ypos;
180.     #elif (DISP_ORIENTATION == 270)
181.     temp = Ypos;
182.     Ypos = Xpos;
183.     Xpos = MAX_Y - 1 - temp;
184.     #endif
185.     LCD_WriteReg(0x0020, Xpos); //水準位置 X 的設置
186.     LCD_WriteReg(0x0021, Ypos); //垂直位置 Y 的設置
187.     }
188.     /*************************************************************
    *********************
189.     清屏函數，作用是讓整個屏顯示某一種顏色，
190.     *************************************************************
    ******************/
191.     void LCD_Clear(uint16_t Color)
192.     {
193.     uint32_t index=0;
194.     LCD_SetCursor(0,0);    //設置游標的初始位置 X、Y
195.     Clr_Cs;
196.     LCD_WriteIndex(0x0022);//開始向 GRAM 寫資料
197.     for( index = 0; index < MAX_X * MAX_Y; index++ )
198.     {
199.     LCD_WriteData(Color);
200.     }
201.     Set_Cs;
202.     }
203.
204.     int main(void)
205.     {
206.     //延時和初始化系統
207.       LCD_Initializtion();      //LCD 初始化
```

```
208.     //LCD 觸控板初始化
209.       LCD_Clear(Red);        //清屏為紅色
210.     //您可以編寫函數來校準螢幕。
211.       /* Infinite loop */
212.       while (1)
213.       {
214.     //您可以編寫函數，把觸摸點座標顯示在 LCD 上
215.       }
216.     }
```