2.2inch 320x240 Touch LCD (A) User Manual

1. Key Parameters

LCD Controller	BD663474 (supports SPI)
Touch Screen Controller	XPT2046
LCD Type	TFT
LCD Interface	SPI serial
Touch Screen Interface	SPI
Backlight	LED
Colors	262,144
Resolution	320*240 DOTS

2. Working Principle of LCD

The SCK and MISO pins of SPI mode are extended on the 2.2inch 320x240 Touch LCD (A). It can be written only, and cannot be read. Data of SPI is saved as 8-bit, and then sent to BD663474 by EPM3032 (EPM3032 is a programmed CPLD) in parallel transmission. BD663474is working on 80-series 8-bit bus interface (Big-endien) mode. WR and RD are locked as written only, so it cannot be read.

CS: Chip select signal (CS)

RS: Register select signal for 80-series bus interface.

"Low": Index register / Status register

"High": Control register

Not used in serial peripheral interface mode.

WR: Write data to the LCD. It is locked as written only, so it cannot be read.

RD: Read data from the LCD. It is locked as written only, so it cannot be read.

D[17：10]: 8-bit bi-directional data bus (Data of SPI is saved as 8-bit, and then sent to BD663474[17：10] by EPM3032 in parallel transmission)

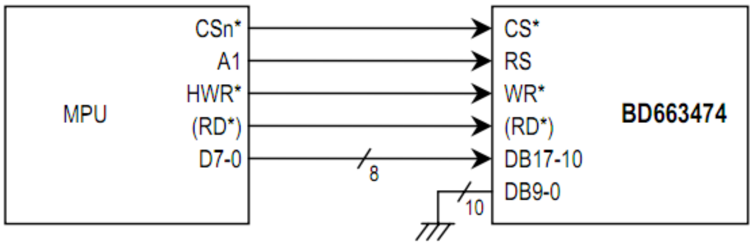
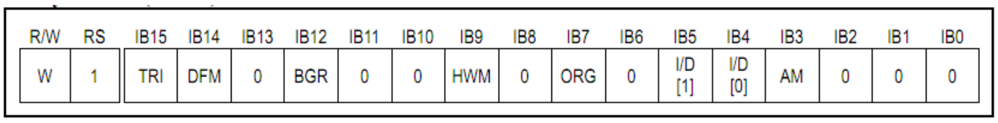


Diagram: Example of interface with 8-bit microcomputer

3. Important Registers introduction of BD663474

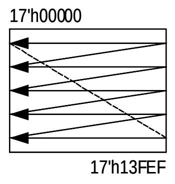
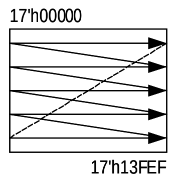
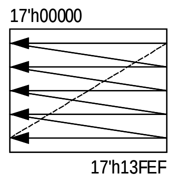
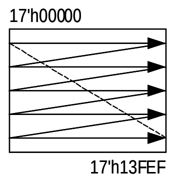
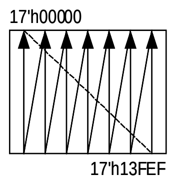
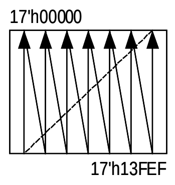
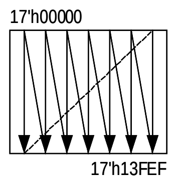
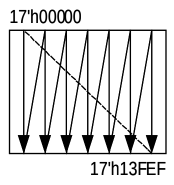
Entry Mode 1 (R003h)



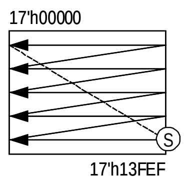
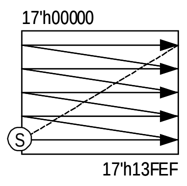
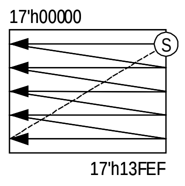
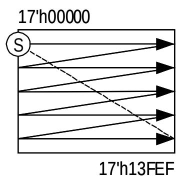
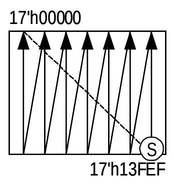
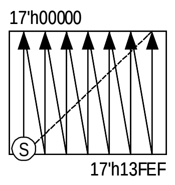
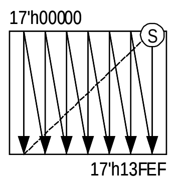
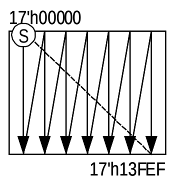
R/W	RS	IB15	IB14	IB13	IB12	IB11	IB10	IB9	IB8	IB7	IB6	IB5	IB4	IB3	IB2	IB1	IB0
W	1	TRI	DFM	0	BGR	0	0	HWM	0	ORG	0	I/D [1]	I/D [0]	AM	0	0	0

Note: For more detail about Entry Mode 1(R003h) . Please refer to BD663474.pdf page 24

AM	This sets the direction of automatically updating the address counter (AC) after data is written to the GRAM. When AM = 0, writes continuously in the horizontal direction. When AM = 1, writes continuously in the vertical direction. When making a window address area, data is written to the GRAM area specified by window addresses in the manner set with I/D and AM.
I/D [1:0]	This sets the automatic increment (+1) or decrement (–1) of the address counter (AC) after data is written to the GRAM. I/D [0] sets the increment or decrement of the lower addresses (AD7-0) in the horizontal direction. I/D [1] sets the increment or decrement of the upper addresses (AD16-8) in the vertical direction.
ORG	When ORG = 1, the start position designated in setting the window address area is shifted in accordance with the I/D bit setting. Set ORG =0 during RAM read and write operation.
HWM	When HWM = 1, data is written to the GRAM at high speed. In high-speed write mode, one line of data is buffered before being written to the GRAM. When HWM = 1, make sure that AM = 0 (horizontal direction) and a data write operation is executed for each line of the window address area. Data will not be written properly in a line where writing is terminated in the middle. Refer to “High-speed RAM write mode” for details.

ORG=0	I/D[1:0] = "00" H-direction: Decrement V-direction: Decrement	I/D[1:0] = "01" H-direction: Increment V-direction: Decrement	I/D[1:0] = "10" H-direction: Decrement V-direction: Increment	I/D[1:0] = "11" H-direction: Increment V-direction: Increment
AM = "0" Horizontal direction				
AM = "1" Vertical direction				

Note: Writing is available only to the GRAM within the window address when setting the window address.

ORG=1	I/D[1:0] = "00" H-direction: Decrement V-direction: Decrement	I/D[1:0] = "01" H-direction: Increment V-direction: Decrement	I/D[1:0] = "10" H-direction: Decrement V-direction: Increment	I/D[1:0] = "11" H-direction: Increment V-direction: Increment
AM = "0" Horizontal direction				
AM = "1" Vertical direction				

Note: When ORG = 1, writing to the RAM starts from the address at the corner (S) in the area.

BGR Reverses the order of dots from RGB to BGR for the 18-bit data to be written to the GRAM.

DFM When IM3-0 = (GND, GND, VccIO, VccIO), sets the data format for three-time transfer RAM writing in conjunction with the TRI bit through the 80-series 8-bit bus interface.
DFM = 0: RGB 18-bit data is written in three byte-boundary transfers.
DFM = 1: RGB 18-bit data is written in three 6-bit transfers.
When IM3-0 = (GND, GND, VccIO, GND), sets the data format for two-time transfer RAM writing in conjunction with the TRI bit through the 80-series 16-bit bus interface.
DFM = 0: RGB 18-bit data is written in two MSB-format transfers.
DFM = 1: RGB 18-bit data is written in two LSB-format transfers.
Always set DFM = 0 when not using an 8-bit or 16-bit interface.

TRI When IM3-0 = (GND, GND, VccIO, VccIO), sets the number of data transfers to the RAM (2 transfers / 3 transfers) through the 80-series 8-bit bus interface.
TRI = 0: 16-bit RAM data is transferred in two transfers.

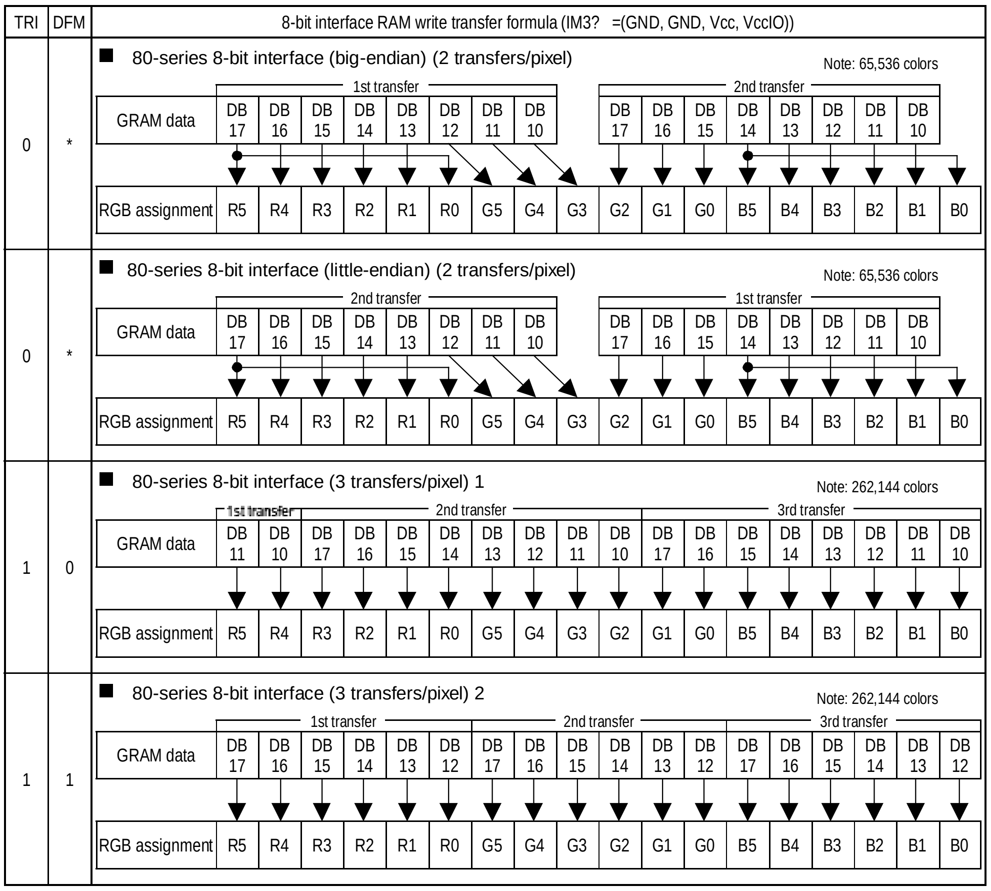
TRI = 1: 18-bit RAM data is transferred in three transfers.

When IM3-0 = (GND, GND, VccIO, GND), set the number of data transfers to the RAM (1 transfer or 2 transfers) through the 80-series 16-bit but interface.

TRI = 0: 16-bit RAM data is transferred in one transfer.

TRI = 1: 18-bit RAM data is transferred in two transfers.

TRI must be set to 0 when not using an 8-bit or 16-bit interface. During GRAM read operation, also set TRI=0.


TRI
DFM
8-bit interface RAM write transfer formula (IM3?   =(GND, GND, Vcc, VccIO))
0 *
80-series 8-bit interface (big-endian) (2 transfers/pixel)
Note: 65,536 colors
1st transfer
2nd transfer
GRAM data
DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10
RGB assignment
R5 R4 R3 R2 R1 R0 G5 G4 G3 G2 G1 G0 B5 B4 B3 B2 B1 B0
0 *
80-series 8-bit interface (little-endian) (2 transfers/pixel)
Note: 65,536 colors
2nd transfer
1st transfer
GRAM data
DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10
RGB assignment
R5 R4 R3 R2 R1 R0 G5 G4 G3 G2 G1 G0 B5 B4 B3 B2 B1 B0
1 0
80-series 8-bit interface (3 transfers/pixel) 1
Note: 262,144 colors
1st transfer
2nd transfer
3rd transfer
GRAM data
DB 11 DB 10 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 11 DB 10
RGB assignment
R5 R4 R3 R2 R1 R0 G5 G4 G3 G2 G1 G0 B5 B4 B3 B2 B1 B0
1 1
80-series 8-bit interface (3 transfers/pixel) 2
Note: 262,144 colors
1st transfer
2nd transfer
3rd transfer
GRAM data
DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12 DB 17 DB 16 DB 15 DB 14 DB 13 DB 12
RGB assignment
R5 R4 R3 R2 R1 R0 G5 G4 G3 G2 G1 G0 B5 B4 B3 B2 B1 B0

Note: Instructions in 8-bit interface mode (IM3-0 = (GND, GND, VccIO, VccIO) are transferred in two 8-bit transfers regardless of TRI and DFM settings.

4. Introduction of 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

5. LCD Pin Description

Table 1: LCD pin description

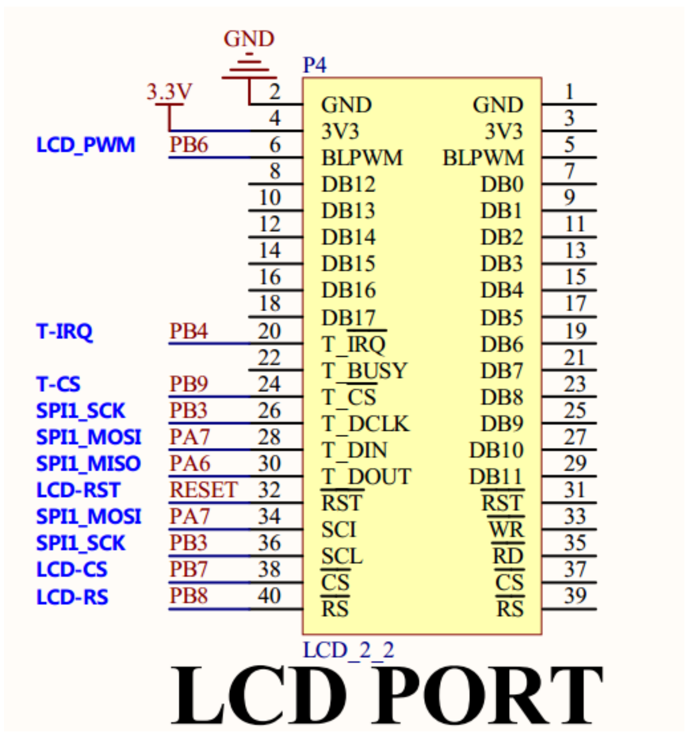
PIN NO.	SYMBOL	DESCRIPTION	FUNCTION
1	GND	Ground	Ground
2			
3	3.3V	3.3V power supply	Connects to 3.3V
4			
5	PWM	Backlight brightness adjustment	Control the backlight brightness via PWM
6			
8	NC	NC	NC
10	NC		
12	NC		
14	NC		
16	NC		
18	NC		
20	NC		
22	NC		
24	NC		
26	NC		
28	NC		
30	NC		
7	NC		
9	NC		
11	NC		
13	NC		
15	NC		
17	NC		
19	T_IRQ	Touch screen interrupt	Low level while the touch screen detects pressing
21	T_BUSY	Touch screen chip busy	
23	T_CS	Touch screen chip select	Low active
25	T_DCLK	Touch screen SPI clock	Connects to SPI SCK
27	T_DIN	Touch screen data input	Connects to SPI MOSI
29	T_DOUT	Touch screen data output	Connects to SPI MISO
31	RESET	Reset the controller chip	Low active

32			
33	MISO	Screen SPI data input (in Serial Mode)	Connects to SPI MOSI
35	SCK	Screen SPI clock (in Serial Mode)	Connects to SPI SCK
34	NC	NC	NC
36	NC	NC	NC
37	CS	LCD chip select (in Serial Mode)	Low active
38			

39	RS	Instruction/Data register selection	RS = 1 : Data Register
			RS = 0 : Instruction Register

40			

Figure 1: Schematic of LCD port



6. Example Analysis

We use Open205R-C development board (MCU STM32F205R onboard, designed by Waveshare) to describe how to use the LCD. You can use the LCD with other similar products.

6.1 Preparations

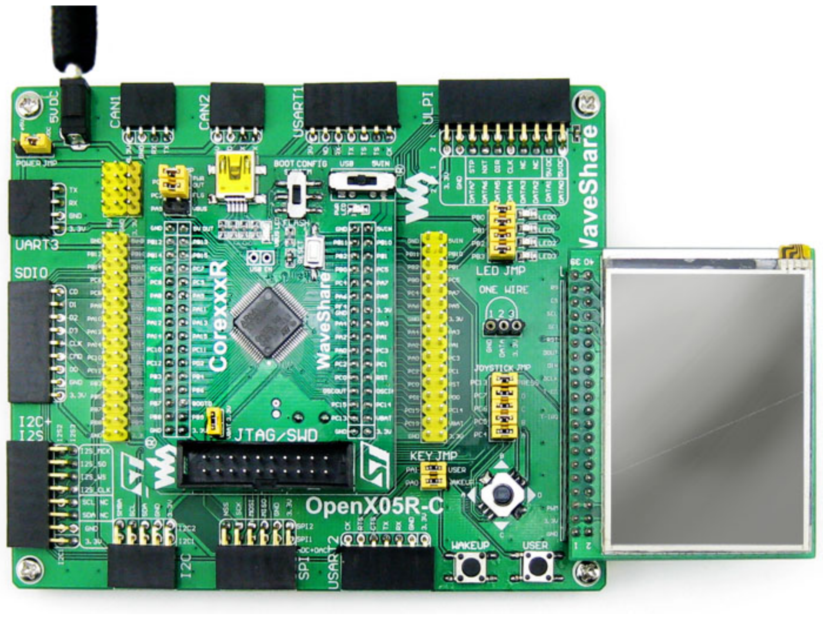
- Open205R-C development board (MCU onboard STM32F205R)
- 2.2inch 320x240 Touch LCD (A)

- 5V power supply (Specification: 5V~2A, outer diameter 3.5mm, inner diameter 1.35mm)
- ST-LINK programmer

6.2 LCD Test Procedure

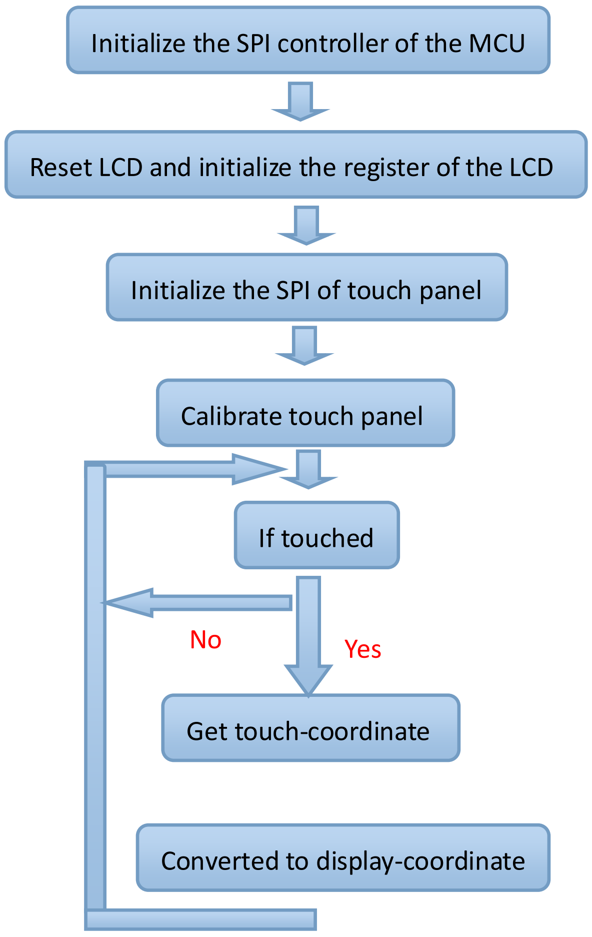
1) Insert LCD to the LCD port of Open205R-C development board as shown in the following figure. The schematic of the port is shown in figure 1. You can connect the LCD to other development board according to figure 1.

Figure 2: How to connect the LCD to the board.



2) Power up the board.
3) Program demo code to the board.
   Demo code description: we supply AVR, PIC, STM8 and STM32 example of the LCD. You can find them from file 2.2inch-320x240-Touch-LCD-A_code.7z (please unzip this file). In this manual, we select the corresponding example STM32, however, you should select the example and programming method according to your board.

6.3 LCD Demo Procedure


Initialize the SPI controller of the MCU
Reset LCD and initialize the register of the LCD
Initialize the SPI of touch panel
Calibrate touch panel
If touched
No
Yes
Get touch-coordinate
Converted to display-coordinate

6.4 Source Code Analysis

```
#define LCD_RST_H() GPIO_SetBits(GPIOB,GPIO_Pin_9) // Set RST to 1
#define LCD_RST_L() GPIO_ResetBits(GPIOB,GPIO_Pin_9)//Set RST to 0

#define LCD_RS_H() GPIO_SetBits(GPIOB,GPIO_Pin_8) //Set RS to 1
#define LCD_RS_L() GPIO_ResetBits(GPIOB,GPIO_Pin_8) //Set RS 0

#define LCD_CS_H() GPIO_SetBits(GPIOB,GPIO_Pin_7) //Set CS to 1
#define LCD_CS_L() GPIO_ResetBits(GPIOB,GPIO_Pin_7) //Set CS to 0

#define TOUCH_nCS_H() GPIO_SetBits(GPIOB,GPIO_Pin_9) //Set CS of
touch to 1
#define TOUCH_nCS_L() GPIO_ResetBits(GPIOB,GPIO_Pin_9) // Set CS of
touch to 0

/*****************************************************************
**************
This is a basic write function. One of the arguments is register address
LCD_Reg and another one is register value LCD_RegValue. LCD_RegValue
will be written in register address by this function.
*****************************************************************
*************/
__inline void LCD_WriteReg(uint16_t LCD_Reg,uint16_t LCD_RegValue)
{
//select command register
TOUCH_nCS_H();
LCD_CS_L();
LCD_RS_L();
SPI_Communication((uint8_t)(LCD_Reg>>8));
SPI_Communication((uint8_t)(LCD_Reg));
LCD_RS_H();
SPI_Communication((uint8_t)(LCD_RegValue>>8));
SPI_Communication((uint8_t)(LCD_RegValue));
LCD_CS_H();
}
/*****************************************************************
**************
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. You can change the value in 0x003 register to switch
display mode. Please refer to Entry Mode 1 (R003h).
```

```
*****************************************************************
*************/
void LCD_Initialization(void)
{
LCD_Configuration();
LCD_RST_L();
delay_ms(10);
LCD_RST_H();
delay_ms(10);
LCD_WriteReg( 0x000, 0x0001 );
delay_ms(10);
/* Power supply settings */
LCD_WriteReg( 0x100, 0x0000 );
LCD_WriteReg( 0x101, 0x0000 );
LCD_WriteReg( 0x102, 0x3110 );
LCD_WriteReg( 0x103, 0xe200 );
LCD_WriteReg( 0x110, 0x009d );
LCD_WriteReg( 0x111, 0x0022 );
LCD_WriteReg( 0x100, 0x0120 );
delay_ms( 20 );
LCD_WriteReg( 0x100, 0x3120 );
delay_ms( 80 );
/* Display control */
LCD_WriteReg( 0x002, 0x0000 );
LCD_WriteReg( 0x001, 0x0100 );
LCD_WriteReg( 0x003, 0x1230 ); // High-speed write mode
// LCD_WriteReg( 0x003, 0x1030 ); // Low-speed write mode
LCD_WriteReg( 0x006, 0x0000 );
LCD_WriteReg( 0x007, 0x0101 );
LCD_WriteReg( 0x008, 0x0808 );
LCD_WriteReg( 0x009, 0x0000 );
LCD_WriteReg( 0x00b, 0x0000 );
LCD_WriteReg( 0x00c, 0x0000 );
LCD_WriteReg( 0x00d, 0x0018 );
LCD_WriteReg( 0x012, 0x0000 );
LCD_WriteReg( 0x013, 0x0000 );
LCD_WriteReg( 0x018, 0x0000 );
LCD_WriteReg( 0x019, 0x0000 );
LCD_WriteReg( 0x203, 0x0000 );
LCD_WriteReg( 0x204, 0x0000 );
LCD_WriteReg( 0x210, 0x0000 );
LCD_WriteReg( 0x211, 0x00ef );
LCD_WriteReg( 0x212, 0x0000 );
LCD_WriteReg( 0x213, 0x013f );
LCD_WriteReg( 0x214, 0x0000 );
LCD_WriteReg( 0x215, 0x0000 );
LCD_WriteReg( 0x216, 0x0000 );
LCD_WriteReg( 0x217, 0x0000 );
LCD_WriteReg( 0x300, 0x5343);
LCD_WriteReg( 0x301, 0x1021);
LCD_WriteReg( 0x302, 0x0003);
LCD_WriteReg( 0x303, 0x0011);
LCD_WriteReg( 0x304, 0x050a);
LCD_WriteReg( 0x305, 0x4342);
LCD_WriteReg( 0x306, 0x1100);
LCD_WriteReg( 0x307, 0x0003);
LCD_WriteReg( 0x308, 0x1201);
LCD_WriteReg( 0x309, 0x050a);

/* RAM access control */
LCD_WriteReg( 0x400, 0x4027 );
LCD_WriteReg( 0x401, 0x0000 );
LCD_WriteReg( 0x402, 0x0000 );
/* First screen drive position (1) */
LCD_WriteReg( 0x403, 0x013f );
/* First screen drive position (2) */
LCD_WriteReg( 0x404, 0x0000 );
LCD_WriteReg( 0x200, 0x0000 );
LCD_WriteReg( 0x201, 0x0000 );
LCD_WriteReg( 0x100, 0x7120 );
LCD_WriteReg( 0x007, 0x0103 );
delay_ms( 10 );
LCD_WriteReg( 0x007, 0x0113 );
LCD_Clear(Red);
}
/*****************************************************************
**************
This is LCD clear function. 0x210 and 0x212 are the begin-coordinate
address while 0x211 and 0x213 are the end-coordinate address.
*****************************************************************
*************/
void LCD_Clear(uint16_t Color)
{
  uint32_t index=0;
  TOUCH_nCS_H();
  LCD_WriteReg( 0x003, 0x1230 ); // High-speed write mode

  LCD_WriteReg(0x210,0x0000); // To set X of begin-coordinate
  LCD_WriteReg(0x212,0x0000); // To set Y of begin-coordinate
  LCD_WriteReg(0x211,0xEF); // To set X of end-coordinate
  LCD_WriteReg(0x213,0x013F); // To set Y of end-coordinate
  LCD_WriteReg(0x200,0x0000);
  LCD_WriteReg(0x201,0x0000);
  LCD_WriteIndex(0x202); //Start to write data to RAM
  LCD_CS_L();
  LCD_RS_H();
  for( index = 0; index < MAX_X * MAX_Y; index++ ) //MAX_X=320, MAX_Y=240
  {
    LCD_WriteData(Color);
  }
  LCD_CS_H();
  LCD_WriteReg( 0x003, 0x1030 ); //Low-speed write mode
}
int main(void)
{
  //System initialization
  LCD_Initialization(); // LCD initialization
  //LCD touch panel initialization
  //You can fill functions to calibrate touch screen.
  while (1)
    {
        //You can fill functions to show touch coordinate on the LCD.
    }
}
```