4.0inch Arduino Mega2560 8&16BIT Module MAR3953 User Manual

Product Description

The product is a 4.0-inch TFT LCD module with 800x480 resolution, 16BIT RGB 65K color display, internal drive IC NT35510, 8-bit and 16-bit parallel port communication, and 8-bit parallel port communication. The module includes LCD display, resistive touch screen, SD card slot and PCB backplane. It supports SD card expansion and can be directly plugged into the Arduino MEGA2560 development board. It can also be used on C51 and STM32 platforms.

Product Features

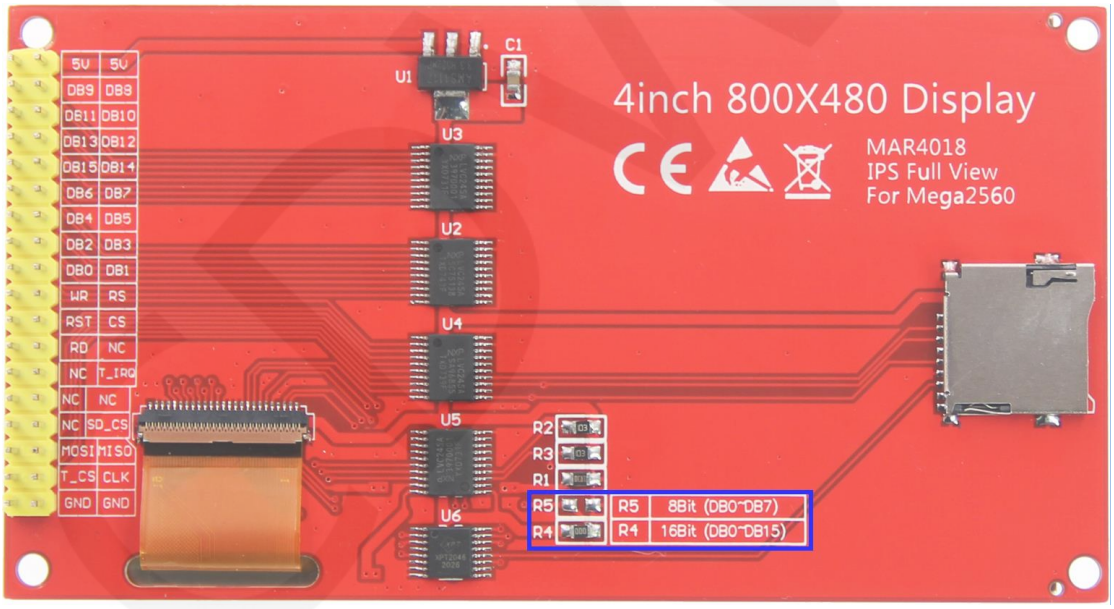
- 4.0-inch color screen, support 16BIT RGB 65K color display, display rich colors
- 800x480 resolution for clear display
- Supports 8-bit and 16-bit parallel bus transmission with fast transfer speed
- On-board 5V/3.3V level-shifting IC compatible with 5V/3.3V operating voltage
- Support Arduino Mage2560 for direct plug-in use
- Support for touch function
- Support SD card function extension
- Provide Arduino libraries and rich sample programs
- Available on C51 and STM32 platforms with a rich sample program
- Military-grade process standards, long-term stable work
- Provide underlying driver technical support

Product Parameters

Name	Description
Display Color	RGB 65K color
SKU	MAR4018
Screen Size	4.0(inch)
Type	TFT

Driver IC	NT35510
Resolution	800*480 (Pixel)
Module Interface	8Bit or 16Bit parallel interface
Active Area	86.40x51.84(mm)
Module PCB Size	58.65x108.48 (mm)
Back Light	8 chip HighLight white LEDs paralle
Operating Temperature	-10℃~60℃
Storage Temperature	-20℃~70℃
Operating Voltage	5V
Power Consumption	TBD
Product Weight	60g

Interface Description



Picture1. Module Pin silkscreen picture

Note:

1. **The module hardware supports 8-bit and 16-bit parallel port data bus mode switching (as shown by the blue box in Picture 1 above), as follows:**

   A. Solder R5 with 0Ω resistor or short circuit directly, and disconnect R4:

select 16-bit data bus mode (default), use DB0~DB15 data pin

B. Solder R4 with 0Ω resistor or short circuit directly, and disconnect R5: select 8-bit data bus mode, use DB0~DB7 data pin

Important Note:

1. **The following pin numbers 1~30 refer to the module pin number of our company with PCB backplane. If you purchase a bare screen, please refer to the pin definition of the bare screen specification, refer to the wiring according to the signal type instead of directly Wire according to the following module pin numbers. For example: LCD_CS is 20 feet on our module, which may be x feet on different sizes of bare screen.**
2. **About VCC supply voltage: If you purchase a module with PCB backplane, VCC/VDD power supply needs to be connected to 5V (module has integrated ultra low dropout 5V to 3.3V circuit), if you buy a bare screen LCD screen, remember to only connect 3.3V.**
3. **About backlight voltage: Modules with PCB backplane are connected to 3.3V, no need to manually access. If you are buying a bare screen, the LEDA is connected to 3.0V-3.3V, and the LEDKx can be grounded.**

Number	Module Pin	Pin description
1	5V	Power pin
2	5V	
3	DB8	Data bus high 8-bit pin
4	DB9	
5	DB10	
6	DB11	
7	DB12	
8	DB13	
9	DB14	

10	DB15	
11	DB7	Data bus low 8-bit pin
12	DB6	
13	DB5	
14	DB4	
15	DB3	
16	DB2	
17	DB1	
18	DB0	
19	RS	LCD register / data selection pin(high level:data, low level:register)
20	WR	LCD write control pin
21	CS	LCD chip select control pin(low level active)
22	RST	LCD reset control pin(low level active)
23	NC	Undefined, reserved
24	RD	LCD read control pin
25	T_IRQ	Touch screen interrupt control pin(low level active)
26	NC	Undefined, reserved
27	NC	
28	NC	
29	SD_CS	Extended reference: SD card select pin
30	NC	Undefined, reserved
31	MISO	SPI bus input pin
32	MOSI	SPI bus output pin
33	TP_CS	Touch screen chip select pin(low level active)
34	EX_CLK	SPI bus clock pin
35	GND	Power ground pin
36	GND	

Hardware Configuration

The LCD module hardware circuit comprises five parts: an LCD display control circuit, a level shift circuit, an SD card control circuit, a touch screen control circuit, and an 8-bit and 16-bit data bus mode switching circuit.

LCD display control circuit for controlling the pins of the LCD, including control pins and data transfer pins.

Level shifting circuit for 5V/3.3V conversion, making the module compatible with 3.3V/5V power supply.

SD card control circuit is used for SD card function expansion, controlling SD card identification, reading and writing.

The touch screen control circuit is used to control touch screen interrupt acquisition, data sampling, AD conversion, data transmission, and the like.

The 8-bit and 16-bit data bus mode switching circuits are used to switch the data bus type (8-bit mode and 16-bit mode). For details, see the red box in Picture 1 above or refer to the module circuit schematic.

working principle

1. Introduction to NT35510 Controller

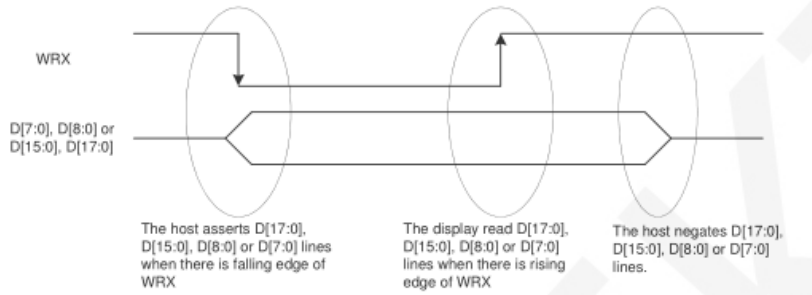
The NT35510 controller is a driver IC for TFT LCDs that supports multiple resolutions: 480*864, 480*854, 480*800, 480*720, 480*640, and 480*1024 (expanded memory required). It has a memory of 1244160 bytes and can support MDDI interface, MIPI interface, 16-bit/18-bit/24-bit RGB interface, 8-bit/16-bit/18-bit/24-bit parallel port, SPI and I2C interface. It supports 8, 65K, 262K and 16.7M RGB color display, display color is very rich, while supporting rotating display and scroll display and video playback, display a variety of ways.

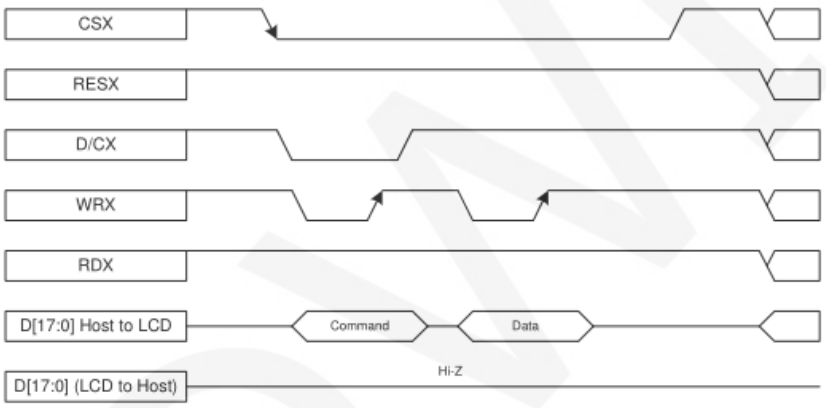
This module uses a 16-bit parallel port to transmit data and 16bit (RGB565) to control a pixel display, so it can display up to 65K colors per pixel. The pixel address setting is performed in the order of the rows and columns, and the incrementing and decreasing

direction is determined by the scanning mode. The NT35510 display method is performed by setting the address and then setting the color value.

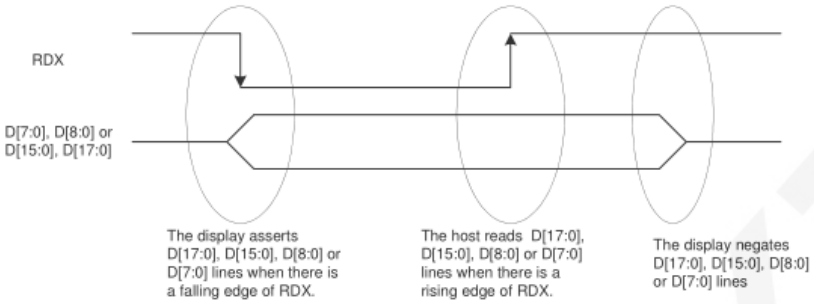
2. Introduction to parallel port communication

The parallel port communication write mode timing is as shown below:

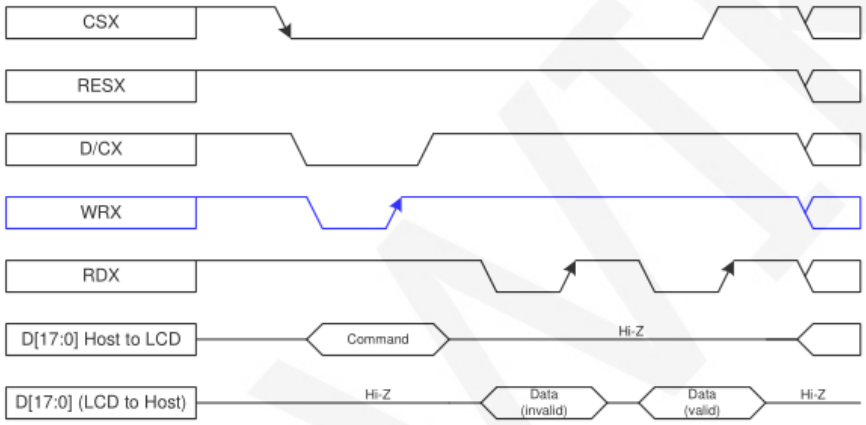




The timing of the parallel port communication read mode is shown in the figure below:



Note: RDX is an unsynchronized signal (It can be stopped).



CSX is a chip select signal for enabling and disabling parallel port communication, active low

RESX is an external reset signal, active low

D/CX is the data or command selection signal, 1-write data or command parameters, 0-write command

WRX is a write data control signal

RDX is a read data control signal

D[X:0] is a parallel port data bit, which has four types: 8-bit, 9-bit, 16-bit, and 18-bit.

When performing a write operation, on the basis of the reset, first set the data or command selection signal, then pull the chip select signal low, then input the content to be written from the host, and then pull the write data control signal low. When pulled high, data is written to the LCD control IC on the rising edge of the write control signal. the chip select signal is pulled high and a data write operation is completed.

When entering the read operation, on the basis of the reset, first pull the chip select signal low, then pull the data or command select signal high, then pull the read data control signal low, and then read the data from the LCD control IC. And then The read data control signal is pulled high, and the data is read out on the rising edge of the read data control signal. Finally, the chip select signal is pulled high, and a data read operation is completed.

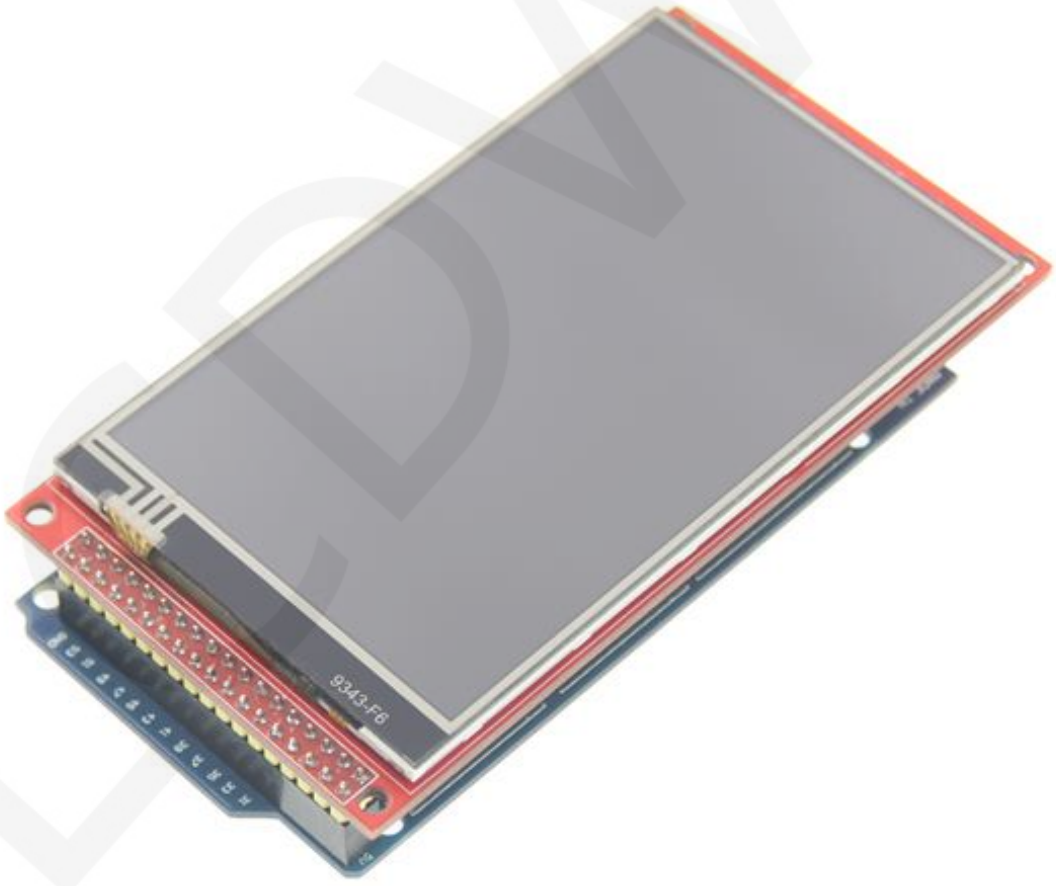
Instructions for use

1. Arduino instructions

Wiring instructions:

See the interface description for pin assignments.

This module can be directly inserted into the Arduino UNO and Mega2560, no need to manually wire, as shown below:



Mega2560 directly inserted picture

Arduino MEGA2560 microcontroller test program directly insert instructions			
Number	Module Pin	Corresponding to MEGA2560 development board direct plug pins	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	not used	22
4	DB9		23
5	DB10		24
6	DB11		25
7	DB12		26
8	DB13		27
9	DB14		28
10	DB15		29
11	DB7	30	
12	DB6	31	
13	DB5	32	
14	DB4	33	
15	DB3	34	
16	DB2	35	
17	DB1	36	
18	DB0	37	
19	RS	38	
20	WR	39	
21	CS	40	
22	RST	41	
23	NC	not used	
24	RD	43	
25	T_IRQ	44	
26	NC	not used	
27	NC		

28	NC	
29	SD_CS	48
30	NC	not used
31	MISO	50
32	MOSI	51
33	TP_CS	53
34	EX_CLK	52
35	GND	GND
36	GND	

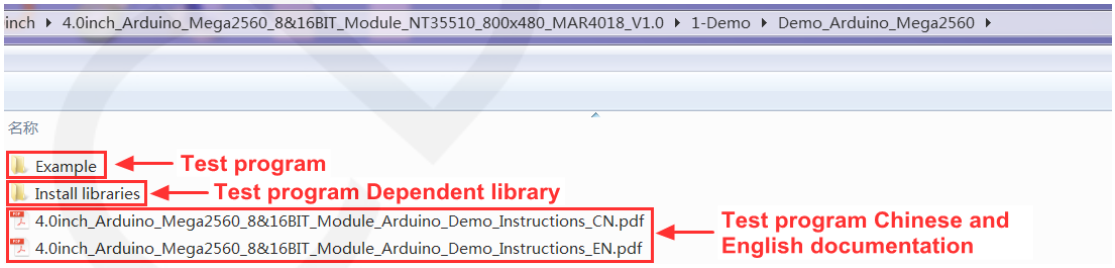
Operating Steps：

A. Insert the LCD module directly into the Arduino MCU according to the above wiring instructions, and power on;

B. Copy the dependent libraries in the Install libraries directory of the test package to the libraries folder of the Arduino project directory (if you do not need to depend on the libraries, you do not need to copy them);

C. Open the directory where the Arduino test program is located and select the example you want to test, as shown below:

(Please refer to the test program description document in the test package for the test program description)



D. Open the selected sample project, compile and download.

The specific operation methods for the Arduino test program relying on library copy, compile and download are as follows:

http://www.lcdwiki.com/res/PublicFile/Arduino_IDE_Use_Illustration_EN.pdf

E. If the LCD module displays characters and graphics normally, the program runs

Successfully;

2. C51 instructions

Wiring instructions:

See the interface description for pin assignments.

STC89C52RC microcontroller test program wiring instructions			
Number	**Module Pin**	**Corresponding to STC89 development board wiring pin**	
		8-bit mode	**16-bit mode**
1	**5V**	5V	
2	**5V**		
3	**DB8**	No need to connect	P20
4	**DB9**		P21
5	**DB10**		P22
6	**DB11**		P23
7	**DB12**		P24
8	**DB13**		P25
9	**DB14**		P26
10	**DB15**		P27
11	**DB7**	P37	
12	**DB6**	P36	
13	**DB5**	P35	
14	**DB4**	P34	
15	**DB3**	P33	
16	**DB2**	P32	
17	**DB1**	P31	
18	**DB0**	P30	
19	**RS**	P12	
20	**WR**	P11	
21	**CS**	P13	
22	**RST**	P14	

23	NC	No need to connect
24	RD	P10
25	T_IRQ	No need to connect (cannot test touch)
26	NC	No need to connect
27	NC	
28	NC	
29	SD_CS	No need to connect
30	NC	No need to connect
31	MISO	No need to connect (cannot test touch)
32	MOSI	No need to connect (cannot test touch)
33	CLK	No need to connect (cannot test touch)
34	T_CS	No need to connect (cannot test touch)
35	GND	GND
36	GND	

STC12C5A60S2 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to STC12 development board wiring pin	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	P20
4	DB9		P21
5	DB10		P22
6	DB11		P23
7	DB12		P24
8	DB13		P25
9	DB14		P26
10	DB15		P27
11	DB7	P07	

12	DB6	P06
13	DB5	P05
14	DB4	P04
15	DB3	P03
16	DB2	P02
17	DB1	P01
18	DB0	P00
19	RS	P12
20	WR	P11
21	CS	P13
22	RST	P33
23	NC	No need to connect
24	RD	P10
25	T_IRQ	P40
26	NC	No need to connect
27	NC	
28	NC	
29	SD_CS	No need to connect
30	NC	No need to connect
31	MISO	P35
32	MOSI	P34
33	CLK	P36
34	T_CS	P37
35	GND	GND
36	GND	

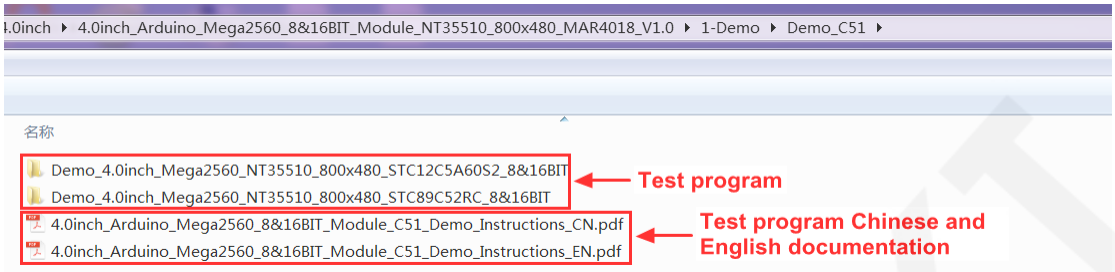
Operating Steps：

A. Connect the LCD module and the C51 MCU according to the above wiring instructions, and power on;

B. Open the directory where the C51 test program is located and select the example

to be tested, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected test program project, compile and download;

detailed description of the C51 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/C51_Keil%26stc-isp_Use_Illustration_EN.pdf

D. If the LCD module displays characters and graphics normally, the program runs successfully

3. STM32 instructions

Wiring instructions:

See the interface description for pin assignments.

STM32F103RCT6 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to Mini STM32 development board wiring pin(using FSMC bus)	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	PB8
4	DB9		PB9
5	DB10		PB10
6	DB11		PB11
7	DB12		PB12
8	DB13		PB13
9	DB14		PB14

10	**DB15**		PB15
11	**DB7**	PB7	
12	**DB6**	PB6	
13	**DB5**	PB5	
14	**DB4**	PB4	
15	**DB3**	PB3	
16	**DB2**	PB2	
17	**DB1**	PB1	
18	**DB0**	PB0	
19	**RS**	PC8	
20	**WR**	PC7	
21	**CS**	PC9	
22	**RST**	PC4	
23	**NC**	No need to connect	
24	**RD**	PC6	
25	**T_IRQ**	PC1	
26	**NC**	No need to connect	
27	**NC**		
28	**NC**		
29	**SD_CS**	No need to connect	
30	**NC**	No need to connect	
31	**MISO**	PC2	
32	**MOSI**	PC3	
33	**CLK**	PC0	
34	**T_CS**	PC13	
35	**GND**	GND	
36	**GND**		

STM32F103ZET6 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to Elite STM32 development board wiring pin(using FSMC bus)	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	PE11
4	DB9		PE12
5	DB10		PE13
6	DB11		PE14
7	DB12		PE15
8	DB13		PD8
9	DB14		PD9
10	DB15		PD10
11	DB7	PE10	
12	DB6	PE9	
13	DB5	PE8	
14	DB4	PE7	
15	DB3	PD1	
16	DB2	PD0	
17	DB1	PD15	
18	DB0	PD14	
19	RS	PG0	
20	WR	PD5	
21	CS	PG12	
22	RST	MCU Reset Pin	
23	NC	No need to connect	
24	RD	PD4	
25	T_IRQ	PF10	
26	NC	No need to connect	

27	NC	
28	NC	
29	SD_CS	No need to connect
30	NC	No need to connect
31	MISO	PB2
32	MOSI	PF9
33	CLK	PB1
34	T_CS	PF11
35	GND	GND
36	GND	

STM32F407VGT6 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to STM32F407VxT6 development board wiring pin(using FSMC bus)	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	PE11
4	DB9		PE12
5	DB10		PE13
6	DB11		PE14
7	DB12		PE15
8	DB13		PD8
9	DB14		PD9
10	DB15		PD10
11	DB7	PE10	
12	DB6	PE9	
13	DB5	PE8	
14	DB4	PE7	
15	DB3	PD1	
16	DB2	PD0	

17	DB1	PD15
18	DB0	PD14
19	RS	PD11
20	WR	PD5
21	CS	PD7
22	RST	MCU Reset Pin
23	NC	No need to connect
24	RD	PD4
25	T_IRQ	PB1
26	NC	No need to connect
27	NC	
28	NC	
29	SD_CS	No need to connect
30	NC	No need to connect
31	MISO	PB2
32	MOSI	PC4
33	CLK	PB0
34	T_CS	PC13
35	GND	GND
36	GND	

STM32F407ZGT6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to Explorer STM32 development board wiring pin(using FSMC bus)	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	PE11
4	DB9		PE12
5	DB10		PE13
6	DB11		PE14

7	**DB12**		PE15
8	**DB13**		PD8
9	**DB14**		PD9
10	**DB15**		PD10
11	**DB7**	PE10	
12	**DB6**	PE9	
13	**DB5**	PE8	
14	**DB4**	PE7	
15	**DB3**	PD1	
16	**DB2**	PD0	
17	**DB1**	PD15	
18	**DB0**	PD14	
19	**RS**	PF12	
20	**WR**	PD5	
21	**CS**	PG12	
22	**RST**	MCU Reset Pin	
23	**NC**	No need to connect	
24	**RD**	PD4	
25	**T_IRQ**	PB1	
26	**NC**	No need to connect	
27	**NC**		
28	**NC**		
29	**SD_CS**	No need to connect	
30	**NC**	No need to connect	
31	**MISO**	PB2	
32	**MOSI**	PF11	
33	**CLK**	PB0	
34	**T_CS**	PC13	
35	**GND**	GND	
36	**GND**		

STM32F429IGT6, STM32F767IGT6, STM32H743IIT6 microcontroller test program wiring instructions			
Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin(using FSMC bus)	
		8-bit mode	16-bit mode
1	5V	5V	
2	5V		
3	DB8	No need to connect	PE11
4	DB9		PE12
5	DB10		PE13
6	DB11		PE14
7	DB12		PE15
8	DB13		PD8
9	DB14		PD9
10	DB15		PD10
11	DB7	PE10	
12	DB6	PE9	
13	DB5	PE8	
14	DB4	PE7	
15	DB3	PD1	
16	DB2	PD0	
17	DB1	PD15	
18	DB0	PD14	
19	RS	PD13	
20	WR	PD5	
21	CS	PD7	
22	RST	MCU Reset Pin	
23	NC	No need to connect	
24	RD	PD4	
25	T_IRQ	PH7	
26	NC	No need to connect	

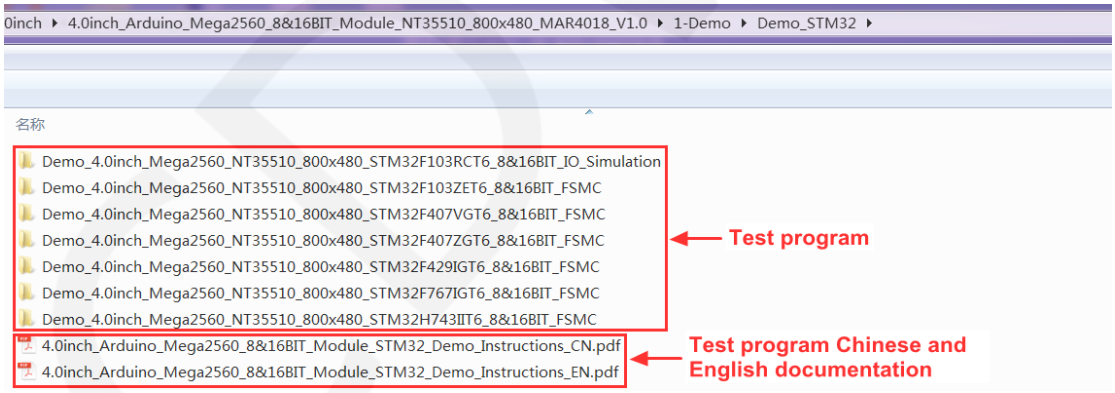
27	NC	
28	NC	
29	SD_CS	No need to connect
30	NC	No need to connect
31	MISO	PG3
32	MOSI	PI3
33	CLK	PH6
34	T_CS	PI8
35	GND	GND
36	GND	

Operating Steps：

A. Connect the LCD module and the STM32 MCU according to the above wiring instructions, and power on;

B. Open the directory where the STM32 test program is located and select the example to be tested, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected test program project, compile and download;

detailed description of the STM32 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_EN.pdf

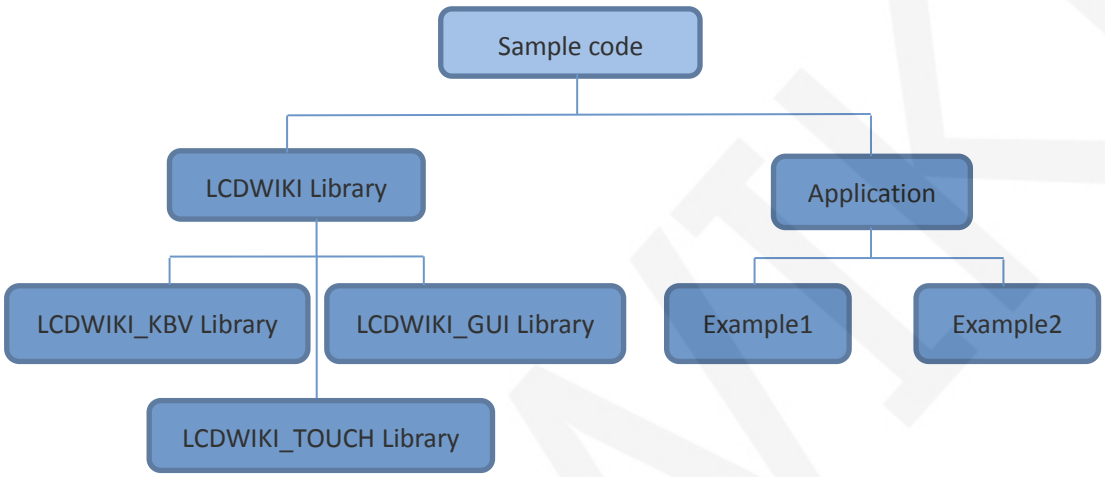
D. If the LCD module displays characters and graphics normally, the program runs successfully；

Software Description

1. Code Architecture

A. Arduino code architecture description

The code architecture is shown below:



Arduino's test program code consists of two parts: the LCDWIKI library and application code.

The LCDWIKI library contains three parts: LCDWIKI_KBV library, LCDWIKI_GUI library, and LCDWIKI_TOUCH library.

The application contains several test examples, each with different test content;

LCDWIKI_KBV is the underlying library, which is associated with hardware. It is mainly responsible for operating registers, including hardware module initialization, data and command transmission, pixel coordinates and color settings, display mode configuration, etc;

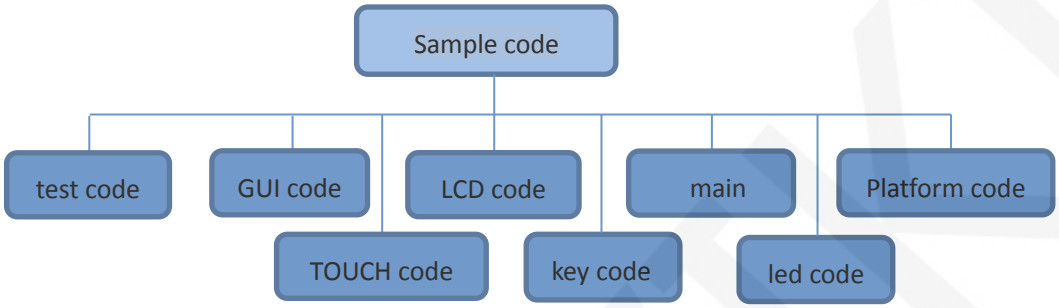
LCDWIKI_GUI is the middle layer library, which is responsible for drawing graphics and displaying characters using the API provided by the underlying library;

LCDWIKI_TOUCH is the underlying library of touch screens, mainly responsible for touch interrupt detection, touch data sampling and AD conversion, and touch data transmission.

The application is to use the API provided by the LCDWIKI library to write some test examples and implement Some aspect of the test function;

B. C51 and STM32 code architecture description

The code architecture is shown below:



The Demo API code for the main program runtime is included in the test code;

LCD initialization and related bin parallel port write data operations are included in the LCD code;

Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code;

The main function implements the application to run;

Platform code varies by platform;

Touch screen related operations are included in the touch code;

The key processing related code is included in the key code (the C51 platform does not have a button processing code);

The code related to the led configuration operation is included in the led code;

2. GPIO definition description

A. Arduino test program GPIO definition description

The module is plugged into the Arduino Mage2560, so it is not allowed to modify the GPIO port definition.

B. C51 test program GPIO definition description

The C51 test program GPIO definition is placed in the lcd.h file as shown below(Take the STC12C5A60S2 microcontroller test program as an example):

```
//IO connect
#define  LCD_DataPortH P2     //High 8-bit data port,Only use the up
#define  LCD_DataPortL P0     //Low 8-bit data port,The lower 8 bits
sbit LCD_RS = P1^2;      //Data/command switching
sbit LCD_WR = P1^1;     //Write control
//sbit LCD_RD = P1^0;         //read control
sbit LCD_CS = P1^3;    //Chip Select
sbit LCD_RESET = P3^3;         //reset
//sbit LCD_BL=P3^2;    //Backlight control,If you do not need control
```

Parallel pin definition needs to select the whole set of GPIO port groups, such as P0, P2, etc., so that when transferring data, the operation is convenient.Other pins can be defined as any free GPIO.

The touch screen GPIO port definition is placed in touch.h, as shown below (only 12C5A60S2 can test touch)

```
//IO连接
sfr    P4     = 0xC0;
sbit DCLK     =     P3^6;
sbit TCS          =     P3^7;
sbit DIN        =     P3^4;
sbit DOUT       =     P3^5;
sbit Penirq     =     P4^0;    //检测触摸屏响应信号
```

The GPIO definition of the touch screen can be modified and can be defined as any other free GPIO.

If the microcontroller does not have a P4 GPIO group, you can define penirq as another GPIO.

C. STM32 test program GPIO definition description

STM32 FSMC test program lcd screen GPIO is defined by FSMC bus. The related definition method can refer to FSMC bus description data. Its GPIO definition is placed in lcd.h file as shown below (take STM32F103ZET6 microcontroller FSMC test program as an example):

```
//////////////////////////////////////////////////////////////////////////////
//-----------------LCD端口定义----------------
#define LED      0      //背光控制引脚        PB0

//QDtech全系列模块采用了三极管控制背光亮灭，用户也可以接PWM调节背光亮度
#define LCD_LED PBout(LED) //LCD背光
//如果使用官方库函数定义下列底层，速度将会下降到14帧每秒，建议采用我司推荐方法
//以下IO定义直接操作寄存器，快速IO操作，刷屏速率可以达到28帧每秒！

//LCD地址结构体
typedef struct
{
  vu16 LCD_REG;
  vu16 LCD_RAM;
} LCD_TypeDef;

//使用NOR/SRAM的 Bank1.sector4,地址位HADDR[27,26]=11 A10作为数据命令区分线
//注意设置时STM32内部会右移一位对其!
#define LCD_BASE        ((u32)(0x6C000000 | 0x000007FE))
#define LCD             ((LCD_TypeDef *) LCD_BASE)
```

STM32 IO simulation test program lcd screen GPIO definition is placed in the lcd.h file, as shown below (take STM32F103RCT6 microcontroller IO simulation test program as an example):

```
//////////////////////////////////////////////////////////////////////////////
//-----------------LCD端口定义----------------
#define GPIO_TYPE  GPIOC  //GPIO组类型
#define LED      10         //背光控制引脚      PC10
#define LCD_CS   9        //片选引脚          PC9
#define LCD_RS   8        //寄存器/数据选择引脚 PC8
#define LCD_RST  4        //复位引脚          PC4
#define LCD_WR   7        //写引脚            PC7
#define LCD_RD   6        //读引脚            PC6

//PB0~15,作为数据线
//注意：如果使用8位模式数据总线，则液晶屏的数据高8位是接到MCU的高8位总
//举例：如果接8位模式则本示例接线为液晶屏DB10-DB17对应接至单片机GPIOB_
//举例：如果是16位模式：DB0-DB7分别接GPIOB_Pin0-GPIOB_Pin7,DB10-DB17对
#define DATAOUT(x) GPIOB->ODR=x; //数据输出
#define DATAIN     GPIOB->IDR;   //数据输入
```

Data parallel port pin definition needs to select a complete set of GPIO port groups, such as PB, when transferring data, it is convenient to operate.Other pins can be defined as any free GPIO.

The GPIO definition related to the STM32 touch screen is placed in the touch.h file as shown below (take the STM32F103RCT6 microcontroller IO simulation test program as an example):

```
//与触摸屏芯片连接引脚
//与触摸屏芯片连接引脚
#define PEN   PCin(1)     //PC1   INT
#define DOUT  PCin(2)      //PC2   MISO      PC2--PB14
#define TDIN  PCout(3)     //PC3   MOSI      PC3--PB15
#define TCLK  PCout(0)     //PC0   SCLK    PC0--PB13
#define TCS   PCout(13)    //PC13  CS
```

If you use the IO simulation test program, you can modify the values in the parentheses. All pin definitions can be modified and can be defined as any other free GPIO.

If the FSMC test program is used, the touch screen GPIO cannot be modified because the GPIO pins on the development board are fixed by the in-line connection.

3. Parallel port communication code implementation

A. Arduino test program parallel port communication code implementation

If the 8-bit mode related code is used in the mcu_8bit_magic.h file of the LCDWIKI_KBV library, as shown below:

```
#define CMASK    0xFF
#define write8(d) {\
        PORTC = d;WR_STROBE;}
#define read8(dst) {\
        RD_ACTIVE; DELAY7; \
        dst = PINC;RD_IDLE;}
#define setWriteDir() {DDRC |= CMASK;}
#define setReadDir()  {DDRC &= ~CMASK;}
```

If the 16-bit mode related code is used in the mcu_16bit_magic.h file of the LCDWIKI_KBV library, as shown below:

```
// Data write strobe, ~2 instructions and always inline
#define WR_STROBE { WR_ACTIVE; WR_IDLE; }
#define RD_STROBE {RD_IDLE; RD_ACTIVE;RD_ACTIVE;RD_ACTIVE;}
#define write16(x) { write_16(x) }
#define read16(dst) { read_16(dst) }
#define writeCmd8(x){ CD_COMMAND; write8(x); CD_DATA;  }
#define writeData8(x){  write8(x) }
#define writeCmd16(x){ CD_COMMAND; write16(x); CD_DATA; }
#define writeData16(x){ write16(x) }

#define write_16(x)    { PORTA = (x) >> 8; PORTC = x; WR_STROBE;}
#define write8(x)     { PORTC = x; WR_STROBE;}
```

B. C51 test program parallel port communication code implementation

The relevant code is implemented in the LCD.c file as shown below:

```
void LCD_write(u8 HVAL,u8 LVAL)
{
  LCD_CS = 0;
  LCD_WR = 0;
  LCD_DataPortH = HVAL;
  LCD_DataPortL = LVAL;
  LCD_WR = 1;
  LCD_CS = 1;
}

u16 LCD_read(void)
{
  u16 d;
  LCD_CS = 0;
  LCD_RD = 0;
  delay_us(1); //delay 1 us
  d = LCD_DataPortH;
  d = (d<<8)|LCD_DataPortL;
  LCD_RD = 1;
  LCD_CS = 1;
  return d;
}
```

Implemented 8-bit and 16-bit commands and 8-bit and 16-bit data write and read

C. STM32 test program parallel port communication code implementation

The STM32 test program parallel port communication code is implemented in the LCD.c file.The FSMC test program is implemented as shown below:

```
u16 LCD_read(void)
{
  vu16 data;  //防止被优化
  data=LCD->LCD_RAM;
  return data;
}

/*****************************************************************************
 * @name       :void LCD_WR_REG(u16 data)
 * @date       :2018-08-09
 * @function   :Write an 16-bit command to the LCD screen
 * @parameters :data:Command value to be written
 * @retvalue   :None
******************************************************************************/
void LCD_WR_REG(u16 data)
{
  LCD->LCD_REG=data; //写入要写的寄存器序号
}

/*****************************************************************************
 * @name       :void LCD_WR_DATA(u16 data)
 * @date       :2018-08-09
 * @function   :Write an 16-bit data to the LCD screen
 * @parameters :data:data value to be written
 * @retvalue   :None
******************************************************************************/
void LCD_WR_DATA(u16 data)
{
  LCD->LCD_RAM=data; //写入要写的数据
}
```

The IO simulation test program is implemented as shown below:

```
void LCD_write(u16 VAL)
{
  LCD_CS_CLR;
  DATAOUT(VAL);
  LCD_WR_CLR;
  LCD_WR_SET;
  LCD_CS_SET;
}

u16 LCD_read(void)
{
  u16 data;
  LCD_CS_CLR;
  LCD_RD_CLR;
  delay_us(1);//延时1us
  data = DATAIN;
  LCD_RD_SET;
  LCD_CS_SET;
  return data;
}
```

The IO analog test program implements 8- and 16-bit commands and 8- and 16-bit data write and read.

The FSMC test program implements 16-bit commands and 16-bit data write and read.

4. touch screen calibration instructions

A. Arduino test program touch screen calibration instructions

Arduino touch screen calibration needs to run the touch_screen_calibration program first, and then calibrate according to the prompts. After the calibration is passed, the calibration parameters displayed on the screen need to be written into the cali_para.h file of the LCDWIKI_TOUCH library, as shown below:

```
#define XFAC       1307 //663 //852
#define XOFFSET    -30//(-13) //(-14)
#define YFAC       2062//894 //1284
#define YOFFSET    -17//(-30)
```

B. C51 test program touch screen calibration instructions

The C51 touch screen calibration needs to execute the Touch_Adjust test item (only available in the STC12C5A60S2 test program), as shown below:

```
//循环进行各项测试
while(1)
{
  main_test();     //测试主界面
  Test_Color();      //简单刷屏填充测试
  Test_FillRec();    //GUI矩形绘图测试
  Test_Circle();     //GUI画圆测试
  Test_Triangle();    //GUI三角形填充测试
  English_Font_test();//英文字体示例测试
  Chinese_Font_test();//中文字体示例测试
  Pic_test();      //图片显示示例测试
  Rotate_Test();
//不使用触摸或者模块本身不带触摸，请屏蔽下面触摸屏测试
  Touch_Test();    //触摸屏手写测试
//需要触摸校准时，请将触摸手写测试屏蔽，将下面触摸校准测试项打开
//  Touch_Adjust();  //触摸校准
}
```

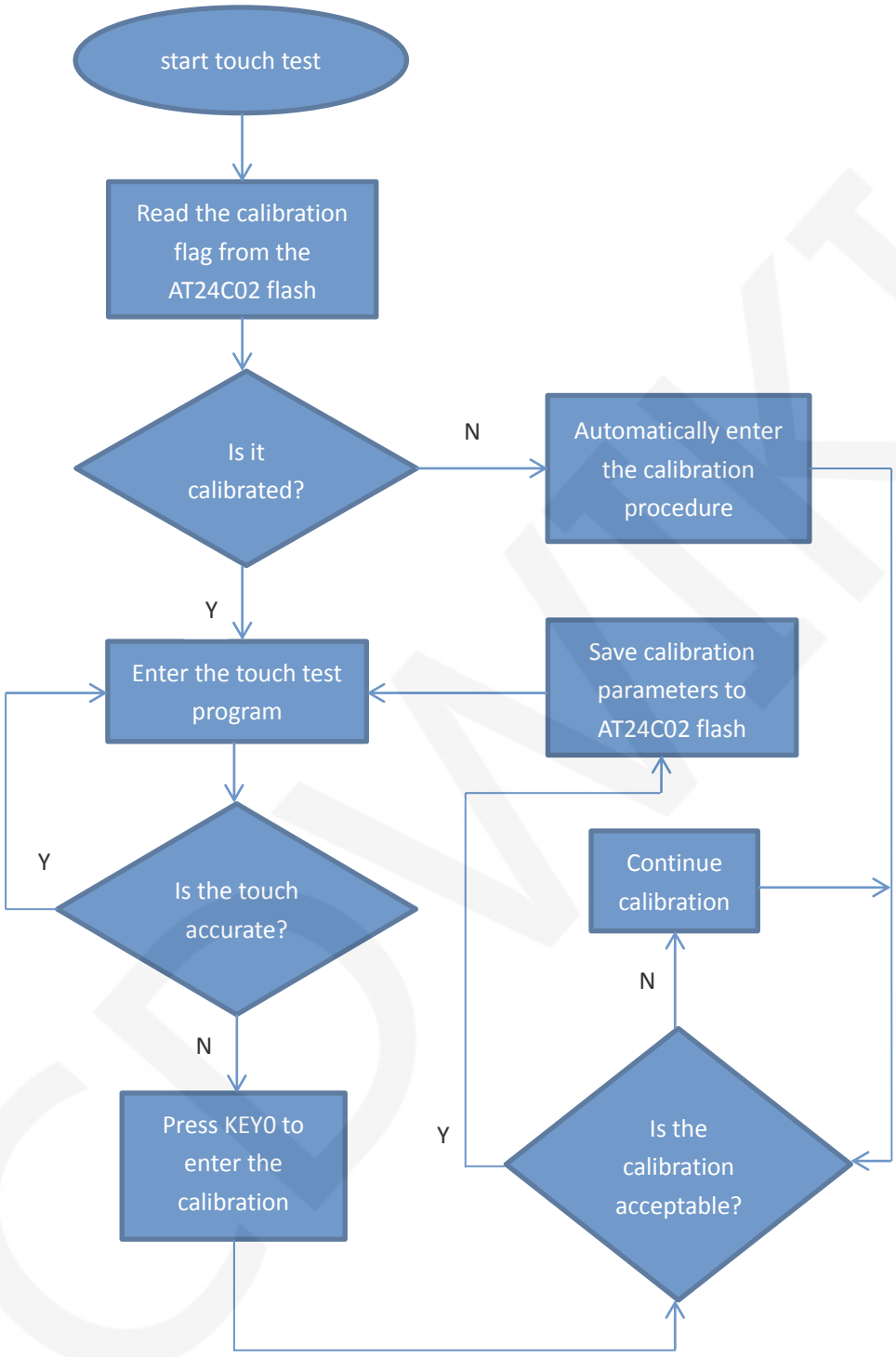
After the touch calibration is passed, you need to save the calibration parameters displayed on the screen in the touch.c file, as shown below:

```
//***因触摸屏批次不同等原因，默认的校准参数值可能会引起触摸
u16 vx=11738,vy=7736;   //比例因子，此值除以1000之后表示多少
u16 chx=3905,chy=246;//默认像素点坐标为0时的AD起始值
//***因触摸屏批次不同等原因，默认的校准参数值可能会引起触摸
```

C. STM32 test program touch screen calibration instructions

The STM32 touch screen calibration program automatically recognizes whether calibration is required or manually enters calibration by pressing a button.

It is included in the touch screen test item. The calibration mark and calibration parameters are saved in the AT24C02 flash. If necessary, read from the flash. The calibration process is as shown below:


start touch test
Read the calibration flag from the AT24C02 flash
Is it calibrated?
N
Automatically enter the calibration procedure
Y
Enter the touch test program
Save calibration parameters to AT24C02 flash
Y
Is the touch accurate?
Continue calibration
N
N
Press KEY0 to enter the calibration
Y
Is the calibration acceptable?

Common software

This set of test examples requires the display of Chinese and English, symbols and pictures, so the modulo software is used. There are two types of modulo software: Image2Lcd and PCtoLCD2002. Here is only the setting of the modulo software for the test program.

The **PCtoLCD2002** modulo software settings are as follows:

Dot matrix format select Dark code

the modulo mode select the progressive mode

Take the model to choose the direction (high position first)

Output number system selects hexadecimal number

Custom format selection C51 format

The specific setting method is as follows:

http://www.lcdwiki.com/Chinese_and_English_display_modulo_settings

Image2Lcd modulo software settings are shown below:

The Image2Lcd software needs to be set to horizontal, left to right, top to bottom, and low position to the front scan mode.