南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB10 Resistive touch screen

2024 Fall
wangq9@mail.sustech.edu.cn

CONTENTS

1 Principle Description

2 Control principle Description

3 How to Program

4 Practice

01

Principle Description

1. Principle Description

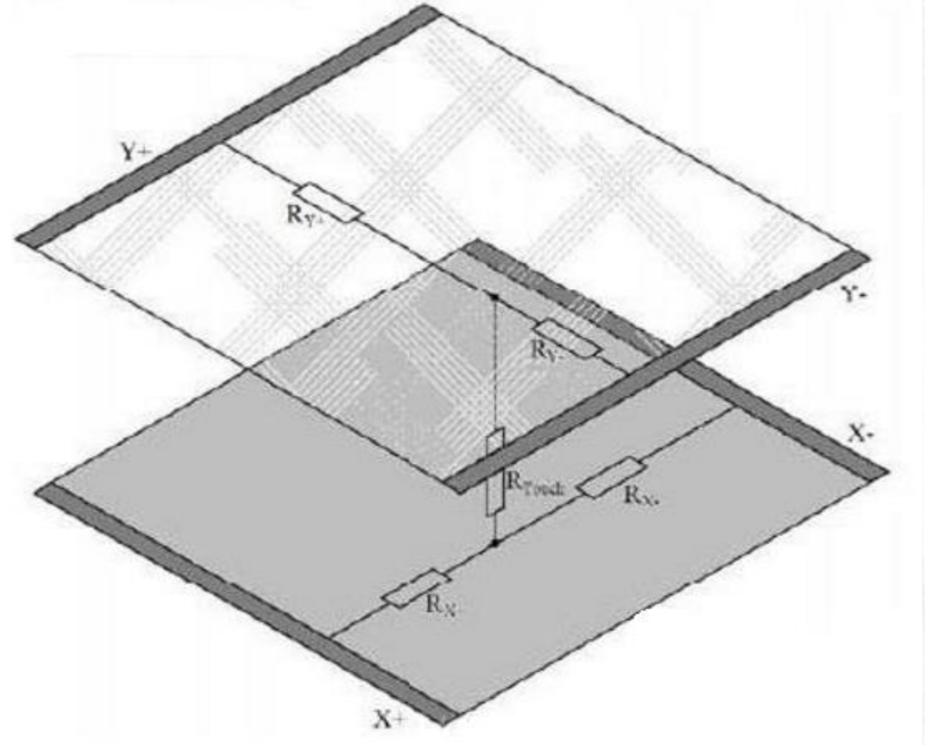
-- Classify of touch screen

- According to the working principle of the touch screen and the media used to transmit information
  - Resistive: accurate positioning, single touch (used by Mini development board)
  - Capacitive: multi touch
  - Infrared type: low price, distortion under curved surface
  - Surface acoustic wave type: Solves various shortcomings, but if there are water drops and dust on the screen surface, the touch screen will become dull

1. Principle Description

-- Working principle of touch screen

- The resistive screen is divided into two layers (layer X and layer Y), and the middle is separated by an isolation pivot.
- The adjacent surfaces of the upper and lower layers are coated with ITO coating, which is conductive.
- The circuits are connected on both sides of layer X and layer Y respectively.
- The resistance of X- to X+ on layer X and Y- to Y+ on layer Y is uniformly distributed.

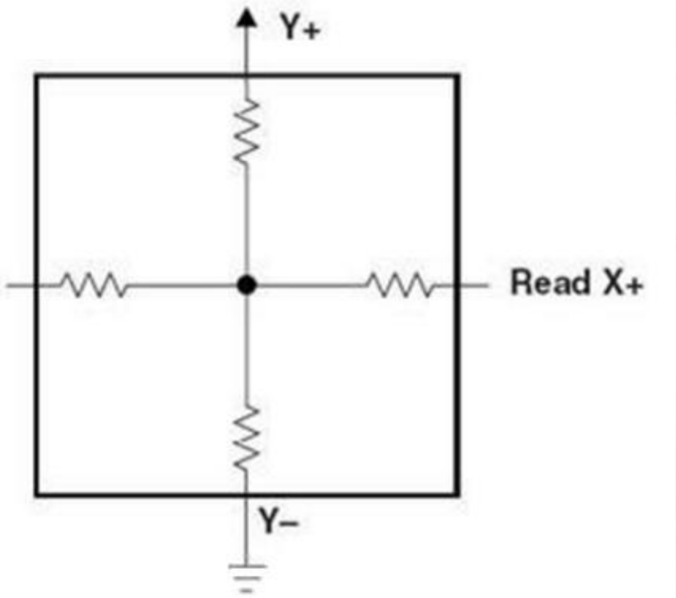


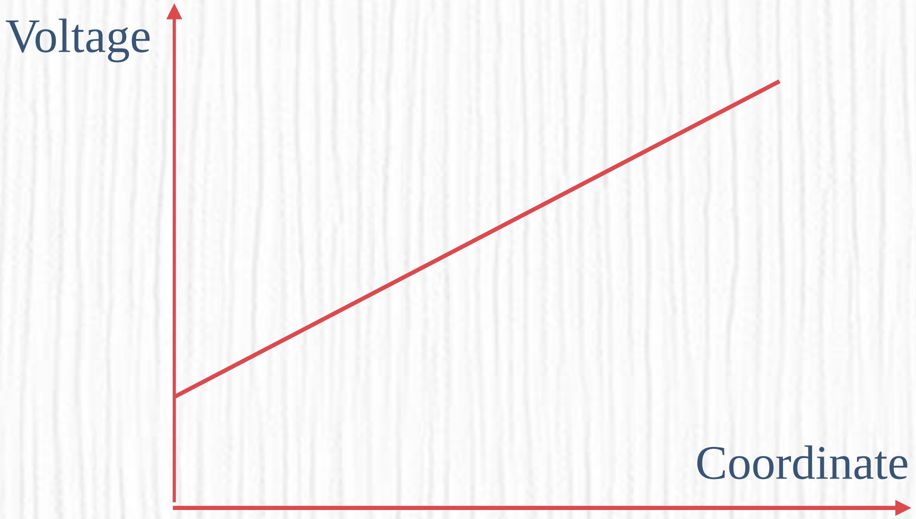
Schematic diagram

1. Principle Description
-- Measuring touch points

- When calculating the Y coordinate, apply the driving voltage V to the Y+ electrode, and Y - ground. The chip measures the voltage at the contact point through X+.

- Since ITO layer conducts electricity uniformly, the ratio of contact voltage to V voltage is equal to the ratio of contact Y coordinate to screen height. Therefore, the relationship equation between voltage and distance can be obtained during actual measurement.



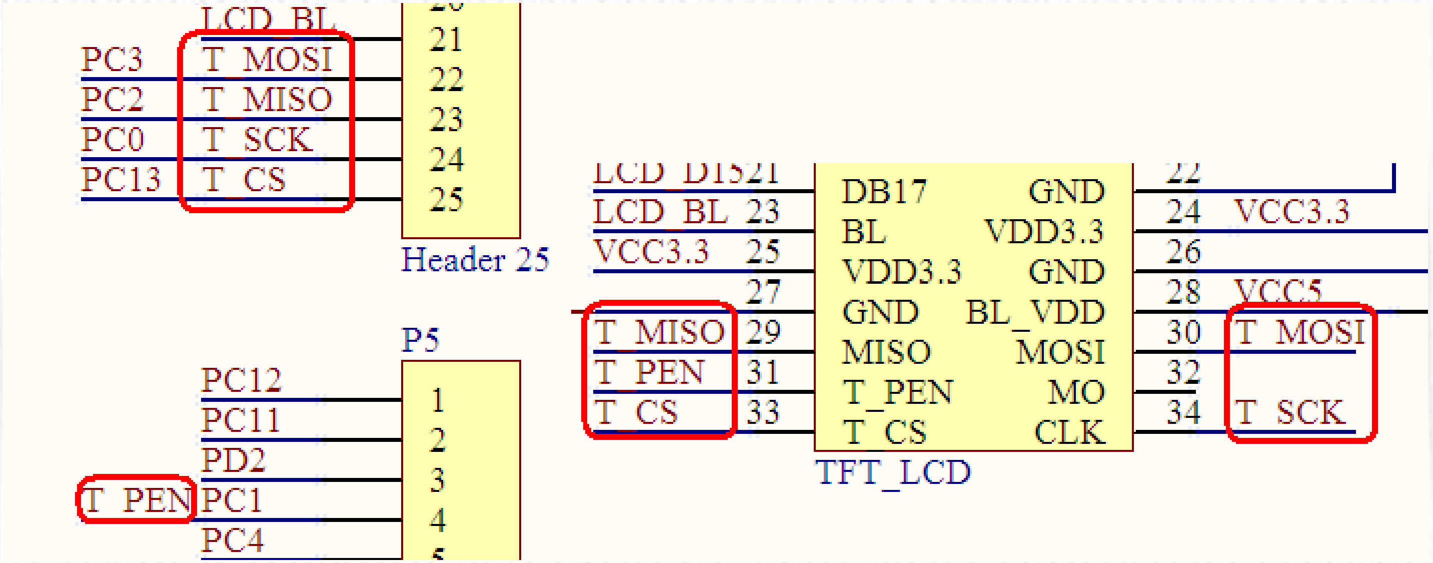


02

Control Principle Description

2. Control Principle Description
-- STM32 touch screen

- ALIENTEK TFTLCD selects 4-wire resistive touch screen.
- The built-in touch screen control chip of TFTLCD is XPT2046.
- T_CS: chip selection
- T_CLK: clock signal
- T_MISO: data reading
- T_MOSI: data output
- T_PEN: interrupt output



Circuit connection diagram between touch screen and STM32

2. Control Principle Description -- XPT2046 chip

- 4-wire touch screen controller.
- 12 bit resolution, 125KHz conversion rate progressive approximation A/D converter.
- Support low-voltage I/O interface from 1.5V to 5.25V.
- The pressed screen position can be found by performing two A/D conversions.
- It can also measure the pressure applied to the touch screen.
- The internal 2.5V reference voltage can be used as auxiliary input, temperature measurement and battery monitoring mode. The voltage range of battery monitoring can be from 0V to 6V.
- A temperature sensor is integrated into the chip.

2. Control Principle Description -- SPI protocol

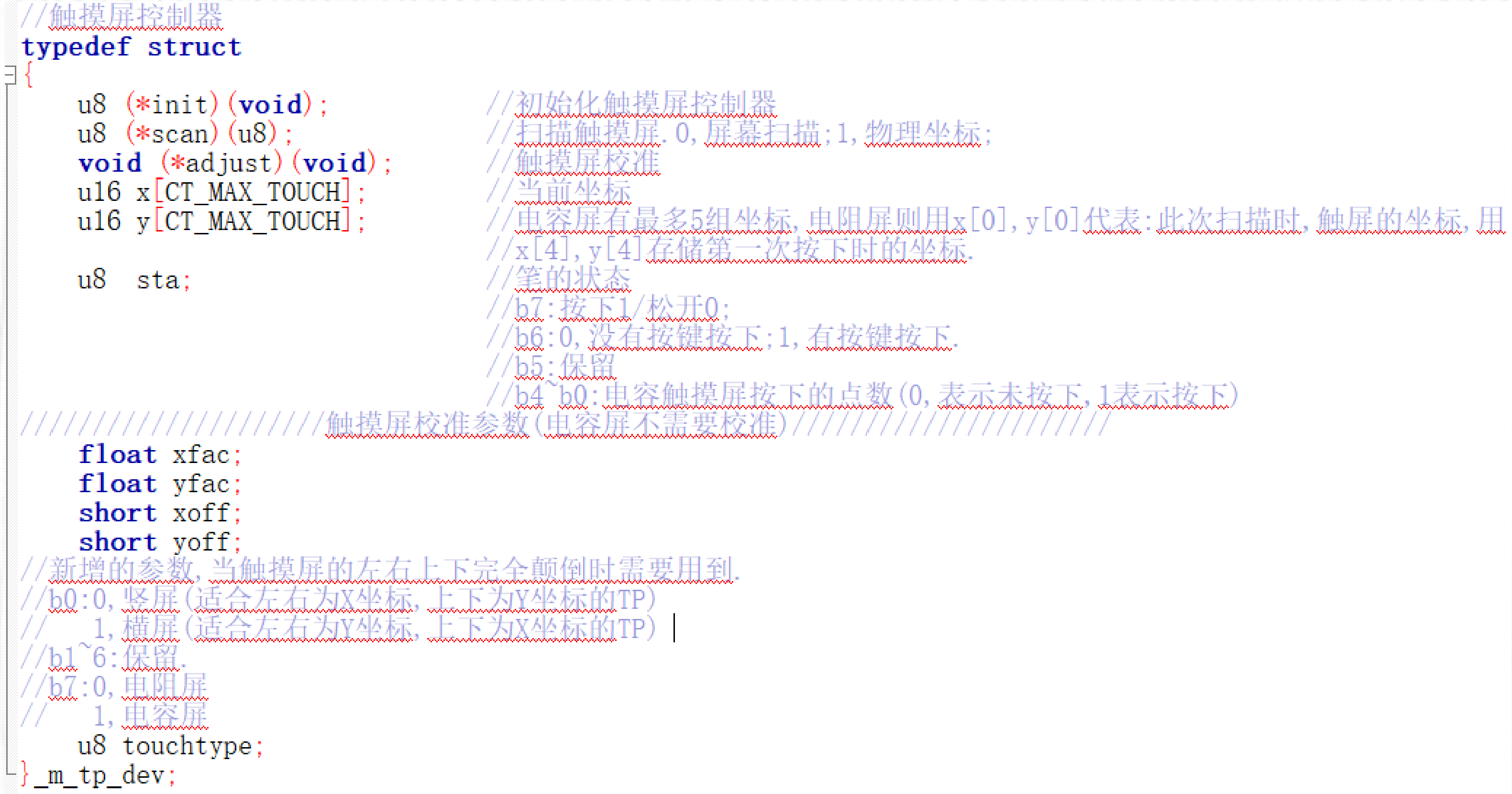
- Serial Peripheral interface
- A high-speed, full duplex, synchronous communication bus.
- Four wires are used.
- Meaning of SPI physical interface:
- SCLK: clock signal, generated by the master device, and the master and slave devices transmit data according to the clock.
- CS: Slave device selects signal, which is controlled by master device. For multiple slave devices, the master device selects which slave device to communicate with, and the cs signal of the slave device is pulled down.
- MISO: Master device data input and slave device data output.
- MOSI: Master device data output, slave device data input.
- SPI interface is mainly used between EEPROM, FLASH, real-time clock, AD converter, digital signal processor and digital signal decoder.

03

How to Program

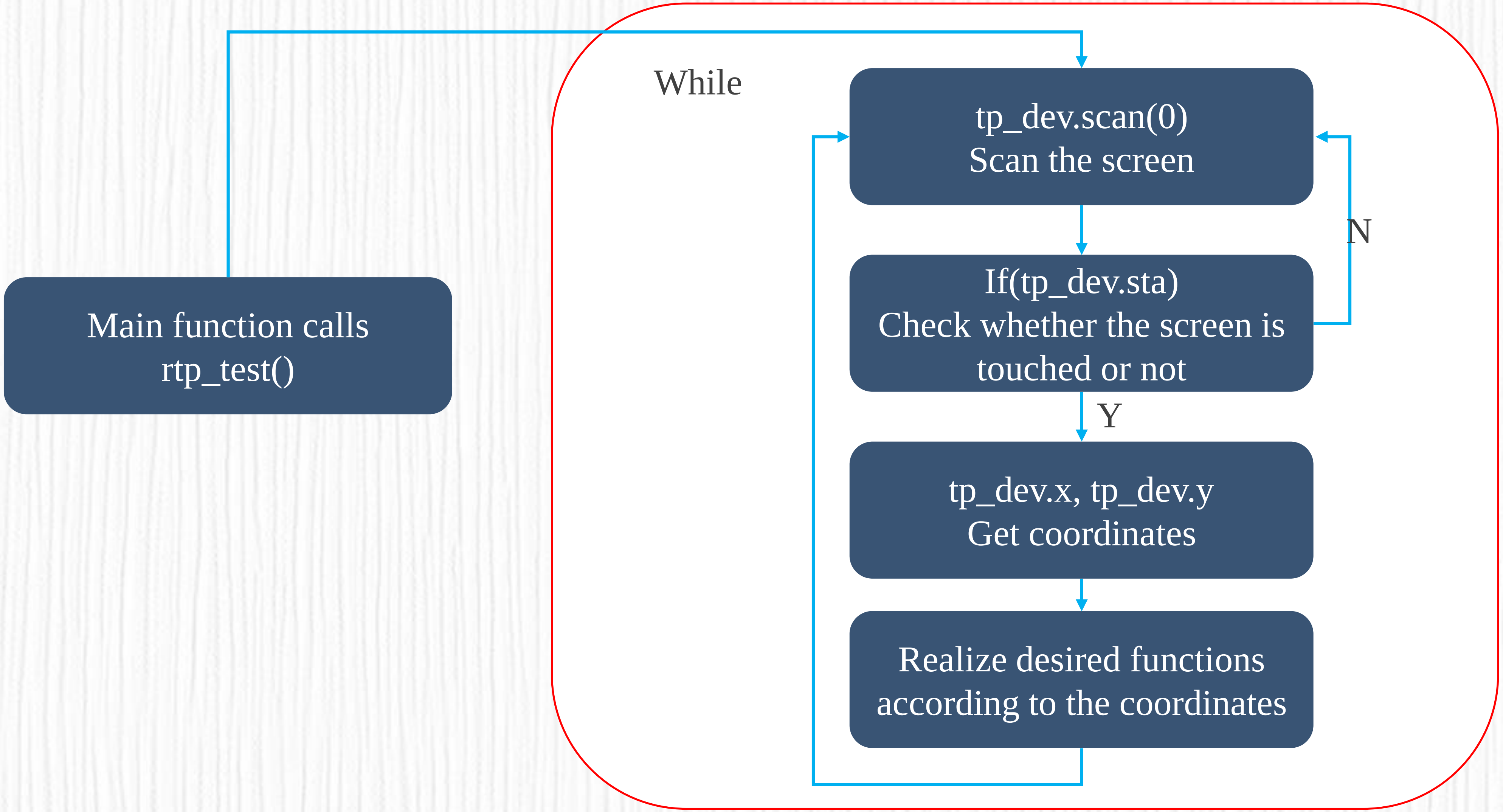
3. How to program
-- Touch screen control structure

- The driving code is implemented in *touch.h* and *touch.c*
- The structure *_m_tp_dev* is defined in *touch.h*



```
//触摸屏控制器
typedef struct
{
    u8 (*init)(void);              //初始化触摸屏控制器
    u8 (*scan)(u8);                //扫描触摸屏.0,屏幕扫描;1,物理坐标;
    void (*adjust)(void);          //触摸屏校准
    u16 x[CT_MAX_TOUCH];           //当前坐标
    u16 y[CT_MAX_TOUCH];           //电容屏有最多5组坐标,电阻屏则用x[0],y[0]代表:此次扫描时,触屏的坐标,用
                                   //x[4],y[4]存储第一次按下时的坐标.
    u8  sta;                       //笔的状态
                                   //b7:按下1/松开0;
                                   //b6:0,没有按键按下;1,有按键按下.
                                   //b5:保留
                                   //b4~b0:电容触摸屏按下的点数(0,表示未按下,1表示按下)
////////////////////触摸屏校准参数(电容屏不需要校准)//////////////////////
    float xfac;
    float yfac;
    short xoff;
    short yoff;
//新增的参数,当触摸屏的左右上下完全颠倒时需要用到.
//b0:0,竖屏(适合左右为X坐标,上下为Y坐标的TP)
//   1,横屏(适合左右为Y坐标,上下为X坐标的TP)
//b1~6:保留.
//b7:0,电阻屏
//   1,电容屏
    u8 touchtype;
}_m_tp_dev;
```

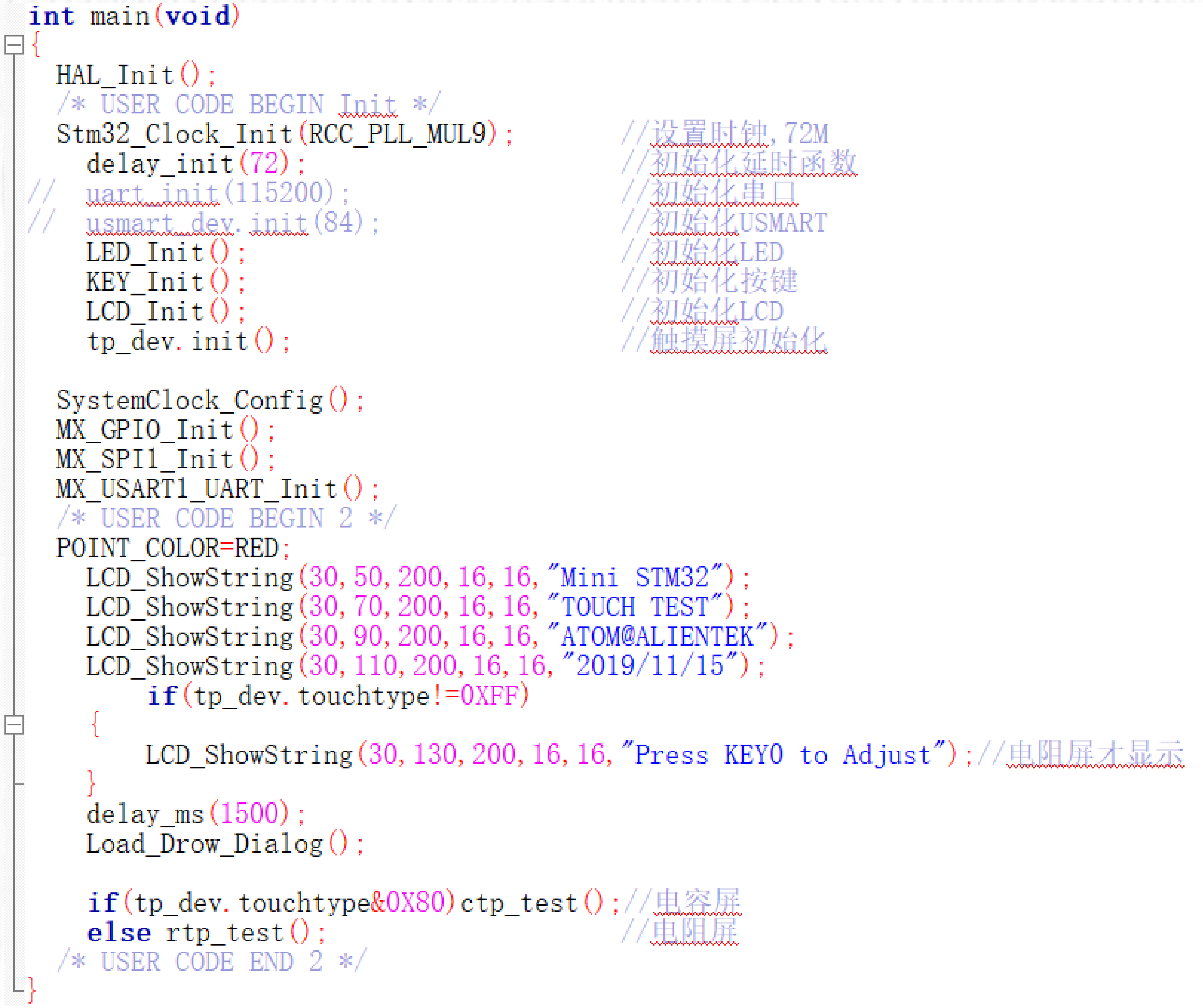
3. How to program
-- Working process



3. How to program
-- Some functions

- TP_Scan
  - Touch screen key scanning
  - Parameter: tp (0:screen coordinates; 1:physical coordinates)
  - Return value: 0:no touch on the screen; 1:the screen has been touched
- TP_Adjust
  - Touch screen calibration code
  - Get four parameters
  - Calibration parameters are saved in AT24CXX chip
- TP_Init
  - Touch screen initialization
  - Return value: 0: no calibration of touch screen; 1:touch screen is calibrated

3. How to program
-- Main function

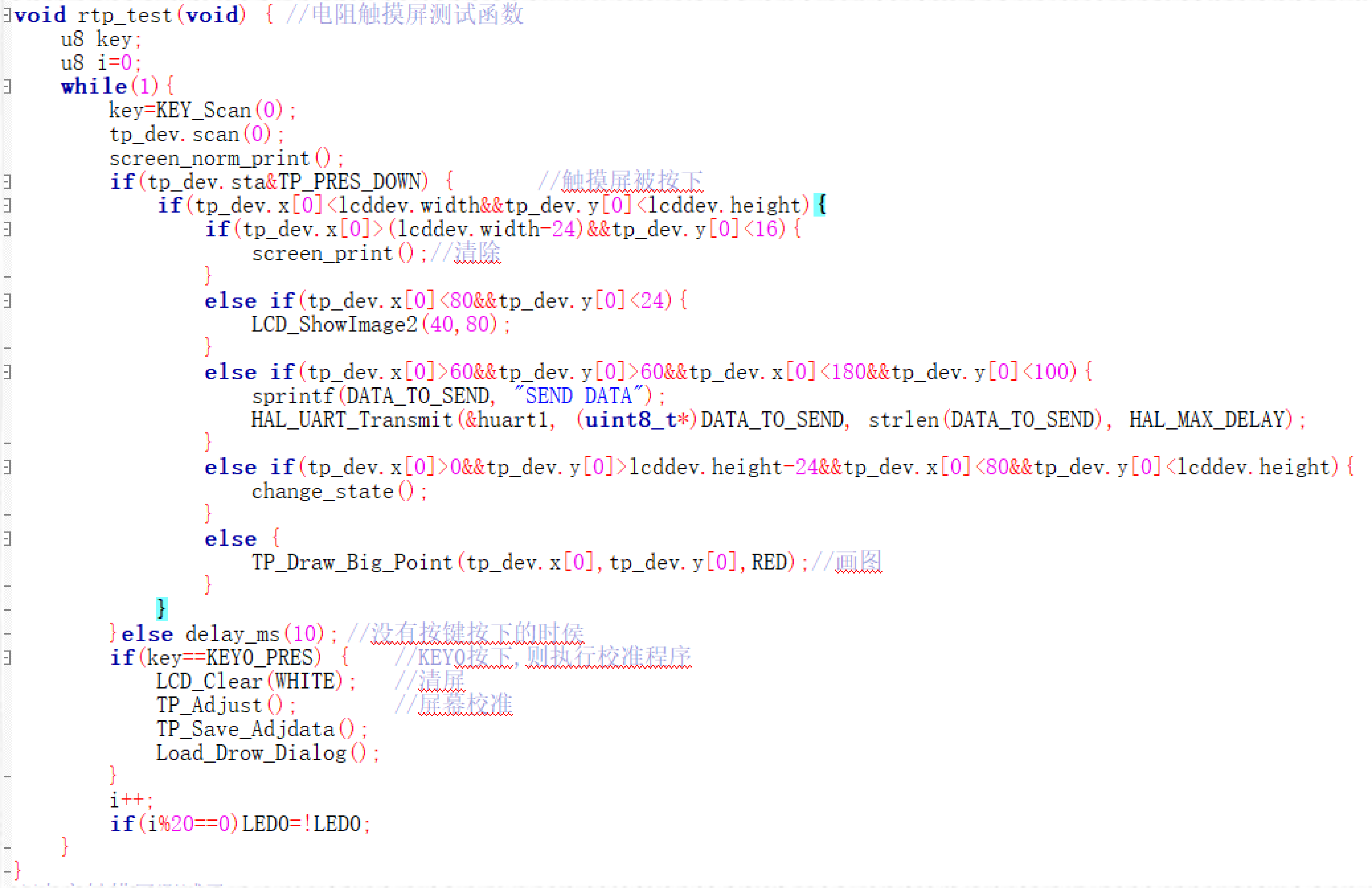


```c
int main(void)
{
  HAL_Init();
  /* USER CODE BEGIN Init */
  Stm32_Clock_Init(RCC_PLL_MUL9);       //设置时钟,72M
    delay_init(72);                     //初始化延时函数
//  uart_init(115200);                  //初始化串口
//  usmart_dev.init(84);                //初始化USMART
    LED_Init();                         //初始化LED
    KEY_Init();                         //初始化按键
    LCD_Init();                         //初始化LCD
    tp_dev.init();                      //触摸屏初始化

  SystemClock_Config();
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  POINT_COLOR=RED;
    LCD_ShowString(30,50,200,16,16,"Mini STM32");
    LCD_ShowString(30,70,200,16,16,"TOUCH TEST");
    LCD_ShowString(30,90,200,16,16,"ATOM@ALIENTEK");
    LCD_ShowString(30,110,200,16,16,"2019/11/15");
        if(tp_dev.touchtype!=0XFF)
    {
        LCD_ShowString(30,130,200,16,16,"Press KEY0 to Adjust");//电阻屏才显示
    }
    delay_ms(1500);
    Load_Drow_Dialog();

    if(tp_dev.touchtype&0X80)ctp_test();//电容屏
    else rtp_test();                    //电阻屏
  /* USER CODE END 2 */
}
```

3. How to program
-- rtp_test function



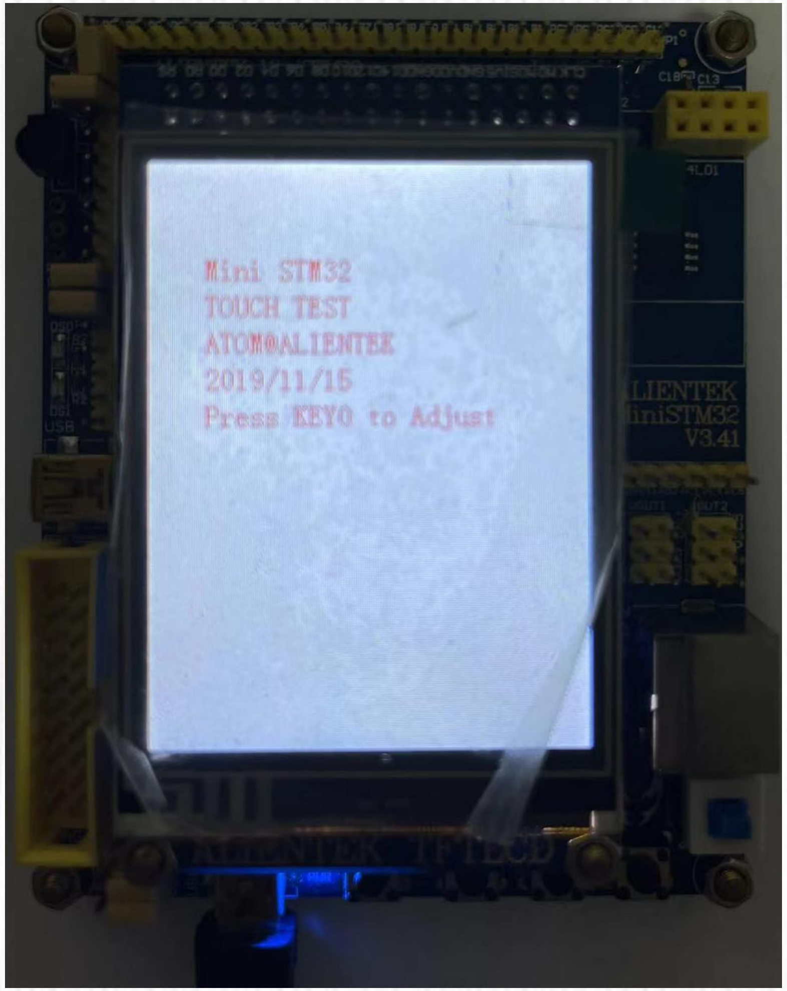
```c
void rtp_test(void) { //电阻触摸屏测试函数
    u8 key;
    u8 i=0;
    while(1){
        key=KEY_Scan(0);
        tp_dev.scan(0);
        screen_norm_print();
        if(tp_dev.sta&TP_PRES_DOWN) {         //触摸屏被按下
            if(tp_dev.x[0]<lcddev.width&&tp_dev.y[0]<lcddev.height){
                if(tp_dev.x[0]>(lcddev.width-24)&&tp_dev.y[0]<16){
                    screen_print();//清除
                }
                else if(tp_dev.x[0]<80&&tp_dev.y[0]<24){
                    LCD_ShowImage2(40,80);
                }
                else if(tp_dev.x[0]>60&&tp_dev.y[0]>60&&tp_dev.x[0]<180&&tp_dev.y[0]<100){
                    sprintf(DATA_TO_SEND, "SEND DATA");
                    HAL_UART_Transmit(&huart1, (uint8_t*)DATA_TO_SEND, strlen(DATA_TO_SEND), HAL_MAX_DELAY);
                }
                else if(tp_dev.x[0]>0&&tp_dev.y[0]>lcddev.height-24&&tp_dev.x[0]<80&&tp_dev.y[0]<lcddev.height){
                    change_state();
                }
                else {
                    TP_Draw_Big_Point(tp_dev.x[0],tp_dev.y[0],RED);//画图
                }
            }
        }else delay_ms(10); //没有按键按下的时候
        if(key==KEY0_PRES) {      //KEY0按下,则执行校准程序
            LCD_Clear(WHITE);     //清屏
            TP_Adjust();          //屏幕校准
            TP_Save_Adjdata();
            Load_Drow_Dialog();
        }
        i++;
        if(i%20==0)LED0=!LED0;
    }
}
```

3. How to program
-- References

- Video：https://www.bilibili.com/video/BV1Lx411Z7Qa?p=67&vd_source=ce498dc8db119fb370d9404aff550452

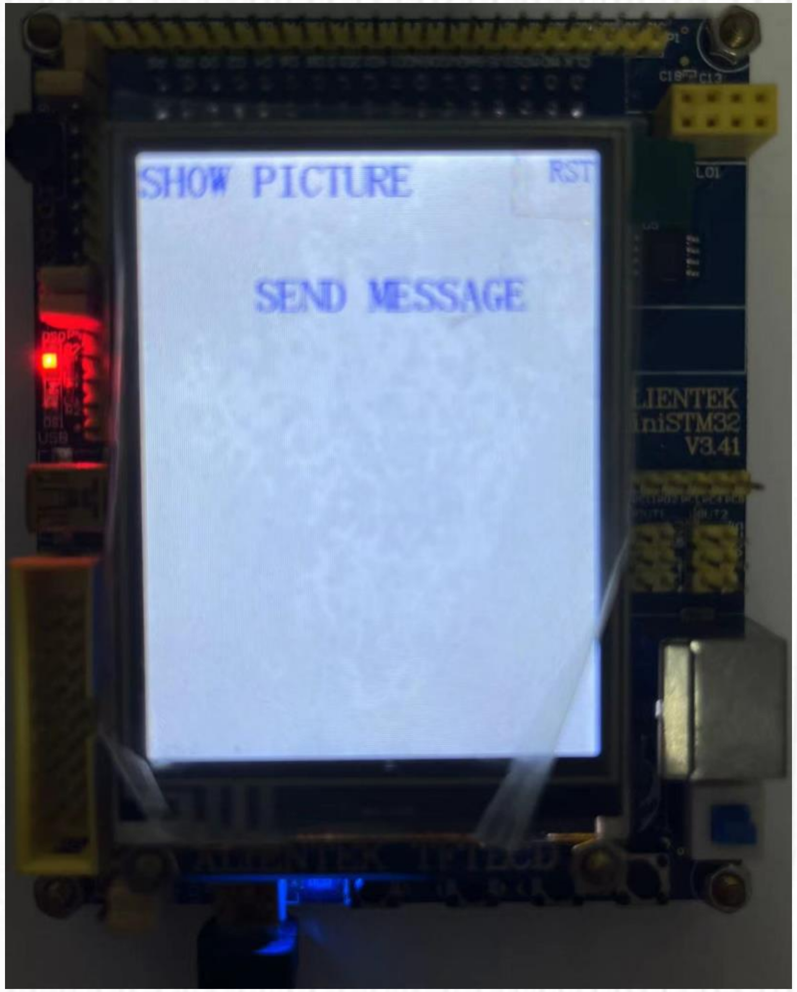
- Information download：http://www.openedv.com/docs/boards/stm32/zdyz_stm32f103_mini.html
http://www.openedv.com/docs/boards/stm32/zdyz_stm32f103_miniV4.html (V4 version)

3. How to program
-- Results 1



```
Mini STM32
TOUCH TEST
ATOM@ALIENTEK
2019/11/15
Press KEY0 to Adjust
```

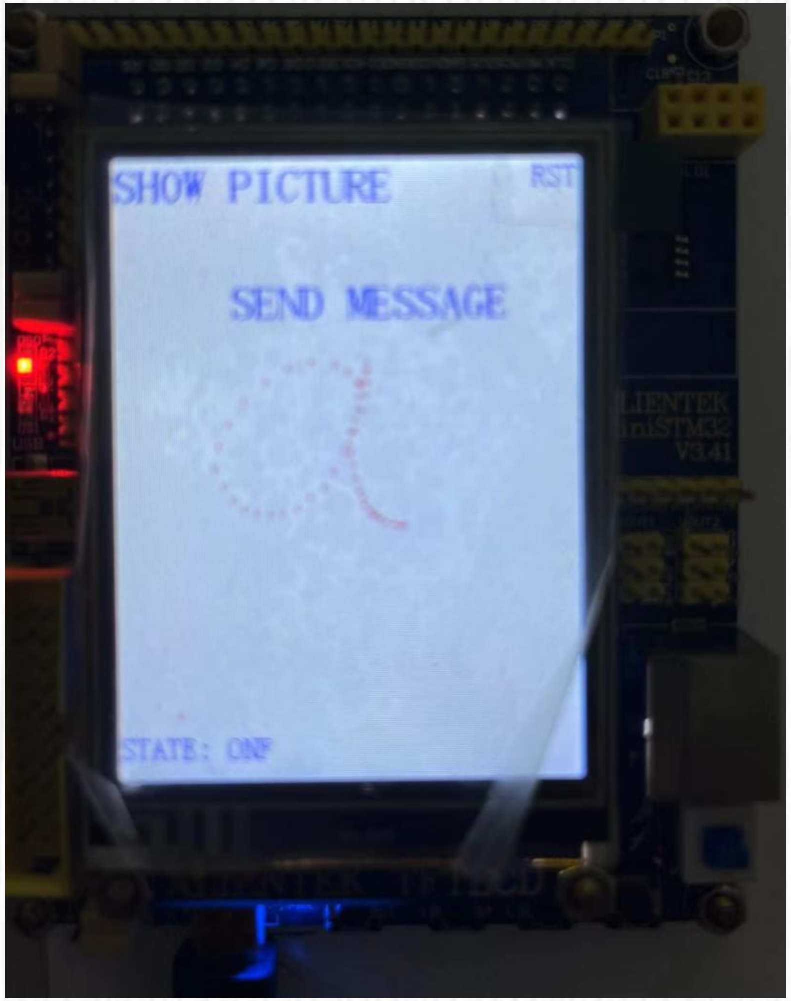
Adjust interface



```
SHOW PICTURE          RST

SEND MESSAGE
```

Main interface



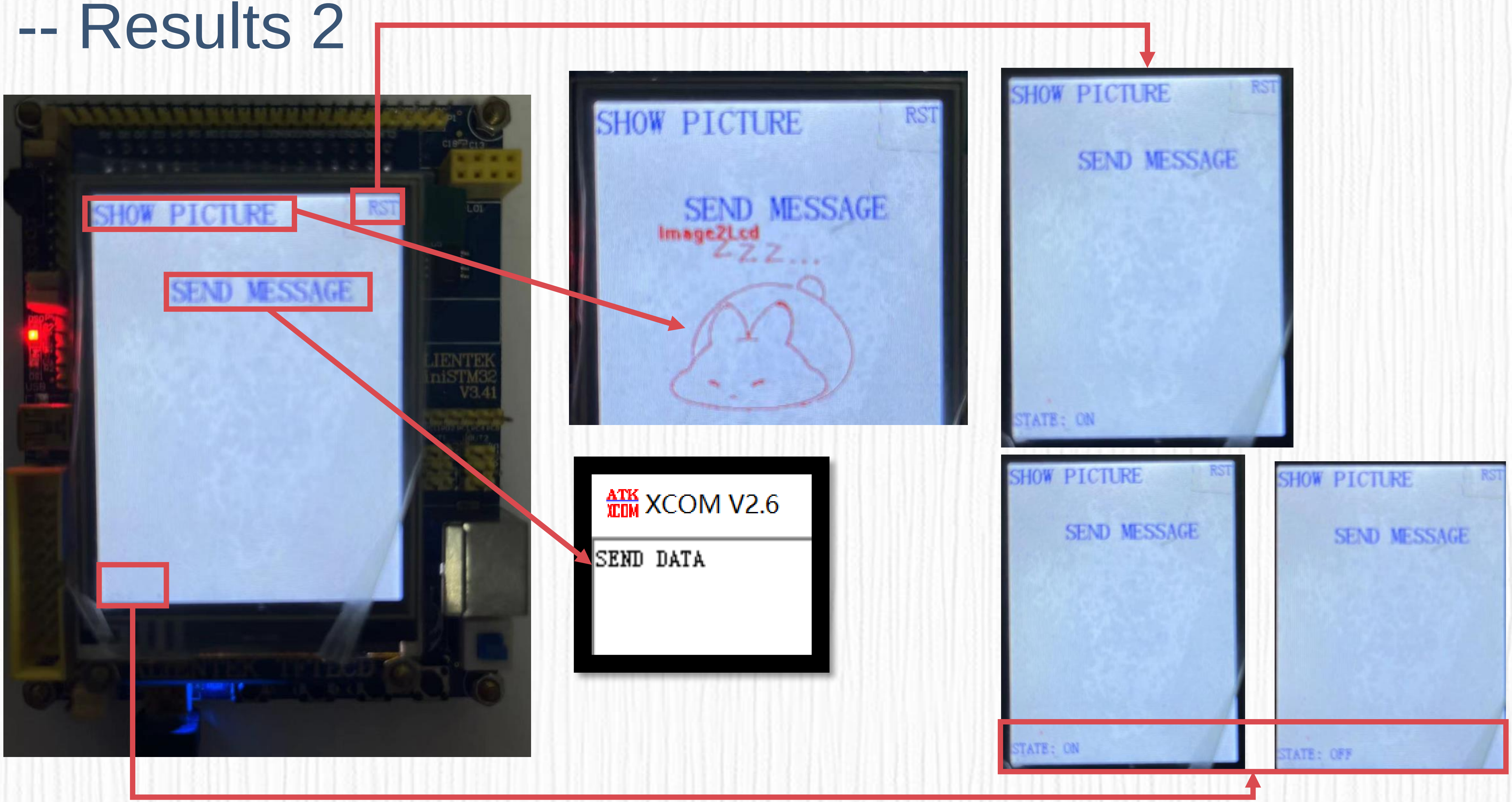
```
SHOW PICTURE          RST

SEND MESSAGE

STATE: ONF
```

Writing on screen

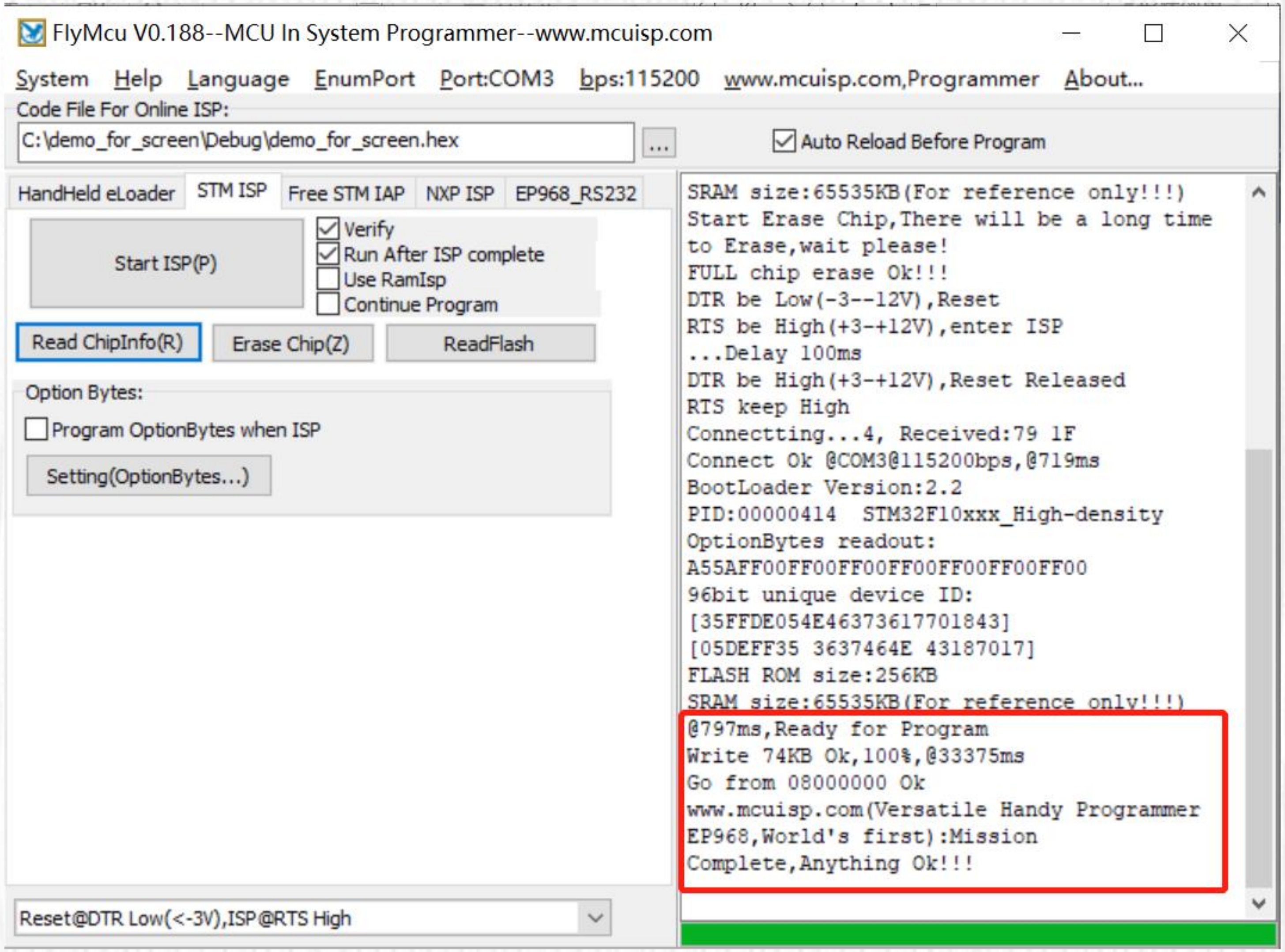
3. How to program -- Results 2



XCOM V2.6

```
SEND DATA
```

3. How to program
-- Programming complete



Code File For Online ISP: C:\demo_for_screen\Debug\demo_for_screen.hex

☑ Auto Reload Before Program

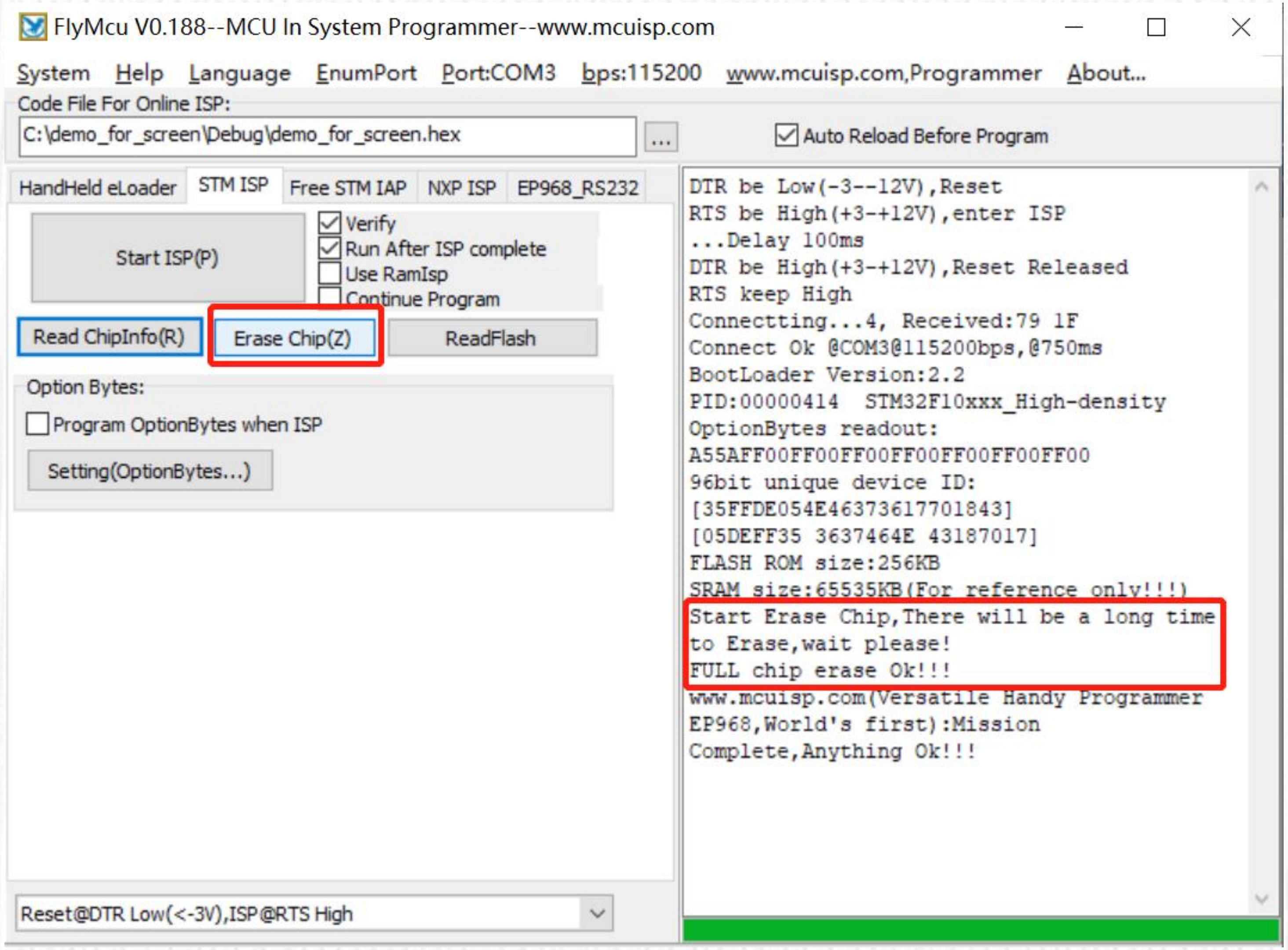
☑ Verify
☑ Run After ISP complete
☐ Use RamIsp
☐ Continue Program

Option Bytes:
☐ Program OptionBytes when ISP

Reset@DTR Low(<-3V),ISP@RTS High

```
SRAM size:65535KB(For reference only!!!)
Start Erase Chip,There will be a long time
to Erase,wait please!
FULL chip erase Ok!!!
DTR be Low(-3--12V),Reset
RTS be High(+3-+12V),enter ISP
...Delay 100ms
DTR be High(+3-+12V),Reset Released
RTS keep High
Connectting...4, Received:79 1F
Connect Ok @COM3@115200bps,@719ms
BootLoader Version:2.2
PID:00000414  STM32F10xxx_High-density
OptionBytes readout:
A55AFF00FF00FF00FF00FF00FF00FF00
96bit unique device ID:
[35FFDE054E46373617701843]
[05DEFF35 3637464E 43187017]
FLASH ROM size:256KB
SRAM size:65535KB(For reference only!!!)
@797ms,Ready for Program
Write 74KB Ok,100%,@33375ms
Go from 08000000 Ok
www.mcuisp.com(Versatile Handy Programmer
EP968,World's first):Mission
Complete,Anything Ok!!!
```

3. How to program
-- Chip erase


FlyMcu V0.188--MCU In System Programmer--www.mcuisp.com
System Help Language EnumPort Port:COM3 bps:115200 www.mcuisp.com,Programmer About...
Code File For Online ISP:
C:\demo_for_screen\Debug\demo_for_screen.hex
...
☑ Auto Reload Before Program
HandHeld eLoader STM ISP Free STM IAP NXP ISP EP968_RS232
Start ISP(P)
☑ Verify
☑ Run After ISP complete
□ Use RamIsp
□ Continue Program
Read ChipInfo(R)
Erase Chip(Z)
ReadFlash
Option Bytes:
□ Program OptionBytes when ISP
Setting(OptionBytes...)
Reset@DTR Low(<-3V),ISP@RTS High

```
DTR be Low(-3--12V),Reset
RTS be High(+3-+12V),enter ISP
...Delay 100ms
DTR be High(+3-+12V),Reset Released
RTS keep High
Connectting...4, Received:79 1F
Connect Ok @COM3@115200bps,@750ms
BootLoader Version:2.2
PID:00000414  STM32F10xxx_High-density
OptionBytes readout:
A55AFF00FF00FF00FF00FF00FF00FF00
96bit unique device ID:
[35FFDE054E46373617701843]
[05DEFF35 3637464E 43187017]
FLASH ROM size:256KB
SRAM size:65535KB(For reference only!!!)
Start Erase Chip,There will be a long time
to Erase,wait please!
FULL chip erase Ok!!!
www.mcuisp.com(Versatile Handy Programmer
EP968,World's first):Mission
Complete,Anything Ok!!!
```

04

Practice

4. Practice

- Run the demo on MiniSTM32 board.
- Complete a new buttons to realize an independently designed function to show a true 16bit color picture.

TIPS: How to Display a true 16bit color picture

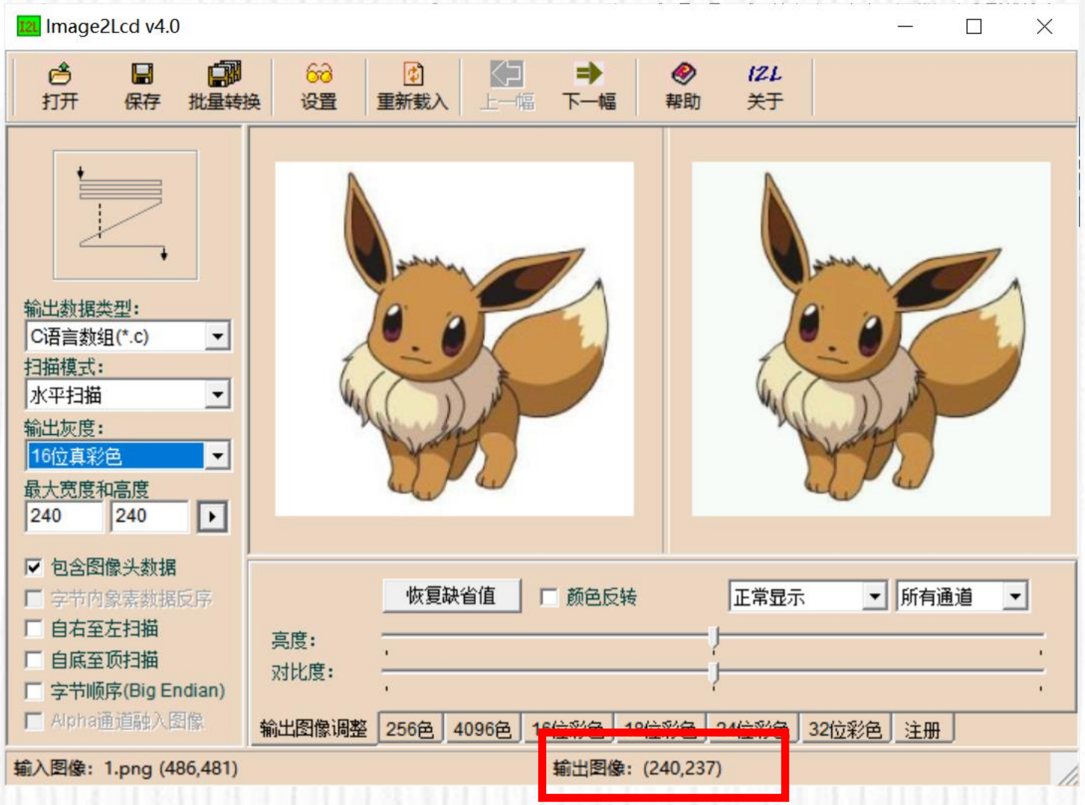
- Step1： Add the following function and prototype into lcd.c and lcd.h (lcd_v4.c and lcd_v4.h for V4 board.)

```
void LCD_ShowPicture(uint16_t x,uint16_t y,uint16_t column,uint16_t row,unsigned short *pic)
{
uint16_t m,h;
uint16_t *data=(uint16_t *)pic;
    for(h=0+y;h<row+y;h++) //60
    {
        for(m=0+x;m<column+x;m++) //180
        {
            LCD_Fast_DrawPoint(m,h,*data++);
        }
    }
}
```

```
void LCD_ShowPicture(uint16_t x,uint16_t y,uint16_t column,uint16_t row,unsigned short *pic);
```

TIPS: How to Display a true 16bit color picture

- Step2： Load the Image into Image2Lcd(download from blackboard), set the correct size and save it as C file



输出数据类型: C语言数组(*.c)

扫描模式: 水平扫描

输出灰度: 16位真彩色

最大宽度和高度 240 240

☑ 包含图像头数据
□ 字节内象素数据反序
□ 自右至左扫描
□ 自底至顶扫描
□ 字节顺序(Big Endian)
□ Alpha通道融入图像

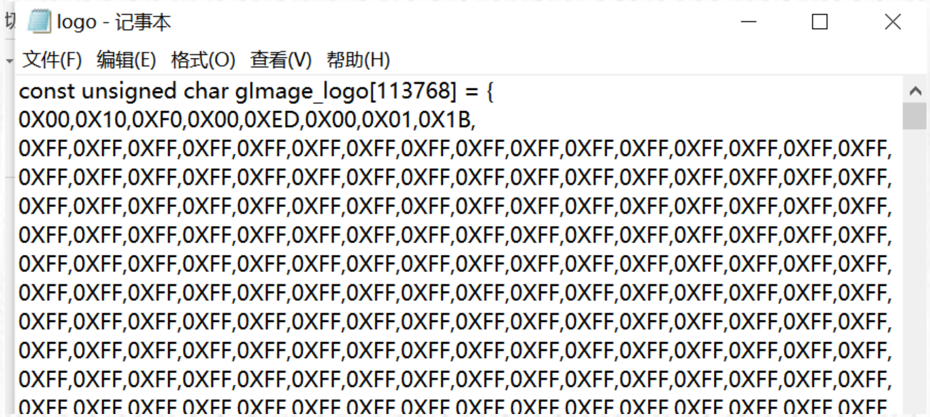
□ 颜色反转 正常显示 所有通道

亮度:

对比度:

输入图像: 1.png (486,481)

输出图像: (240,237)

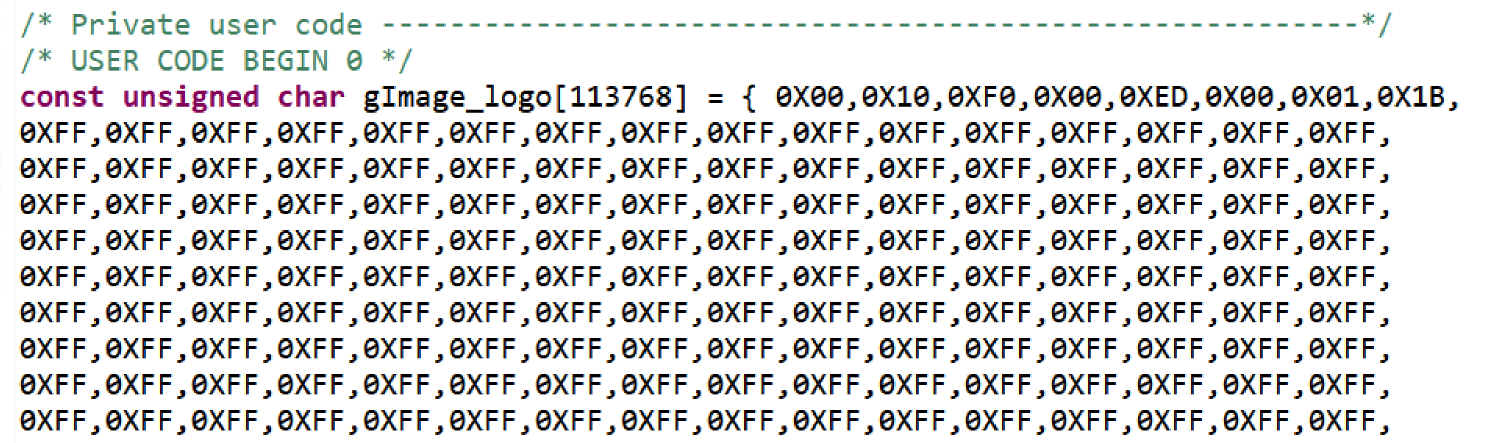


```
const unsigned char gImage_logo[113768] = {
0X00,0X10,0XF0,0X00,0XED,0X00,0X01,0X1B,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF
```

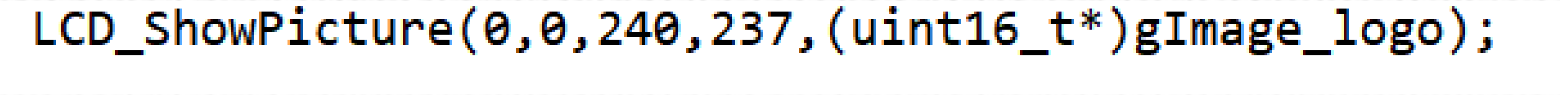
(240， 237) -> size of the picture

TIPS: How to Display a true 16bit color picture

- Step3： Copy the char array declaration into main.c and display the picture using `LCD_ShowPicture()`



```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
const unsigned char gImage_logo[113768] = { 0X00,0X10,0XF0,0X00,0XED,0X00,0X01,0X1B,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
```



```
LCD_ShowPicture(0,0,240,237,(uint16_t*)gImage_logo);
```