3.7.screen display

1. learning target

1.Learn to understand the basics of iic communication.

2.Learn how to use the onboard 9.6-inch screen for displaying Chinese characters and character strings

IIC introduction of STM32F1:

The IIC bus is a bidirectional two-wire continuous bus that provides communication lines between integrated circuits. It means a protocol for completing the exchange of information between integrated circuits or functional units.

The IIC module receives and sends data, and converts data from serial to parallel, or parallel to serial. Interrupts can be enabled or disabled. The interface is connected to the IIC bus through a data pin (SDA) and a clock pin (SCL). Allows connection to standard (up to 100kHz) or fast (up to 400kHz) IIC buses. (The data line SDA and the clock SCL form a serial bus, which can send and receive data).

The IIC bus has three types of signals in the process of transmitting data, namely: start signal (START), stop (end) signal (STOP), and response signal (ACK). Second, it is idle when no data transfer is in progress.

The following is a simple interpretation of IIC communication combined with STM32f103

1.idle state:

For the initialization of SCL and SDA, we should set it to high level

2.When data transfer starts (start condition):

Start signal: When SCL is high level, SDA jumps from high to low. Here we should pay attention that the start signal is a level jump timing signal, not a level signal. From the above figure, we can see that SDA and SCL At the beginning, they are all high. When SCL is high, delay for a period of time to pull SDA low, and then wait for a period of time for SCL to become low.

3.end of data transfer：

End signal: When SCL is high level, SDA jumps from low to high, ending data transmission. The end (stop) signal is also a level jump timing signal. As can be seen from the above figure, a sign is required at the end of data transmission, indicating that the data transmission has been completed. First, we need to pull both SCL and SDA low, and wait for a while (SCL changes from low to high) to pull SCL high. During SCL is high SDA becomes high (so the following code will first pull SCL high and then pull SDA high)

4.Response signal: After receiving 8bit data, the IC that receives data sends a specific low-level pulse to the IC that sends data, indicating that the data has been received. After the CPU sends a signal to the controlled unit, it waits for the controlled unit to send a response signal. After receiving the response signal, the CPU makes a judgment on whether to continue to transmit the signal according to the actual situation. If no response signal is received, it is judged that the controlled unit is faulty.

5.slave PC reply：

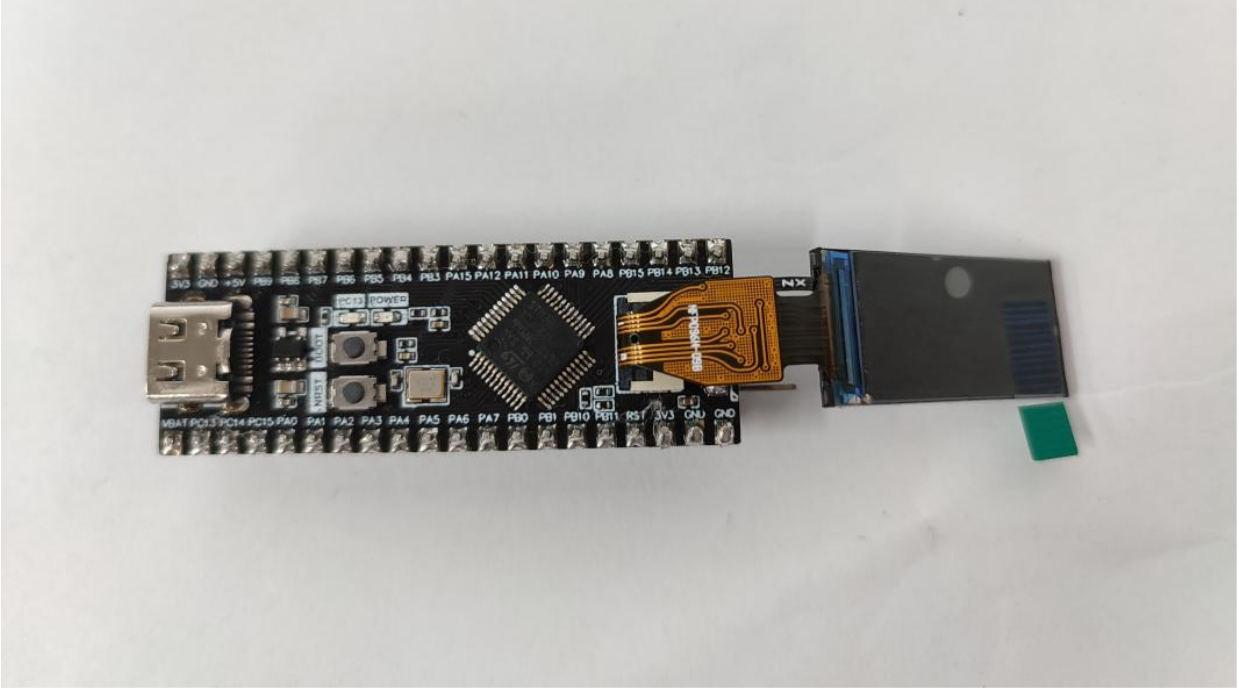
Pull both SDA and SCL high, so that you can better judge whether the receiver returns a response signal, and judge whether the received signal is low in the while loop (that is, return a response), if not, wait for a while, within the specified time If it returns 1 without returning a response, it means that the response failed to be received.

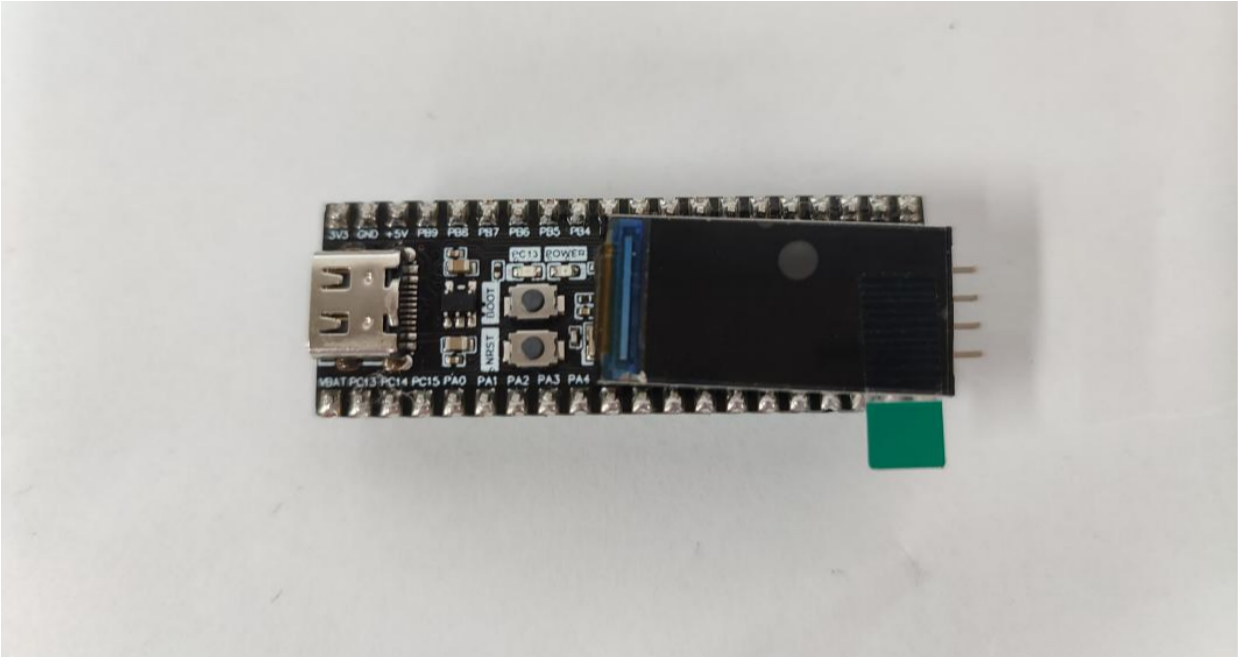
6.master Reply

When the slave generates a response, the master changes from sending to receiving, and the slave changes from receiving to sending. Afterwards, data is sent by the slave, received by the master, each reply is generated by the master, and the clock signal is still generated by the master. If the host wants to terminate this transmission, it sends a non-response signal, and then the host generates a stop condition.

2. Hardware wiring connection

Note: If you do not purchase the package that includes the screen, you need to purchase the screen used in the experiment separately, otherwise you will not be able to complete this course due to the lack of important accessories.





Physical diagram of screen connection

3. program analysis

LCD initialization code：

```c
void LCD_Init(void)
{
  LCD_GPIO_Init();
  LCD_RES_Clr();
  delay_ms(100);
  LCD_RES_Set();
  delay_ms(100);
  LCD_BLK_Clr();
  delay_ms(100);
  LCD_WR_REG(0x11);     //Sleep out
  delay_ms(120);                //Delay 120ms
  LCD_WR_REG(0xB1);     //Normal mode
  LCD_WR_DATA8(0x05);
  LCD_WR_DATA8(0x3C);
  LCD_WR_DATA8(0x3C);
  LCD_WR_REG(0xB2);     //Idle mode
  LCD_WR_DATA8(0x05);
  LCD_WR_DATA8(0x3C);
  LCD_WR_DATA8(0x3C);
  LCD_WR_REG(0xB3);     //Partial mode
  LCD_WR_DATA8(0x05);
  LCD_WR_DATA8(0x3C);
  LCD_WR_DATA8(0x3C);
  LCD_WR_DATA8(0x05);
  LCD_WR_DATA8(0x3C);
  LCD_WR_DATA8(0x3C);
  LCD_WR_REG(0xB4);     //Dot inversion
  LCD_WR_DATA8(0x03);
  LCD_WR_REG(0xC0);     //AVDD GVDD
  LCD_WR_DATA8(0xAB);
  LCD_WR_DATA8(0x0B);
  LCD_WR_DATA8(0x04);
  LCD_WR_REG(0xC1);     //VGH VGL
  LCD_WR_DATA8(0xC5);   //C0
  LCD_WR_REG(0xC2);     //Normal Mode
  LCD_WR_DATA8(0x0D);
  LCD_WR_DATA8(0x00);
  LCD_WR_REG(0xC3);     //Idle
  LCD_WR_DATA8(0x8D);
  LCD_WR_DATA8(0x6A);
  LCD_WR_REG(0xC4);     //Partial+Full
  LCD_WR_DATA8(0x8D);
```

```
    LCD_WR_DATA8(0x09);
    LCD_WR_DATA8(0x0F);
    LCD_WR_DATA8(0x25);
    LCD_WR_DATA8(0x36);
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0x08);
    LCD_WR_DATA8(0x04);
    LCD_WR_DATA8(0x10);
    LCD_WR_REG(0xE1);     //negative gamma
    LCD_WR_DATA8(0x0A);
    LCD_WR_DATA8(0x0D);
    LCD_WR_DATA8(0x08);
    LCD_WR_DATA8(0x07);
    LCD_WR_DATA8(0x0F);
    LCD_WR_DATA8(0x07);
    LCD_WR_DATA8(0x02);
    LCD_WR_DATA8(0x07);
    LCD_WR_DATA8(0x09);
    LCD_WR_DATA8(0x0F);
    LCD_WR_DATA8(0x25);
    LCD_WR_DATA8(0x35);
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0x09);
    LCD_WR_DATA8(0x04);
    LCD_WR_DATA8(0x10);
    LCD_WR_REG(0xFC);
    LCD_WR_DATA8(0x80);
    LCD_WR_REG(0x3A);
    LCD_WR_DATA8(0x05);
    LCD_WR_REG(0x36);
    if(USE_HORIZONTAL==0)LCD_WR_DATA8(0x08);
    else if(USE_HORIZONTAL==1)LCD_WR_DATA8(0xC8);
    else if(USE_HORIZONTAL==2)LCD_WR_DATA8(0x78);
    else LCD_WR_DATA8(0xA8);
    LCD_WR_REG(0x21);     //Display inversion
    LCD_WR_REG(0x29);     //Display on
    LCD_WR_REG(0x2A);     //Set Column Address
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0x1A);   //26
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0x69);    //105
    LCD_WR_REG(0x2B);     //Set Page Address
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0x01);     //1
    LCD_WR_DATA8(0x00);
    LCD_WR_DATA8(0xA0);     //160
    LCD_WR_REG(0x2C);
}
```

screen fill code:

```
void LCD_Fill(u16 xsta,u16 ysta,u16 xend,u16 yend,u16 color)
{
    u16 i,j;
    LCD_Address_Set(xsta,ysta,xend-1,yend-1);
    for(i=ysta;i<yend;i++)
    {
        for(j=xsta;j<xend;j++)
        {
            LCD_WR_DATA(color);
        }
    }
}
```

string display:

```
void LCD_ShowString(u16 x,u16 y,const u8 *p,u16 fc,u16 bc,u8 sizey,u8 mode)
{
    while(*p!='\0')
    {
        LCD_ShowChar(x,y,*p,fc,bc,sizey,mode);
        x+=sizey/2;
        p++;
    }
}
```

Chinese character display code:

```
void LCD_ShowChinese16x16(u16 x,u16 y,u8 *s,u16 fc,u16 bc,u8 sizey,u8 mode)
{
    u8 i,j,m=0;
    u16 k;
    u16 HZnum;
    u16 TypefaceNum;
    u16 x0=x;
  TypefaceNum=(sizey/8+((sizey%8)?1:0))*sizey;
    HZnum=sizeof(tfont16)/sizeof(typFNT_GB16);
    for(k=0;k<HZnum;k++)
    {
        if ((tfont16[k].Index[0]==*(s))&&(tfont16[k].Index[1]==*(s+1)))
        {
            LCD_Address_Set(x,y,x+sizey-1,y+sizey-1);
            for(i=0;i<TypefaceNum;i++)
            {
                for(j=0;j<8;j++)
                {
                    if(!mode)
                    {
                        if(tfont16[k].Msk[i]&(0x01<<j))LCD_WR_DATA(fc);
                        else LCD_WR_DATA(bc);
                        m++;
                        if(m%sizey==0)
                        {
                            m=0;
                            break;
                        }
                    }
                    else
                    {
                        if(tfont16[k].Msk[i]&(0x01<<j)) LCD_DrawPoint(x,y,fc);
                        x++;
                        if((x-x0)==sizey)
                        {
                            x=x0;
                            y++;
                            break;
                        }
                    }
                }
            }
        }
        continue;
    }
}
```

image display code:

```
void LCD_ShowPicture(u16 x,u16 y,u16 length,u16 width,const u8 pic[])
{
    u16 i,j;
    u32 k=0;
    LCD_Address_Set(x,y,x+length-1,y+width-1);
    for(i=0;i<length;i++)
    {
        for(j=0;j<width;j++)
        {
            LCD_WR_DATA8(pic[k*2]);
            LCD_WR_DATA8(pic[k*2+1]);
            k++;
        }
    }
}
```

main function code：

```
int main(void)
{
    delay_init();
    LCD_Init();
    LCD_Fill(0,0,LCD_W,LCD_H,WHITE);
    LCD_ShowPicture(20,45,120,29,gImage_pic1);
    LCD_ShowString(10,0,"Hello!",BLACK,WHITE,16,0);
    LCD_ShowChinese(50,20,"亚博智能",BLACK,WHITE,16,0);
    while(1)
    {

    }
}
```

The function realized by the main function is very simple, first call the LCD initialization function, fill the screen with white background, and then use the function to display character strings, Chinese characters and pictures.

4.Experimental phenomena

After the program is downloaded, the 0.96-inch screen will display "Hello!", "亚博智能" and the logo of the yahboom company.