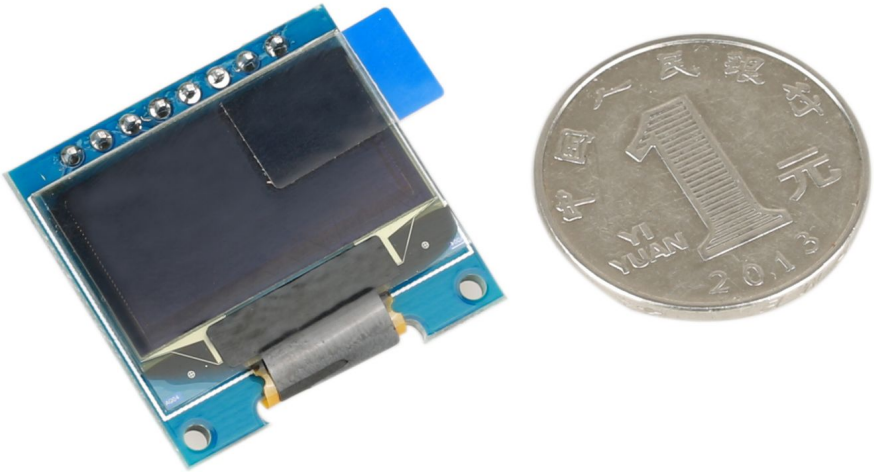
Touch Your Future!
www.yfrobot.com

# 0.96’ OLED(带字库)

## 使用手册 V1.0




YFRobot 荣誉出品

2015 年 11 月 16 日

# 版 权 声 明

# 目　录

# 1 0.96'OLED 简介

OLED(Organic Light-Emitting Diode)：有机发光二极管又称为有机电激光显示，由美籍华裔教授邓青云在实验室中发现，由此展开了对 OLED 的研究。OLED 显示技术具有自发光的特性，采用非常薄的有机材料涂层和玻璃基板，当有电流通过时，这些有机材料就会发光，而且 OLED 显示屏幕可视角度大，并且能够节省电能。OLED 由于同时具备自发光、不需背光源、对比度高、厚度薄、视角广、反应速度快、可用于挠曲面板、使用温度范围广、结构及制程简单等优异之特性，被认为下一代平面显示器新兴应用技术。

最先接触的 12864 屏都是 LCD 的，需要背光，功耗较高，而 OLED 的功耗低，更加适合小系统；由于两者发光材料的不同，在不同的环境中，OLED 的显示效果更佳。更多的 OLED 与 LCD 的比较，可以百度，了解更多信息。

此手册我们将详细介绍 YFROBOT 0.96'OLE（带字库）的使用方法，并提供详细的例程详解。

该模块特点：

1、三色可选，模块有两种单色和黄蓝双色两种颜色可选，单色为纯白色和纯蓝色，双色为黄蓝双色；

2、超小尺寸，显示尺寸为 0.96 寸，模块尺寸为 27mm（长）*26mm（宽）*4mm（高）；

3、高分辨率，分辨率为 128*64；

4、两种接口模式，4 线串行 SPI 接口模式，IIC 接口模式；

5、带字库，可显示标准的国标简体（GB2312）汉字、8*16 点 ASCII 粗体字库、7*8 点 ASCII 字库、5*7 点 ASCII 字库。

6、兼容 3.3V~5V 电压。

YFROBOT 0.96'OLED 模块通过外部排针和单片机通讯，各引脚详细描述如下：

| 序号 | 名称 | 说明 |
|---|---|---|
| 1 | GND | 地 |
| 2 | VCC | 电源，3.3V~5V |
| 3 | CLK | 4 线 ISP 接口模式：时钟线<br>IIC 接口模式：时钟线<br>GT20L16S1Y 的时钟线 |
| 4 | DIN | 4 线 ISP 接口模式：串行数据线<br>IIC 接口模式：数据线<br>GT20L16S1Y 的串行数据输入端口 |
| 5 | D/C | 4 线 ISP 接口模式：命令/数据标志位<br>IIC 接口模式：接 GND |
| 6 | CS1 | 4 线 ISP 接口模式：OLED 片选<br>IIC 接口模式：接 GND |
| 7 | SO | GT20L16S1Y 的串行数据输出端口 |
| 8 | CS2 | GT20L16S1Y 的片选端口 |

此模块引出 IIC 通讯端口，只需将图中标识的部位短接，就可使用 IIC 通讯模式。教程暂时未提供 IIC 的通讯例程。

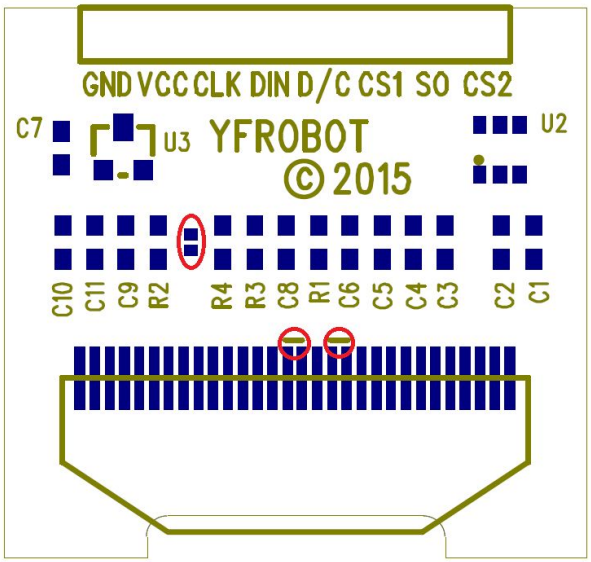



图 1.1　IIC 接口模式设置

# 2 OLED 点亮原理

0.96’OLED 模块（以下简称 OLED）显示屏由两个部分构成，有效的显示部分和驱动芯片 SSD1306，此芯片被固化在液晶显示玻璃的背部。额外设计字库模块，方便显示不同字体的字符和汉字。

## 2.1 取模

这里我们介绍一款非常好用的取模软件：PCtoLCD2002 完美版。他可以提供各种字符，包括汉字（字体和大小都可以设置）阵取模，且取模的方式也是可以设置的，常用的取模方式，该软件都支持。该软件也支持图形取模，图片的格式需为 BMP 格式。

此软件在“字符图片取模软件”文件夹中。

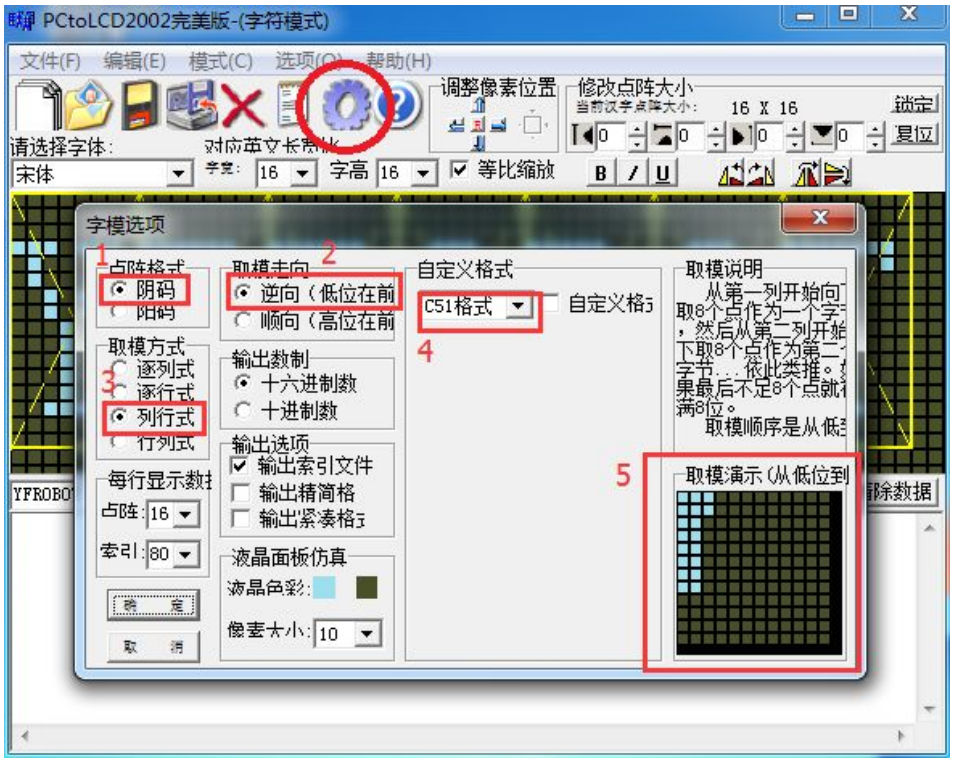



图 2.1.1 取模设置

打开取模软件，如图 2.1.1 所示，点击圆圈处图标打开设置，安照方框 1、2、3、4 处设置，方框 5 为取模的示意图，可以对照上方的“取模说明”，理解示意图。“每行显示数”表示生成字模后，每行包含的字节个数。点击“确定”，保存并推出设置。

输入“YFROBOT”，点击“生成字模”，下方生成数组。

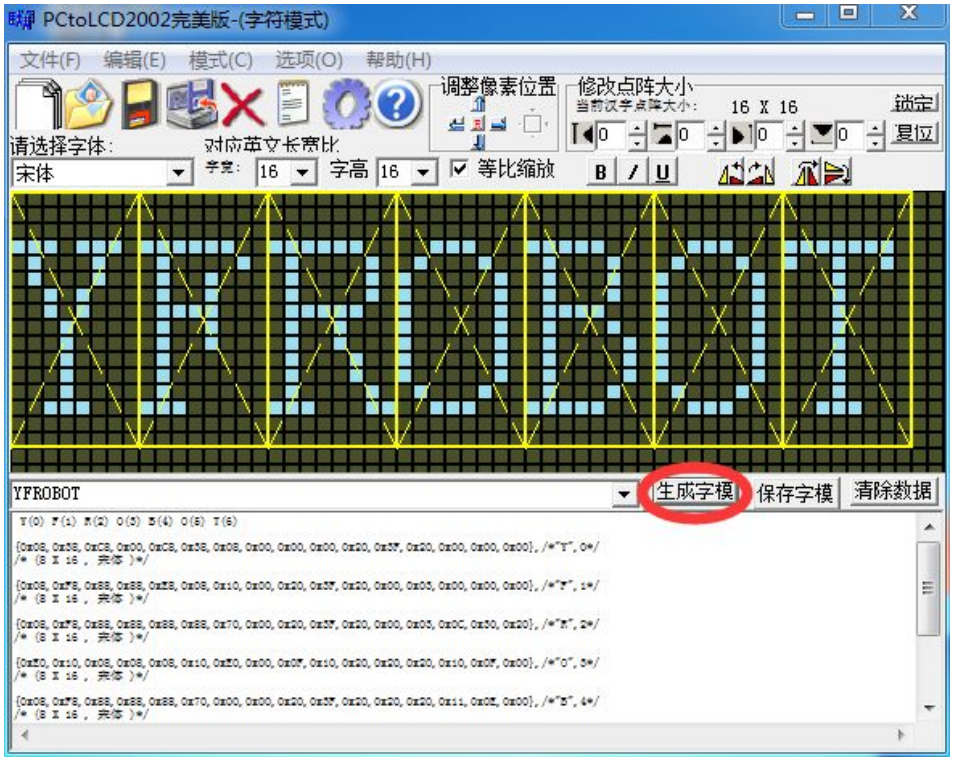



图 2.1.2　生成字模

“Y”生成的数组为：

{0x08,0x38,0xC8,0x00,0xC8,0x38,0x08,0x00,0x00,0x00,0x20,0x3F,0x20,0x00,0x00,0x00}

结合数组与图 2.1.3，可以看出取模的设置，和生成的字模是相对应的。从第一列开始，向下取 8 个点作为一个字节，然后从第二列开始，向下取 8 个点作为第二个字节，以此类推。如果最后不足 8 个点，就补满 8 位。

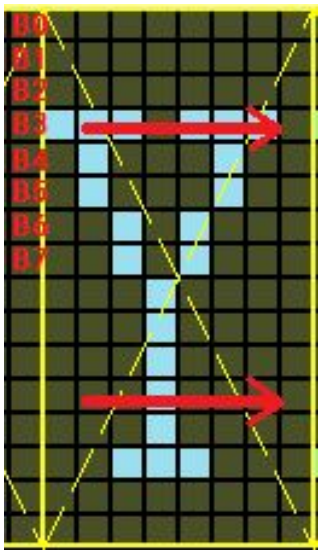



图 2.1.3　取模数序

再介绍图片取模。软件只能识别 BMP 格式的，我们需要将其它格式的图片转换为 BMP 格式，并设置合适的像素。在文件夹“字符图片取模软件→实验图片”，选择图片“yfrobot”右击鼠标，“打开方式→画图”，打开图片，这样就可以对图片进行简单的设置。如图 2.1.4：

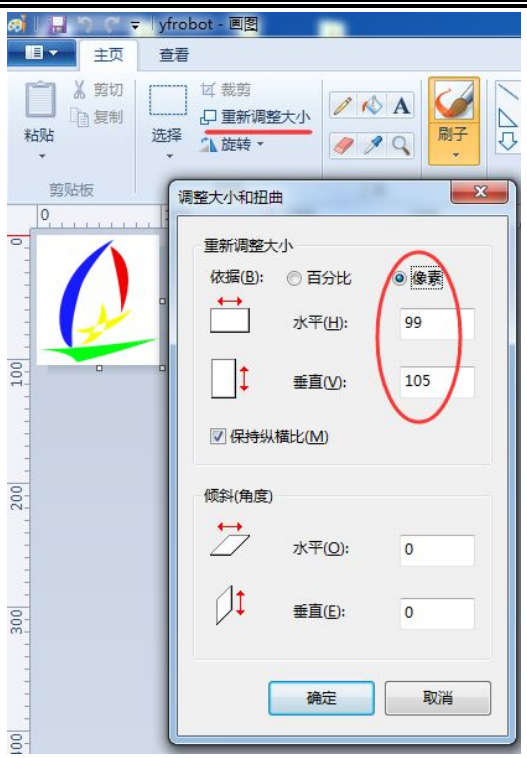



图 2.1.4　设置图片像素

0.96'OLED 的显示大小为 128×64 的，我们设置图片像素为 60×64，然后将图片另存为 BMP 格式。


图 2.1.5　另存为 BMP 格式

打开取模软件，点击圆圈处图标，导入图片，取模方式和上面相同，导入图片后，点击"生成字模"，生成一个多维数组，将部分大括号"｛｝"删除，变为一维数组，得到图片

显示点阵数组。

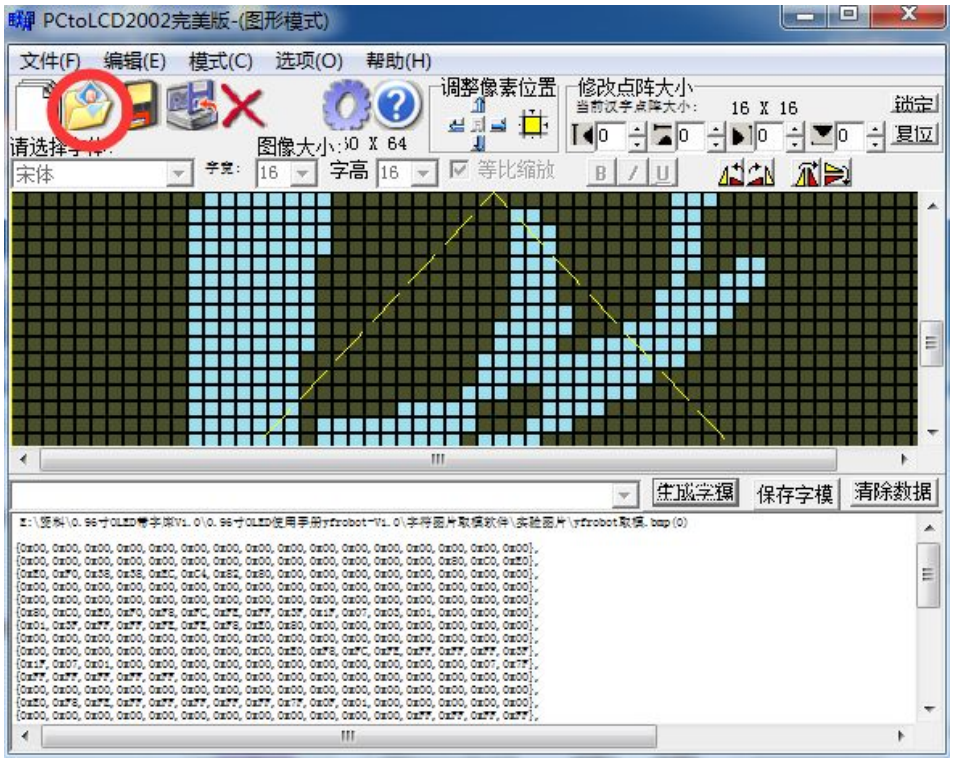



图 2.1.6　图片取模

## 2.2　显示部分

我们先介绍显示部分，显示部分由 128×64 个像素点组成，看下图 2.2.1：

行列控制线与驱动端口（Column 和 Segment）之间的关系，可以看到驱动口 COM63、31、62、30……32、0 分别对应行 ROW1、2、3……63、64。驱动端口 SEG127、126……1、0 分别对应列 Column1、2……127、128。这样驱动芯片就可以驱动对应的点，使得某个点亮或者灭。

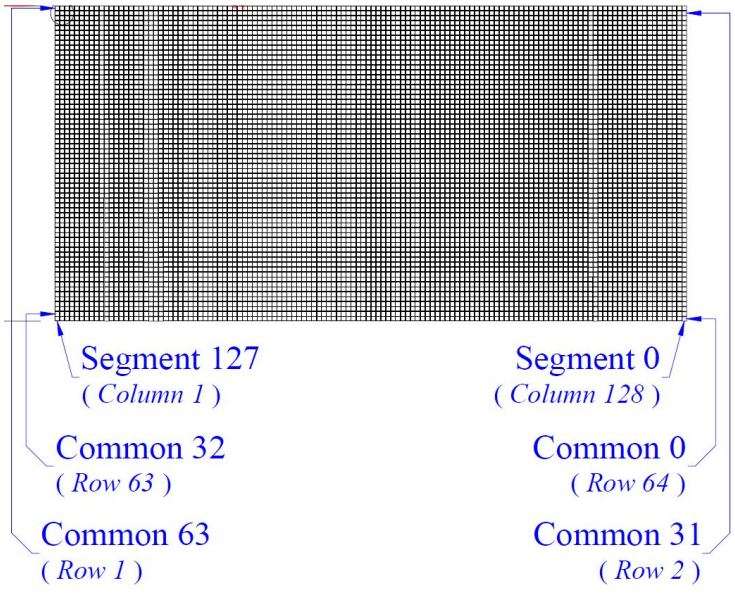



图 2.2.1　行列控制线与驱动端口（Column 和 Segment）之间的关系

以此为基础，我们建立一个显示坐标系，如图 2.2.2，X 轴 Y 轴分别对应列和行，建立此坐标系主要目的是为了更加方便指定显示位置，方便编程。


图 2.2.2　显示坐标系

## 2.3　驱动芯片 SSD1306

SSD1306 是 OLED 核心驱动芯片，已经被封装好固化在显示玻璃中。单片机与 SSD1306 通讯，然后 SSD1306 驱动 OLED，使 OLED 固定的点被驱动点亮。

本节主要讲解数据存储与建立的显示坐标之间的关系。其他寄存器配置详见《SSD1306-Revision 1.1》。看此节时请参考此手册，帮助您理解 SSD1306 的设置

在 SSD1306 的内部有一个 Graphic Display Data RAM (GDDRAM：图形显示数据内存)，它有 128×8 字节，即 128×64 个 Bits，每个 Bits 分别对应 OLED 的 128×64 个点，也就是每个像素点对应一个 Bits。这些字节分别存储在 PAGE0~PAGE7 中，每页存储 128 个字节，如图 2.3.1：

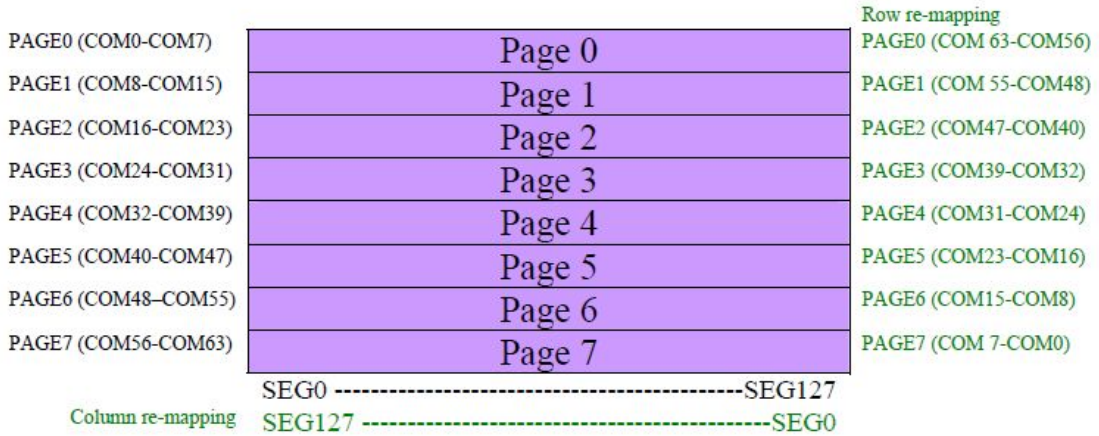



图 2.3.1　SSD1306 中 GDDRAM 页的机构

图 2.3.1 中端口 COM 和端口 SEG 与 GDDRAM 的对应关系是可以软件设置的。

在软件设置的时候用到这个函数：OLED_WCMD(u8 cmd);//向 SSD1306 写命令。这里不详细介绍，在程序设计中详细介绍。

当写入一个字节到 GDDRAM 时，这字节又是怎么存储的呢，请看 2.3.2：

每一个 Bit 存储在一个方格中，再看红色圈出部位，每个方格代表着每个像素点，这样，GDDRAM 和像素点之间就构成了联系。

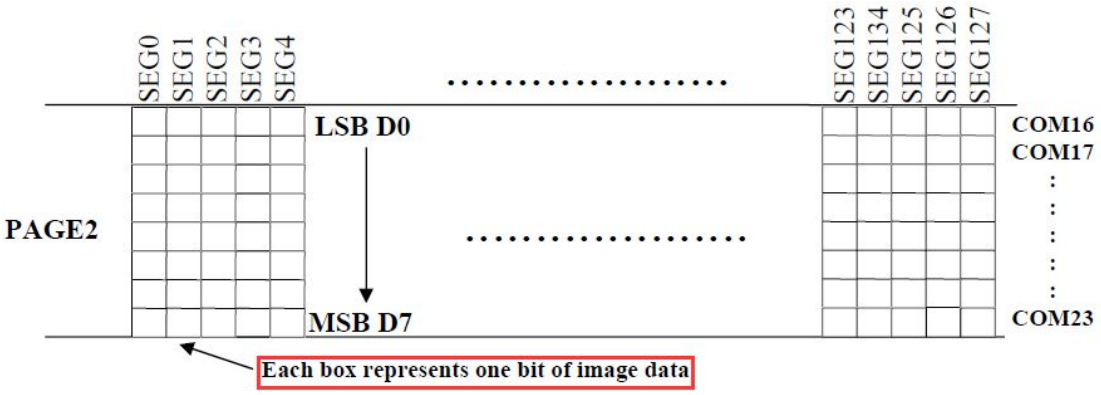



图 2.3.2　GDDRAM 内部存储

这里简单介绍一下《SSD1306-Revision 1.1》，第 9 节主要介绍 SSD1306 的设置方式，第 10 节详细介绍设置的效果，在设置 SSD1306 时，需要将这两章结合起来看。

首先我们设置内存寻址模式：

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0<br>0 | 20<br>A[1:0] | 0<br>* | 0<br>* | 1<br>* | 0<br>* | 0<br>* | 0<br>* | 0<br>$A_1$ | 0<br>$A_0$ | Set Memory<br>Addressing Mode | A[1:0] = 00b, Horizontal Addressing Mode<br>A[1:0] = 01b, Vertical Addressing Mode<br>A[1:0] = 10b, Page Addressing Mode (RESET)<br>A[1:0] = 11b, Invalid |

图 2.2.3　内存寻址模式配置

写命令：

```
OLED_WCMD(0x20);  //设置内存寻址模式
OLED_WCMD(0x02);  //[1:0],00，列地址模式;01，行地址模式;10,页地址模式;默认 10;
```

先发送“设置内存寻址模式”，紧接着发送寻址模式的命令。设置 A[1:0]为 10B，设置为页寻址模式，寻址模式示意图如下图所示，在给 GDDRAM 些数据时，每写一个数据，列地址指针自动加 1，到最后一个地址（COL127）时，地址指针指向列的首地址（COL0），页地址不改变。

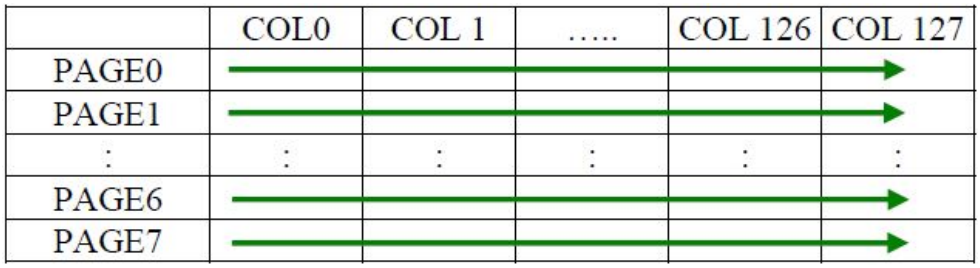



图 2.3.4　页寻址模式地址指针寻址方向

根据图 2.3.1，我们设置：

```
OLED_WCMD(0xA1);//段重定义设置,bit0:0,column0->SEG0;bit0:1,column0->SEG127;
OLED_WCMD(0xC8);  //设置 COM 扫描方向;
OLED_WCMD(0xDA); //设置 COM 硬件引脚配置　[5:4]配置　配置 COM（SSD1306）与 ROW（液晶）的连接关系
OLED_WCMD(0x12);　　//[5:4]配置　配置 COM（SSD1306）与 ROW（液晶）的连接关系
```

首先设置列 Column 与段 Segment 之间的关系，

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | A0/A1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | $X_0$ | Set Segment Re-map | A0h, X[0]=0b: column address 0 is mapped to SEG0 (RESET)<br>A1h, X[0]=1b: column address 127 is mapped to SEG0 |

图 2.3.5　OLED_WCMD(0xA1)意义

再设置 COM，COM 信号端口配置需和 OLED 的硬件配置相匹配，并设置 COM 的输出扫描方向（蓝色箭头），它的顺序为 COM63、31、62、30……32、0。

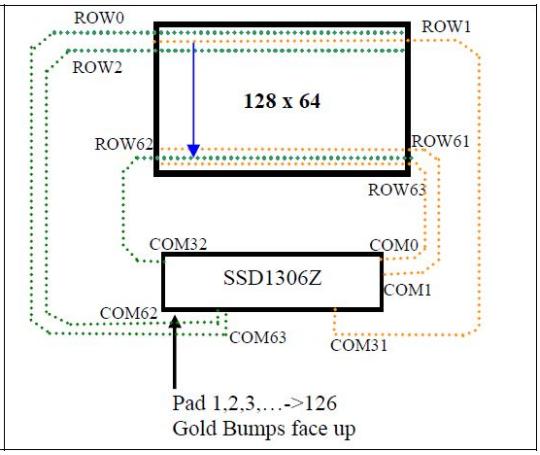
| | |
|---|---|
| 7 Alternative COM pin configuration (DAh A[4] =1)<br>COM output Scan direction: from COM63 to COM0(C8h)<br>Disable COM Left/Right remap (DAh A[5] =0) |  |

图 2.3.6　配置 COM 端与显示 ROW 之间的关系

通过这样的设置，使得 GDDRAM 与建立的坐标系，构成如下关系，如图 2.3.7 所示：

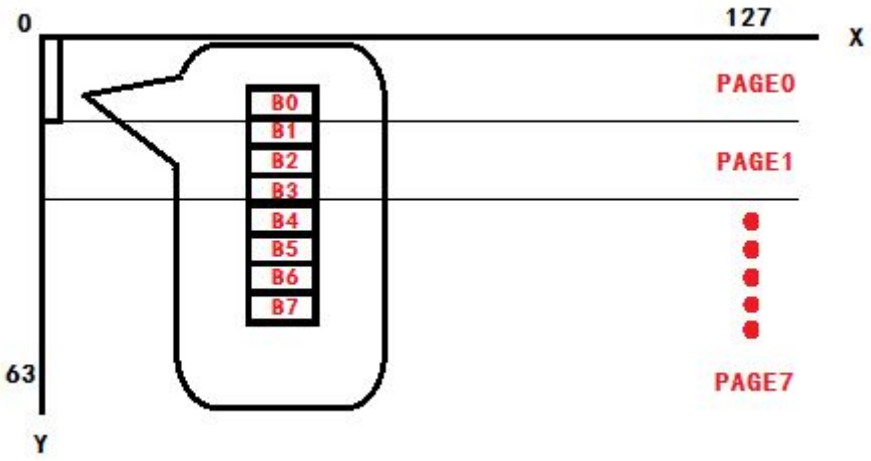



图 2.3.7　坐标系与 GDDRAM 的关系

## 2.4　GT20L16S1Y 字库芯片

GT20L16S1Y（以下简称 GT20）是一款内含 15X16 点阵的汉字库芯片，支持 GB2312 国标简体汉字（含有国家信标委合法授权）、ASCII 字符。排列格式为竖置横排。用户通过字符内码，利用手册提供的方法计算出该字符点阵在芯片中的地址，可从该地址连续读出字符点阵信息。

| 分类 | 字库内容 | 编码体系（字符集） | 字符数 |
|---|---|---|---|
| 汉字及字符 | 15X16 点 GB2312 标准点阵字库 | GB2312 | 6763+376 |
| | 8X16 点国标扩展字符 | GB2312 | 126 |
| ASCII 字符 | 5X7 点 ASCII 字符 | ASCII | 96 |
| | 7X8 点 ASCII 字符 | ASCII | 96 |
| | 8X16 点 ASCII 字符 | ASCII | 96 |
| | 8X16 点 ASCII 粗体字符 | ASCII | 96 |
| | 16 点阵不等宽 ASCII 方头（Arial）字符 | ASCII | 96 |
| | 16 点阵不等宽 ASCII 白正（TimesNewRoman）字符 | ASCII | 96 |

图 2.4.1　芯片所含内容

汉字及字符的内码详见《GB2312 简体中文编码表》，用户只需要知道内码，就可以计算出该字符点阵在芯片的地址，然后从就可从该地址连续读出点阵信息用于显示。

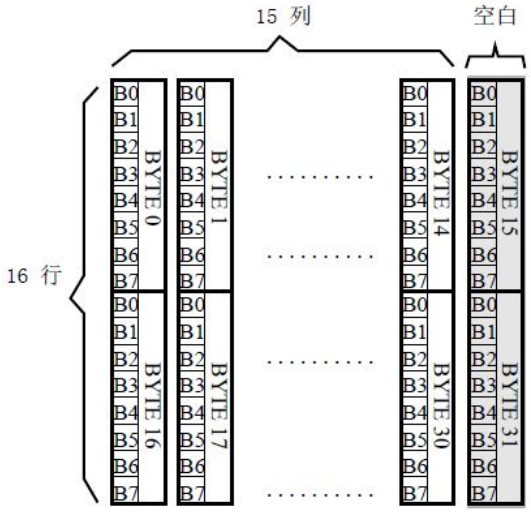



图 2.4.2　15×16 点汉字排列格式

图 2.4.2 为一个汉字的排列格式，由 32 个字节（BYTE0-BYTE31）来表示，32 个字节中，有 BTYE15、BTYE31 是空白，里面无有效信息，但在读取时是连续的，不可省略。这样的存储方式和上面介绍的取模方式相同。

可以看到一个字节的顺序和 GDDRAM 中的字节顺序是相同的，低位在上，高位在下，和建立的坐标系相对应。

不同的字符在 GT20 中的存储位置有对应的算法，就以 15×16 点 GB2312 标准点阵字库为例。

GBCode 标识汉字的内码；

MSB 标识汉字内码 GBCode 的高 8bits；

LSB 标识汉字内码 GBCode 的低 8bits；

Address（24bit）表示汉字或 ASCII 字符点阵在芯片 GT20 中的字节地址；

BaseAdd 为点阵数据在芯片 GT20 中的起始地址。

计数方法：

```
BaseAdd=0;
if(MSB ==0xA9 && LSB >=0xA1)
      Address = (282 + (LSB - 0xA1 ))*32+BaseAdd;
else if(MSB >=0xA1 && MSB <= 0xA3 && LSB >=0xA1)
```

```
        Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
        Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;
```

"啊"的内码为 B0A1

在 GT20 中的起始地址 Address=( (0xB0-0xB0)*94 + (A1-A1)+846 )*32 +0。

其它字库详见手册《GT20L16S1Y 用户手册 V35》。

# 3 通讯方式

OLED 模块中有两个模块需要通讯，第一个是 SSD1306，另一个是 GT20L16S1Y，下面我们将详细介绍。

## 3.1 SSD1306 通讯

SSD1306 提供多种通讯方式，6800 并行通讯、8080 并行通讯、4 线 SPI 通讯、3 线 SPI 通讯、IIC 通讯，但是此 OLED 模块只引出了 4 线 SPI 通讯和 IIC 通讯，默认 4 线 SPI 通讯，例程中暂时未提供 IIC 通讯模式例程。

这里我们详细介绍 4 线 SPI 通讯，下图为时序图：

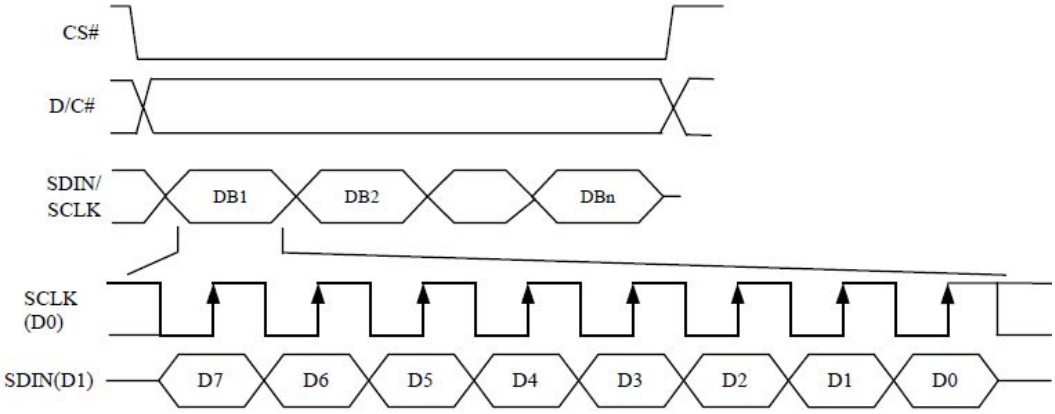



图 3.1.1　4 线 SPI 通讯

CS：片选信号，低电平有效；

DC：命令数据标志位，0：写命令；1：写数据；

SCLK：串行时钟线，上升沿时，数据或命令被写入。在 4 线模式下，D0 作为串行时钟线；

SDIN：串行数据线。在 4 线模式下，D1 作为串行时钟线。

根据上述时序，写出以下两个函数分别是 void OLED_WCMD(u8 cmd)//写命令、void OLED_WDAT(u8 dat)//写数据。

```
/**********************************************
Function:  void OLED_WCMD(u8 cmd)
Description: 向 SSD1306 写命令
Input:          cmd：要写入的命令
Others:         OLED_DC=0：表示写命令；=1 写数据
**********************************************/
void OLED_WCMD(u8 cmd)
{
    u8 i;
    OLED_DC=0; //写命令
    OLED_CS=0;
    for(i=0;i<8;i++)
```

```
        {
            OLED_CLK=0;
            if(cmd&0x80)OLED_DIN=1;
            else OLED_DIN=0;
            OLED_CLK=1;
            cmd<<=1;
        }
        OLED_CS=1;
        OLED_DC=1;
    }
    /*********************************************************
    Function:  void OLED_WDAT(u8 dat)
    Description: 向 SSD1306 写数据
    Input:          dat：要写入的数据
    Others:         OLED_DC=0：表示写命令；=1 写数据
    *********************************************************/
    void OLED_WDAT(u8 dat)
    {
        u8 i;
        OLED_DC=1; //写数据
        OLED_CS=0;
        for(i=0;i<8;i++)
        {
            OLED_CLK=0;
            if(dat&0x80)OLED_DIN=1;
            else OLED_DIN=0;
            OLED_CLK=1;
            dat<<=1;
        }
        OLED_CS=1;
        OLED_DC=1;
    }
```

通过这两个函数就可以实现单片与 SSD1306 之间的通讯。

## 3.2 GT20L16S1Y 通讯

GT20L16S1Y（以下简称 GT20）通讯为 4 线 SPI 模式，对芯片操作有两种方式，一种是一般读取，另一种是快速读取点阵数据，例程中我们使用的是一般读取。

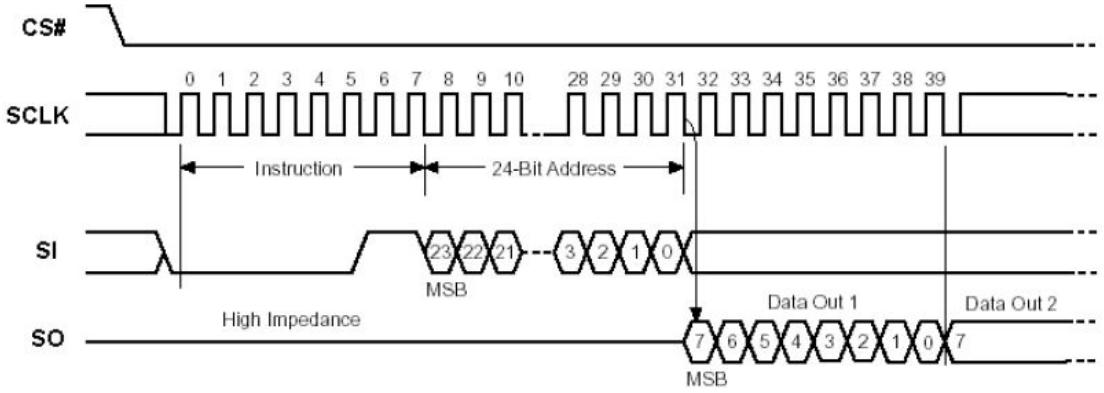



图 3.2.1　GT20 一般读取

信号意义：

CS：片选信号，所有串行数据传输开始于 CS 下降沿，CS 在传输期间必须保持低电平，在两条指令之间必须为高电平；

SCLK：串行时钟信号，上升沿移入，在下降沿移出：

SI：串行数据输入，该信号用来把数据串行输入 GT20，数据在时钟的上升沿移入；

SO：串行数据输出，该信号用来把数据从 GT20 串行输出，数据在时钟下降沿移出。

时序解读：

1、首先把片选信号 CS 变低，紧跟着的是 1 个字节的命令（Instruction）字 03H 和 3 个字节的地址，通过引脚 SI 移位输入，每一位在串行时钟 SCLK 上升沿被锁存；

2、然后该地址的字节数据通过串行数据输出引脚 SO 移位输出，每一位在串行时钟 SCLK 下降沿被移出。根据字符所占字节个数，读出相应的字节个数；

3、读取字节数据后，片选信号 CS 变为高，结束本次操作。

如果片选信号 CS 持续保持为低，则下一个地址的字节数据继续通过串行数据输出引脚 SO 移位输出，所以两条指令之间必须把 CS 拉高。

根据上述通讯规则，写成一下两个函数：

```
/*****************************************************
Function:    void GT20_WCMD(u8 cmd)
Description:   向 GT20L16 写命令
Input:              u8 cmd  命令符
Others:             时钟上升沿，数据输入 GT20
*****************************************************/
void GT20_WCMD(u8 cmd)
{
      u8 i;
      for(i=0;i<8;i++)
      {
            GT20_CLK=0;
            if(cmd&0x80) GT20_SI=1;
            else GT20_SI=0;
```

```
            GT20_CLK=1;
            cmd<<=1;
        }
    }
    /*****************************************************
    Function:    static u8 GT20_RDAT( )
    Description:   读 GT20L16 中字库的汉字或字符数据
    Output:          data  一个字节，用于显示汉字或字符
    Others:          时钟下降沿，读 GT20 数据
    *****************************************************/
    static u8 GT20_RDAT(void)
    {
        u16 i;
        u8 data=0x00;
        GT20_CLK = 1;
        for(i=0;i<8;i++)
        {
            GT20_CLK = 1;
            data = data<<1;
            GT20_CLK = 0;
            if(GT20_SO)
                data = data+1;
        }
        return data;
    }
```

因为这两个函数不是一个完整的指令，所以在这两个函数中未使用片选信号 GT20_CS。在第 5 节中，将详细介绍取一个汉字或字符的点阵字模。

# 4 连接方式

这节介绍模块输出引脚和芯片引脚之间的关系，以及端口在程序中的命名。

GND：接地；

VCC：兼容 3.3V 和 5V。

电源不可接反和过压！

根据不同的单片机和工程需求可以对端口自由变动，根据 YFROBOT 提供的例程，进行如下初始化：

丝印及程序中端口命名有如下关系：

表 4.1 端口及函数命名

| PCB 丝印 | SSD1306 4SPI 通讯 | | SSD1306 IIC 通讯 | | GT20 SPI 通讯 | | 单片机端口 |
|---|---|---|---|---|---|---|---|
| | 端口名 | 函数定义 | 端口名 | 函数定义 | 端口名 | 函数定义 | |
| CLK | SCLK | OLED_CLK | SCL | | SCLK | GT20_CLK | PC8 |
| DIN | SDIN | OLED_DIN | SDA | | SI | GT20_SI | PC9 |
| D/C | DC | OLED_DC | 接 GND | | | | PC10 |
| CS1 | CS | OLED_CS | 接 GND | | | | PC11 |
| SO | | | | | SO | GT20_SO | PC12 |
| CS2 | | | | | CS | GT20_CS | PC13 |

```
/*******************************************************
Function:   void OLED_Init(void)
Description:  驱动端口初始化、OLED 与 GT20 驱动
                C8  (输出) -- CLK       时钟
                C9  (输出) -- DIN       数据   SSD1306 输入、GT20L16 输入
                C10 (输出) -- D/S    命令/数据
                C11 (输出) -- CS1    SSD1306 片选
                C12 (输入) -- SO     数据输入，GT20L16 输出
                C13 (输出) -- CS2    GT20L16 片选
Calls by:   void OLED_SetInit(void)
*******************************************************/
void OLED_Init(void)
{
    RCC->APB2ENR|=1<<4;      //使能 PORTC 时钟
    GPIOC->CRH&=0XFF000000;  //输出
    GPIOC->CRH|=0X00383333;

    GPIOC->ODR|=0XF<<8;          //输入   高电平
    GPIOC->ODR&=~(1<<12);
}
```

Oled.h 文件中，端口定义：

```
//-----------------OLED 端口定义----------------
#define OLED_CLK PCout(8)
#define OLED_DIN PCout(9)
#define OLED_DC    PCout(10)
#define OLED_CS    PCout(11)

#define GT20_CLK PCout(8)
#define GT20_SI    PCout(9)
#define GT20_SO    PCin(12)
#define GT20_CS    PCout(13)
```

# 5 程序设计

完整程序请详见《STM32 OLED 显示实验》，这里主要介绍程序的核心部分。

首先需要对 SSD1306 进行初始化，保证 OLED 能够被 SSD1306 有效驱动

```
/**********************************************
Function:  void OLED_Init(void)
Description: 驱动端口初始化、SSD1306 初始化。
Calls:          void OLED_Init(void);   //驱动端口初始化
                void OLED_WCMD(u8 cmd);  //向 SSD1306 写命令
**********************************************/
void OLED_SetInit(void)
{
    OLED_Init();                //端口初始化
    OLED_WCMD(0xAE);    //关闭显示
    OLED_WCMD(0xD5);    //设置时钟分频因子,震荡频率
    OLED_WCMD(0x50);     //[3:0],分频因子;[7:4],震荡频率
    OLED_WCMD(0xA8);    //设置驱动路数
    OLED_WCMD(0x3F);        //默认 0X3F(1/64)
    OLED_WCMD(0xD3);    //设置显示偏移
    OLED_WCMD(0x00);        //默认为 0
    OLED_WCMD(0x40);    //设置显示开始行 [5:0],行数.
    OLED_WCMD(0x8D);    //电荷泵设置
    OLED_WCMD(0x14);    //bit2，开启/关闭
    OLED_WCMD(0x20);  //设置内存寻址模式
    OLED_WCMD(0x02);// [1:0],00，列地址模式;01，行地址模式;10,页地址模式;默认
10;
    OLED_WCMD(0xA1);  //段重定义设置,bit0: 0,column 0->SEG 0;   1,column 0->SEG
127;
    OLED_WCMD(0xC8);  //设置 COM 扫描方向;bit3:0,普通模式;1,重定义模式
COM[N-1]->COM0;N:驱动路数
    OLED_WCMD(0xDA); //设置 COM 硬件引脚配置    [5:4]配置    配置 COM
（SSD1306）与 ROW（液晶）的连接关系
    OLED_WCMD(0x12);        //[5:4]配置   配置 COM（SSD1306）与 ROW（液晶）
的连接关系
    OLED_WCMD(0x81);  //对比度设置
    OLED_WCMD(0xEF);  //1~255;默认 0X7F (亮度设置,越大越亮)
    OLED_WCMD(0xD9);  //设置预充电周期
    OLED_WCMD(0xF1);  //[3:0],PHASE 1;[7:4],PHASE 2;
```

```
        OLED_WCMD(0xDB); //设置 VCOMH  电压倍率
        OLED_WCMD(0x30);  //[6:4] 000,0.65*vcc;001,0.77*vcc;011,0.83*vcc;
        OLED_WCMD(0xA4); //全局显示开启;bit0:1,开启;0,关闭;(白屏/黑屏)
        OLED_WCMD(0xA6); //设置显示方式;bit0:1,反相显示;0,正常显示
        OLED_WCMD(0xAF); //开启显示
        OLED_Clear();           //清屏
    }
```

关于 SSD1306 寄存器的配置请详见手册《SSD1306-Revision 1.1》。

设计一个显存数组 OLED_GRAM[128][8]，和 SSD1306 中图形显示数据内存 GDDRAM 相对应。

```
    //OLED 的显存
    //存放格式如下.
    //PAGE0   [0]0 1 2 3 ... 127          位高低顺序   D0
    //PAGE1   [1]0 1 2 3 ... 127                       D1
    //PAGE2   [2]0 1 2 3 ... 127                       D2
    //PAGE3   [3]0 1 2 3 ... 127                       D3
    //PAGE4   [4]0 1 2 3 ... 127                       D4
    //PAGE5   [5]0 1 2 3 ... 127                       D5
    //PAGE6   [6]0 1 2 3 ... 127                       D6
    //PAGE7   [7]0 1 2 3 ... 127                       D7
    u8 OLED_GRAM[128][8];
    /*****************************************************
    Function:  void OLED_Refresh_Gram(void)
    Description:   更新显存到 OLED，将 OLED_GRAM 中的内容更新到
                    SSD1306 的 Graphic Display Data RAM (GDDRAM)
                    每次显示新内容时都需要调用此函数。
    *****************************************************/
    void OLED_Refresh_Gram(void)
    {
        u8 i,n;
        for(i=0;i<8;i++)
        {
            OLED_WCMD(0xB0+i);//设置页地址（0~7）每页中的 GDDRAM 列地址是自动增加的。
            OLED_WCMD(0x00);   //设置显示位置—列低地址
            OLED_WCMD(0x10);   //设置显示位置—列高地址
            for(n=0;n<128;n++)
                OLED_WDAT(OLED_GRAM[n][i]); //#define OLED_DATA 1 //写数据
```

```
        }
    }
```

在 2.2 节中，已经详细介绍了设置 GDDRAM 与像素点之间的关系，这里只需要将 OLED 的显存写入 GDDRAM 就可以了，每次改变显示内容的时候只需要改变数组 OLED_GRAM[128][8]中值就可，再将值更新到 GDDRAM。

点亮 OLED 上的指定像素点：

```
/**********************************************************
Function:  void OLED_DrawPoint(u8 x,u8 y,u8 t)
Description: 在屏幕上任一位置画点，填充或清空
Input:       x：0~127
             y:0~63
             t:1  填充  0,清空
**********************************************************/
void OLED_DrawPoint(u8 x,u8 y,u8 t)
{
    u8 pos,bx,temp=0;
    if(x>127||y>63)return;//超出范围了.
    pos=y/8;
    bx=y%8;
    temp=1<<bx;
    if(t)OLED_GRAM[x][pos]|=temp;
    else OLED_GRAM[x][pos]&=~temp;
}
```

算法比较简单，请结合坐标系和 GDDRAM 关系理解里面的算法。

```
/**********************************************************
Function: void OLED_Photo( u8 x, u8 y, u8 len, u8 wide, const u8 *pic, u8 mode)
Description:从任意点开始画指点大小的图片
Calls：  void OLED_DrawPoint(u8 x,u8 y,u8 t) //任意坐标画点
Input:    u8 x，u8 y：起始点。
          u8 len：长，x 轴方向
          u8 wide：宽，y 轴方向
          u8 *pic：图片数组
          u8 mode：字符显示区域为 0 或 1
**********************************************************/
void OLED_Picture( u8 x, u8 y, u8 len, u8 wide, const u8 *pic, u8 mode)
{
    u8 x_pos,y_pos,temp;
    u8 i,j;
```

```
        x_pos = x;
        y_pos = y;
        for(j=0; j<len; j++)
        {
            for(i=0;i<wide;i++)
            {
                if(i%8==0)
                {
                    temp = *(pic+(i/8)*len+j);
                }
                if(temp&0x01)       OLED_DrawPoint( x_pos+j,y_pos+i,mode);
                else  OLED_DrawPoint( x_pos+j,y_pos+i,!mode);                //某“点”写 0，
或写 1
                temp >>=1;
            }
        }
    }
```

图片的取模方式和 2.1 节介绍的取模方式相同。图片“yfrobot”的点阵模数组存储在文件“picture.h”中。

以上都是操作 SSD1306，使得液晶上的显示，下面介绍通过读取 GT20 中点阵字模，通过 OLED 显示。

```
    /*****************************************************
    Function:  void GT20_GetBytes(u8 addrH, u8 addrM, u8 addrL,u8 *fontBuf,u8 dataLen)
    Description:  读取点阵字模
    Calls:          void GT20_WCMD(u8 cmd)          //向 GT20L16 写命令
                    static u8 GT20_RDAT(void)   //读 GT20L16 中字库的汉字或字符数据
    Input:      u8 addrH    高地址
                u8 addrM    中地址
                u8 addrL    低地址
                u8 *fontBuf    数组存储指针
                u8 dataLen      汉字或字符  点阵字模字节个数
    Others:   汉字和字符存储的字节不同，根据实际情况进行修改
    *****************************************************/
    void GT20_GetBytes(u8 addrH, u8 addrM, u8 addrL,u8 *fontBuf,u8 dataLen)
    {
        u8 i;
        GT20_CS = 1;
        OLED_CS = 1;
```

```
    GT20_CS = 0;     //片选  低电平为有效电平
    GT20_CLK = 0;
    GT20_WCMD(0x03);   //通讯起始命令字（03H）
    GT20_WCMD(addrH);
    GT20_WCMD(addrM);
    GT20_WCMD(addrL);
    for(i=0;i<dataLen; i++)
    {
        *(fontBuf+i) = GT20_RDAT( );
    }
    delay_us(2);
    GT20_CS = 1;
}
```

不同字符，使用的字节个数不同，15×16 点汉字需要 32 个字节，5×7 点 ASCII 的信息需要 8 个字节，根据字节个数的不同，设置“dataLen”值的大小。

如何显示这些字符？我们把这些字符看作是不同规格的图片，15×16 点汉字，就看作 15×16 的图片。

```
/*****************************************************
Function: void OLED_GT20Photo( u8 x, u8 y, u8 len, u8 wide, u8 *pic, u8 mode)
Description:从任意点开始画指点大小的图片
Calls：  void OLED_DrawPoint(u8 x,u8 y,u8 t) //任意坐标画点
Input:    u8 x，u8 y：起始点。
          u8 len：长，x 轴方向
          u8 wide：宽，y 轴方向
          u8 *pic：图片数组
          u8 mode：字符显示区域为 0 或 1
*****************************************************/
void OLED_GT20Photo( u8 x, u8 y, u8 len, u8 wide, u8 *pic, u8 mode)
{
    u8 x_pos,y_pos,temp;
    u8 i,j;
    x_pos = x;
    y_pos = y;
    for(j=0; j<len; j++)
    {
        for(i=0;i<wide;i++)
        {
            if(i%8==0)
```

```
                {
                    temp = *(pic+(i/8)*len+j);
                }
                if(temp&0x01)       OLED_DrawPoint( x_pos+j,y_pos+i,mode);
                else  OLED_DrawPoint( x_pos+j,y_pos+i,!mode);              //某“点”写 0，
或写 1
                temp >>=1;
            }
        }
    }
```

通过这个函数，就可以将得到的汉字或字符数组，更新到 OLED_GRAM[128][8]数组中相应位置，从而被显示。

```
    /*****************************************************
    Function:  void OLED_Show8*16BoldString(u8 x,u8 y,u8 *p,u8 mode)
    Description: 在任意点显示 8*16 点 ASCII 粗体字符串
    Calls：void GT20_GetBytes(u8 addrH, u8 addrM, u8 addrL,u8 *fontBuf,u8 dataLen); //读取
点阵字模
             void OLED_GT20Photo( u8 x, u8 y, u8 len, u8 wide, u8 *pic, u8 mode);        //从
任意点开始画指点大小的图片
    Input:    u8 x，u8 y：起始点。
              u8 *text：图片数组
              u8 mode：字符显示区域为 0 或 1
    *****************************************************/
    void OLED_Show8X16BoldString(u8 x,u8 y,u8 *text,u8 mode)
    {
        u8 i = 0;
        u8 addrH, addrM, addrL;
        u8 FONTBUF[16];
        while(text[i]>0x00)
        {
            if((text[i]>=0x20)&&(text[i]<=0x7E))
            {
                /*8*16 点 ASCII 粗体在字库 IC 中的地址由以下公式来计算：*/
                /*Address = (ASCIICode －0x20 ) * 16+BaseAdd;BaseAdd=0x3CF80*/
                /*由于担心 8 位单片机有乘法溢出问题，所以分三部取地址*/
                FONTADDR = (text[i]- 0x20)*16+0x3CF80;

                addrH = (FONTADDR&0xff0000)>>16;   //地址的高 8 位,共 24 位
```

```
                addrM = (FONTADDR&0xff00)>>8;        //地址的中 8 位,共 24 位
                addrL = FONTADDR&0xff;               //地址的低 8 位,共 24 位
                GT20_GetBytes(addrH,addrM,addrL,FONTBUF,16 );//取 16 个字节的数据，
存到"FONTBUF[16]"
                OLED_GT20Photo( x, y, 8, 16, FONTBUF, mode);      //更新 GRAM，显示
为填充
                i++;
                x+=8;
            }
            else
                i++;
        }
    }
```

通过字符内码，计算出字符数组在 GT20 中的存储地址，通过函数 GT20_GetBytes(u8 addrH, u8 addrM, u8 addrL,u8 *fontBuf,u8 dataLen);读取点阵字模数组，再通过函数 OLED_GT20Photo( u8 x, u8 y, u8 len, u8 wide, u8 *pic, u8 mode)字符数组更新到 OLED_GRAM[128][8]数组中。在使用此函数后，都需要调用 OLED_Refresh_Gram(void)

更多的显示方式请详见工程，在这就不一一例举。

正确的连接器件，下载程序，OLED 中就可显示您想要的效果。

# 附录 A：更新说明

V1.0(201501120：1、使用 4 线 SIP 串口通讯；

2、读取 GT20L16S1Y 字库芯片，并显示。

# 附录 B：联系方式

YFROBOT 电子工作室

地址：江苏省淮安市工业园区纬六路 20 号　　邮编：223000

电话：0517-84899889　　传真：0517-84899889

网址：www.yfrobot.com　　Email：yfrobot@126.com