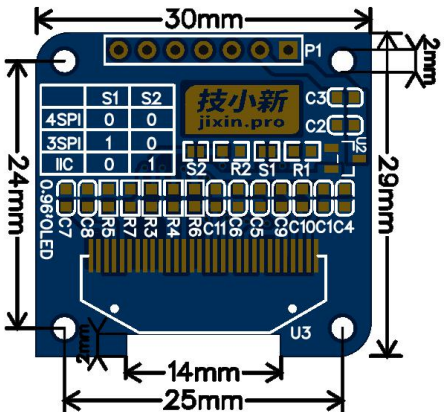
0.96’OLED（7Pin）模块学习手册（STM8S 版）

1、模块介绍

1.1 模块特点

0.96’OLED（7Pin）模块采用 SSD1306 为主芯片，像素为 128*64，通讯方式可选择 SPI 或 IIC（地址默认 0x78），引脚完全兼容 IIC（即在设置为 IIC 模式时，可只接 4 根线），默认是 4-Wire SPI 通讯模式，自发光自由视角，功耗低。主要特点如下：

- 兼容 3.3V 或 5V 电源输入
- 兼容 3.3V 或 5V IO 口电平
- 通讯方式可选择 SPI（4-Wire 或 3-Wire）/IIC



0.96’OLED（7Pin）模块

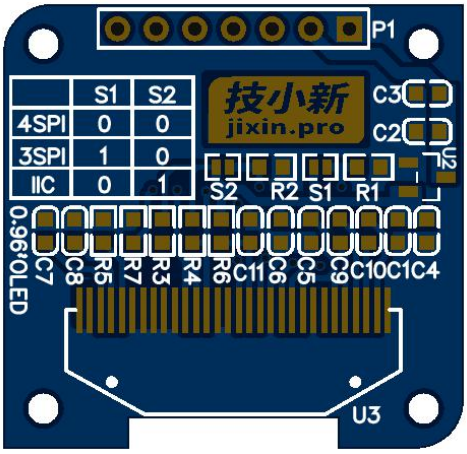
1.2 模块接口引脚

Symbol（符号）	Type（类型）	Description（描述）
GND	电源	接地引脚
VCC	电源	电源输入引脚
D0	输入	时钟输入
D1	输入/输出	数据输入/应答输出
RES	输入	复位信号输入
DC	输入	数据/命令选择输入
CS	输入	片选引脚

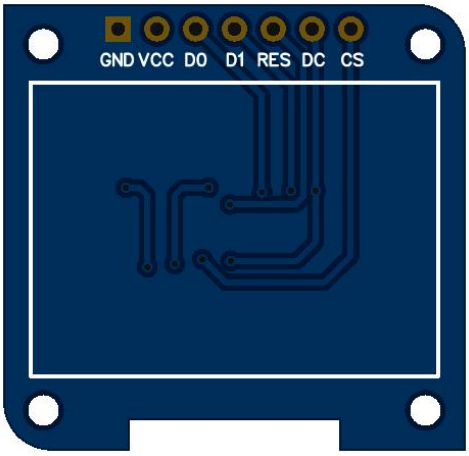
模块引脚接口功能表

1.3 模块丝印

模块上的每一个器件都有一个相对应的丝印在上面，可以通过 BOM 表中的器件编号在板子找到器件的位置。



模块正面丝印图



模块正面丝印图

在模块的正面丝印中，S1 与 S2 分别是两个焊盘，作用是选择模块与 MCU 的通讯方式（不焊为 0，焊上为 1，默认 S1 与 S2 是不焊上的，模式为 4-WireSPI），通讯方式选择如下表：

通讯模式	S1	S2
4-WireSPI（4 线 SPI）	0	0
3-WireSPI（3 线 SPI）	1	0
IIC	0	1

通讯方式选择表

2、模块用途

OLED，即有机发光二极管（ Organic Light Emitting Diode）。 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、可用于挠曲性面板、使用温度范围广、构造及制程较简单等特性，被认为是下一代的平面显示器新兴应用技术 。

0.96’OLED（7Pin）模块采用 SSD1306 为驱动芯片，并集成在 OLED 面板中，像素为 128*64，通讯方式可选择 SPI（4-WireSPI 与 3-WireSPI）/IIC，功耗低，自发光自由视角。模块带有稳压芯片，VCC 输入范围 3.3V~5V。接口兼容 0.96’OLED（4Pin）模块。

2.1 小型智能设备显示屏

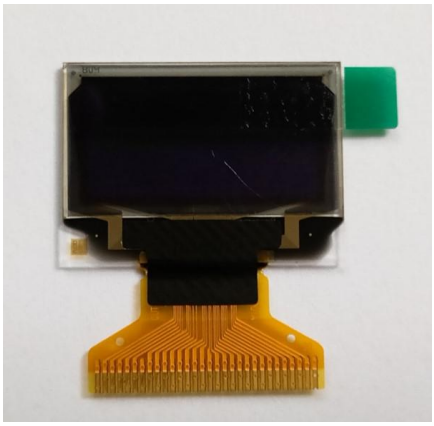
OLED 有低功耗、视角广、分辨率高与自发光的特点在显示器中有很大的优势，但目前由于技术与成本的原因，目前更多是应用在小型的智能设备中，如智能手环等。

2.2 人机交互界面

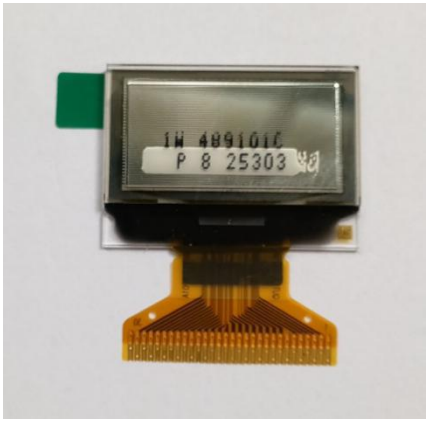
在调试设备或者测试数据时，有时候需要实时观察数据的变化，加入显示屏可以把观察设备的运行情况，数据变化等。在成本和难易程度上，OLED 显示屏是非常适合初学者去学习与应用的。

3.硬件设计

硬件设计主要介绍以 SSD1306 芯片为核心，设计出一个通讯方式可选（4-WireSPI/3-WireSPI/IIC）的 OLED 模块（也就是该模块）。这里用的是全智景一款 0.96 寸的 SSD1306，官方手册与技术手册可以在技新网的产品中心 https://www.jixin.pro/product/527.html 下面的教程与资料中下载。特别说明一下，在实际中见到的 OLED 屏中已经把 SSD1306 内嵌到里面了，由于 SSD1306 芯片的驱动管脚有限，一般做成 0.96 寸 120*64 及以下的规格，如下图：

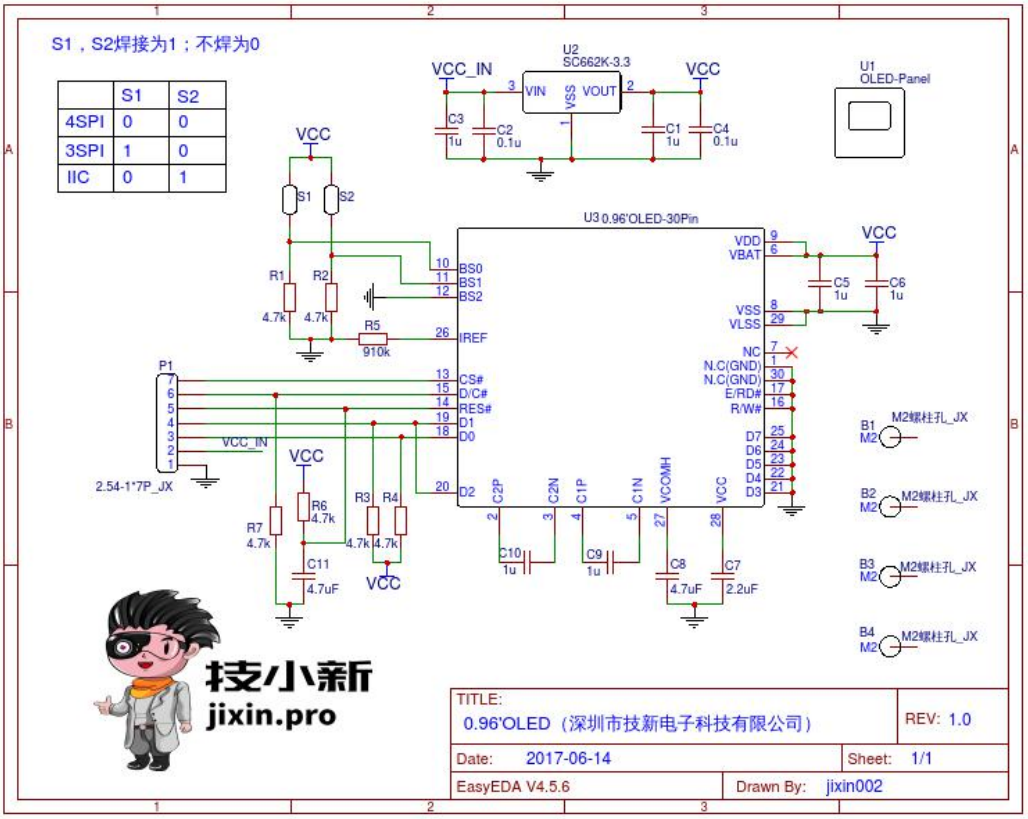


SSD1306 正面图



SSD1306 背面图

3.1 模块原理图



模块原理图

3.2 SSD1306 引脚功能表

<table>
<tr><th>引脚号码</th><th>符号</th><th>类型</th><th>功能</th></tr>
<tr><td colspan="4">电源相关引脚</td></tr>
<tr><td>9</td><td>VDD</td><td>电源</td><td>逻辑电源
这是一个供电引脚，必须连接到外部电源。</td></tr>
<tr><td>8</td><td>VSS</td><td>电源</td><td>逻辑地
这是一个地引脚，它作为逻辑电平的参考引脚，必须链接到外部地。</td></tr>
<tr><td>28</td><td>VCC</td><td>电源</td><td>OEL 板供电
它必须为芯片电源电压正极引脚，在使用整流器时要在该引脚与 V_{SS} 之间接入一个电容；如果不使用整流器时必须接到外部电源。</td></tr>
<tr><td>29</td><td>VLSS</td><td>电源</td><td>模拟地
这是一个模拟地引脚，它应该连接到外部 V_{SS}。</td></tr>
<tr><td colspan="4">驱动器相关引脚</td></tr>
<tr><td>26</td><td>IREF</td><td>输入</td><td>亮度调节电流参考
这是 segment 参考电流引脚，该引脚与 V_{SS} 之间应该接入一个电阻，设置电流在最大值 12.5uA</td></tr>
<tr><td>27</td><td>VCOMH</td><td>输出</td><td>COM 信号输出高电平电压
这是一个 COM 信号输出高电平电压的输入引脚，在这个引脚与 V_{SS} 之间应该介入一个电容</td></tr>
<tr><td colspan="4">DC/DC 转换相关引脚</td></tr>
<tr><td>6</td><td>VBAT</td><td>电源</td><td>DC/DC 转换电路电源
这是 DC/DC 转换器内部缓冲区的电源引脚，
使用时必须与外部电源连接；不使用时接到 V_{DD}</td></tr>
<tr><td>4/5
2/3</td><td>C1P/C1N
C2P/C2N</td><td>输入</td><td>反相电容器的正端
反相电容器的负端
使用电荷泵电路时两端需要接一个电容，在不使用时必须浮空</td></tr>
<tr><td colspan="4">通讯接口配置相关引脚</td></tr>
<tr><td>10
11
12</td><td>BS0
BS1
BS2</td><td>输入</td><td>通讯方式选择
这些引脚由 MCU 接口选择输入，选择如下
<table>
<tr><th></th><th>BS0</th><th>BS1</th><th>BS2</th></tr>
<tr><td>IIC</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>3-wire SPI</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>4-wire SPI</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>8-bit 68xx Parallel</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>8-bit 80xx Parallel</td><td>0</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>14</td><td>RES#</td><td>输入</td><td>控制器与驱动器的电源复位</td></tr>
</table>

			此引脚是复位信号的输入，低电平的时候复位，在正常运行时应该将此引脚保持高电平
13	CS#	输入	**片选** 此引脚是片选输入引脚，只有在此引脚被拉低是使能后才能与 MCU 进行通讯
15	D/C#	输入	**数据/命令控制** 此引脚是数据/命令控制引脚。当此引脚被拉高时，D7~D0 的输入作为数据；当此引脚被拉低时，D7~D0 的输入将传送到命令寄存器。 在串行接口模式下，此引脚被拉高时，SDIN 的输入作为数据；此引脚被拉低时 SDIN 的输入将传送到命令寄存器。 在 IIC 模式，此引脚作为从机选择 SA0 位。
17	E/RD#	输入	**读写使能/读** 此引脚由 MCU 输入。当作为 68xx 系列通讯接口时，此引脚作为使能信号，此引脚拉高和 CS#引脚拉低时读/写操作是可行的。 当作为 80xx 系列通讯接口时，此引脚作为读信号引脚，当此引脚是拉低与 CS#引脚拉低时，读操作是可行的。 在串行或 IIC 模式下，此引脚必须接 V_{SS}。
16	R/W#	输入	**读写选择/写** 此引脚由 MCU 输入。当作为 68xx 系列通讯接口时，此引脚作为读/写选择，此引脚拉高为读模式，拉低为写模式。 当选择为 80xx 通讯接口时，此引脚作为写信号引脚，当此引脚是拉低与 CS#引脚拉低时，读操作是可行的。 在串行或 IIC 模式下，此引脚必须接 V_{SS}。
18~25	D0~D7	输入/输出	**主机数据输入/输出总线** 这些引脚作为 8 位双向的数据总线连接到单片机的数据总线中。 当选择为串行通讯方式时，D1 将作为串行数据输入 SDIN，D0 将作为串行书中输入 SCLK。 当选择为 IIC 模式时，D2 与 D1 将一起作为 SDA_{OUT} 与 SDA_{in} 使用，D0 作为串行时钟输入 SCL 没有使用的引脚除了在串行模式下的 D2，必须连接到 V_{SS}
保留			

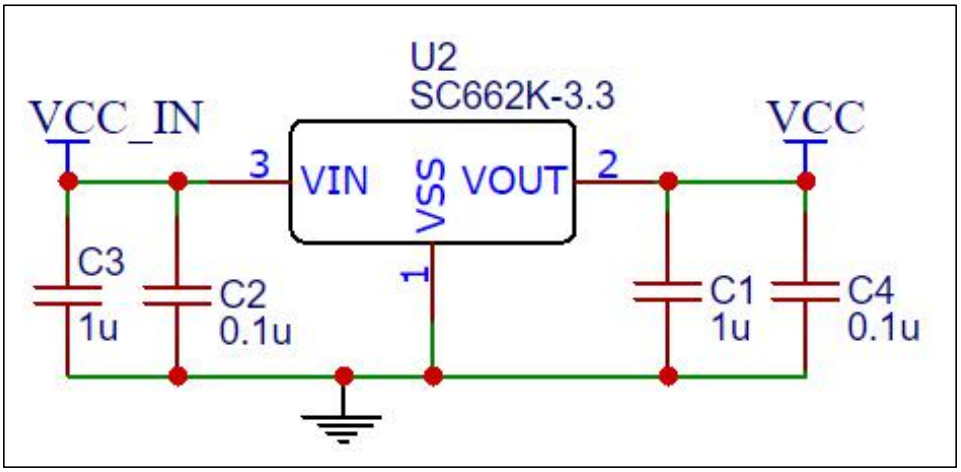
7	N.C.	悬空	**保留引脚** 悬空
1,30	N.C.(GND)	悬空	**保留引脚** 减少功能引脚的压降影响，作为防静电保护电路作用必须接到外部地

上引脚功能表是根据官方手册翻译过来的，详细内容可参考官方手册。0.96 寸的 SSD1306 的引脚引出有 30 个，根据引脚的作用分为电源、驱动、DC/DC、通讯等，另外还有一些保留的引脚，电路设计根据引脚功能分为电源电路设计，通讯方式选择电路设计，通讯接口电路设计和其他部分电路设计。

3.3 模块的电源电路设计

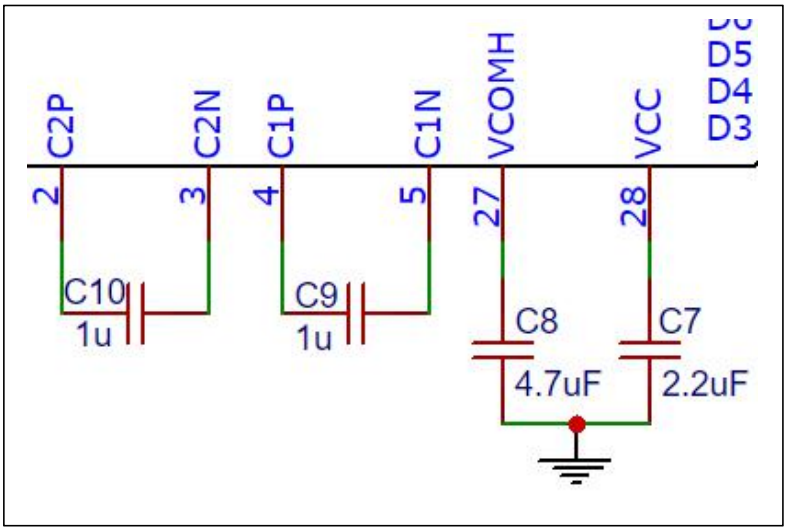
SSD1306 的电源有 V_{CC}：显示屏工作电压；V_{DD}：逻辑电路工作电压。V_{CC} 的工作电压比较高，但是有两种供电方式：第一种是外部给 V_{CC} 供电，电压范围 8.5V~9.5V；第二种是使用 SSD1306 的内部 DC/DC 电路生成的电压对 V_{CC} 进行供电，供电范围 7.0V~7.5V（可参考官方手册 2、3.2 部分内容）。一般的系统的工作电压是 3.3V 或 5V，所以电源电路设计采用 SSD1306 的内部 DC/DC 设计（可参考官方手册 1.62、部分内容），VCC 由内部的 DC/DC 电路供电，只需要给 V_{BAT} 与 V_{DD} 供电，供电电压为 3.3V。

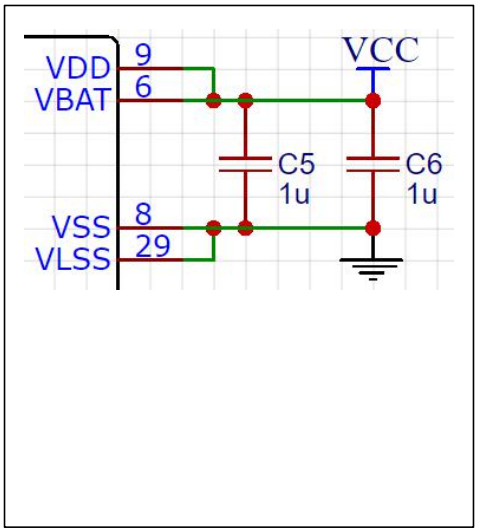
确定了供电的电压为 3.3V 后，为了兼容 5V 的系统电路设计一个 LDO 电路可以将 5V 稳压成 3.3V 给 SSD1306 供电，这样既能兼容 3.3V 输入也能兼容 5V 输入。



LDO 稳压电路

使用 SSD1306 内部 DC/DC 电路需要对其相关的引脚进行设计，如 VBAT 要接外部电源，C1P/C1N 与 C2P/C2N 要接一个电容等，还有其他电源相关引脚（参考引脚功能表与官方手册 1.62 内容）。

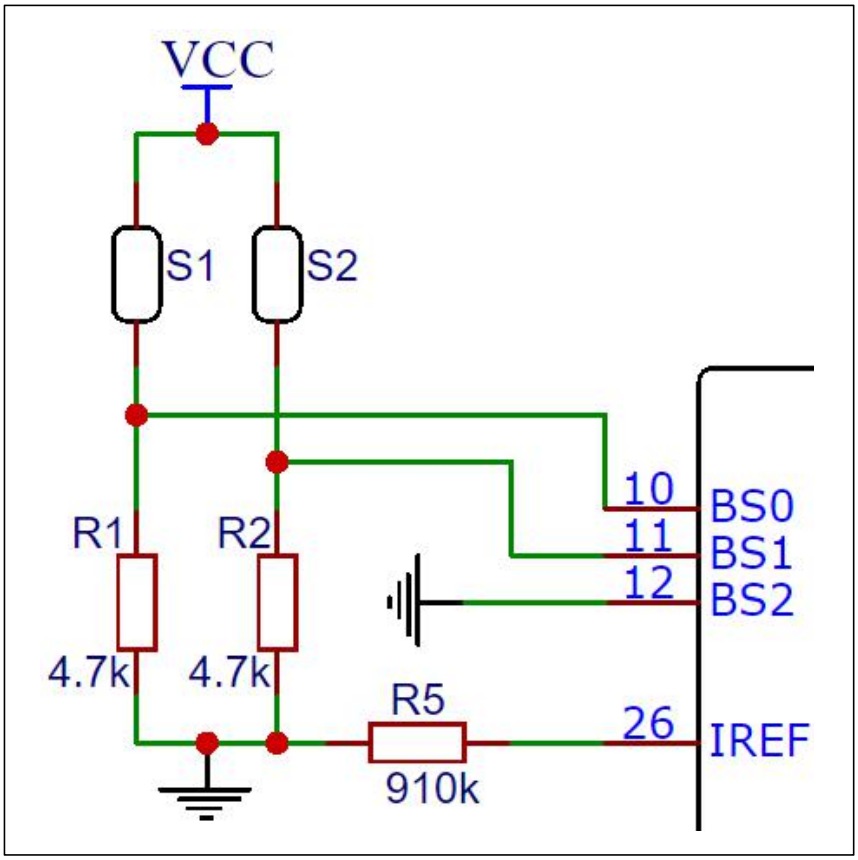




电源设计相关引脚电路

3.4 通讯方式选择电路设计

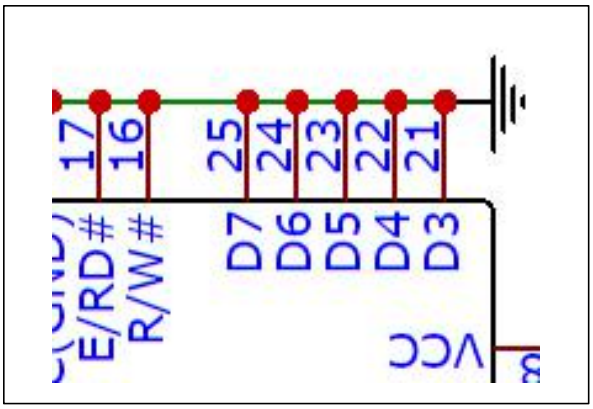
本次设计为串行通讯方式（4-WireSPI、3-WireSPI、IIC）可选择，与通讯方式有关的引脚有 BS0、BS1 与 BS2 引脚。因为不涉及并行通讯方式，所以 BS2 引脚直接接地，BS1 与 BS1 通过电阻和焊盘（S1，S2）构成可选择通讯方式电路。默认两焊盘是不焊接，BS0 与 BS1 接一电阻到 GND 为 0，通讯方式是 4-WireSPI。如焊上 S1，BS0 接到 VCC 为 1，通讯方式为 3-WireSPI。



通讯方式选择电路

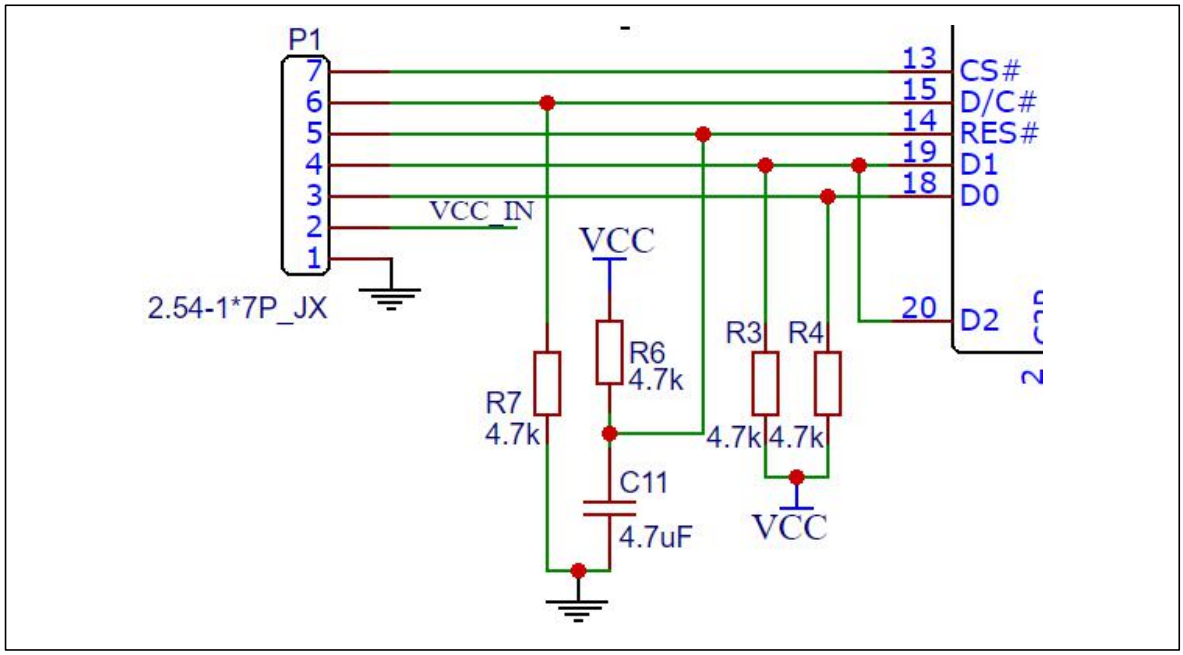
3.5 通讯接口电路设计

模块的通讯方式为串行通讯方式，需要用到的通讯接口由 D0、D1、D2、CS#、RES#、D/C#，其他没有用到的数据接口引脚根据引脚功能表要求接地处理。



在 4-WireSPI 通讯接口中，D0 作为通讯的时钟输入信号（SCLK），D1 作为通讯的数据输入信号（SDIN），RES#作为复位信号引脚，D/C#作为数据/命令选择引脚，CS#作为片选信号引脚。在 3-WireSPI 通讯接口中，D0 作为通讯的时钟输入信号（SCLK），D1 作为通讯的数据输入信号（SDIN），RES#作为复位信号引脚，D/C#应该拉低，CS#作为片选信号引脚。在 IIC 通讯接口中，D0 作为通讯的时钟输入信号（SCLK），D1 作为通讯的数据输入信号（SDIN），D2 必须与 D1 连接到一起，否则接收不到应答信号，D/C#引脚作为地址号的第 0 位（SA0）。

在设计中把 D0、D1、CS#、RES#、D/C#等引脚引出，可实现 4-WireSPI 与 3-WireSPI 的通讯接口。为了兼容 IIC 接口，根据技术手册要求，把 D0 与 D1 分别接一个上拉电阻，D2 与 D1 短接。考虑到 IIC 接口只需要 4 根线（VCC，GND，SCL，SDA）就可以实现通讯，对 RES#引脚设计一个上电复位电路（上电时 RES#引脚保持 3us 低电平即可），D/C#引脚通过一个电阻接到 GND（SA0 = 0）。电路图如下：

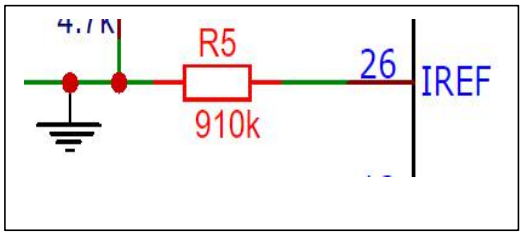


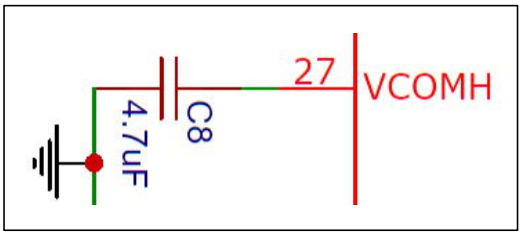
通讯接口电路

作为 4-WireSPI 接口时，D0 与 D1 分别接了上拉电阻，RES#是低电平复位，平常为高电平，D/C#接一个电阻下拉，对通讯时序不产生影响。作为 3-WireSPI 接口时，D/C#接了一个电阻下拉，默认低电平，对通讯时序不产生影响。作为 IIC 接口时，D0 与 D1 分别接上拉电阻，D2 与 D1 短接到一起可产生应答信号，D/C#引脚默认为 0（SA0 = 0），RES#引脚可实现上电复位，只需接 4 根线就能实现 IIC 通讯。

3.6 其他部分电路设计

在引脚功能表中有两个引脚 IREF 与 VCOMH，根据官方手册分别接一个 910K 电阻与 4.7uF 电容到 GND。7 脚是悬空引脚，1 脚与 30 脚接地作为静电保护功能。





注：

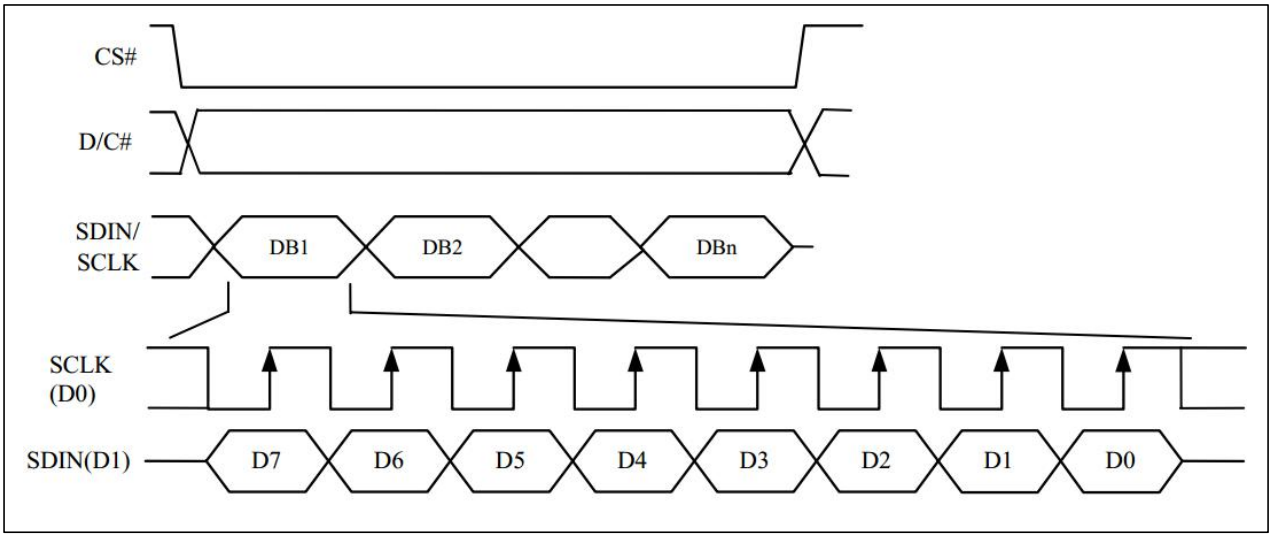
电路设计参考官方推荐电路，器件选用也是根据官方推荐，如果想了解更多的电路设计方案可参考官方手册中的 3.3.2.2~3.3.5.2 内容。最后引出的接口中，丝印与 SSD1306 的对应关系为：VCC-->VCC_IN，GND-->GND，D0-->D0，D1-->D1，RES-->RES#，DC-->D/C#，CS-->CS#。

4、软件设计

模块的通讯方式有多种，这里只讲解 4-WireSPI 通讯并用软件模拟的方式实现。官方手册对软件部分介绍的比较少，这里主要参考技术手册的第 8 章的内容来讲解通讯协议。本次的软件设计是基于 STM8S105C6T6 最小系统板来实现模拟 SSD1306 的 4-WireSPI 协议，程序部分可以分为两部分来理解：通讯协议的实现与显示数据。了解通讯协议可以在各个平台的单片机进行程序移植，数据显示用的函数与命令都是可以直接拿来用的，不需要修改。下面会详细介绍。

4.1 模块的 4-WireSPI 时序

SSD1306 的 4-WireSPI 的时序图如下：



4-WireSPI 时序图

根据 SSD1306 的 4-WireSPI 的时序图，使用 4-WireSPI 协议对 SSD1306 写入一字节数据的操作顺序如下（数据是高位先发送）：

1) 拉低 CS#引脚（CS 输出低），选通器件开始通讯。
2) 对 SSD1306 写入命令/数据。D/C#引脚拉低：后面写入的字节是命令字节；D/C#拉高：后面写入的字节是数据字节。
3) 在时钟线（D0）低电平时，器件把数据位赋予信号线（D1），拉高时钟线（D0）产生上升沿，在上升沿时数据位被发送出去。一次发送的数据大小是 8 位（一个字节），需要连续发送 8 次，然后拉高 CS#，结束这次通讯。

4.2 模块与 STM8S105C6T6 的硬件连接

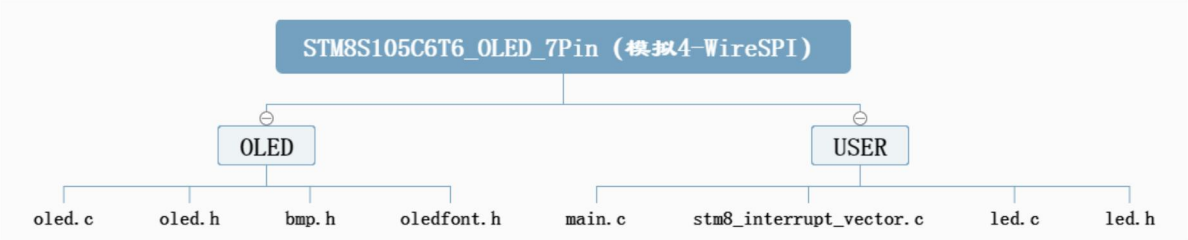
模块支持 3.3V~5V 的电压输入，技新的 STM8S105C6T6 最小系统板有 3.3V 与 5V 的供电模式，两者之间是兼容的，可以直接 VCC 对 VCC（如果系统板采用 3.3V 供电 VCC 就是 3.3V），GND 对 GND 连接。4-WireSPI 通讯方式的引脚分接法：D0 接 PC5，D1 接 PC6，DC 接 PE7，CS 接 PC4。RES 接 PE6。如下：

0.96'OLED（7Pin）模块	STM8S105C6T6
GND	GND
VCC	VCC（3V3/5V）
D0	PC5
D1	PC6
RES	PE6
DC	PE7
CS	PC4

模块与 STM8S105C6T6 接线对应表

4.3 模块在 STM8S105C6T6 上的应用

与本手册配套的例程是 STM8S105C6T6_OLED_7Pin（4-WireSPI），采用的是模拟 4-WireSPI 方式与模块进行通讯（例程可在技新网的产品中心 https://www.jixin.pro/product/527.html 下的教程与资料中下载）。工程文件的主要结构下：



STM8S105C6T6_OLED_7Pin（模拟 4-WireSPI）工程文件结构图

工程文件夹下包含两个子文件夹：OLED 文件夹（存放 oled 相关文件）和 USER 文件夹（STM8S105C6T6 的工程文件夹），这里主要介绍 OLED 文件夹。OLED 文件夹下一共有 4 个文件，其中 oledfont.h 文件存放 ASCII 的数组与汉字取模的数组，bmp.h 文件存放的是图片取模的数组（取模方式后面会介绍）。oled.c 文件

与 oled.h 文件内容包含了模块与单片机的接口定义与初始化、模拟 4-WireSPI 时序的实现、各种显示函数实现。

协议采用 IO 口模拟的方式，要对用到的 IO 口进行相应的初始化后才能使用。IO 口的初始化用一个函数封装，初始化后对引脚的输出高低电平进行宏定义，这样使得代码更利于修改。STM8S105C6T6 与模块的接口的 IO 口的输入输出宏定义在 oled.h 文件下，IO 口的初始化在 oled.c 文件下：

```
7
8   /*--------------------引脚定义--------------------------*/
9   #define SPI_MOSI_HIGH       (PC_ODR |= 0x40)    //PC6（D1）输出高
10  #define SPI_MOSI_LOW        (PC_ODR &= 0xbf)    //PC6（D1）输出低
11
12  #define SPI_SCK_HIGH        (PC_ODR |= 0x20)    //PC5（D0）输出高
13  #define SPI_SCK_LOW         (PC_ODR &= 0xdf)    //PC5（D0）输出低
14
15  #define SPI_CS_HIGH         (PC_ODR |= 0x10)    //PC4（CS）输出高
16  #define SPI_CS_LOW          (PC_ODR &= 0xef)    //PC4（CS）输出低
17
18  #define OLED_RES_HIGH       (PE_ODR |= 0x40)    //PE6（RES）输出高
19  #define OLED_RES_LOW        (PE_ODR &= 0xbf)    //PE6（RES）输出低
20
21  #define OLED_DC_HIGH        (PE_ODR |= 0x80)    //PE7（DC）输出高
22  #define OLED_DC_LOW         (PE_ODR &= 0x7f)    //PE7（DC）输出低
23
```

oled.h 文件下的接口宏定义

```
55
56  /*
57      @brief          初始化OLED与单片机的IO接口
58      @param          无
59      @retval         无
60   */
61  static void OLED_GPIO_Init(void)
62  {
63      PC_DDR |= 0x70; //将 PC4 PC5 PC6 口设为输出
64      PC_CR1 |= 0x70; //将 PC4 PC5 PC6 口设为推挽
65      PC_CR2 &= 0x8f; //将 PC4 PC5 PC6 口设为推挽
66
67      PE_DDR |= 0xc0; //将 PE6 PE7 口设为输出
68      PE_CR1 |= 0xc0; //将 PE6 PE7 口设为推挽
69      PE_CR2 &= 0x3f; //将 PE6 PE7 口设为推挽
70  }
71
```

oled.c 文件下的 IO 口初始化

对于同一款系列的单片机，比如 STM8S105 系列，根据模块与单片机的硬件连接，只需要修改上述的 IO 口宏定义与 IO 口初始化部分的内容就可以直接使用本例程了，其他的都不需要修改。最后打开工程，把程序下载到板子看就可以看到 OLED 模块显示。

4.4 通讯协议的实现

根据 4-WireSPI 的时序图，来实现 OLED 模块通讯的模拟 4-WireSPI，代码在 oled.c 文件中实现：

```
72
73  /*
74      @brief          模拟SPI发送一个字节
75      @param          data: 发送的数据
76      @retval         无
77   */
78  static void SPI_Write_Byte(unsigned char data)
79  {
80      unsigned char i;     //定义变量
81      for(i = 0; i < 8; i++)  //循环8次
82      {
83          SPI_SCK_LOW;     //将时钟线拉低
84          delay(1);    //延迟
85
86          if(data & 0x80) //数据从高位-->低位依次发送
87              SPI_MOSI_HIGH;  //数据为为1
88          else
89              SPI_MOSI_LOW;   //数据位为0
90
91          data <<= 1; //数据左移1位
92
93          delay(1);    //延迟
94          SPI_SCK_HIGH;    //时钟线拉高，把数据发送出去
95          delay(1);    //延迟
96      }
97
98  }
99
```

模拟 4-WireSPI 写一字节数据

对于 SSD1306 来说，数据有命令与数据之分（如果接受到的是命令，SSD1306 就会把接收到的下一个字节当作命令转移到命令寄存器中；如果接受到的是数据，SSD1306 就会把接收到的下一个字节当作数据存放到 GDDRAM 中）。当 DC 脚为高电平时，后面接收到的字节就是数据；当 DC 脚为低电平是，后面接受到的字节就是命令。用一个函数通过判断传入的参数来调用发送数据或命令函数，其代码如下如所示：

```
128
129 /*
130     @brief          对OLED写入一个字节
131     @param          dat: 数据
132                     cmd: 1，写该数据；0，写入命令
133     @retval         无
134  */
135 void OLED_WR_Byte(unsigned char dat,unsigned char cmd)
136 {
137     if(cmd)  //如果cmd为高，则发送的是数据
138         OLED_DC_HIGH;    //将DC拉高
139     else    //如果cmd为低，则发送的是命令
140         OLED_DC_LOW;     //将DC拉低
141
142     SPI_CS_LOW; //片选拉低，选通器件
143
144     SPI_Write_Byte(dat); //发送数据
145
146     SPI_CS_HIGH; //片选拉高，关闭器件
147     OLED_DC_HIGH; //DC拉高，空闲时为高电平
148 }
149
```

向 SSD1306 写入数据/命令

在 oled.h 文件中，根据写入命令/数据函数定义了一个宏定义，当传入的参数 cmd = OLED_CMD，则写

入的是命令；当传入的参数 cmd = OLED_DATA，则写入的是数据：

```
24
25  /*definition----------------------------------------*/
26  #define OLED_CMD  0 //写命令
27  #define OLED_DATA 1 //写数据
```

数据/命令的宏定义

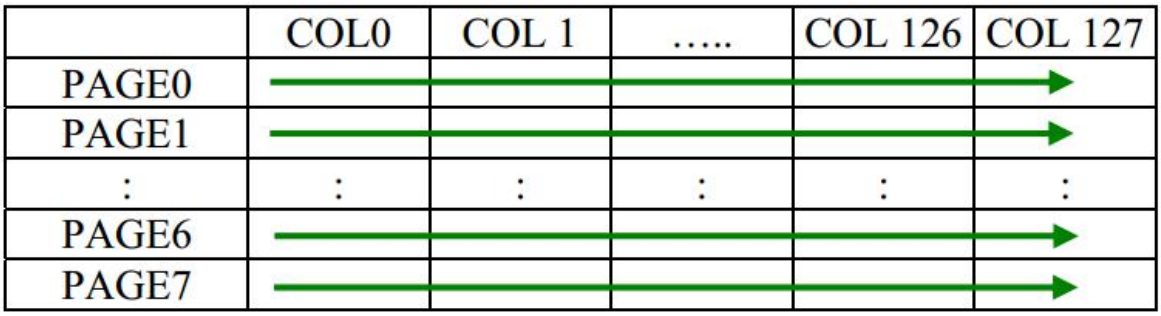
使用的是串行通讯协议（4-WirtSPI 通讯协议）时，只能对模块进行写入。在 oled.c 文件中还有一个重要的函数就是 void OLED_Init(void)，OLED 初始化函数，虽然这个函数看起来比较长，配置了很多东西，但其实在官方手册中的 4.2.2 中已经给出了，可以直接使用。

```
353
354 /*
355     @brief          OLED初始化函数
356     @param          无
357     @retval         无
358  */
359 void OLED_Init(void)
360 {
361
362     OLED_GPIO_Init();   //初始化相应的IO口
363
364
365     OLED_RES_HIGH;
366     delay_ms(100);
367     OLED_RES_LOW;
368     delay_ms(200);//延迟，由于单片机上电初始化比OLED快，所以必须加上延迟，等待OLED上电初始化完成
369     OLED_RES_HIGH;
370     delay_ms(200);
371
372     OLED_WR_Byte(0xAE,OLED_CMD);    //关闭显示
373
374     OLED_WR_Byte(0x00,OLED_CMD);    //设置低列地址
375     OLED_WR_Byte(0x10,OLED_CMD);    //设置高列地址
376     OLED_WR_Byte(0x40,OLED_CMD);    //设置起始行地址
377     OLED_WR_Byte(0xB0,OLED_CMD);    //设置页地址
378
379     OLED_WR_Byte(0x81,OLED_CMD);    // 对比度设置，可设置亮度
380     OLED_WR_Byte(0xFF,OLED_CMD);    //  265
381
382     OLED_WR_Byte(0xA1,OLED_CMD);    //设置段（SEG）的起始映射地址，column的127地址是SEG0的地址
383     OLED_WR_Byte(0xA6,OLED_CMD);    //正常显示，0xa7逆显示
384
385     OLED_WR_Byte(0xA8,OLED_CMD);    //设置驱动路数（16~64）
386     OLED_WR_Byte(0x3F,OLED_CMD);    //64duty
387
388     OLED_WR_Byte(0xC8,OLED_CMD);    //重映射模式，COM[N-1]~COM0扫描
389
390     OLED_WR_Byte(0xD3,OLED_CMD);    //设置显示偏移
391     OLED_WR_Byte(0x00,OLED_CMD);    //无偏移
392
393     OLED_WR_Byte(0xD5,OLED_CMD);    //设置震荡器分频
394     OLED_WR_Byte(0x80,OLED_CMD);    //使用默认值
395
396     OLED_WR_Byte(0xD9,OLED_CMD);    //设置 Pre-Charge Period
397     OLED_WR_Byte(0xF1,OLED_CMD);    //使用官方推荐值
398
399     OLED_WR_Byte(0xDA,OLED_CMD);    //设置 com pin configuartion
400     OLED_WR_Byte(0x12,OLED_CMD);    //使用默认值
401
402     OLED_WR_Byte(0xDB,OLED_CMD);    //设置 Vcomh，可调节亮度（默认）
403     OLED_WR_Byte(0x40,OLED_CMD);    ////使用官方推荐值
404
405     OLED_WR_Byte(0x8D,OLED_CMD);    //设置OLED电荷泵
406     OLED_WR_Byte(0x14,OLED_CMD);    //开显示
407
408     OLED_WR_Byte(0xAF,OLED_CMD);    //开启OLED面板显示
409     OLED_Clear();       //清屏
410     OLED_Set_Pos(0,0);   //设置数据写入的起始行、列
411 }
412
```

OLED 初始化函数

4.5 SSD1306 功能函数的应用

SSD1306 初始化后就可以对 SSD1306 写入数据进行显示了，在这之前要介绍一下 SSD1306 的寻址方式，也就是 SSD1306 是怎么把数据显示到屏上。SSD1306 有 3 中寻址模式：页寻址模式、水平寻址模式、垂直寻址模式。模块所使用的屏大小为 128*64 个像素点，X 轴方向为 128，Y 轴方向为 64，也就是有 64 行 128 列。SSD1306 把屏的 Y 轴的 64 个点分为 8PAGE 和 128 列，每 PAGE 大小为 8 也就是一个字节的数据大小，数据写入就是以字节的方式写入。SSD1306 初始化时设为页寻址模式，页寻址方式下的寻址指针移动如下图（详细参考技术手册 10.1.3）：



SSD136 页寻址模式下的寻址指针移动图

寻址方式决定了写入数据的方式。如在屏中显示一个汉字，汉字一般大小为 16*16，占 2PAGE 和 16COL，用上图的寻址方式写入会从第一页（PAGE）的第一列（COL）开始填充数据，当第一页的所有列全填充完后需要手动把寻址指针指向第二页继续把第二页的列填充完成（字模取模方式就是根据寻址方式决定的）。

SSD1306 的显示函数都是根据页寻址模式写的，结合 oledfont.h 文件中的字库（ASCII 码与中文）与 bmp.h 文件中的图片字库使用（字库的取模后面会介绍）。功能函数的实现在 oled.c 文件中，oled.h 文件中有声明，外部可以调用 oled.h 文件里声明的函数。

```
35
36  /*declaration----------------------------------------------*/
37  void OLED_WR_Byte(unsigned char dat,unsigned char cmd);   //OLED写字节函数
38  void OLED_Display_On(void); //开显示函数
39  void OLED_Display_Off(void);//关显示函数
40  void OLED_Init(void);   //OLED初始化函数
41  void OLED_Clear(void);  //清屏函数
42  void OLED_ShowChar(unsigned char x,unsigned char y,unsigned char chr); //显示字符函数
43  //在指定的位置，显示一个指定数的长度大小函数
44  void OLED_ShowNum(unsigned char x,unsigned char y,unsigned int num,unsigned char len,unsigned char size2);
45  void OLED_ShowString(unsigned char x,unsigned char y, unsigned char *p);     //在指定位置开始显示字符串函数
46  void OLED_Set_Pos(unsigned char x, unsigned char y);        //画点函数
47  void OLED_ShowCHinese(unsigned char x,unsigned char y,unsigned char no); //声明在指定位置显示汉字函数
48  //在指定范围显示图片函数
49  void OLED_DrawBMP(unsigned char x0, unsigned char y0,unsigned char x1, unsigned char y1,unsigned char BMP[]);
50  void OLED_Scroll(void); //滚动函数
51
```

oled.h 文件中的函数声明

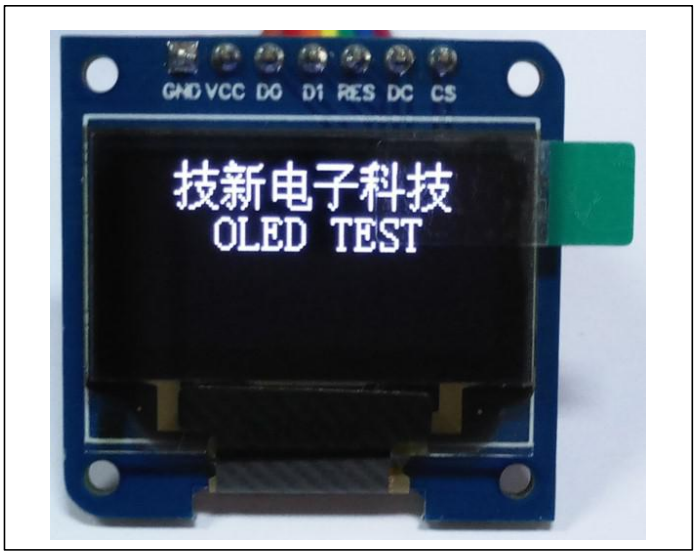
最后在主函数中调用函数把 oledfont.h 中的字库数据或 bmp.h 中的图片字库数据显示到 OLED 屏中，在调用函数显示之前要对 SSD1306 进行初始化与清屏操作。操作流程如下：

例程主函数如下：

```
29  main()
30  {
31      CLK_CKDIVR = 0x00;  //f_HSI = f_HSI RC 输出    f_CPU = f_MASTER    16M
32      LED_Init(); //LED初始化
33
34
35      OLED_Init();     //OLED初始化
36      OLED_Clear();    //OLED清屏
37
38      OLED_ShowString(30,2,"OLED TEST");//OLED显示  OLED TEST
39
40      OLED_ShowCHinese(16,0,0);    //OLED显示  技
41      OLED_ShowCHinese(32,0,1);    //OLED显示  新
42      OLED_ShowCHinese(48,0,2);    //OLED显示  电
43      OLED_ShowCHinese(64,0,3);    //OLED显示  子
44      OLED_ShowCHinese(80,0,4);    //OLED显示  科
45      OLED_ShowCHinese(96,0,5);    //OLED显示  技
46
47      while (1)
48      {
49          PG_ODR ^= 0x03; //异或 对 PG0 PG1 口取反
50          delay_ms(300);  //延迟300MS
51      }
52  }
53
```

主函数图

把模块与 STM8S105C6T6 的引脚连接好后，将程序下载进去就可以看到效果：



效果图

4.6 注意事项

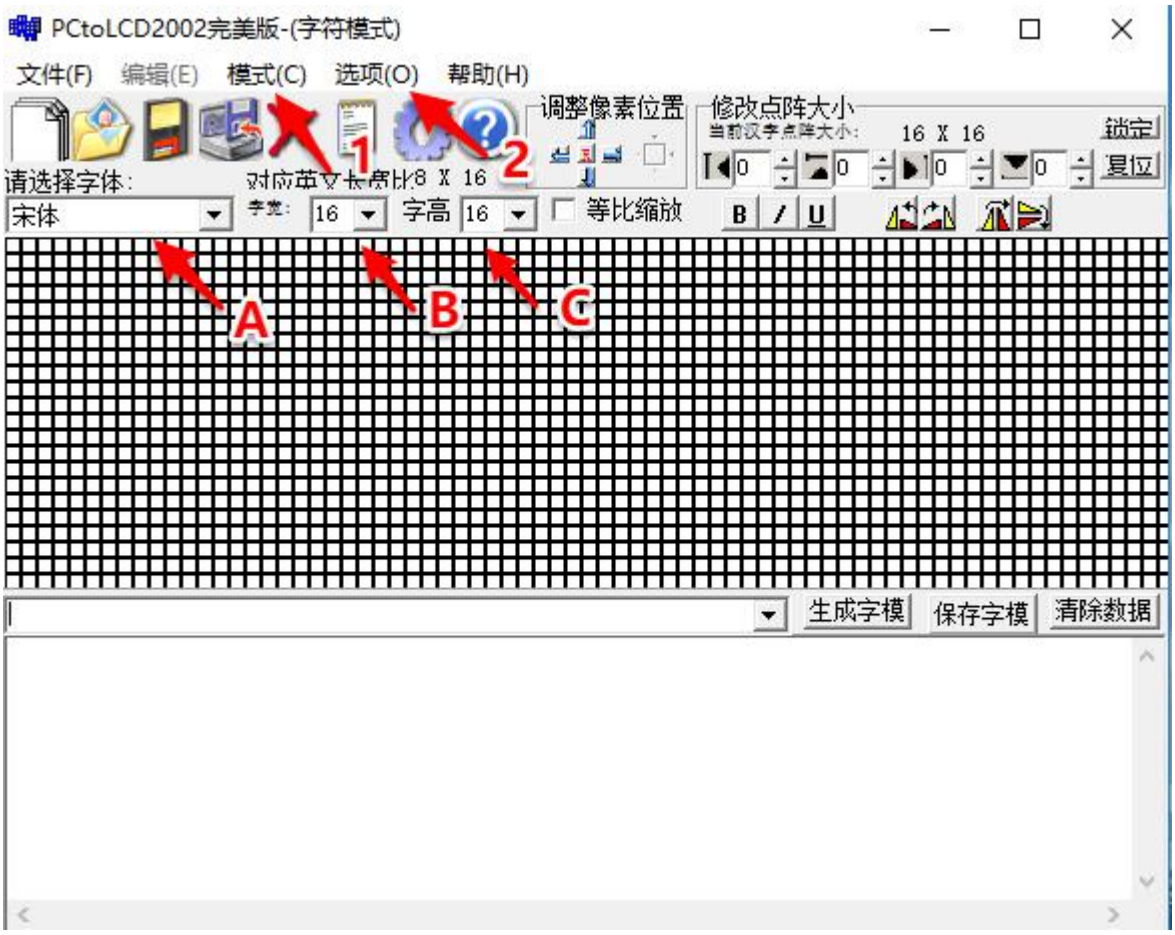
对 SSD1306 写入的数据是存放在 SSD1306 的 GDDRAM 中，掉电不会保存。对 SSD1306 写入的命令配置参数，下次上电会保留上次的配置。STM8S105C6T6 的例程采用的 IDE 是 STVD+SDVP，寄存器开发。SSD1306 还有许多的功能命令，在技术手册的第 9 章与第 10 章有详细的介绍。

5、取模软件的用法

ASCII、汉字、图片的字模都是使用 PCtoLCD2002 字模软件生成的，软件可在技新网的产品中心 https://www.jixin.pro/product/527.html 下的教程与资料中下载。ASCII 的字模已经取好在 oledfont.h 文件中，可以直接使用（函数默认使用的是 8*16 大小）。如果用户想显示汉字与图片，那么就需要自己通过取模软件取模出来。该取模软件可以取 bmp 格式的图片与汉字的字模。取出的字模分别按照 oledfont.h 文件中的汉字格式与 bmp.h 文件的图片格式存放到其中。这里的取模模式都是根据 SSD1306 的寻址方式决定的。

5.1 汉字的取模方法

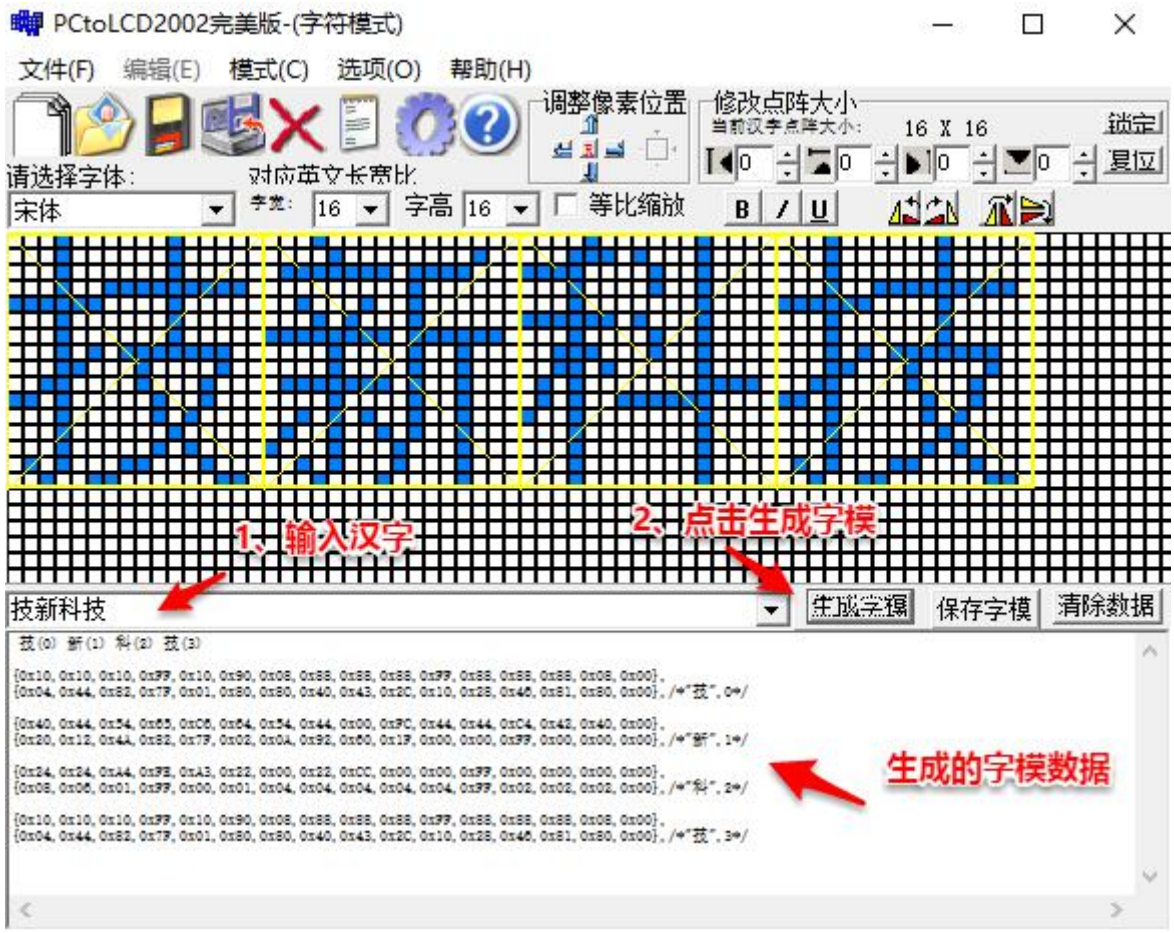
打开 PCtoLCD2002.exe 软件



箭头 A 是选择字体，可根据个人爱好选择。箭头 B 与箭头 C 是选择字宽与字高，都选择选择 16，因为显示汉字的函数就是按照这个大小编写的。点击箭头 1 指向的模式选项，选择字符模式

点击箭头 2 的选项，会弹出一个字模选项，把字模选项的参数配置为下图格式，然后点击确定按钮

最后输入汉字，点击生成字模，下方就会生成字模数据



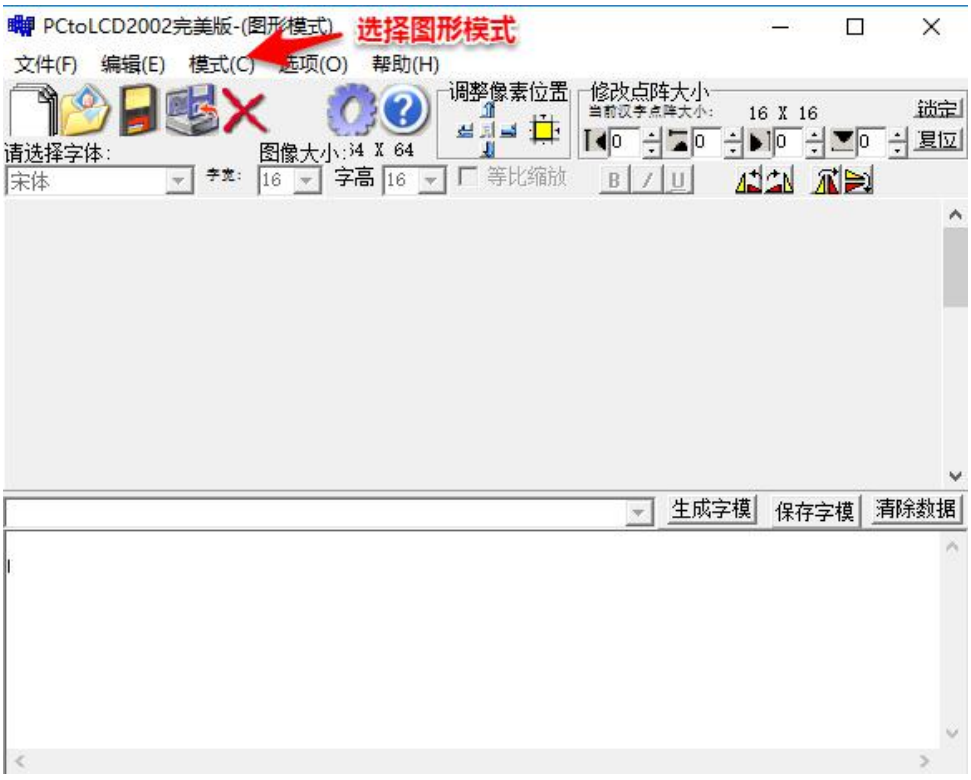
复制生成的字模，存放到 oledfont.h 文件的 const unsigned char Hzk[][16]数组下

```
204
205   const unsigned char  Hzk[][16]={
206
207   {0x10,0x10,0x10,0xFF,0x10,0x90,0x08,0x88,0x88,0x88,0xFF,0x88,0x88,0x88,0x08,0x00},
208   {0x04,0x44,0x82,0x7F,0x01,0x80,0x80,0x40,0x43,0x2C,0x10,0x28,0x46,0x81,0x80,0x00},/*"技",0*/
209
210   {0x40,0x44,0x54,0x65,0xC6,0x64,0x54,0x44,0x00,0xFC,0x44,0x44,0xC4,0x42,0x40,0x00},
211   {0x20,0x12,0x4A,0x82,0x7F,0x02,0x0A,0x92,0x60,0x1F,0x00,0x00,0xFF,0x00,0x00,0x00},/*"新",1*/
212
213   {0x00,0x00,0xF8,0x88,0x88,0x88,0x88,0xFF,0x88,0x88,0x88,0x88,0xF8,0x00,0x00,0x00},
214   {0x00,0x00,0x1F,0x08,0x08,0x08,0x08,0x7F,0x88,0x88,0x88,0x88,0x9F,0x80,0xF0,0x00},/*"电",2*/
215
216   {0x80,0x82,0x82,0x82,0x82,0x82,0x82,0xE2,0xA2,0x92,0x8A,0x86,0x82,0x80,0x80,0x00},
217   {0x00,0x00,0x00,0x00,0x00,0x40,0x80,0x7F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"子",3*/
218
219   {0x24,0x24,0xA4,0xFE,0xA3,0x22,0x00,0x22,0xCC,0x00,0x00,0xFF,0x00,0x00,0x00,0x00},
220   {0x08,0x06,0x01,0xFF,0x00,0x01,0x04,0x04,0x04,0x04,0x04,0xFF,0x02,0x02,0x02,0x00},/*"科",4*/
221
222   {0x10,0x10,0x10,0xFF,0x10,0x90,0x08,0x88,0x88,0x88,0xFF,0x88,0x88,0x88,0x08,0x00},
223   {0x04,0x44,0x82,0x7F,0x01,0x80,0x80,0x40,0x43,0x2C,0x10,0x28,0x46,0x81,0x80,0x00},/*"技",5*/
224
225   };
226
```

汉字字模

5.2 图片的取模方法

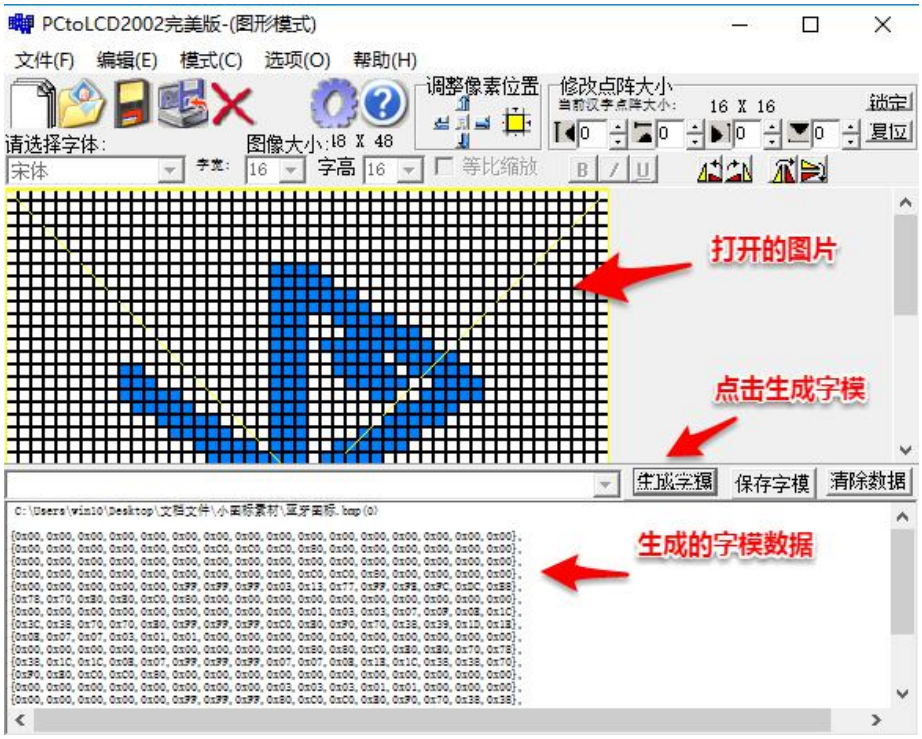
打开 PCtoLCD2002.exe 软件，在模式选项中选择图形模式



点击选项(O)，把字模选项的参数配置为下图格式，然后点击确定按钮

点击文件(F)-->打开(O)，打开 bmp 格式的图片，图片大小不要超过屏的大小 128*64，图片的颜色尽量为黑白或两种色差明显的颜色，方便软件识别

打开图片后点击生成字模按钮，下方就会生成图片字模的数据



如果用户想自己绘制图片，点击文件(F)-->新建(N)-->输入图片尺寸（不要超过 128*64）-->确定

点击确定之后，在生成的区域中通过鼠标右键在上面画点，按住左键并拖动可以连续画点，点鼠标右键可以取消画的点，按住鼠标右键并拖动可以连续取消。画完后点击生成字模，在下方就可以生成相应的字模数据