# OLED12864Lib使用教程

参考资料：

- ssd1306——oled驱动芯片手册
- oled12864中景园版本手册

手册阅读记录(解读)：

- 阅读记录

# 文件说明

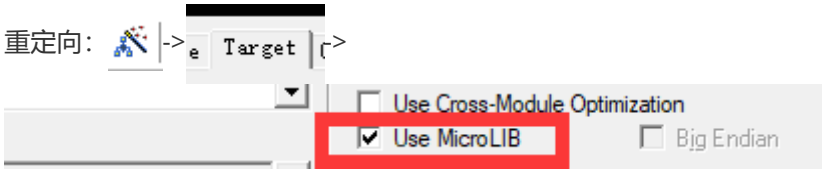
- oled12864.c/.h主要文件
- oledfont.c/.h字体字号文件

## 文件解析

### 1.oled12864.h

```
/* 用户配置 ------------------------------------------------------------------------------------------------------------------------------------------*/
/* OLED功能选项 --------------------------------------------------------------------------------------------------------------------------------------*/


/* 开启重定向流与否 ----------------------------------------------------------*/
#define PRINT2SCREEN
/* 开启重定向流与否 ----------------------------------------------------------*/


/* 选择通信接口方式 ----------------------------------------------------------*/
//#define _OLED_BY_HW_I2C
//#define _OLED_BY_SW_I2C
#define _OLED_BY_SPI
/* 选择通信接口方式 ----------------------------------------------------------*/

/* OLED功能选项 --------------------------------------------------------------------------------------------------------------------------------------*/




/* 接口资源配置 --------------------------------------------------------------------------------------------------------------------------------------*/
#if (defined _OLED_BY_HW_I2C)
  #include "i2c.h"
/* 硬件I2C配置 ---------------------------------------------------------*/
  #define OLED_hi2cn &hi2c1
/* 硬件I2C配置 ---------------------------------------------------------*/
  #define HW_I2C_Transmit(__HANDLE,__ADDRESS,__BUFFER,__SIZE) HAL_I2C_Master_Transmit(__HANDLE,__ADDRESS,(uint8_t*)__BUFFER,__SIZE,0xffff)
#elif (defined _OLED_BY_SW_I2C)
  #include "gpio.h"
/* 硬件I2C配置 ---------------------------------------------------------*/

/* 软件I2C配置 ---------------------------------------------------------*/
/*设置指南：
  *SCL和SDA可以使用任意一个GPIO管脚
  *最好设置成开漏上拉超高速模式
  *APB2时钟设置为36即可，超过了可能不工作
  */
  #define SCL_PORT GPIOA
  #define SCL_PIN GPIO_PIN_5

  #define SDA_PORT GPIOA
  #define SDA_PIN GPIO_PIN_7
/* 软件I2C配置 ---------------------------------------------------------*/
  #define OLED_SCL_RESET() HAL_GPIO_WritePin(SCL_PORT,SCL_PIN,GPIO_PIN_RESET)
  #define OLED_SCL_SET()   HAL_GPIO_WritePin(SCL_PORT,SCL_PIN,GPIO_PIN_SET)

  #define OLED_SDA_RESET() HAL_GPIO_WritePin(SDA_PORT,SDA_PIN,GPIO_PIN_RESET)
  #define OLED_SDA_SET()   HAL_GPIO_WritePin(SDA_PORT,SDA_PIN,GPIO_PIN_SET)
#elif (defined _OLED_BY_SPI)
  #include "spi.h"
/* 硬件SPI配置 ---------------------------------------------------------*/
```

用户配置中：

1. 重定向： 


   重定向流可以使得printf或fputc等Cstdlib的函数的数据流重定向到屏幕输出。

   这样就可以使用printf来输出信息到屏幕上，方便调试和格式化输出。

2. 通信接口选择：有硬件SPI、硬件I2C、软件I2C的选择。

1. 硬件SPI刷新率最高，但占用引脚多,通信速率可以达到外设时钟APXClock的1半,如F4的全速有42M bit/s
2. 硬件I2C刷新率低，但占用引脚少，只要2根。速率只有100K~2M
3. 软件I2C刷新率低，占用引脚低，且不受是否有SPI和I2C外设限制，使用普通GPIO口就可以调用

3. 接口资源配置
    1. 配置使用的i2c句柄或spi句柄
    2. 配置管脚

### 2.oledfont.h

```
/* 使用默认字号 ---------------------------------------------------------*/
#define DEFAULT_FONT 16
/* 使用默认字号 ---------------------------------------------------------*/

/* 选择字体 -------------------------------------------------------------*/
#define SONG
//#define TERMINAL
/* 选择字体 -------------------------------------------------------------*/
```

见图，如果打开默认字号，则统统默认使用默认字号

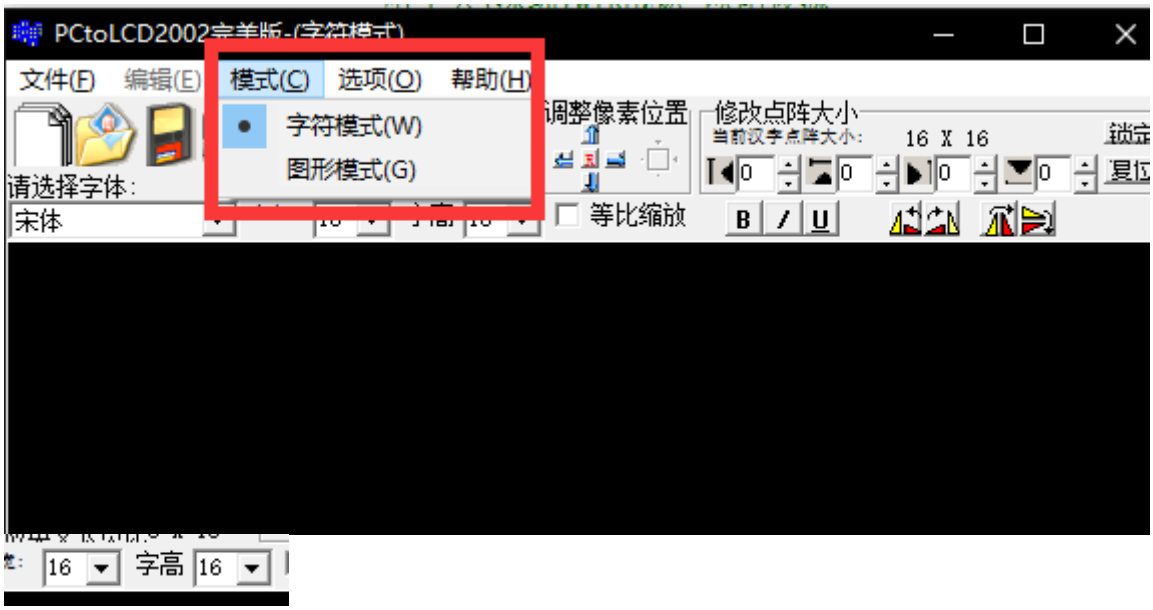
## 函数功能说明

使用的坐标系是像素坐标系(左上角0点，→x ↓y)

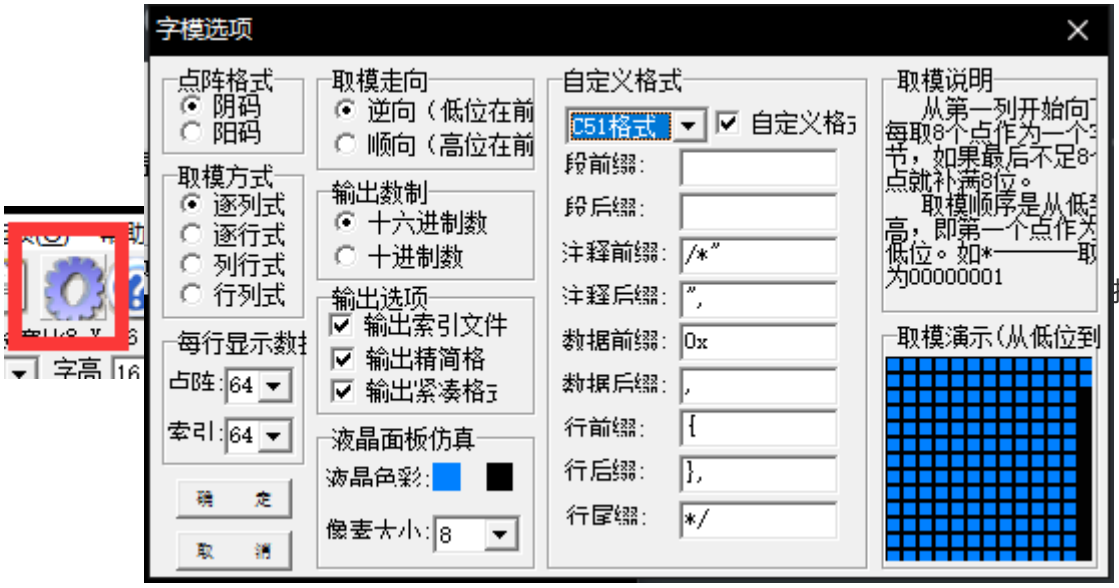
1. 各种通信接口最终同一使用 OLED_WriteRead_Byte(uint8_t data,uint8_t CorD) 发送指令到OLED控制器ssd1306上
2. holed.Draw.
    1. Point(x坐标，y坐标，擦写标记) 擦写标记：擦POINT_MOVE 写：POINT_DRAW
    ```
    #define POINT_DRAW 1
    #define POINT_MOVE 0
    ```
    2. Line(起点x，起点y，终点x，终点y)
    3. Char(字符，字号，字符左上角坐标x，字符左上角坐标y)
    4. String("字符串"，字号，字符左上角坐标x，字符左上角坐标y) 通过判断'\0'计数故不用输入字数
    5. Buffer(字符串数组头指针，字符串数组元素个数，字号，字符左上角坐标x，字符左上角坐标y) 补'\0'调用String
    6. BMP(bmp数据数组指针)
3. holed.Set.
    1. Brightness(设置亮度) 输入值在0-15 即 0x00——0x0F之间
    2. IsForce_FullScreen(0/1)输入1强制全屏点亮(无视GDDRAM内容)
    3. Horizontal_Flip(0/1)输入1水平翻转
    4. Vertical_Flip(0/1)输入1垂直翻转
    5. Inverse_Display(0/1)反显与否 即写的是黑色还是蓝色
    6. Display_OFF()
    7. Display_ON()
    8. **Refresh() 最重要的一个！！！以上所有作画需要使用本函数将GDDRAM数组内容写入oled控制器ssd1306的GDDRAM中。GDDRAM(Graphic Display Data random-access memory)**

## 再开发说明：

### 改变字体




设置字体大小16x16就是库中的1608表示16高08宽，英文是中文的一半宽。




按图配置

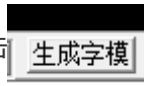
下面这串字符是可显示字符(注意开头！前还有1个空格也算，共95个)

!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_`abcdefghijklmnopqrstuvwxyz{|}~

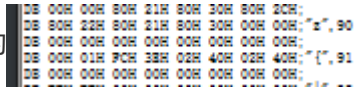
复制到



选择喜欢的字体。点击




将生成的



复制黏贴到oledfont.c 中

再复制之前先到oledfont.h中

```
/* 选择字体 ---------------------------------------------------------*/
#define SONG
//#define TERMINAL
/* 选择字体 ---------------------------------------------------------*/
```

增加刚才自定义字体的宏名

然后

```
#include "oledfont.h"
#if defined SONG
const unsigned char asc2_set_1005[][10]={
const unsigned char asc2_set_1206[][12]={
const unsigned char asc2_set_1407[][14]={
const unsigned char asc2_set_1608[][16]={
const unsigned char asc2_set_2010[][30]={
const unsigned char asc2_set_2412[][36]={
#elif defined TERMINAL
const unsigned char asc2_set_1005[][10]={
const unsigned char asc2_set_1206[][12]={
const unsigned char asc2_set_1407[][14]={
const unsigned char asc2_set_1608[][16]={
const unsigned char asc2_set_2010[][30]={
#elif defined xxx
const unsigned char asc2_set_1005[][10]={
const unsigned char asc2_set_1206[][12]={
const unsigned char asc2_set_1407[][14]={
const unsigned char asc2_set_1608[][16]={
const unsigned char asc2_set_2010[][30]={
const unsigned char asc2_set_2412[][36]={
#endif
```

将xxx改成修改对应字体，然后将数据复制到对应字号的字体的数组中。

## 字号自定义

已经有12、14、16、20、24的字号，更大的字体需要很大的数组储存(如1个64*64的字符就需要64 / 8 x 32 = 256字节存储空间 ）而10号字太小了糊的看不见。

如果需要添加，在以上数组中多加1个就好了。对应的在

```
void OLED_PutChar(uint8_t c,uint8_t size,uint8_t x,uint8_t y)
{

  if( !( c >=' ' &&  c<='~' ) )
    return;
  #ifdef DEFAULT_FONT
  size=DEFAULT_FONT;
  #endif
  uint8_t byte, len,  idx1=c-' ', y0=y, row=1,  col=0,  row_num=size/8 + (size%8?1:0 ) ;

  len= row_num *(size/2);

  for( uint8_t idx2=0 ; idx2  < len ; idx2=row_num*col+row-1 )
  {
    switch(size)
    {
      case 10:
        byte=asc2_set_1005[idx1][idx2];break;
      case 12:
        byte=asc2_set_1206[idx1][idx2];break;
      case 14:
        byte=asc2_set_1407[idx1][idx2];break;
      case 16:
        byte=asc2_set_1608[idx1][idx2];break;
      case 20:
        byte=asc2_set_2010[idx1][idx2];break;
      case 24:
        byte=asc2_set_2412[idx1][idx2];break;
    }
    row++;
```

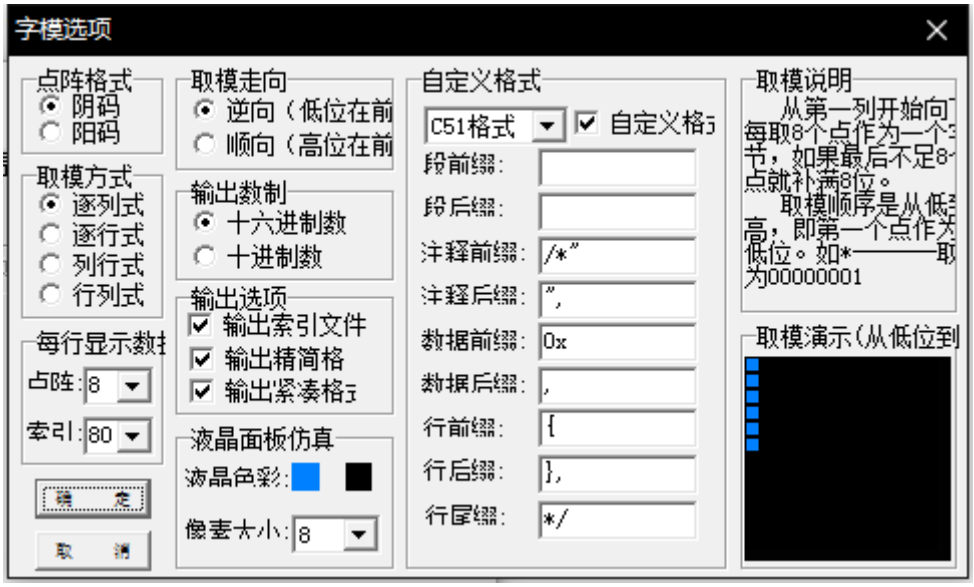
中添加新的。

## 图片播放

图片需要时二值化的。注：图像处理可以使用 

挺好用的。

取模软件使用图片模式。

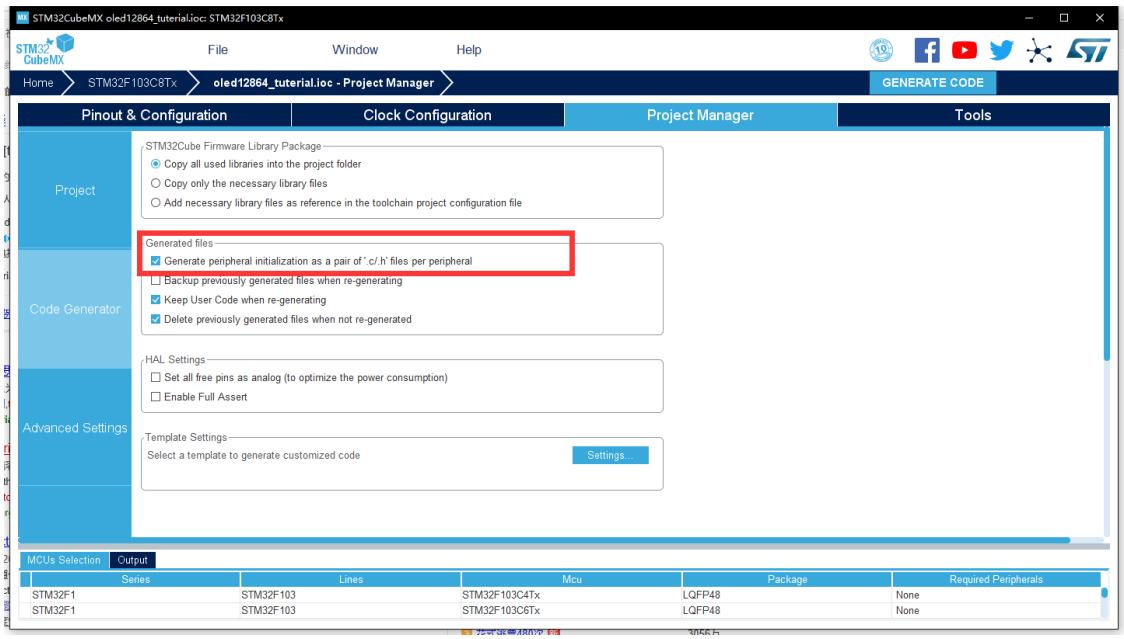



然后http://pic.55.la/网站中将图片格式转换成BMP格式的，然后生成模。将数组存下来调用

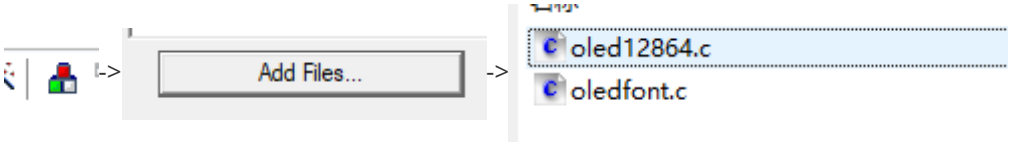
holed.Draw.BMP() 然后刷新屏幕即可。

# 实例

1.




记得勾选生成外设初始化文件。

2. 


3. 包含头文件

```
/* Includes ------------------------------
#include "main.h"
#include "spi.h"
#include "gpio.h"

/* Private includes ----------------------
/* USER CODE BEGIN Includes */
#include "oled12864.h"
/* USER CODE END Includes */
```

4. 初始化

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_SPI1_Init();
/* USER CODE BEGIN 2 */
holed.Init();
/* USER CODE END 2 */
```

5. 

```
printf("Velocity:%3.2f\nBattery:%3.2f%%",12.3,45.6);
holed.Set.Refresh();
```

重定向到屏幕并刷新

6. 效果(默认字号)。可见只需要添加头文件，初始化，就可以打印到屏幕上方便调试。


7. 

```
holed.Draw.BMP(bmp1);
holed.Set.Refresh();
```