2025/8/2

WEGUI RGB接口使用手册V0.5

目录

[1. 驱动设置相关 2](#_Toc205060149)

[1.1.设置笔刷颜色RGB\_Set\_Driver\_Colour 2](#_Toc205060150)

[1.2.设置刷图驱动笔刷Lcd\_Set\_Driver\_Mode 2](#_Toc205060151)

[1.3.设置高级刷图驱动模式的”限定区域” Lcd\_Set\_Driver\_Box 2](#_Toc205060152)

[2.绘图相关 3](#_Toc205060153)

[2.1.画一个点Lcd\_Draw\_Pixl 3](#_Toc205060154)

[2.2.画一条直线Lcd\_Draw\_Line 3](#_Toc205060155)

[2.3.画圆/圆的一部分Lcd\_Draw\_Circel\_part 3](#_Toc205060156)

[2.4.填充圆/圆的一部分Lcd\_Fill\_Circel\_part 4](#_Toc205060157)

[2.5.描画矩形Lcd\_Draw\_Box 5](#_Toc205060158)

[2.6.填充矩形Lcd\_Fill\_Box 5](#_Toc205060159)

[2.7.描画圆角矩形Lcd\_Draw\_RBox 5](#_Toc205060160)

[2.8.填充圆角矩形Lcd\_Fill\_RBox 6](#_Toc205060161)

[2.9.放置图片Lcd\_Draw\_Bitmap 6](#_Toc205060162)

[2.10.放置一个ASCII字符LcdD\_Draw\_Ascii 6](#_Toc205060163)

[2.12.放置一个UNICODE字符Lcd\_Draw\_Unicode 6](#_Toc205060164)

[2.13.放置一个UNICODE字符Lcd\_Draw\_Unicode 7](#_Toc205060165)

[2.14.放置字符串Lcd\_Draw\_UTF8\_String 7](#_Toc205060166)

[2.15.获取字符串文字占用的宽度LCD\_Get\_UTF8\_XLen 7](#_Toc205060167)

[2.16.获取字符串中换行的次数LCD\_Get\_UTF8\_YLine 8](#_Toc205060168)

[2.17.填充缓存Lcd\_Fill\_GRAM 8](#_Toc205060169)

[2.18.清除缓存Lcd\_Clear\_GRAM 8](#_Toc205060170)

[3.接口函数 9](#_Toc205060171)

[3.1.初始化函数lcd\_driver\_Init 9](#_Toc205060172)

[3.2.刷屏函数LCD\_Refresh 9](#_Toc205060173)

[4.详解刷屏函数LCD\_Refresh 10](#_Toc205060174)

[4.1.概述 10](#_Toc205060175)

[4.2.使用demo 10](#_Toc205060176)

[4.3.单色OLED全屏刷新函数demo 11](#_Toc205060177)

[4.4.多色TFT全屏刷新函数demo 11](#_Toc205060178)

# 驱动设置相关

## 1.1.设置笔刷颜色RGB\_Set\_Driver\_Colour

|  |  |
| --- | --- |
| 函数名称 | void RGB\_Set\_Driver\_Colour(uint8\_t num, uint16\_t colour); |
| 功能说明 | 仅TFT彩色屏幕有效, 单色屏幕无效  设置num笔刷的颜色为colour |
| 应用示例 | *//设置Writer\_0的笔刷颜色为0x1234*  RGB\_Set\_Driver\_Colour(0, 0x1234);  *//设置Writer\_1的笔刷颜色为0xFFFF*  RGB\_Set\_Driver\_Colour(2, 0xFFFF); |

## 1.2.设置刷图驱动笔刷Lcd\_Set\_Driver\_Mode

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Set\_Driver\_Mode(lcd\_driver\_mode\_t mode) |
| 功能说明 | 指定驱动刷图笔刷   1. Write\_0 使用0色笔刷, 即lcd\_driver.colour[0] 2. Write\_1 使用1色笔刷, 即lcd\_driver.colour[1] 3. Write\_2 使用2色笔刷, 即lcd\_driver.colour[2] 4. Write\_inv 普通反写笔刷   1位时0->1或1->0  2位色时0(0B00)->3(0B11) 或1(0B01)->2(0B10)...  3位色时0(0B000)->7(0B111)或2(0B001)->6(0B110)...   1. Write\_0\_inBox使用0色高级笔刷, 基础功能与Write\_0一致 2. Write\_1\_inBox 高级写1色高级笔刷, 基础功能与Write\_1一致 3. Write\_inv\_inBox 高级反写笔刷, 基础功能与Write\_inv一致   其中高级模式只在“限定区域”绘图有效, 超出”限定区域”不刷图, 设置“限定区域”由lcd\_driver\_Write\_inv\_inBox函数设定 |
| 应用示例 | void Lcd\_Set\_Driver\_Mode (lcd\_driver\_Write\_0); |

## 1.3.设置高级刷图驱动模式的”限定区域” Lcd\_Set\_Driver\_Box

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Set\_Driver\_Box(uint16\_t x\_min ,uint16\_t y\_min ,uint16\_t x\_max,uint16\_t y\_max) |
| 功能说明 | 设置高级模式的“限定区域“ |
| 应用示例 | *//设置 (0,10)到(50,60)的矩形区域为高级笔刷的＂限定区域＂*  Lcd\_Set\_Driver\_Box (0,10,50,60); |

# 2.绘图相关

## 2.1.画一个点Lcd\_Draw\_Pixl

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Pixl(int16\_t x,int16\_t y); |
| 功能说明 | 绘制一个像素点 |
| 应用示例 | Lcd\_Draw\_Pixl(1,2); |

## 2.2.画一条直线Lcd\_Draw\_Line

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Line(int16\_t x1,int16\_t y1,int16\_t x2,int16\_t y2); |
| 功能说明 | 从(x1,y1)到(x2,y2)绘制一条直线 |
| 应用示例 | Lcd\_Draw\_Line(0,0,127,63); |

## 2.3.画圆/圆的一部分Lcd\_Draw\_Circel\_part

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Circel\_part(int16\_t x0,int16\_t y0,uint8\_t r,circle\_part\_t cPart); |
| 功能说明 | 圆心位置(x0,y0);半径r;圆的区域 cPart  其中cPart分为圆形的8个部分, 也可任意组合，如下为circle\_part\_t的枚举  C\_RU = 0x01,//右上八分之一圆  C\_UR = 0x02,//上右八分之一圆  C\_UL = 0x04,//上左八分之一圆  C\_LU = 0x08,//左上八分之一圆  C\_LD = 0x10,//左下八分之一圆  C\_DL = 0x20,//下左八分之一圆  C\_DR = 0x40,//下右八分之一圆  C\_RD = 0x80,//右下八分之一圆  C\_HL = C\_UL | C\_LU | C\_LD | C\_DL ,//左半圆  C\_HR = C\_UR | C\_RU | C\_RD | C\_DR ,//右半圆  C\_HU = C\_RU | C\_UR | C\_UL | C\_LU ,//上半圆  C\_HD = C\_LD | C\_DL | C\_DR | C\_RD ,//下半圆  C\_QLU = C\_UL | C\_LU,//左上角四分圆  C\_QLD = C\_LD | C\_DL,//左下角四分圆  C\_QRU = C\_UR | C\_RU,//右上角四分圆  C\_QRD = C\_DR | C\_RD,//右下角四分圆  C\_ALL = 0xFF,//完整的园 |
| 应用示例 | *//以 (50,50)为圆心，画半径为10的完整的圆的外边框*  Lcd\_Draw\_Circel\_part(50,50,10, C\_ALL);  *//以 (50,50)为圆心，画半径为10的圆的外边框的右上八分之一部分*  Lcd\_Draw\_Circel\_part(50,50,10, C\_RU);  *//以 (50,50)为圆心，画半径为10的圆的外边框的左半圆和右上八分之一部分*  Lcd\_Draw\_Circel\_part(50,50,10, C\_HL|C\_RU); |

## 2.4.填充圆/圆的一部分Lcd\_Fill\_Circel\_part

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Fill\_Circel\_part (int16\_t x0,int16\_t y0,uint8\_t r,circle\_part\_t cPart); |
| 功能说明 | 圆心位置(x0,y0);半径r;圆的区域 cPart  其中cPart分为圆形的8个部分, 也可任意组合，如下为circle\_part\_t的枚举  C\_RU = 0x01,//右上八分之一圆  C\_UR = 0x02,//上右八分之一圆  C\_UL = 0x04,//上左八分之一圆  C\_LU = 0x08,//左上八分之一圆  C\_LD = 0x10,//左下八分之一圆  C\_DL = 0x20,//下左八分之一圆  C\_DR = 0x40,//下右八分之一圆  C\_RD = 0x80,//右下八分之一圆  C\_HL = C\_UL | C\_LU | C\_LD | C\_DL ,//左半圆  C\_HR = C\_UR | C\_RU | C\_RD | C\_DR ,//右半圆  C\_HU = C\_RU | C\_UR | C\_UL | C\_LU ,//上半圆  C\_HD = C\_LD | C\_DL | C\_DR | C\_RD ,//下半圆  C\_QLU = C\_UL | C\_LU,//左上角四分圆  C\_QLD = C\_LD | C\_DL,//左下角四分圆  C\_QRU = C\_UR | C\_RU,//右上角四分圆  C\_QRD = C\_DR | C\_RD,//右下角四分圆  C\_ALL = 0xFF,//完整的园 |
| 应用示例 | *//以 (50,50)为圆心，填充半径为10的完整的圆*  Lcd\_Fill\_Circel\_part(50,50,10, C\_ALL);  *//以 (50,50)为圆心，填充半径为10的圆的的右上八分之一*  Lcd\_Fill\_Circel\_part (50,50,10, C\_RU);  *//以 (50,50)为圆心，填充半径为10的圆的左半圆和右上八分之一的组合*  Lcd\_Fill\_Circel\_part (50,50,10, C\_HL|C\_RU); |

## 2.5.描画矩形Lcd\_Draw\_Box

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Box(int16\_t x\_min,int16\_t y\_min, int16\_t x\_max, int16\_t y\_max); |
| 功能说明 | 起点坐标:(x\_min,y\_min)  终点坐标:(x\_max,y\_max) |
| 应用示例 | *//在(0,0)到(63,63)画一个矩形边框*  Lcd\_Draw\_Box(0,0,63,63); |

## 2.6.填充矩形Lcd\_Fill\_Box

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Fill\_Box(int16\_t x\_min,int16\_t y\_min, int16\_t x\_max, int16\_t y\_max); |
| 功能说明 | 起点坐标:(x\_min,y\_min)  终点坐标:(x\_max,y\_max) |
| 应用示例 | *//在(0,0)到(63,63)填充一个矩形*  Lcd\_Fill\_Box(0,0,63,63); |

## 2.7.描画圆角矩形Lcd\_Draw\_RBox

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_RBox(int16\_t x\_min,int16\_t y\_min, int16\_t x\_max, int16\_t y\_max, int8\_t r); |
| 功能说明 | 起点坐标:(x\_min,y\_min)  终点坐标:(x\_max,y\_max)  圆角半径:r |
| 应用示例 | *//在(0,0)到(63,63)画一个四倒圆角3个像素的矩形边框*  Lcd\_Draw\_RBox (0,0,63,63,3); |

## 2.8.填充圆角矩形Lcd\_Fill\_RBox

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Fill\_RBox(int16\_t x\_min,int16\_t y\_min, int16\_t x\_max, int16\_t y\_max, int8\_t r); |
| 功能说明 | 起点坐标:(x\_min,y\_min)  终点坐标:(x\_max,y\_max)  圆角半径:r |
| 应用示例 | *//在(0,0)到(63,63)填充一个四倒圆角3个像素的矩形*  Lcd\_Fill\_RBox (0,0,63,63,3); |

## 2.9.放置图片Lcd\_Draw\_Bitmap

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Bitmap(int16\_t x0,int16\_t y0,uint8\_t sizex,uint8\_t sizey,uint8\_t BMP[]); |
| 功能说明 | 起点坐标:(x0,y0)  图片宽度: sizex  图片高度: sizey  图片数组: BMP[] |
| 应用示例 | *//在(0,0)画一个宽128高64的BMP数组的图片*  Lcd\_Draw\_Bitmap(0,0,128,64,BMP[]); |

## 2.10.放置一个ASCII字符LcdD\_Draw\_Ascii

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Ascii(int16\_t x,int16\_t y,uint8\_t chr); |
| 功能说明 | 起点坐标:(x,y)  字符: chr |
| 应用示例 | *//在(0,0)坐标画一个ASCII字符A*  Lcd\_Draw\_Ascii(0,0,’A’); |

## 2.12.放置一个UNICODE字符Lcd\_Draw\_Unicode

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_Unicode(int16\_t x,int16\_t y,unicode\_t unicode\_id); |
| 功能说明 | 起点坐标:(x,y)  字符id: unicode\_id |
| 应用示例 | *//在(0,0)坐标画一个字符unicode字符”中”(Unicode = 0x4e2d)*  unicode\_t id;  id.u16 = 0x2d4e; *//注意大小端模式*  Lcd\_Draw\_Unicode (0,0,id);  *//在(0,0)坐标画一个字符unicode字符”文”(Unicode = 0x4e2d)*  unicode\_t id;  id.u8[0] = 0x4e; *//注意大小端模式*  id.u8[1] = 0x2d;  Lcd\_Draw\_Unicode (0,0,id); |

## 2.13.放置一个UNICODE字符Lcd\_Draw\_Unicode

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_int32(int16\_t x,int16\_t y,int16\_t num); |
| 功能说明 | 起点坐标:(x,y)  数字: unm |
| 应用示例 | *//在(0,0)写数字1234*  Lcd\_Draw\_Unicode (0,0,1234);  *//在(0,0)写数字-4567*  Lcd\_Draw\_Unicode (0,0,-4567); |

## 2.14.放置字符串Lcd\_Draw\_UTF8\_String

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Draw\_UTF8\_String(int16\_t x,int16\_t y,uint8\_t \*p); |
| 功能说明 | 起点坐标:(x,y)  字符串指针: \*p  当字符串里包含”\r\n”或”\r”或”\n”时自动换行，  换行距离由lcd\_driver.newline\_high的值决定 |
| 应用示例 | *//在(0,0)开始画字符*  Lcd\_Draw\_UTF8\_String(0,0,(uint8\_t\*)”AB中文混搭CD\nnewline换行”); |

## 2.15.获取字符串文字占用的宽度LCD\_Get\_UTF8\_XLen

|  |  |
| --- | --- |
| 函数名称 | uint16\_t Lcd\_Get\_UTF8\_XLen(uint8\_t \*p); |
| 功能说明 | 字符串指针: \*p  根据系统设置的字体计算出字符串在显示屏上占用的宽度 |
| 应用示例 | width = Lcd\_Get\_UTF8\_XLen (”AB中文混搭CD\nnewline换行”); |

## 2.16.获取字符串中换行的次数LCD\_Get\_UTF8\_YLine

|  |  |
| --- | --- |
| 函数名称 | uint16\_t Lcd \_Get\_UTF8\_YLine(uint8\_t \*p); |
| 功能说明 | 字符串指针: \*p  返回字符串中出现”\r\n”或’\r’或’\n’的次数 |
| 应用示例 | i1 = Lcd\_Get\_UTF8\_YLine(”A\r\n”); *//i1 = 1;*  i2 = Lcd\_Get\_UTF8\_YLine(”A\r\nbbb\r”); *//i2 = 2;*  i3 = Lcd\_Get\_UTF8\_YLine(”A\r\nBBB\nCCC\rDDD”); *//i3 = 3;* |

## 2.17.填充缓存Lcd\_Fill\_GRAM

|  |  |
| --- | --- |
| 函数名称 | void OLED\_Fill\_GRAM(uint8\_t n); |
| 功能说明 | 将缓存的值全部替换成value, 即TFT彩屏刷新后将呈现刷成笔刷n的颜色 |
| 应用示例 | *//缓存全部填充成0*  Lcd\_Fill\_value\_GRAM(0);  *//缓存全部填充成1F*  Lcd\_Fill\_value\_GRAM(1);  *//缓存全部填充成2*  Lcd\_Fill\_value\_GRAM(2); |

## 2.18.清除缓存Lcd\_Clear\_GRAM

|  |  |
| --- | --- |
| 函数名称 | void Lcd\_Clear\_GRAM(void); |
| 功能说明 | 用0x00对缓存进行快速填充, 即TFT彩屏刷新后将呈现刷成笔刷0的颜色 |
| 应用示例 | Lcd\_Clear\_GRAM(); |

# 3.接口函数

## 3.1.初始化函数lcd\_driver\_Init

|  |  |
| --- | --- |
| 函数名称 | void lcd\_driver\_Init(void); |
| 功能说明 | 包含字体设置和开机驱动模式设置  需要开机时初始化一次 |
| 应用示例 | lcd\_driver\_Init(); |

## 3.2.刷屏函数LCD\_Refresh

|  |  |
| --- | --- |
| 函数名称 | uint8\_t LCD\_Refresh(void) |
| 功能说明 | 在全屏缓存模式下返回0  在页缓存模式下返回下次刷新的行的位置, 即返回0代表刷新完毕 |
| 应用示例 | do   {  While(LCD\_is\_Busy()!=0);*//等待屏幕接口*    Lcd\_Clear\_GRAM();*//清空缓存*    Lcd \_Draw\_Ascii (0,0,'A'); *//画图*   }while(LCD\_Refresh());*//发送缓存 直到发完整个屏幕* |

# 4.详解刷屏函数LCD\_Refresh

## 4.1.概述

LCD\_Refresh(void)函数基本应用方式已在上节描述，该函数为“接口类型”函数，即不同的嵌入式单片机型号，不同的屏幕通讯硬件接口(如软硬件I2C和软硬件SPI)，和不同的屏幕类型（如OLED和TFT），以及不同的驱动方式（如动态刷新和全局刷新驱动方式），其实现代码均不一致.本历程中使用了stm32f103为主控，适配了6种通讯接口，三种屏幕类型，四种驱动方式，供读者移植.如下表 1 展示了demo中已适配的所有接口，每一个所支持得接口都有一个刷屏函数，demo中通过宏定义切换

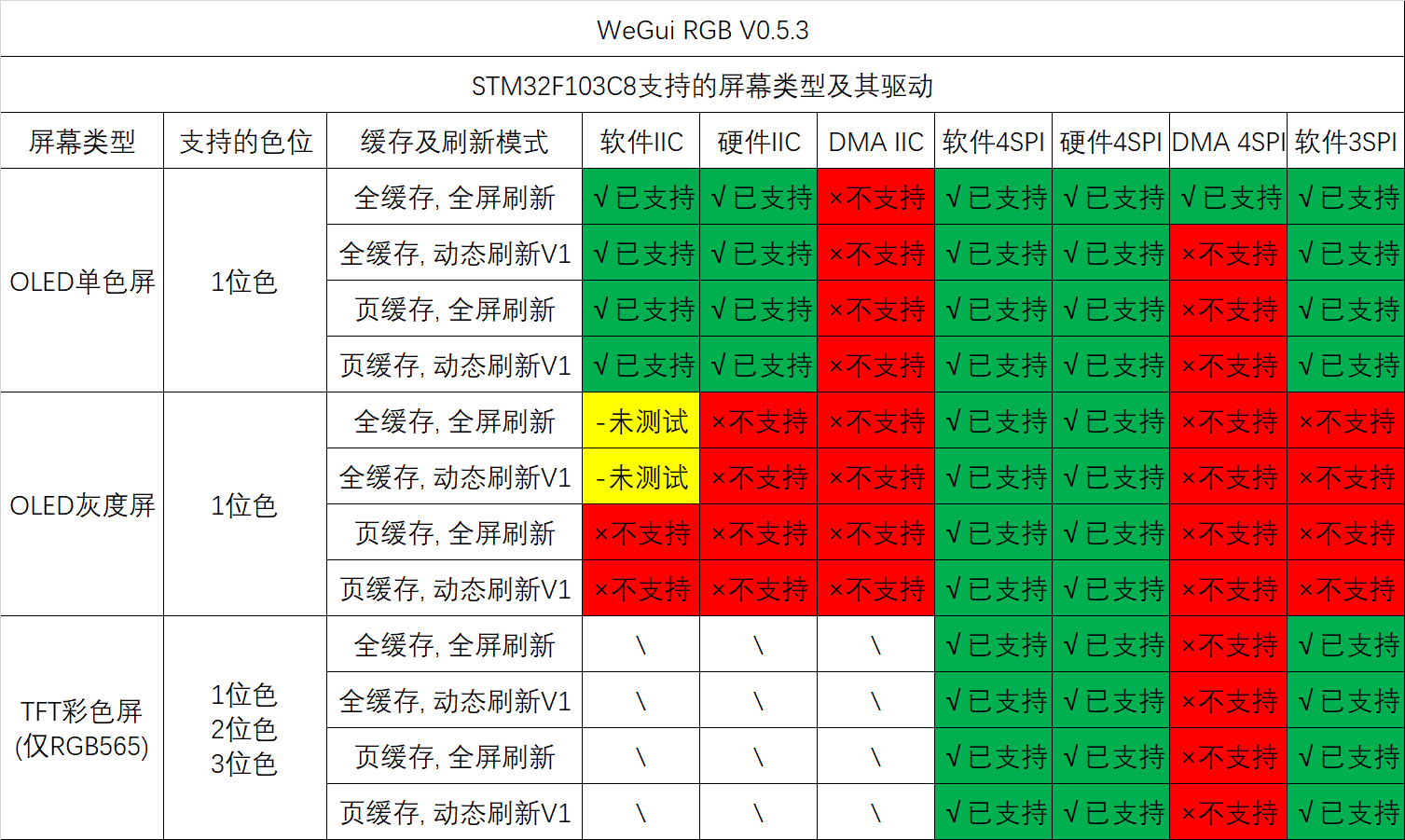


表 1

## 4.2.使用demo

do

 {

While(LCD\_is\_Busy()!=0);*//等待屏幕接口*

  Lcd\_Clear\_GRAM();*//清空缓存*

  Lcd\_Draw\_Ascii (0,0,'A'); *//画图*

Lcd\_Draw\_Ascii (0,0,'B'); *//画图*

 }while(LCD\_Refresh());*//发送缓存*

## 4.3.单色OLED全屏刷新函数demo

#define SCREEN\_WIDTH 128

#define SCREEN\_HIGH 64

uint8\_t LCD\_Refresh(void)

{

 uint8\_t ypage;

 for(ypage=0;ypage<((SCREEN\_HIGH+7)/8);ypage++)

 {

  LCD\_Set\_Addr(0,ypage);

  LCD\_Send\_nDat(&lcd\_driver.LCD\_GRAM[ypage][0][0],SCREEN\_WIDTH); *//显存*

 }

 return 0;

}

## 4.4.多色TFT全屏刷新函数demo

*//多色TFT*

#define LCD\_COLOUR\_BIT 2

#define SCREEN\_WIDTH 128

#define SCREEN\_HIGH 160

uint8\_t LCD\_Refresh(void)

{

 uint16\_t x,y;

 LCD\_Set\_Addr(0,0,SCREEN\_WIDTH-1,SCREEN\_HIGH-1);

 LCD\_DC\_Set();

 LCD\_CS\_Clr();

 for(y=0;y<SCREEN\_HIGH;y++)

 {

  uint8\_t ypage=y/8;

  uint8\_t mask=0x01<<(y%8);

  uint8\_t \*gram=&lcd\_driver.LCD\_GRAM[ypage][0][0]; *//显存*

  for(x=0;x<SCREEN\_WIDTH;x++)

  {

   uint8\_t c=0;

   #if (LCD\_COLOUR\_BIT>=1)*//---1位色---*

    if(\*gram++&mask){c+=1;}

   #endif

   #if (LCD\_COLOUR\_BIT>=2)*//---2位色---*

    if(\*gram++&mask){c+=2;}

   #endif

   #if (LCD\_COLOUR\_BIT>=3)*//---3位色---*

    if(\*gram++&mask){c+=4;}

   #endif

   SPI\_send\_16bitByte(lcd\_driver.colour[c]);

  }

 }

 LCD\_CS\_Set();

 return 0;

}