编　　制：北京万邦易嵌科技有限公司 - 嵌入式事业部 蒋政伟

版　　本：v1.3

修改时间：2022 年 11 月 18 日

一、OLED 简介

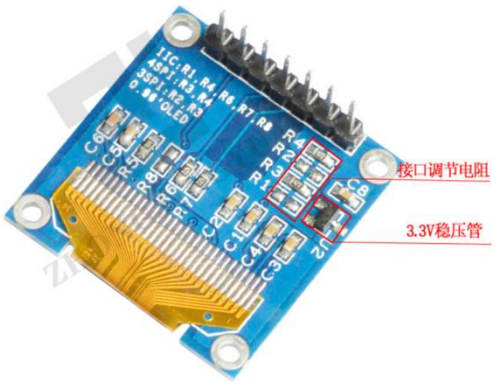
OLED 显示屏是利用有机电自发光二极管制成的显示屏。由于同时具备自发光有机电激发光二极管，不需背光源、对比度高、厚度薄、视角广、反应速度快、可用于挠曲性面板、使用温度范围广、构造及制程较简单等优异之特性;与 LCD 屏相比，在某些方面 OLED 屏更占优势：

1、发光方式，OLED 屏的发光靠像素发光，而 LCD 屏的发光是借助背光模块，因此后者需要内置一个背光模组，整体较厚，暂时没有成熟的屏下指纹方案，前者可配合超声波、光学等模块实现成熟的屏下指纹识别；

2、显示参数方面，OLED 屏的发光特性使它的功耗比 LCD 屏低，较 LCD 屏更加省电，OLED 屏延迟更低，视角更广以及色彩更艳丽，视觉效果比 LCD 屏好；

3、成本方面，目前 OLED 屏成本相比 LCD 较高，一般用于中高端的手机产品中；

4、优劣方面，两者没有绝对的优劣，但 OLED 更轻薄且可弯折的特性，使其成为未来便携设备的显示屏幕发展方向，但目前仍未能完全克服烧屏的情况。

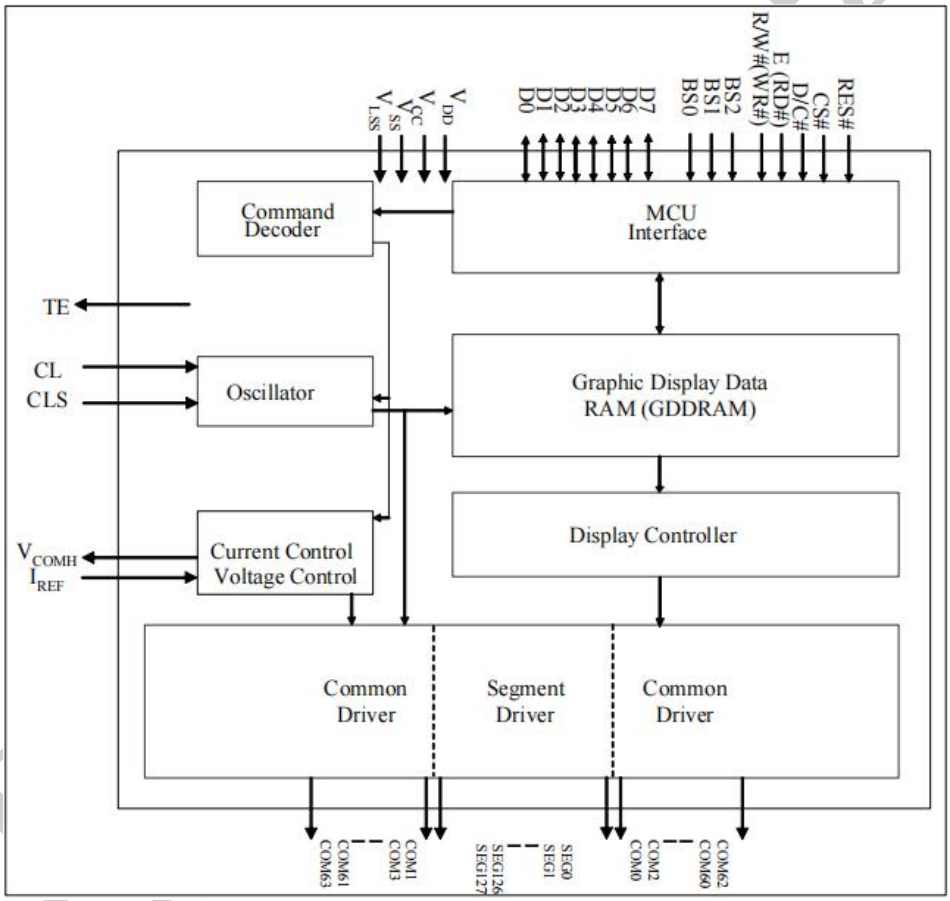


产品参数	
尺寸	0.96 寸
分辨率	128x64
接口类型	SPI、IIC 接口(默认采样 SPI 接口,如果需要 IIC 接口可自行切换电阻位置来调整)
控制芯片	SSD1306

二、SSD1306 简介

SSD1306 是一个单芯片 CMOS OLED/PLED 驱动器，具有有机/聚合物发光二极管点矩阵图形显示系统的控制器。它由 8 页,每页 128 个字节构成。该集成电路是为普通阴极型 OLED 面板设计的。SSD1306 嵌入了对比度控制、显示 RAM 和振荡器，这减少了外部组件的数量和功耗。它有 256 步的亮度控制。数据/命令由通用单片机通过可选择的硬件 6800/8000 系列兼容的并行接口、I2C 接口或串行外围接口发送。

2.1 硬件结构图



2.2 接口选择

SSD1306 单片机接口由 8 个数据引脚和 5 个控制引脚组成。

- 下表总结了不同接口模式下的引脚分配情况：

Pin Name / Bus Interface	Data/Command Interface								Control Signal				
	D7	D6	D5	D4	D3	D2	D1	D0	E	R/W#	CS#	D/C#	RES#
8-bit 8080	D[7:0]								RD#	WR#	CS#	D/C#	RES#
8-bit 6800	D[7:0]								E	R/W#	CS#	D/C#	RES#
3-wire SPI	Tie LOW					NC	SDIN	SCLK	Tie LOW		CS#	Tie LOW	RES#
4-wire SPI	Tie LOW					NC	SDIN	SCLK	Tie LOW		CS#	D/C#	RES#
I^2C	Tie LOW					SDA_{OUT}	SDA_{IN}	SCL	Tie LOW			SA0	RES#

- BS[2:0]的硬件选择可以设置不同的 MCU 模式,BS[2:0]的设置如下表：(0-VSS/1-VDD)

SSD1306 Pin Name	I^2C Interface	6800-parallel interface (8 bit)	8080-parallel interface (8 bit)	4-wire Serial interface	3-wire Serial interface
BS0	0	0	0	0	1
BS1	1	0	1	0	0
BS2	0	1	1	0	0

即用户可以通过对 BS0/BS1/BS2 进行不同的焊接以选择不同的接口模式。

2.3 串行接口(4 线 SPI)

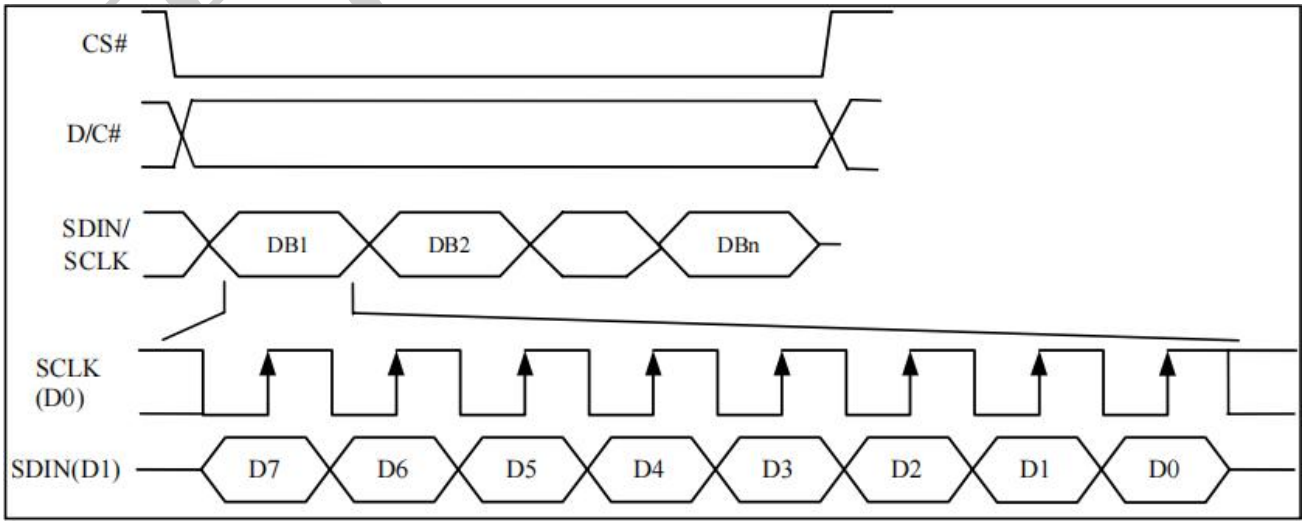
4 线串行接口由串行时钟 SCLK、串行数据：SDIN、D/C#、CS#组成。在 4 线 SPI 模式下，D0 作为 SCLK，D1 作为 SDIN。对于未使用的数据引脚，D2 应该保持打开状态。从 D3 到 D7、E 和 R/W#（WR#）#的引脚可以连接到外部接地。

其中 CS#作为片选引脚，在数据通信的过程中应一致保持低电平状态；而 D/C#作为控制引脚，主要负责选择当前的接口的功能：写命令或写数据；当 D/C#为低电平时表明此时是写命令模式，而 D/C#为高电平时表明此时是写数据模式。

Function	E(RD#)	R/W#(WR#)	CS#	D/C#	D0
Write command	Tie LOW	Tie LOW	L	L	↑
Write data	Tie LOW	Tie LOW	L	H	↑

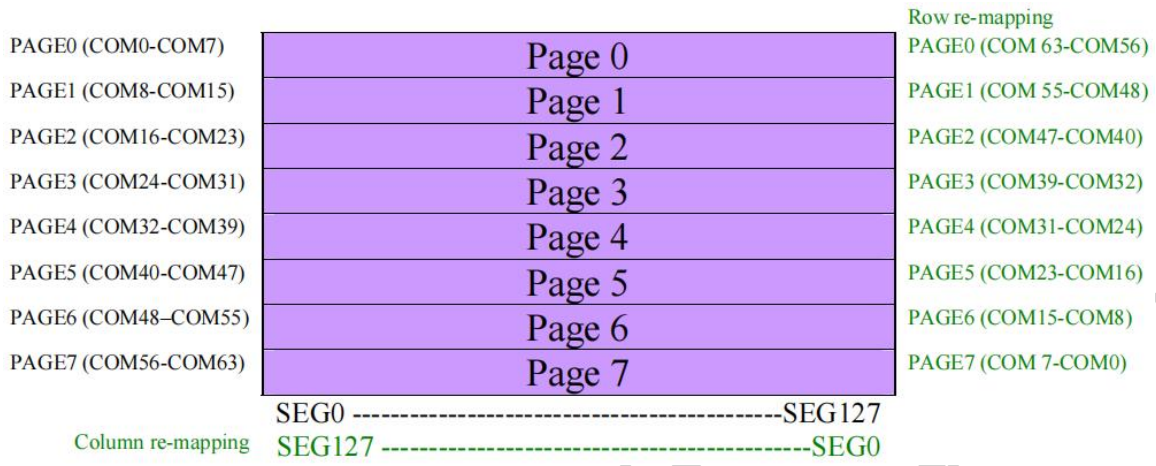
SDIN 在 SCLK 的每个上升沿上按 D7,D6,…D0 的顺序被移位到一个 8 位移位寄存器中。D/C# 在每 8 个时钟上采样，移位寄存器中的数据字节写入图形显示数据在同一时钟内的 RAM (GDDRAM)或命令寄存器。

在串行模式下，只允许进行写操作。下面是数据写入的过程：

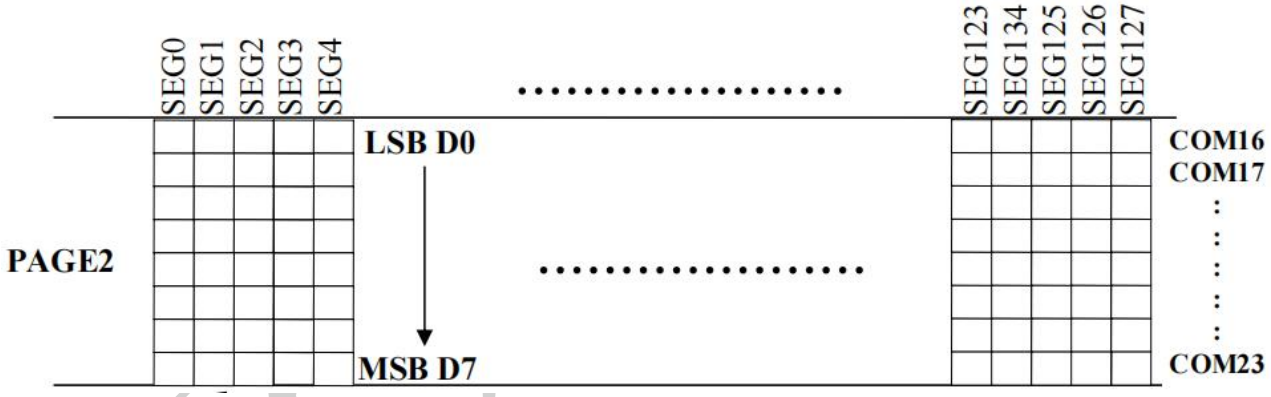


2.4 图形显示数据 RAM(GDDRAM)

GDDRAM 是一个保存要显示的位模式的位映射静态 RAM。内存的大小为 128 × 64 位，内存分为 8 页，PAGE0～PAGE7，每页 128 字节，用于单色 128x64 点阵显示，如下图所示：



当一个数据字节被写入 GDDRAM 时，当前列同一页的所有行图像数据都被填充（即列地址指针指向的整个列（8 位）被填充）。数据位 D0 写入顶部行，而数据位 D7 写入底部行。



2.5 常用命令介绍

序号	指令 HEX	各位描述 D7	D6	D5	D4	D3	D2	D1	D0	功能说明
0	81	1	0	0	0	0	0	0	1	**[设置对比度]** A 的值越大屏幕越亮
	A[7:0]	A7	A6	A5	A4	A3	A2	A1	A0	A 的取值范围：0x00~0xFF
1	AE/AF	1	0	1	0	1	1	1	X0	**[显示屏开关] AE**,关闭 / AF,开启
2	8D	1	0	0	1	1	1	0	1	**[电荷泵开关]**A[2]=0,关闭 / A[2]=1,开启
	A[7:0]	*	*	0	1	0	A2	0	0	
3	B0~B7	1	0	1	1	0	A2	A1	A0	**[设置页地址]** A[2:0]=0~7 对应 0~7 页
4	00~0F	0	0	0	0	A3	A2	A1	A0	**[设置列地址低四位]** 设置 8 位列地址的低四位
5	10~1F	1	1	1	1	A3	A2	A1	A0	**[设置列地址高四位]** 设置 8 位列地址的高四位

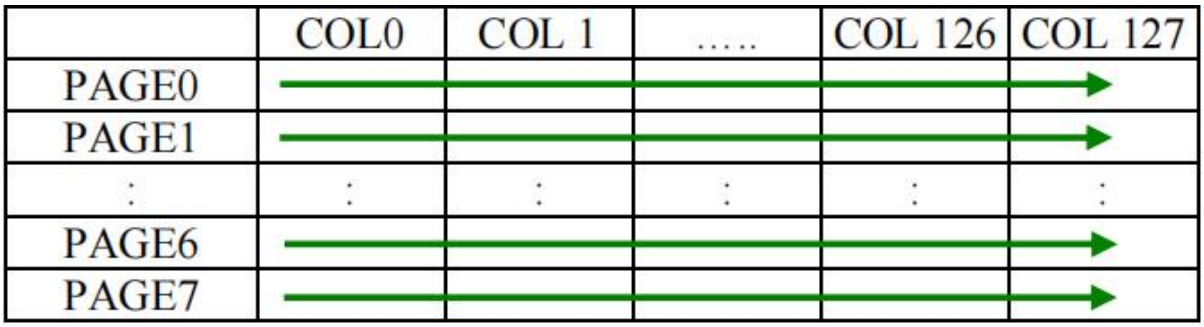
6	26/27	0	0	1	0	0	1	1	X0	**[连续水平滚动模式]** 26,向右滚动 / 27,向左滚动 （按 1 列进行水平滚动） A[7:0]、E[7:0] 、F[7:0] 虚拟字节（固定值） B[2:0] 设置起始页地址 0~7 页 C[2:0] 根据帧频率设置每个滚动间的时间间隔（帧数越少越流畅） 000-5 帧/001-64 帧/010-128 帧/011-256 帧 100-3 帧/100-4 帧 /110-25 帧 /111-2 帧 D[2:0] 设置结束页地址 0~7 页
	A[7:0]	0	0	0	0	0	0	0	0	
	B[2:0]	*	*	*	*	*	B2	B1	B0	
	C[2:0]	*	*	*	*	*	C2	C1	C0	
	D[2:0]	*	*	*	*	*	D2	D1	D0	
	E[7:0]	0	0	0	0	0	0	0	0	
	F[7:0]	1	1	1	1	1	1	1	1	
7	29/2A	0	0	1	0	1	0	X1	X0	**[连续水平垂直滚动模式]** 29,垂直和右水平滚动 / 2A,垂直和左水平滚动 A[7:0] 虚拟字节（固定值） B[2:0] 设置起始页地址 0~7 页 C[2:0] 根据帧频率设置每个滚动间的时间间隔（详细数据参考命令 6） D[2:0] 设置结束页地址 0~7 页 E[5:0] 设置垂直滚动偏移量 0~63 行
	A[7:0]	0	0	0	0	0	0	0	0	
	B[2:0]	*	*	*	*	*	B2	B1	B0	
	C[2:0]	*	*	*	*	*	C2	C1	C0	
	D[2:0]	*	*	*	*	*	D2	D1	D0	
	E[5:0]	*	*	E5	E4	E3	E2	E1	E0	
8	A3	1	0	1	0	0	0	1	1	**[设置垂直滚动区域]** A[5:0] 设置垂直滚动起始行 (0~63) B[6:0] 设置垂直滚动行数 注意：A+B <= 64
	A[5:0]	*	*	A5	A4	A3	A2	A1	A0	
	B[6:0]	*	B6	B5	B4	B3	B2	B1	B0	
8	2E	0	0	1	0	1	1	1	0	**[停止滚动]** 停止后须重写 RAM 中的数据
9	2F	0	0	1	0	1	1	1	1	**[启动滚动]** 不支持同时运行多种滚动模式

2.5.1 设置内存寻址模式(20h)

在 SSD1306 中，有 3 种不同的内存寻址模式：页面寻址模式、水平寻址模式和垂直寻址模式。此命令将内存寻址的方式设置为上述三种模式之一。

2.5.1.1 页面寻址模式(A[1:0]=10b)

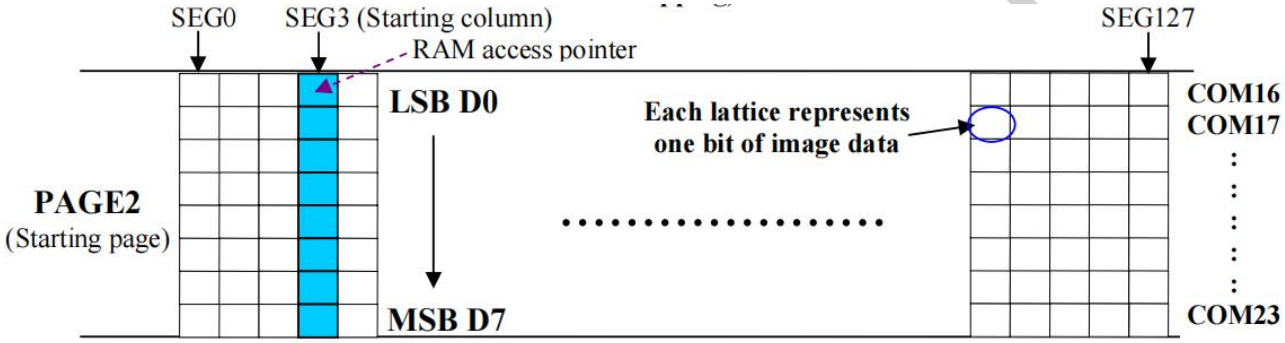
在页面寻址模式下，读写显示 RAM 后，列地址指针自动增加 1。如果列地址指针到达列结束地址，则列地址指针将被重置为列起始地址，而页面地址指针将不会被更改。用户必须设置新的页面和列地址，才能访问下一页的 RAM 内容。页面寻址模式的移动顺序和列地址点如图所示：



在正常显示数据 RAM 读写和页面寻址模式下，需要执行以下步骤来定义启动 RAM 访问指针位置：

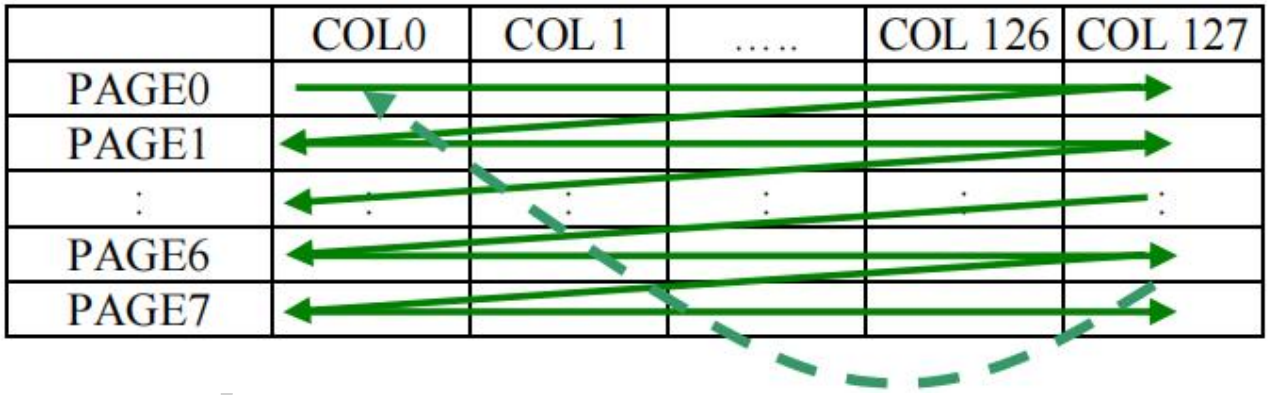
- 通过命令 B0h 到 B7h 设置目标显示位置的页面开始地址。
- 通过命令 00h~0Fh 设置指针的下起始列地址。
- 通过命令 10h~1Fh 设置指针的上起始列地址。

例如,如果页面地址设置为 B2h,下列地址为 03h,上列地址为 10h,则表示起始列为 PAGE2 的 SEG3。RAM 访问指针的位置如图所示。输入数据字节将被写入第 3 列的 RAM 位置。



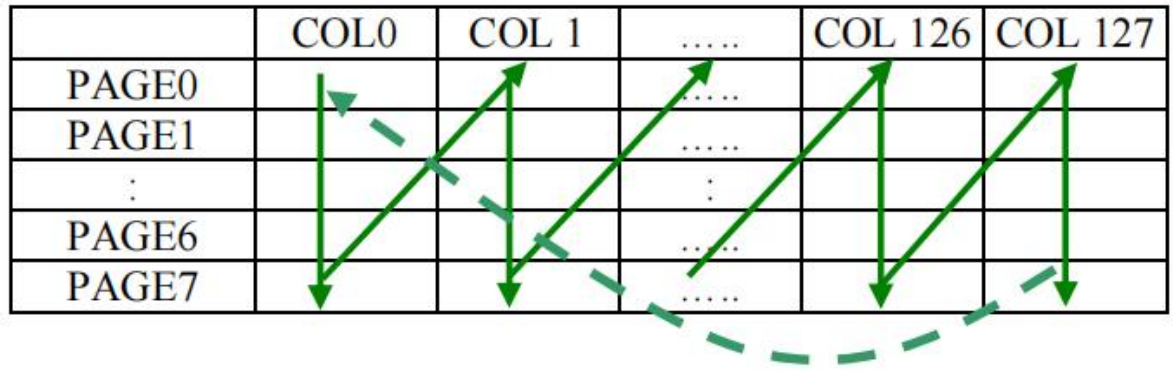
2.5.1.2 水平寻址模式(A[1:0]=00b)

在水平寻址模式下，在显示器读写 RAM 后，列地址指针自动增加 1。如果列地址指针到达列结束地址，则将列地址指针重置为列起始地址，页地址指针增加 1。水平寻址模式下的页面和列地址点的移动顺序如图所示。当列和页面地址指针都到达结束地址时，指针将被重置为列开始地址和页面开始地址:



2.5.1.3 垂直寻址模式(A[1:0]=01b)

在垂直寻址模式下，读写显示 RAM 后，页面地址指针自动增加 1。如果页面地址指针到达页面结束地址，则页面地址指针重置为页面起始地址，列地址指针增加 1。垂直寻址模式的页面和列地址点的移动顺序如图所示。当列和页面地址指针都到达结束地址时，指针将被重置为列开始地址和页面开始地址：



在正常显示数据 RAM 读或写和水平/垂直寻址模式下，需要执行以下步骤来定义 RAM 访问指针位置：

- 通过命令 21h 设置目标显示位置的列开始和结束地址。
- 通过命令 22h 设置目标显示位置的页面开始和结束地址。

2.7 初始化序列

```
OLED_WR_Byte(0xAE, OLED_CMD);//--关闭显示
OLED_WR_Byte(0x00, OLED_CMD);//--set low column address
OLED_WR_Byte(0x10, OLED_CMD);//--set high column address
OLED_WR_Byte(0x40, OLED_CMD);//--set start line address
OLED_WR_Byte(0x81, OLED_CMD);//--设置屏幕对比度寄存器寄存器
OLED_WR_Byte(0xCF, OLED_CMD);//--设置对比度[0x00~0xFF]
OLED_WR_Byte(0xA1, OLED_CMD);//--Set SEG/Column Mapping     0xa0 左右反置 0xa1 正常
OLED_WR_Byte(0xC8, OLED_CMD);//--Set COM/Row Scan Direction 0xc0 上下反置 0xc8 正常
OLED_WR_Byte(0xA6, OLED_CMD);//--set normal display
OLED_WR_Byte(0xA8, OLED_CMD);//--set multiplex ratio(1 to 64)
OLED_WR_Byte(0x3f, OLED_CMD);//--1/64 duty
OLED_WR_Byte(0xD3, OLED_CMD);//--set display offset
OLED_WR_Byte(0x00, OLED_CMD);//--not offset
OLED_WR_Byte(0xd5, OLED_CMD);//--set display clock divide ratio/oscillatorfrequency
OLED_WR_Byte(0x80, OLED_CMD);//--set divide ratio, Set Clock as 100 Frames/Sec
OLED_WR_Byte(0xD9, OLED_CMD);//--set pre-charge period
OLED_WR_Byte(0xF1, OLED_CMD);//--Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
OLED_WR_Byte(0xDA, OLED_CMD);//--set com pins hardware configuration
OLED_WR_Byte(0x12, OLED_CMD);
OLED_WR_Byte(0xDB, OLED_CMD);//--set vcomh
OLED_WR_Byte(0x40, OLED_CMD);//--Set VCOM Deselect Level
OLED_WR_Byte(0x20, OLED_CMD);//--设置寻址方式 (0x00/0x01/0x02)
OLED_WR_Byte(0x02, OLED_CMD);//--页面寻址模式(页内列地址自增)
OLED_WR_Byte(0x8D, OLED_CMD);//--set Charge Pump enable/disable
OLED_WR_Byte(0x14, OLED_CMD);//--set(0x10) disable
OLED_WR_Byte(0xA4, OLED_CMD);//--Disable Entire Display On (0xa4/0xa5)
OLED_WR_Byte(0xA6, OLED_CMD);//--设置显示模式(0Xa6 正常模式亮为 1/0xa7 反显模式亮为 0)
OLED_WR_Byte(0xAF, OLED_CMD);//--开启显示
```

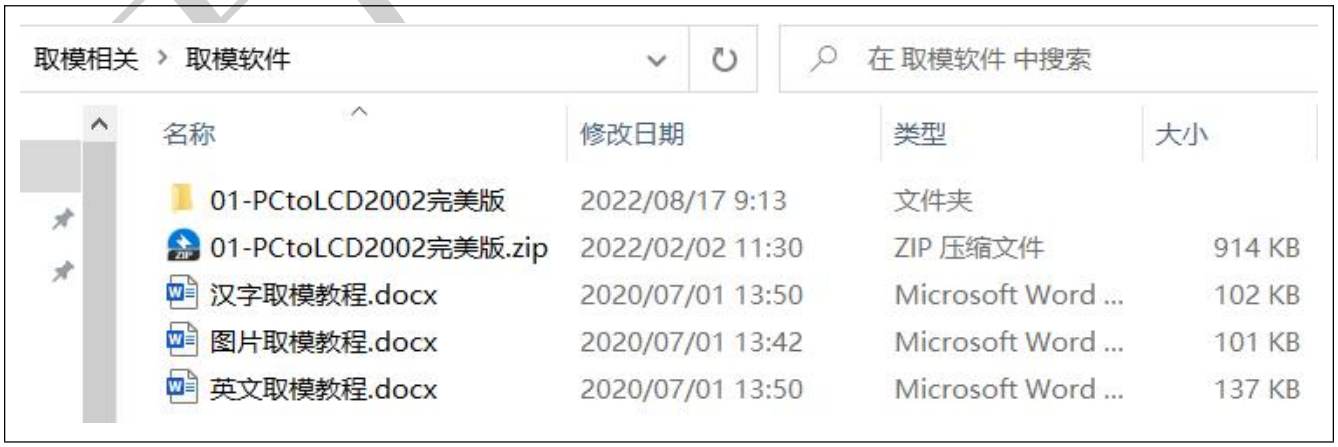
三、OLED 基础功能应用

SPI 接口定义			
序号	符号	引脚	说明
1	GND	GND	电源地
2	VCC	VCC	电源正(3.3~5v)
3	D0	PB14	SPI 时钟线
4	D1	PB13	SPI 数据线
5	RES#	PB12	复位线
6	DC#	PB1	数据/命令选择线
7	CS#	PA7	SPI 片选信号线(该引脚不能悬空,否则会导致 OLED 不能稳定工作)

3.1 设置坐标

```
/**
 * 函数功能：设置坐标
 * 形参说明：page - 页地址[0~7]
 *           col  - 列地址[0~127]
 * 返 回 值：
 * 寻址方式：页面寻址/页内寻址(模式) -- 特点:页内列地址会自增
 */
void OLED_Set_Coord(u8 page, u8 col)
{
    //设置页地址
    SPI_Write_Byte(0xB0+page, OLED_CMD);
    //设置列地址：
    SPI_Write_Byte(0x00|(col&0x0F), OLED_CMD); //设置列地址的低四位[0x00 ~ 0x0F]
    SPI_Write_Byte(0x10|(col>>4), OLED_CMD);   //设置列地址的高四位[0x10 ~ 0x1F]
}
```

3.2 数据取模



3.3 数据显示

- 核心函数(基于取模方式进行数据还原)

```
/**
 * 函数功能：显示单个数据(把取模后的数据还原到屏幕上)
 * 形参说明：page - 页地址  col - 列地址
            w - 单个数据的宽度  h - 单个数据的高度
            pbuff - 该数据取模后的字节数组
 * 返 回 值：
 * 取模方式：阴码,列行式,逆向(低位在前),16 进制,C51 格式
 */
void OLED_Display_Data(u8 page, u8 col, u8 w, u8 h, const char *buff)
{
    u8 i, j;
    for (i=0; i<h/8; i++) //控制页数
    {
        //设置坐标
        page++;
        //坐标超出屏幕有效范围时自动退出
        if (page > OLED_WIDTH-1 || col > OLED_HEIGHT-1) return;
        OLED_Set_Coord(page, col);
        //写数据
        for (j=0; j<w; j++) //控制页内要写多少个字节数据
        {
            SPI_Write_Byte(buff[i*w+j], OLED_DAT);
        }
    }
}
```

四、OLED 高级功能应用

4.1 构造显存

```
static u8 OLED_GRAM[OLED_WIDTH/8][OLED_HEIGHT];  //构造和显示屏等大的显示缓存区
```

4.2 画点函数

```
void OLED_Draw_Point(u8 x, u8 y, u8 c)
{
    if(x > OLED_WIDTH-1 || y > OLED_HEIGHT-1) return;
    if(c)
        OLED_GRAM[y/8][x] |= (1<<(y%8));
    else
        OLED_GRAM[y/8][x] &= ~(1<<(y%8));
}
```

4.3 显存刷新函数

```
void OLED_GRAM_Refresh(void)
{
    u8 i, j;
    for(i=0; i<OLED_HEIGHT/8; i++)
    {
        //设置坐标
        SPI_Write_Byte(0xB0+i, OLED_CMD);//设置页地址
        SPI_Write_Byte(0x00, OLED_CMD);  //设置列地址的低四位[0x00 ~ 0x0F]
        SPI_Write_Byte(0x10, OLED_CMD);  //设置列地址的高四位[0x10 ~ 0x1F]
        for(j=0; j<OLED_WIDTH; j++)
        {
            SPI_Write_Byte(OLED_GRAM[i][j], OLED_DAT);
        }
    }
}
```

4.4 数据显示

```

/**
 * 函数功能：把数据写的显存中
 * 形参说明：x - 横坐标
             y - 纵坐标
             w - 宽度
            h - 高度
            buff - 取模的数据
            mode - 0 正常模式(亮 1/灭 0)  1 反显模式(亮 0/灭 1)
 * 返 回 值：
 * 取模方式：阴码,列行式,逆向(低位在前),16 进制,C51 格式
 */
void OLED_Draw_Data(u8 x, u8 y, u8 w, u8 h, const char *buff, u8 mode)
{
    u16 i, j; //至少是 16bit
    u8 x0 = x;
    u8 y0 = y;
    while(h%8) h++;         //高度补全
    for(i=0; i<w*h/8; i++)  //计算要写入多少个字节
    {
        //取一个字节
        u8 tmp = buff[i];
        //往显存写入一个字节的数据
        for(j=0; j<8; j++)
        {
            if(mode) //反显模式
```

```
                OLED_Draw_Point(x, y, (tmp>>j & 0x01) ? 0:1);
            else     //正常模式
                OLED_Draw_Point(x, y, (tmp>>j & 0x01) ? 1:0);
            y++; //
        }
        x++;
        if(x - x0 == w) //判断当前页的数据有没有写完
        {
            x = x0;     //x 复位
            y0 += 8;    //y0 更新
        }
        y = y0;
    }
}
```

4.5 多级目录显示框架

```
//1-构建结构体
typedef struct
{
    unsigned char CurrentTaskNum;   //当前任务编号:页码
    unsigned char Up;    //上键
    unsigned char Down;  //下键
    unsigned char Return;    //返回
    unsigned char Enter; //确认键
    void (*CurrentTaskFunc)(void);  //当前操作（函数指针）
}taskInfo;

//2-任务调度表
taskInfo taskTable[] =
{
   {0, 0, 0, 0, 0, fun1},

   {1, 3, 2, 1, 4, fun1},
   {2, 1, 3, 2, 5, fun2},
   {3, 2, 1, 3, 6, fun3},
};

//3-申请基本变量
u8 taskIndex = 0; //初始任务
void (*taskFunc)(void);

//4-调度
switch(KEY_GetVal())
{
    case KEY1_PRES:  taskIndex = taskTable[taskIndex].Up;    break;  //前进
```

```
    case KEY2_PRES:  taskIndex = taskTable[taskIndex].Return;break;  //后退
    case KEY3_PRES:  taskIndex = taskTable[taskIndex].Down;  break;  //返回
    case KEY4_PRES:  taskIndex = taskTable[taskIndex].Enter; break;  //确认
}
//更新任务
taskFunc = taskTable[taskIndex].CurrentTaskFunc;
OLED_GRAM_Init();    //初始化 GRAM 显存
(*taskFunc)();
OLED_GRAM_Refresh();//刷新 GRAM
```

五、点阵字库

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

5.1 6x8 点阵字库

```
//取模方式：阴码、逆向、列行式、C51 格式
const unsigned char Ascii_6x8_ZIK[][6] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00,//
    0x00, 0x00, 0x00, 0x2f, 0x00, 0x00,// !
    0x00, 0x00, 0x07, 0x00, 0x07, 0x00,// "
    0x00, 0x14, 0x7f, 0x14, 0x7f, 0x14,// #
    0x00, 0x24, 0x2a, 0x7f, 0x2a, 0x12,// $
    0x00, 0x62, 0x64, 0x08, 0x13, 0x23,// %
    0x00, 0x36, 0x49, 0x55, 0x22, 0x50,// &
    0x00, 0x00, 0x05, 0x03, 0x00, 0x00,// '
    0x00, 0x00, 0x1c, 0x22, 0x41, 0x00,// (
    0x00, 0x00, 0x41, 0x22, 0x1c, 0x00,// )
    0x00, 0x14, 0x08, 0x3E, 0x08, 0x14,// *
    0x00, 0x08, 0x08, 0x3E, 0x08, 0x08,// +
    0x00, 0x00, 0x00, 0xA0, 0x60, 0x00,// ,
    0x00, 0x08, 0x08, 0x08, 0x08, 0x08,// -
    0x00, 0x00, 0x60, 0x60, 0x00, 0x00,// .
    0x00, 0x20, 0x10, 0x08, 0x04, 0x02,// /
    0x00, 0x3E, 0x51, 0x49, 0x45, 0x3E,// 0
    0x00, 0x00, 0x42, 0x7F, 0x40, 0x00,// 1
    0x00, 0x42, 0x61, 0x51, 0x49, 0x46,// 2
    0x00, 0x21, 0x41, 0x45, 0x4B, 0x31,// 3
    0x00, 0x18, 0x14, 0x12, 0x7F, 0x10,// 4
    0x00, 0x27, 0x45, 0x45, 0x45, 0x39,// 5
    0x00, 0x3C, 0x4A, 0x49, 0x49, 0x30,// 6
    0x00, 0x01, 0x71, 0x09, 0x05, 0x03,// 7
    0x00, 0x36, 0x49, 0x49, 0x49, 0x36,// 8
    0x00, 0x06, 0x49, 0x49, 0x29, 0x1E,// 9
```

```
    0x00, 0x00, 0x36, 0x36, 0x00, 0x00,// :
    0x00, 0x00, 0x56, 0x36, 0x00, 0x00,// ;
    0x00, 0x08, 0x14, 0x22, 0x41, 0x00,// <
    0x00, 0x14, 0x14, 0x14, 0x14, 0x14,// =
    0x00, 0x00, 0x41, 0x22, 0x14, 0x08,// >
    0x00, 0x02, 0x01, 0x51, 0x09, 0x06,// ?
    0x00, 0x32, 0x49, 0x59, 0x51, 0x3E,// @
    0x00, 0x7C, 0x12, 0x11, 0x12, 0x7C,// A
    0x00, 0x7F, 0x49, 0x49, 0x49, 0x36,// B
    0x00, 0x3E, 0x41, 0x41, 0x41, 0x22,// C
    0x00, 0x7F, 0x41, 0x41, 0x22, 0x1C,// D
    0x00, 0x7F, 0x49, 0x49, 0x49, 0x41,// E
    0x00, 0x7F, 0x09, 0x09, 0x09, 0x01,// F
    0x00, 0x3E, 0x41, 0x49, 0x49, 0x7A,// G
    0x00, 0x7F, 0x08, 0x08, 0x08, 0x7F,// H
    0x00, 0x00, 0x41, 0x7F, 0x41, 0x00,// I
    0x00, 0x20, 0x40, 0x41, 0x3F, 0x01,// J
    0x00, 0x7F, 0x08, 0x14, 0x22, 0x41,// K
    0x00, 0x7F, 0x40, 0x40, 0x40, 0x40,// L
    0x00, 0x7F, 0x02, 0x0C, 0x02, 0x7F,// M
    0x00, 0x7F, 0x04, 0x08, 0x10, 0x7F,// N
    0x00, 0x3E, 0x41, 0x41, 0x41, 0x3E,// O
    0x00, 0x7F, 0x09, 0x09, 0x09, 0x06,// P
    0x00, 0x3E, 0x41, 0x51, 0x21, 0x5E,// Q
    0x00, 0x7F, 0x09, 0x19, 0x29, 0x46,// R
    0x00, 0x46, 0x49, 0x49, 0x49, 0x31,// S
    0x00, 0x01, 0x01, 0x7F, 0x01, 0x01,// T
    0x00, 0x3F, 0x40, 0x40, 0x40, 0x3F,// U
    0x00, 0x1F, 0x20, 0x40, 0x20, 0x1F,// V
    0x00, 0x3F, 0x40, 0x38, 0x40, 0x3F,// W
    0x00, 0x63, 0x14, 0x08, 0x14, 0x63,// X
    0x00, 0x07, 0x08, 0x70, 0x08, 0x07,// Y
    0x00, 0x61, 0x51, 0x49, 0x45, 0x43,// Z
    0x00, 0x00, 0x7F, 0x41, 0x41, 0x00,// [
    0x00, 0x55, 0x2A, 0x55, 0x2A, 0x55,// "\"
    0x00, 0x00, 0x41, 0x41, 0x7F, 0x00,// ]
    0x00, 0x04, 0x02, 0x01, 0x02, 0x04,// ^
    0x00, 0x40, 0x40, 0x40, 0x40, 0x40,// _
    0x00, 0x00, 0x01, 0x02, 0x04, 0x00,// '
    0x00, 0x20, 0x54, 0x54, 0x54, 0x78,// a
    0x00, 0x7F, 0x48, 0x44, 0x44, 0x38,// b
    0x00, 0x38, 0x44, 0x44, 0x44, 0x20,// c
    0x00, 0x38, 0x44, 0x44, 0x48, 0x7F,// d
    0x00, 0x38, 0x54, 0x54, 0x54, 0x18,// e
    0x00, 0x08, 0x7E, 0x09, 0x01, 0x02,// f
    0x00, 0x18, 0xA4, 0xA4, 0xA4, 0x7C,// g
```

```
    0x00, 0x7F, 0x08, 0x04, 0x04, 0x78,// h
    0x00, 0x00, 0x44, 0x7D, 0x40, 0x00,// i
    0x00, 0x40, 0x80, 0x84, 0x7D, 0x00,// j
    0x00, 0x7F, 0x10, 0x28, 0x44, 0x00,// k
    0x00, 0x00, 0x41, 0x7F, 0x40, 0x00,// l
    0x00, 0x7C, 0x04, 0x18, 0x04, 0x78,// m
    0x00, 0x7C, 0x08, 0x04, 0x04, 0x78,// n
    0x00, 0x38, 0x44, 0x44, 0x44, 0x38,// o
    0x00, 0xFC, 0x24, 0x24, 0x24, 0x18,// p
    0x00, 0x18, 0x24, 0x24, 0x18, 0xFC,// q
    0x00, 0x7C, 0x08, 0x04, 0x04, 0x08,// r
    0x00, 0x48, 0x54, 0x54, 0x54, 0x20,// s
    0x00, 0x04, 0x3F, 0x44, 0x40, 0x20,// t
    0x00, 0x3C, 0x40, 0x40, 0x20, 0x7C,// u
    0x00, 0x1C, 0x20, 0x40, 0x20, 0x1C,// v
    0x00, 0x3C, 0x40, 0x30, 0x40, 0x3C,// w
    0x00, 0x44, 0x28, 0x10, 0x28, 0x44,// x
    0x00, 0x1C, 0xA0, 0xA0, 0xA0, 0x7C,// y
    0x00, 0x44, 0x64, 0x54, 0x4C, 0x44,// z
    0x14, 0x14, 0x14, 0x14, 0x14, 0x14,// {
};
```