2.9inch e-Paper Module (B)

用户手册

版本	日期	内容
V1.0	2017.05.17	初版
V2.0	2018.12.05	增加注意事项，程序更新

【说明】

- 使用产品之前，请阅读本册，非正常使用造成的产品损坏，不在保修范围内
- 本手册是基于提供的示例程序，进行介绍说明
- 本手册默认用户使用的是带驱动板的屏幕（无论是购买驱动板或者自己设计驱动板）
- 用户手册以及程序可能会不定期更新，恕不另外通知，尽请谅解

目录

版本说明

本墨水屏有裸屏版本和带驱动板版本。

裸屏版本只有屏幕，使用的时候需要配合驱动板使用，如果您是第一次购买使用微雪墨水屏，建议您购买带驱动板版本，或者再另外购买一款驱动板做测试。

带驱动板的版本，已经将驱动电路集成在背面的 PCB 上面，使用的时候可以使用随屏配送的 8PIN 排线接到单片机控制使用。

裸屏（2.9inch e-Paper (B)）：

带驱动板（2.9inch e-Paper Module (B)）：

注意事项

1. 支持局刷的屏幕，注意使用的时候不能一直用局刷对屏幕进行刷新，需要在做几次局刷之后，对屏幕进行一次全刷清屏。否则会造成屏幕显示效果异常，**无法修复!**
2. 三色的墨水屏在批量的时候，会存在一定的色差，这个是正常现象，建议将屏幕刷白朝上存放，使用前，先上电做几次刷白操作
3. 注意屏幕不能长时间上电，在屏幕不刷新的时候，要将屏幕设置成睡眠模式，或者进行断电处理。否则屏幕长时间保持高电压状态，会损坏膜片，**无法修复!**
4. 使用墨水屏的时候，建议刷新时间间隔至少是 180s，并且至少每 24 小时做一次刷新，如果长期不使用墨水屏的话，要将墨水屏刷白存放。（具体储存环境需求参考数据手册）
5. 屏幕进入睡眠模式之后，会忽略发送的图片数据，只有重新初始化才能正常刷新
6. 控制 0x3C 寄存器可以调节边框颜色，在例程中可以调节 Border Waveform Control 寄存器或者 VCOM AND DATA INERTVAL SETTING 进行设置
7. 如果发现制作的图片数据在屏幕上显示错误，建议检查一下图片大小设置是否正确，调换一下宽度和高度设置再试一下。
8. 屏幕无法在阳光直射的环境下工作，注意刷新过程最好在室内完成
9. 墨水屏的工作电压要求是 3.3V，如果您购买的是裸屏的话，设计电路的时候如果需要配合 5V 工作环境的话，建议做一下电平转换处理。带驱动版版本（新版本）加入了电平处理电路，可以同时支持 3.3V 和 5V 工作环境， 老版本只能支持 3.3V 工作环境，使用的时候可以先确认一下版本。（PCB 上带有芯片的一般是新版本）
10. 屏幕的 FPC 排线比较脆弱，注意使用的时候沿屏幕水平方向弯曲排线，不可以沿屏幕垂直方向弯曲排线
11. 墨水屏屏幕较为脆弱，注意尽量避免跌落，碰撞，用力按压。

12. 我们建议客户拿到屏幕之后，先用我们提供的示例程序，使用对应的开发板进行测试

产品简介

- 本产品是 2.9inch 电子墨水屏模块，分辨率为 296x128，带有内部控制器，使用 SPI 接口进行通信，可显示红黑白三种颜色
- 墨水屏具有功耗低，可视角度大，阳光下仍可清晰显示等优点[1]，常用于货架标签，工业仪表等显示应用。
- 本产品提供有基于 Arduino, Raspberry Pi 和 STM32 三款开发板/卡片式电脑的示例程序

产品参数

- 工作电压： 3.3V/5V[2] (裸屏版本只支持 3.3V)
- 工作温度：0~50°C
- 通信接口：3-wire SPI/4-wire SPI (默认 4-wire SPI)
- 外形尺寸：
  - 驱动版：89.5mm × 38mm
  - 裸屏：66.89mm × 29.05mm
- 显示尺寸：66.89mm × 29.05mm
- 点距：0.138 × 0.138
- 分辨率： 296 × 128
- 显示颜色：红、黑、白
- 刷新时间[3]：
  - 全局刷新：15s

[1] 由于屏幕没有防紫外线层，所以不支持直接在阳光直照的环境下工作刷新的。

[2] 注意这里的 5V 是指整个系统的工作电平都是 5V，如果供电 5V，其他工作电平是 3.3V 的话，屏幕是不能正常工作的

[3] 该数据为实验数据，实际使用的时候受通信情况和使用环境影响，可能存在误差

- 功耗[4]：
  - 刷新功耗：26.4mW(typ.)
  - 待机功耗：<0.017mW
- 可视角度：>170°

接口说明（驱动板）

VCC ：3.3V/5V

GND ：GND

DIN ：SPI 通信 MOSI 引脚

CLK ：SPI 通信 SCK 引脚

CS ：SPI 片选引脚（低电平有效）

DC ：数据/命令控制引脚（高电平表示数据， 低电平表示命令）

RST ：外部复位引脚（低电平复位）

BUSY ：忙状态输出引脚（低电平表示忙）

[4] 该数据为实验数据，实际使用的时候受使用环境影响，可能存在误差

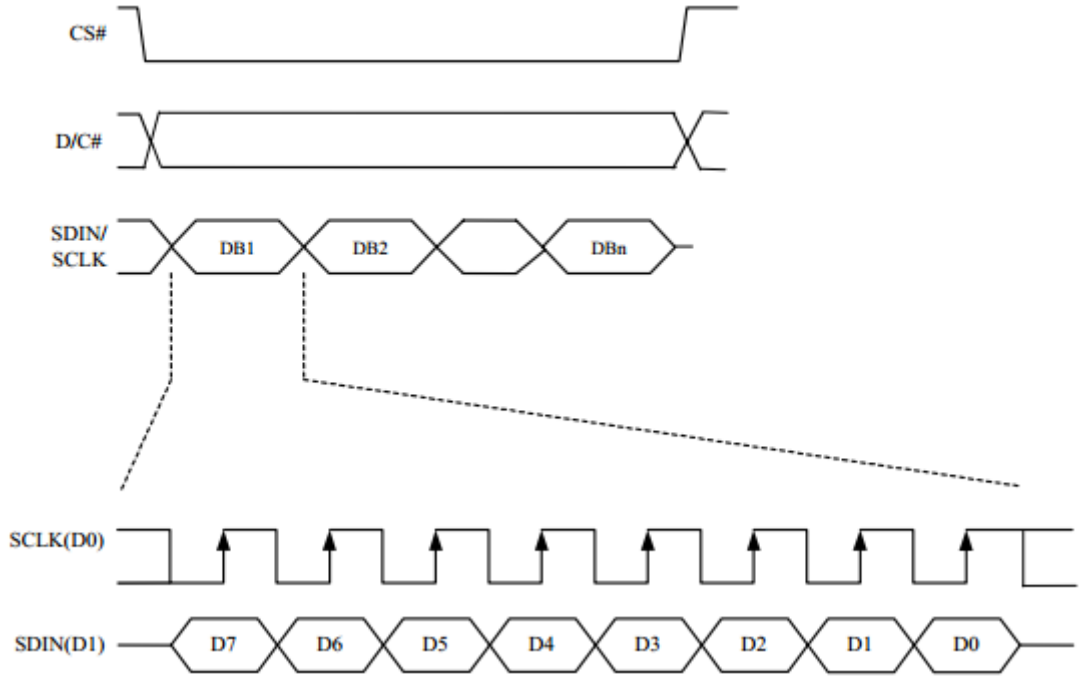
工作原理

器件介绍

本产品使用的电子纸采用“微胶囊电泳显示”技术进行图像显示，其基本原理是悬浮在液体中的带电纳米粒子受到电场作用而产生迁移。电子纸显示屏是靠反射环境光来显示图案的，不需要背光，即使是在阳光底下，电子纸显示屏依然清晰可视，可视角度几乎达到了 180°。因此，电子纸显示屏非常适合阅读。

通信协议（4-WIRE SPI）

墨水屏的通信方式是 SPI[5] （详情参考数据手册）



注：与传统的 SPI 协议不同的地方是：由于是只需要显示，故而将从机发往主机的数据线（MISO）进行了隐藏，只保留了主机发往从机的数据线，也就是 MOSI。

CS 为从机片选， 仅当 CS 为低电平时，芯片才会被使能。

DC 为芯片的数据/命令控制引脚，当 DC = 0 时写入命令；DC = 1 时写入数据。

[5] 关于标准的 SPI 协议，有需要的客户可以自行搜索网上资料了解

SCLK 为 SPI 通信时钟。

SDIN 为 SPI 通信主机发往从机数据。

对于 SPI 通信而言，数据是有传输时序的，即时钟相位（CPHA）与时钟极性(CPOL)的组合：

1) CPOL 的高低决定串行同步时钟的空闲状态电平，CPOL = 0，空闲电平为低电平。CPOL 对传输影响不大；

2) CPHA 的高低决定串行同步时钟是在第一个时钟跳变沿还是第二个时钟跳变沿数据被采集，当 CPHL = 0，在第一个跳变沿进行数据采集；

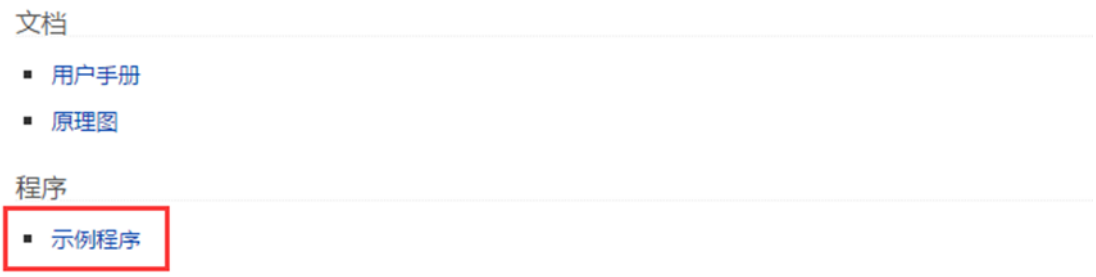
这两者组合就成为四种 SPI 通信方式，通常使用 SPI0，即 CPHL = 0，CPOL = 0。

从图中可以看出，当 SCLK 第一个下降沿时开始传输数据，一个时钟周期传输 8bit 数据，使用 SPI0，按位传输,高位在前,低位在后。

示例程序

下载例程

在微雪官网找到对应产品，在产品资料打开下载路径，进入到 wiki 界面，下载示例程序



解压下载到的文件，得到如下文件：

Arduino[6]： Arduino 例程，以 UNO 开发板为例；

RaspberryPi：树莓派例程，包含 BCM2835、WiringPi 和 python 三种例程

STM32： STM32 例程，以 open103Z 开发板为例，基于 STM32F103ZET6

[6] 我们提供的 Arduino 例程，是基于 Arduino UNO 开发板的，客户如果使用其他的 Arduino 开发板注意先确认一下引脚和程序是否完全兼容。

树莓派

我们分别提供了 BCM2835， wiringPi， python2 和 python3 示例程序，用户可以根据自己的需求参考和使用示例程序。

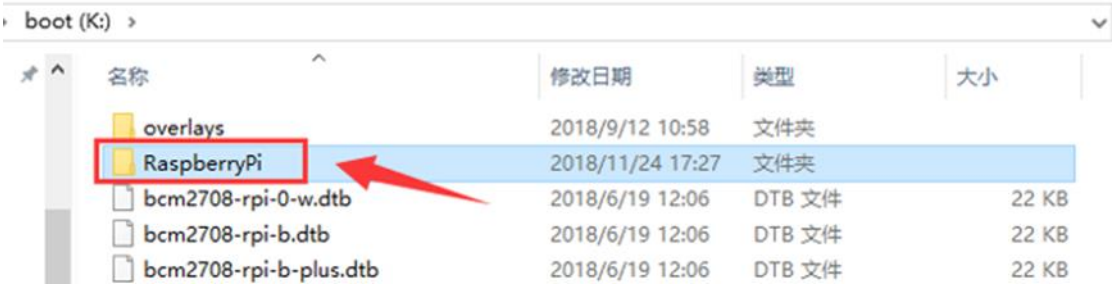
硬件连接

e-Paper	树莓派	
	BCM2835 编号	**物理引脚**
3.3V	3.3V	3.3V
GND	GND	GND
DIN	MOSI	19
CLK	SCLK	23
CS	CE0	24
DC	25	22
RST	17	11
BUSY	24	18

复制程序到树莓派

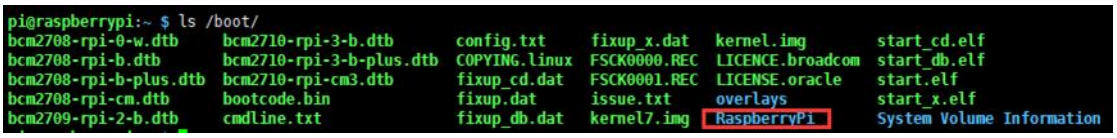
1. 将安装有 Raspbian 镜像的 SD 卡插入到电脑中，

2. 将之前下载下来的程序中的 RaspberryPi 文件夹复制到 SD 卡的根目录下



3. 将 SD 卡插入到树莓派，并上电，查看/boot 目录可以看到刚刚复制进去的文件

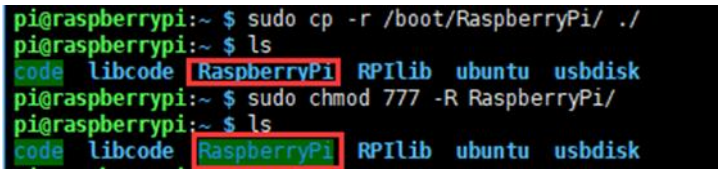
ls /boot



4. 将文件复制到用户目录下，并修改权限：

sudo cp -r /boot/RaspberryPi/ ./

sudo chmod 777 -R RaspberryPi/



安装函数库（需要联网操作）

使用示例程序前，需要先安装必要的函数库，否则程序无法正常使用

安装 BCM2835

函数库获取链接：http://www.airspayce.com/mikem/bcm2835/

进入 BCM2835 的官网下载最新的函数库安装包，并复制到树莓派上面。运行一下指令进行安装。

```
sudo tar zxvf bcm2835-1.xx.tar.gz

cd bcm2835-1.xx

sudo ./configure

make

sudo make check

sudo make install
```

注意：指令中的 xx 代表的是你下载的安装包的版本号，比如，如果下载的是 bcm2835-1.52,

那么第一条指令执行的应该是 sudo tart zxvf bcm2835-1.52.tar.gz

安装 WIRINGPI

在树莓派终端直接运行以下指令安装：

```
sudo apt-get install git

sudo git clone git://git.drogon.net/wiringPi

cd wiringPi

sudo ./build
```

安装 PYTHON2 函数库

在终端运行以下指令

```
sudo apt-get install python-pip

sudo apt-get install python-pip

sudo apt-get install python-imaging

sudo pip install spidev

sudo pip install RPi.GPIO
```

安装 PYTHON3 函数库

在树莓派终端运行指令

```
sudo apt-get install python3-pip

sudo apt-get install python-imaging

sudo pip3 install spidev

sudo pip3 install RPi.GPIO

sudo pip3 install Pillow
```

安装 Pillow 如果报错: ImportError: libopenjp2.so.7: cannot open shared object file: No such file or directory，则先执行如下指令：sudo apt-get install libopenjp2-7-dev

运行程序

安装好函数库之后，将程序复制到树莓派中，并进入对应的文件夹中：

运行 bcm2835 程序：

```
cd ~/RaspberryPi/bcm2835    #进入程序所在目录

make                        #编译程序

sudo ./epd                  #运行程序
```

运行 wiringpi 程序：

```
cd ~/RaspberryPi/wiringpi   #进入程序所在目录

make                        #编译程序

sudo ./epd                  #运行程序
```

运行 python2 程序：

```
cd ~/RaspberryPi/python2     #进入程序所在目录

sudo python main.py          #运行程序
```

运行 python3 程序：

```
cd ~/RaspberryPi/python3     #进入程序所在目录

sudo python3 main.py         #运行程序
```

预期结果

1) 整个屏幕全部刷新成白色

2) 显示一张全局图片

3) 画圈画线，显示两种大小的字符

刷新显示的时候是先刷新黑色，后刷新红色的

可以按 ctrl+c 退出程序

ARDUINO

注意：这里提供的程序是基于 Module 版本或者 e-Paper driver HAT 版本的。如果购买了 e-paper shield 驱动板，可以参考 e-paper shield 的例程，这里不再做另外说明

硬件连接

硬件连接到开发板 UNO PLUS，注意，如果你接入的 Arduino 开发板不是 Arduino UNO 的话，可能需要根据具体硬件情况修改 SPI 引脚（DIN 和 CLK）的连接。

e-Paper	Arduino UNO
3.3V	3V3
GND	GND
DIN	D11
CLK	D13
CS	D10
DC	D9
RST	D8
BUSY	D7

运行程序

1. 确保你已经安装了 1.8.5 版本以上的 Arduino IDE 软件。
2. 打开 Arduino 工程文件
3. 设置好开发板和对应的串口
4. 将程序下载到开发板中

预期结果.

由于使用 UNO PLUS 开发板，其全局变量空间只有 2Kb，因此不能创建一个完整的图像缓存，通过分多次传输数据的方式进行全局显示：

运行了程序之后，会显示一张图片。（先显示黑色部分，后显示红色部分）

STM32

本例程使用的开发板为 Open103Z. 例程基于 HAL 库

硬件连接

e-Paper	Open103Z
3.3V	3V3
GND	GND
DIN	PA7
CLK	PA5
CS	PA4
DC	PA2
RST	PA1
BUSY	PA3

运行程序

1. 运行本程序需要安装 keil v5
2. 打开 STM32 工程
3. 选择好下载器类型
4. 编译并将程序下载到开发板

预期结果

4) 整个屏幕全部刷新成白色

5) 显示一张全局图片

6) 画圈画线，显示两种大小的字符

刷新显示的时候是先刷新黑色，后刷新红色的

程序说明

版本说明

本次例程为 V2.0 版本，其中主要添加了如下功能：

1） 增加中文显示字符功能

2） 增加读取图片功能

3） 树莓派增加 python3 例程

4） 优化了画图函数

5） 修正 e-paper 不能正常进入睡眠模式的问题

C 代码分析

这里基于树莓派 BCM2835 例程做分析

1. 初始底层管脚与 SPI

   DEV_ModuleInit（）；

2. 初始化墨水屏，并清屏

```
if(EPD_Init() != 0) {
    printf("e-Paper init failed\r\n");
}
EPD_Clear();
DEV_Delay_ms(500);
```

3. 新建一个图片缓存，并设置缓存的参数

```
  DEV_Delay_ms(500);

  //Create a new image cache named IMAGE_BW and fill it with white
  UBYTE *BlackImage, *RedImage;
  UWORD Imagesize = ((EPD_WIDTH % 8 == 0)? (EPD_WIDTH / 8 ): (EPD_WIDTH / 8 + 1)) * EPD_HEIGHT;
  if((BlackImage = (UBYTE *)malloc(Imagesize)) == NULL) {
      printf("Failed to apply for black memory...\r\n");
      exit(0);
  }
  if((RedImage = (UBYTE *)malloc(Imagesize)) == NULL) {
      printf("Failed to apply for red memory...\r\n");
      exit(0);
  }
  printf("NewImage:BlackImage and RedImage\r\n");
  Paint_NewImage(BlackImage, EPD_WIDTH, EPD_HEIGHT, 270, WHITE);
  Paint_NewImage(RedImage, EPD_WIDTH, EPD_HEIGHT, 270, WHITE);
```

首先申请一个图片缓存 BlackImage/RedImage，设置缓存大小为：Imagesize=图片宽度/8*图片高度。

Paint_newImage:新建一整画图，第一个参数为我们前面设置的图片缓存， 第二个和第三个参数分别是设置图片的宽度和高度，第四个参数设置图片的填充颜色

Paint_SelectImage 选择缓存图片

4. 读取图片并显示

```
#if 1   // show bmp
    printf("show windows-----------------------\r\n");
    printf("read black bmp\r\n");
    Paint_SelectImage(BlackImage);
    Paint_Clear(WHITE);
    GUI_ReadBmp("./pic/100x100.bmp", 50, 10);

    printf("read red bmp\r\n");
    Paint_SelectImage(RedImage);
    Paint_Clear(WHITE);
    
    EPD_Display(BlackImage, RedImage);
    DEV_Delay_ms(2000);

    printf("show bmp-----------------------\r\n");
    printf("read black bmp\r\n");
    Paint_SelectImage(BlackImage);
    GUI_ReadBmp("./pic/2in9b-b.bmp", 0, 0);
    printf("read red bmp\r\n");
    Paint_SelectImage(RedImage);
    GUI_ReadBmp("./pic/2in9b-r.bmp", 0, 0);

    EPD_Display(BlackImage, RedImage);
    DEV_Delay_ms(2000);
#endif
```

Paint_SelectImage 选择缓存图片；

Paint_Clear(WHITE)图片填充为白色

GUI_ReadBmp 读取 BMP 图片在指定位置，第一个参数为 BMP 图片的相对地址，第二三个为设置图片的起点 X 与 Y 坐标，如果读取图片大于缓存图片，越界的图片信息将不会显示；

EPD_Display 把图片缓存发送到墨水屏并显示；

5. 读取转换成数组的图片数据

```
#if 1   // show image for array
    printf("show image for array\r\n");
    EPD_Display(gImage_2in9b_b, gImage_2in9b_r);
    DEV_Delay_ms(2000);
#endif
```

Paint_DrawBitMap 为画位图函数，这个函数是直接把整屏的转换成数组的图片发送墨水屏，因为 STM32 与 Arduino 不能直接读取图片，要通过转换成数组的方式来显示

6. 画图，可实现画点、线、框、圆、英文字符、中文字符

```
#if 1   // Drawing on the image
    /*Horizontal screen*/
    //1.Draw black image
    Paint_SelectImage(BlackImage);
    Paint_Clear(WHITE);
    Paint_DrawPoint(10, 80, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
    Paint_DrawPoint(10, 90, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
    Paint_DrawPoint(10, 100, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
    Paint_DrawPoint(10, 110, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
    Paint_DrawLine(20, 70, 70, 120, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
    Paint_DrawLine(70, 70, 20, 120, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
    Paint_DrawRectangle(20, 70, 70, 120, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
    Paint_DrawRectangle(80, 70, 130, 120, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
    Paint_DrawString_EN(10, 0, "waveshare", &Font16, BLACK, WHITE);
    Paint_DrawString_CN(130, 20,"微雪电子", &Font24CN, WHITE, BLACK);
    Paint_DrawNum(10, 50, 987654321, &Font16, WHITE, BLACK);

    //2.Draw red image
    Paint_SelectImage(RedImage);
    Paint_Clear(WHITE);
    Paint_DrawCircle(160, 95, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
    Paint_DrawCircle(210, 95, 20, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
    Paint_DrawLine(85, 95, 125, 95, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
    Paint_DrawLine(105, 75, 105, 115, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
    Paint_DrawString_CN(130, 0,"你好abc树莓派", &Font12CN, BLACK, WHITE);
    Paint_DrawString_EN(10, 20, "hello world", &Font12, WHITE, BLACK);
    Paint_DrawNum(10, 33, 123456789, &Font12, BLACK, WHITE);

    printf("EPD_Display\r\n");
    EPD_Display(BlackImage, RedImage);
    DEV_Delay_ms(2000);
#endif
```

在缓存中，将缓存数据发送到墨水屏上显示；

其中：

Paint_DrawPoint 为画点，第一二个参数表示点的坐标，第三个参数为点的颜色，第四个为点的大小，第五个参数为点的扩充风格；

Paint_DrawLine 为画线，第一二参数为起点坐标，第三四个参数为终点坐标，第五个参数

为颜色，第六个参数为选择实线与虚线，第七个参数为线宽；

Paint_DrawRectangle 为画框，第一二个参数为起点坐标，第三四个参数为终点坐标，第五个参数为填充颜色，第六个参数为是否填充框的内部，第七个参数为线框；

Paint_DrawCircle 为画圆，第一二个参数为圆心坐标，第三个参数为圆的半径，第四个参数为填充颜色，第五个参数为是否填充框的内部，第六个参数为线框；

Paint_DrawString_EN 为显示英文字符，第一二个参数为起点坐标，第三个参数为需要显示的字符，第四个参数为显示的字体大小，第五六个参数为字体背景颜色与字体颜色

Paint_DrawString_CN 为显示中文字符，第一二个参数为起点坐标，第三个参数为需要显示的字符，第四个参数为显示的字体大小，第五六个参数为字体背景颜色与字体颜色。

这里需要注意的是要保存为 GB2312 编码，不然程序无法识别。

Paint_DrawNum 为显示数字，第一二个参数为起点坐标，第三个参数为数字，它是一个 int 型的变量，需要注意传入的参数不要超过变量长度，第四个参数为显示的字体大小，第五六个参数为字体背景颜色与字体颜色

7. 睡眠模式

```
    printf("Goto Sleep mode...\r\n");
    EPD_Sleep();
    free(BlackImage);
    BlackImage = NULL;
    free(RedImage);
    RedImage = NULL;
```

需要模块进入低功耗，则在程序最后加上睡眠模式

PYTHON 分析

1. 实例化墨水屏驱动并初始化

```
try:
    epd = epd2in9b.EPD()
    epd.init()
    print "clear"
    epd.Clear(0xFF)
    
```

2. 使用 python image 库，新建图片缓存

```
# Drawing on the Horizontal image
HBlackimage = Image.new('1', (epd2in9b.EPD_HEIGHT, epd2in9b.EPD_WIDTH), 255)  # 298*126
HRedimage = Image.new('1', (epd2in9b.EPD_HEIGHT, epd2in9b.EPD_WIDTH), 255)  # 298*126    
```

新建缓存为黑白图片，定义长宽和填充颜色

3. 调用 python imageDraw 库画图

```
# Horizontal
print "Drawing"
drawblack = ImageDraw.Draw(HBlackimage)
drawred = ImageDraw.Draw(HRedimage)
font24 = ImageFont.truetype('/usr/share/fonts/truetype/wqy/wqy-microhei.ttc', 24)
drawblack.text((10, 0), 'hello world', font = font24, fill = 0)
drawblack.text((10, 20), '2.9inch e-Paper', font = font24, fill = 0)
drawblack.text((150, 0), u'微雪电子', font = font24, fill = 0)
drawblack.line((20, 50, 70, 100), fill = 0)
drawblack.line((70, 50, 20, 100), fill = 0)
drawblack.rectangle((20, 50, 70, 100), outline = 0)
drawred.line((165, 50, 165, 100), fill = 0)
drawred.line((140, 75, 190, 75), fill = 0)
drawred.arc((140, 50, 190, 100), 0, 360, fill = 0)
drawred.rectangle((80, 50, 130, 100), fill = 0)
drawred.chord((200, 50, 250, 100), 0, 360, fill = 0)
epd.display(epd.getbuffer(HBlackimage), epd.getbuffer(HRedimage))
time.sleep(2)

# Drawing on the Vertical image
LBlackimage = Image.new('1', (epd2in9b.EPD_WIDTH, epd2in9b.EPD_HEIGHT), 255)  # 126*298
LRedimage = Image.new('1', (epd2in9b.EPD_WIDTH, epd2in9b.EPD_HEIGHT), 255)  # 126*298
# Vertical
drawblack = ImageDraw.Draw(LBlackimage)
drawred = ImageDraw.Draw(LRedimage)
font18 = ImageFont.truetype('/usr/share/fonts/truetype/wqy/wqy-microhei.ttc', 18)
drawblack.text((2, 0), 'hello world', font = font18, fill = 0)
drawblack.text((2, 20), '2.9inch epd', font = font18, fill = 0)
drawblack.text((20, 50), u'微雪电子', font = font18, fill = 0)
drawblack.line((10, 90, 60, 140), fill = 0)
drawblack.line((60, 90, 10, 140), fill = 0)
drawblack.rectangle((10, 90, 60, 140), outline = 0)
drawred.line((95, 90, 95, 140), fill = 0)
drawred.line((70, 115, 120, 115), fill = 0)
drawred.arc((70, 90, 120, 140), 0, 360, fill = 0)
drawred.rectangle((10, 150, 60, 200), fill = 0)
drawred.chord((70, 150, 120, 200), 0, 360, fill = 0)
epd.display(epd.getbuffer(LBlackimage), epd.getbuffer(LRedimage))
time.sleep(2)
```

在缓存中画框、线、圆、圆弧、英文字符，并将图片缓存发送至墨水屏并显示

更多用法可以参考 imageDraw

4. 读取图片

```
print "read bmp file"
HBlackimage = Image.open('2in9b-b.bmp')
HRedimage = Image.open('2in9b-r.bmp')
epd.display(epd.getbuffer(HBlackimage), epd.getbuffer(HRedimage))
time.sleep(2)

print "read bmp file on window"
blackimagel = Image.new('1', (epd2in9b.EPD_HEIGHT, epd2in9b.EPD_WIDTH), 255)  # 298*126
redimagel = Image.new('1', (epd2in9b.EPD_HEIGHT, epd2in9b.EPD_WIDTH), 255)  # 298*126
newimage = Image.open('100x100.bmp')
blackimagel.paste(newimage, (50,10))
epd.display(epd.getbuffer(blackimagel), epd.getbuffer(redimagel))
```

第一部分：全屏显示图片

打开图片， 并将图片数据显示

第二部分： 在指定位置显示图片

新建一个和图片一样大小的缓存，将 100x100 的图片复制到这个缓存的制定位置，并将图片缓存发送至墨水屏显示

5. 睡眠模式

```
    epd.sleep()
```

需要模块进入低功耗模式，则在程序最后加上睡眠模式

图片数组制作

2.13inch[7] 墨水屏只支持灰度为二阶即黑白的图片，如果图片灰度过多，是不能完全显示出全部的颜色的。

1） 找一张灰度为二阶的图片，或者使用电脑的画图工具修改一下格式

2） 通过系统自带的画图工具修改图片分辨率为墨水屏的分辨率

3）使用 Image2Lcd.exe 软件生成图片所对应的数组（.c 文件）如下图：

使用 Image2Lcd.exe 打开图片，设置一下参数

输出数据类型为：C 语言数组

扫描模式：垂直扫描

输出灰度：单色

最大宽度和高度：屏幕的分辨率大小

然后保存生成.c 格式的数组。将对应的数组复制到程序中，并调用显示即可。

[7] 这里以 2.13inch 墨水屏黑白款为例进行说明。