2.13inch e-Paper HAT (D)
用户手册

产品概述

- 本品是 2.13 英寸柔性电子墨水屏模块，分辨率为 212 x 104，带有内部控制器，使用 SPI 接口通信，可显示黑白两色。
- 具有功耗低、视角宽等优点，常用于货架标签、工业仪表等显示应用。

特点

- 无需背光，断电可长时间保持最后一屏的显示内容
- 支持局部刷新，平均 0.55S 可刷一帧
- 功耗非常低，基本只在刷新时耗电
- 可使用 e-Paper HAT 与 e-paper Shield 两种驱动板驱动
- 提供完善的配套资料手册(Raspberry/Arduino/STM32 等示例程序)

产品参数

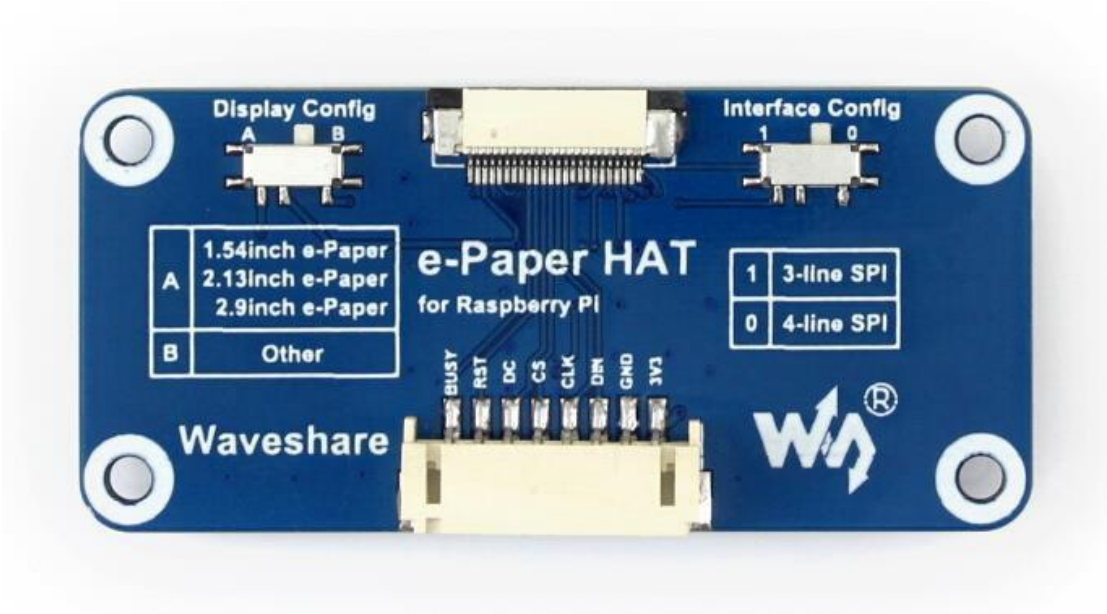
工作电压：	3.3V
通信接口：	3-wire SPI、4-wire SPI
外形尺寸：	59.2(H) × 29.20(V) × 0.3(D)mm
显示尺寸：	48.55(H) × 23.71(V)mm
点距　　：	0.229 × 0.228
分辨率　：	212(H) × 104(V)
显示颜色：	黑、白
灰度等级：	2
全局刷新：	2.3s
刷新功耗：	26.4mW(typ.)
待机功耗：	<0.017mW
可视角度：	>170°

接口说明

VCC：	3.3V
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）

开关设置说明

如下图所示，e-Paper Driver HAT 正面两个拨动开关：Display Config 和 Interface Config，下文将对这两个开关设置做说明。



Display Config 设置

e-Paper Driver HAT 上的 Display Config 开关是为了支持驱动微雪多款 SPI 电子墨水屏裸屏而设置的。开关有 A/B 两端，当接入不同型号的电子墨水屏，Display Config 开关须对应设置如下表：

B (接到以下型号，开关须拨动到 **B** 端**)**	**A (**接到以下型号，开关须拨动到 **A** 端**)**
1.54inch e-Paper (B)	1.54inch e-Paper
1.54inch e-Paper (C)	2.13inch e-Paper
2.13inch e-Paper (B)	2.13inch e-Paper (D)

2.13inch e-Paper (C)	2.9inch e-Paper
2.7inch e-Paper	
2.7inch e-Paper (B)	
2.9 inch e-Paper (B)	
2.9 inch e-Paper (C)	
4.2inch e-Paper	
4.2inch e-Paper (B)	
4.2inch e-Paper (C)	
7.5inch e-Paper	
7.5inch e-Paper (B)	
7.5inch e-Paper (C)	

Interface Config 设置

Interface Config 设置开关可以切换电子墨水屏使用 3-line SPI 或 4-line SPI。

当开关拨动到 1 处，模块工作于 3-line SPI；

当开关拨动到 0 处，模块工作于 4-line SPI；

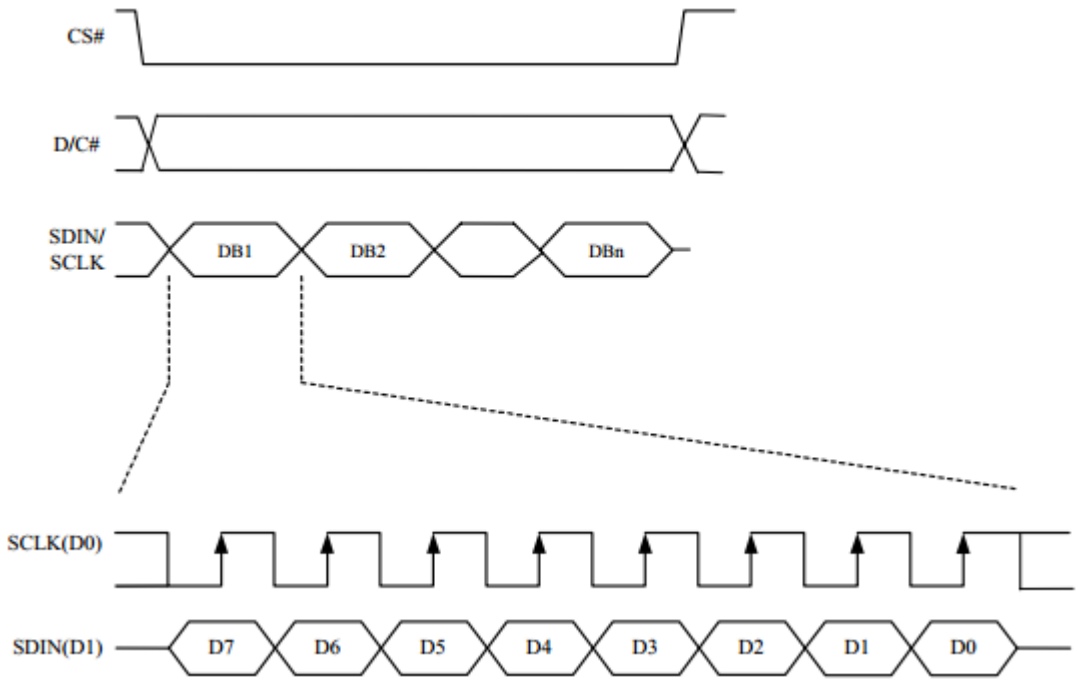
示例程序默认使用 4-line SPI。

硬件说明

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



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进行了隐藏。

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 时写命令，当 DC = 1 时写数据。
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，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，国内通常使用 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。

显示控制

对于此款屏幕，查看数据手册可知显示需要控制两个寄存器，R10H 与 R13H

(9) Data Start Transmission 1 (DTM1) (R10H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Starting data transmission	0	0	0	0	0	1	0	0	0	0
	0	1	Pixel1	Pixel2	Pixel3	Pixel4	Pixel5	Pixel6	Pixel7	Pixel8
	0	1	..	..	..	..	..	..	..	..
	0	1	Pixel(n-7)	Pixel(n-6)	Pixel(n-5)	Pixel(n-4)	Pixel(n-3)	Pixel(n-2)	Pixel(n-1)	Pixel(n)

This command starts transmitting data and write them into SRAM.
In KW mode, this command writes “OLD” data to SRAM.
In KWR mode, this command writes “B/W” data to SRAM.
In Program mode, this command writes “OTP” data to SRAM for programming.

(12) Data Start Transmission 2(DTM2) (R13H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Starting data transmission	0	0	0	0	0	1	0	0	1	1
	0	1	Pixel1	Pixel2	Pixel3	Pixel4	Pixel5	Pixel6	Pixel7	Pixel8
	0	1	..	..	..	..	..	..	..	..
	0	1	Pixel(n-7)	Pixel(n-6)	Pixel(n-5)	Pixel(n-4)	Pixel(n-3)	Pixel(n-2)	Pixel(n-1)	Pixel(n)

This command starts transmitting data and write them into SRAM.
In KW mode, this command writes “NEW” data to SRAM.
In KWR mode, this command writes “RED” data to SRAM.

而此屏幕只能显示黑白两色，即处于 KW 模式，那么寄存器 R10H,需要写入旧图片的数据，寄存器 R13H,需要写入新的图片的数据，由于是全局刷新，那么旧的数据传白屏即可。

由于 2.13 寸柔性电子纸可支持局部刷新。查看数据手册有 R91H,R90H 两个寄存器设置屏幕为局部刷新模式，并且设置显示区域，然后再控制 R10G,R13H。

(36) Partial Window (PTL) (R90H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Set Partial Window	0	0	1	0	0	1	0	0	0	0
	0	1	HRST[7:3]					0	0	0
	0	1	HRED[7:3]					1	1	1
	0	1	-	-	-	-	-	-	-	VRST[8]
	0	1	VRST[7:0]							
	0	1	-	-	-	-	-	-	-	VRED[8]
	0	1	VRED[7:0]							
	0	1	-	-	-	-	-	-	-	PT_SCAN

This command sets partial window.

HRST[7:3]: Horizontal start channel bank. (value 00h~13h)

HRED[7:3]: Horizontal end channel bank. (value 00h~13h). HRED must be greater than HRST.

VRST[8:0]: Vertical start line. (value 000h~127h)

VRED[8:0]: Vertical end line. (value 000h~127h). VRED must be greater than VRST.

PT_SCAN: 0: Gates scan only inside of the partial window.

1: Gates scan both inside and outside of the partial window. (default)

(37) Partial In (PTIN) (R91H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Partial In	0	0	1	0	0	1	0	0	0	1

This command makes the display enter partial mode.

使用方法

使用 e-Paper HAT 或者 e-Paper Shield 均可驱动:

若用于树莓派使用 e-Paper HAT 方便驱动；

用于 Arduino 或者 NUCLEO 使用 e-Paper Shield 方便驱动；

e-Paper HAT

用于树莓派

安装必要的函数库

墨水屏工作于树莓派上面的时候，需要安装必要的函数库（wiringPi、bcm2835、python 库），否则以下的示例程序可能无法正常工作。安装方法详见：

http://www.waveshare.net/wiki/Pioneer600_Datasheets

硬件连接

以下为树莓派 BCM 管脚编码硬件连接（树莓派三代 B/B+）：

e-Paper	树莓派 3 代 B/B+
3.3V	3.3V
GND	GND
DIN	MOSI
CLK	SCLK
CS	CE0
DC	25 (BCM)
RST	17 (BCM)
BUSY	24 (BCM)

预期结果

1) 安装好相应的库后，把对应程序复制至树莓派中，进入对应目录中：
   - **bcm2835：**执行命令：make，将会编译代码，生成一个名为 epd 的执行文件。执行命令：sudo ./epd，程序将会运行。
   - **wiringpi：**执行命令：make，将会编译代码，生成一个名为 epd 的执行文件。执行命令：sudo ./epd，程序将会运行。

- **python**：执行命令：sudo python main.py

屏幕会显示图像。

注意：本模块刷新速度慢，并且刷新过程中会有多次闪烁，请耐心等待。

用于 ARDUINO

硬件连接

硬件连接到开发板 UNO PLUS：

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

预期结果

1) 把示例程序包中 arduino/libraries 目录下的文件复制到 documents\arduino\libraries，该位置可以通过 Arduino IDE → File → Preferences → Sketchbook location 指定。
2) 点击 Upload 上传工程。
3) 屏幕会显示图像。

注意：

- 本模块刷新速度慢，并且刷新过程中会有多次闪烁，请耐心等待。
- Arduino UNO 的 RAM 只有 2K，而本模块完整更新 1 帧图像需要占用 2756 字节的内存，而程序本身需要占用部分内存，因此使用 e-Paper HAT 驱动仅提供静态图像显示程序，若需要其他显示功能可使用 e-Paper Shield 驱动。

用于 STM32 开发板

- 本例程使用的开发板主控为 STM32F103RB。
- 本例程基于 HAL 库，因此可以使用 STM32CubeMX 把示例程序移植到其他 STM 芯片上。
- 本例程在 Keil v5 环境下编译通过。

硬件连接

硬件连接到 STM32F103RB：

e-Paper	STM32F103RB
3.3V	3.3V
GND	GND
DIN	PA7(MOSI)
CLK	PA5(SCK)
CS	PB6
BUSY	PA8
DC	PC7
RST	PA9

预期结果

1) 打开位于 MDK-ARM 目录下的 Keil 工程（epd-demo.uvprojx）。
2) 点击 Build 编译工程。
3) 点击 Download，把工程写入到芯片中。
4) 开发板复位之后，屏幕会显示图像。

- 本模块刷新速度慢，并且刷新过程中会有多次闪烁，请耐心等待。

e-Paper Shield

硬件配置

- Arduino 主板带有 ICSP 接口时，显示模块上的 SPI Config 开关置于 ICSP 方向(默认)
- Arduino 主板没有 ICSP 接口时，显示模块上的 SPI Config 开关分别置于 SCLK\D13，MISO\D12,MOSI\D11

演示及代码分析

本模块提供分别基于 Arduino UNO 和 XNUCLEO-F103RB 的实验例程。

ARDUINO UNO

打开 Arduino 目录，可见如下目录：

把 library 中的 Waveshare_ePaper 目录复制到 arduino 根目录下的 libraries,路径一般为：C:\Program Files (x86)\Arduino\libraries

显示 SD 卡的图片

1. 初始化墨水屏，并将清除墨水屏缓存

```
GUIPAINT paint;
EPD2IN13D epd;
EPD_SDCARD SD;

Dev->System_Init();
Serial.print("2.13inch e-Paper D demo\n");
//Initialize e-Paper
if(epd.EPD_Init() != 0) {
    Serial.print("e-Paper init failed");
    return;
}
epd.EPD_Clear();
```

2. 初始化 SD 卡，需要插上 SD 卡，否则会产生异常

```
//1.Initialize the SD card
SD.SDCard_Init();
```

3. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空

```
//2.Create a new image cache named IMAGE_BW and fill it with white
paint.Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
paint.Paint_Clear(WHITE);
```

4. 读取图片，并将图片数据保存在外部 RAM 中

```
//3.Read BMP images into RAM
SD.SDCard_ReadBMP("f2in13.bmp", 0, 0);
```

5. 将外部 RAM 中数据刷新到墨水屏的缓存中，并打开显示

```
//4.Refresh the picture in RAM to e-Paper
epd.EPD_Display();
Dev->Dev_Delay_ms(2000);
```

显示转换成数组的图片

1. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空
2. 把数组中的数据保存在外部 RAM 中

```
//2.show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
paint.Paint_DrawBitMap(IMAGE_DATA);
```

3. 将外部 RAM 中数据刷新到墨水屏的缓存中，并打开显示

画图

1. 初始化墨水屏，并将清除墨水屏缓存
2. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空
3. 画点，设置点的位置、颜色、大小、填充的方式

```
paint.Paint_DrawPoint(10, 70, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
paint.Paint_DrawPoint(10, 80, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
paint.Paint_DrawPoint(10, 90, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
```

4. 画线，设置线的起始坐标、颜色、实线还是虚线、线的宽度

```
paint.Paint_DrawLine(20, 70, 70, 100, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(70, 70, 20, 100, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(180, 70, 180, 100, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
paint.Paint_DrawLine(165, 85, 195, 85, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
```

5. 画框，设置框的对角线起始坐标、颜色、内部是否填充、线的宽度

```
paint.Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

6. 画圆，设置圆的圆心坐标、半径、颜色、内部是否填充、线的宽度

```
paint.Paint_DrawCircle(180, 85, 15, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawCircle(180, 45, 15, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

7. 显示英文字符，设置字符的起点坐标、显示的字符、字体大小、背景色、字体色

```
paint.Paint_DrawString_EN(10, 2, "Waveshare Electronic", &Font12, BLACK, WHITE);
paint.Paint_DrawString_EN(10, 20, "hello world", &Font12, WHITE, BLACK);
```

8. 显示数字，设置显示数字的起点坐标、数字、字体大小、背景色、字体色

```
paint.Paint_DrawNum(10, 50, 987654321, &Font16, WHITE, BLACK);
```

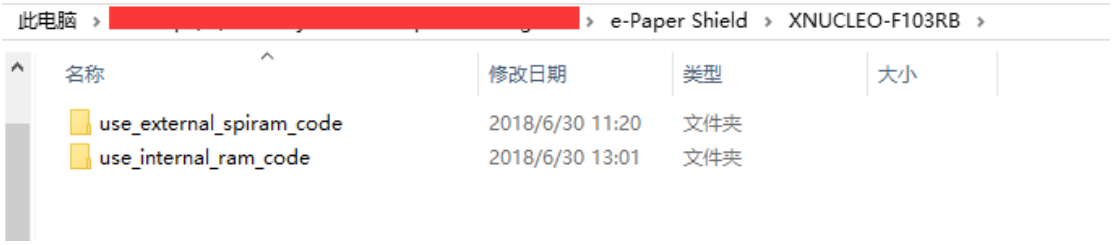
9. 显示时间，设置起点坐标、包含时间的结构体、字体大小、背景色、字体色。（局部刷新）

```
Serial.print("Show time\r\n");
PAINT_TIME nowtime;
nowtime.Hour = 23;
nowtime.Min = 59;
nowtime.Sec = 50;

sFONT font = Font24;
UWORD Xstart = 40;
UWORD Ystart = 40;
for (;;) {
    paint.Paint_DrawTime(Xstart, Ystart, &nowtime, &font, WHITE, BLACK);
    nowtime.Sec = nowtime.Sec + 1;
    if (nowtime.Sec == 60) {
        nowtime.Min = nowtime.Min + 1;
        nowtime.Sec = 0;
        if (nowtime.Min == 60) {
            nowtime.Hour =  nowtime.Hour + 1;
            nowtime.Min = 0;
            if (nowtime.Hour == 24) {
                nowtime.Hour = 0;
                nowtime.Min = 0;
                nowtime.Sec = 0;
            }
        }
    }

    //Brushing the entire screen is not as obvious as painting a window.
    epd.EPD_DisplayPartial(0, 0, Paint_Image.Image_Width, Paint_Image.Image_Height);
}
```

NUCLEO-F103RB

- 本例程使用的开发板主控为 XNUCLEO-F103RB。
- 本例程基于 HAL 库，因此可以使用 STM32CubeMX 把示例程序移植到其他 STM 芯片上。
- 本例程在 Keil v5 环境下编译通过。

对于 STM32 控制而言，有两种例程：一种使用内部 RAM（部分大屏无法显示），一种使用外部 RAM



显示 SD 卡的图片

1. 初始化墨水屏，并将清除墨水屏缓存
2. 初始化 SD 卡，需要插上 SD 卡，否则会产生异常
3. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空
4. 读取图片，并将图片数据保存在外部 RAM 中
5. 将外部 RAM 中数据刷新到墨水屏的缓存中，并打开显示

```
    if(EPD_Init() != 0) {
        printf("e-Paper init failed\r\n");
    } else {
        printf("e-Paper init Successful\r\n");
    }
    EPD_Clear();
    printf("e-Paper clear over...\r\n");
    DEV_Delay_ms(500);
#if 1
    /*show sd card pic*/
    printf("/**************/\r\nshow image for SD card\r\n");
    //1.Initialize the SD card
    SDCard_Init();

    //2.Create a new image cache named IMAGE_BW and fill it with white
    printf("Initialize the cached related properties...\r\n");
    GUI_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
    GUI_Clear(WHITE);

    //3.Read BMP pictures into RAM
    printf("Read the image into the cache...\r\n");
    SDCard_ReadBMP("f2in13.bmp", 0, 0);

    //4.Refresh the picture in RAM to e-Paper
    printf("Refresh the picture in RAM to e-Paper...\r\n/**************/\r\n");
    EPD_DisplayFull();
#endif
```

显示转换成数组的图片

1. 初始化墨水屏，并将清除墨水屏缓存
2. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空
3. 把数组中的数据据保存在外部 RAM 中
4. 将外部 RAM 中数据刷新到墨水屏的缓存中，并打开显示

```
/*show image for array*/
printf("show image for array\r\n");
//1.Create a new image cache named IMAGE_BW and fill it with white
Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
Paint_Clear(WHITE);

printf("Paint_DrawBitMap\r\n");
//2.show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
Paint_DrawBitMap(IMAGE_DATA);

printf("EPD_Display\r\n");
//3.Refresh the picture in RAM to e-Paper
EPD_Display();
Dev_Delay_ms(2000);
```

画图

1. 初始化墨水屏，并将清除墨水屏缓存
2. 新建一张图片缓存， 设置为黑白图片，定义图像的宽度和长度，翻转度数，是否反色，并将缓存清空
3. 画点，设置点的位置、颜色、大小、填充的方式
4. 画线，设置线的起始坐标、颜色、实线还是虚线、线的宽度
5. 画框，设置框的对角线起始坐标、颜色、内部是否填充、线的宽度
6. 画圆，设置圆的圆心坐标、半径、颜色、内部是否填充、线的宽度
7. 显示英文字符，设置字符的起点坐标、显示的字符、字体大小、背景色、字体色
8. 显示数字，设置显示数字的起点坐标、数字、字体大小、背景色、字体色
9. 将外部 RAM 中数据刷新到墨水屏的缓存中，并打开显示

```
//1.Create a new image cache named IMAGE_BW and fill it with white
Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_90, IMAGE_COLOR_POSITIVE);
Paint_Clear(WHITE);

//2.Drawing on the image
Paint_DrawPoint(5, 10, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
Paint_DrawPoint(5, 25, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
Paint_DrawPoint(5, 40, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
Paint_DrawPoint(5, 55, BLACK, DOT_PIXEL_4X4, DOT_STYLE_DFT);

Paint_DrawLine(20, 10, 70, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(70, 10, 20, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(170, 15, 170, 55, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
Paint_DrawLine(150, 35, 190, 35, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);

Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawCircle(170, 35, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawCircle(170, 85, 20, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawString_EN(5, 70, "hello world", &Font16, WHITE, BLACK);
Paint_DrawString_EN(5, 90, "waveshare", &Font20, BLACK, WHITE);

Paint_DrawNum(5, 120, 123456789, &Font20, BLACK, WHITE);

//3.Refresh the picture in RAM to e-Paper
EPD_Display();
Dev_Delay_ms(2000);
```

局部刷新

1. 初始化墨水屏为局部刷新
2. 填充定义的时间结构体的时分秒
3. 清除某个窗口的数据
4. 显示时间，设置起点坐标、包含时间的结构体、字体大小、背景色、字体色。
5. 把外部 RAM 中的数据发送到

```
#if 1
  printf("/**************/\r\nPartial display routine\r\n");
    GUI_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_90, IMAGE_COLOR_POSITIVE);
    GUI_Clear(WHITE);
    EPD_DisplayPartial(0, 0, GUI_Image.Image_Width, GUI_Image.Image_Height);
    DEV_Delay_ms(500);

    printf("Show time\r\n");
    GUI_TIME nowtime;
    nowtime.Hour = 23;
    nowtime.Min = 59;
    nowtime.Sec = 50;

    sFONT font = Font24;
    UWORD Xstart = 40;
    UWORD Ystart = 40;
    for (;;) {
        GUI_DrawTime(Xstart, Ystart, &nowtime, &font, WHITE, BLACK);
        nowtime.Sec = nowtime.Sec + 1;
        if (nowtime.Sec == 60) {
            nowtime.Min = nowtime.Min + 1;
            nowtime.Sec = 0;
            if (nowtime.Min == 60) {
                nowtime.Hour =  nowtime.Hour + 1;
                nowtime.Min = 0;
                if (nowtime.Hour == 24) {
                    nowtime.Hour = 0;
                    nowtime.Min = 0;
                    nowtime.Sec = 0;
                }
            }
        }

        //Brushing the entire screen is not as obvious as painting a window.
        EPD_DisplayPartial(0, 0, GUI_Image.Image_Width, GUI_Image.Image_Height);
        DEV_Delay_ms(450);//Analog clock 1s
    }
#endif
```

显示图片

有直接和间接两种方法可以让模块显示图片，

直接显示图片：通过函数直接读取图片的数据并解码缓存，然后把缓存的数据发送到电子纸缓存中。关于该方法的实现，树莓派和使用 e-Paper Shield 的代码均实现此功能。

间接显示图片：用电脑把图片转换成对应的数组，然后把该数组以.c 文件的形式直接嵌入到程序中。本节将会讲解如何把一张图片转换成对应的数组。

1) 打开 Windows 系统自带的画图工具，新建图片，像素设置成 104 x 212。
2) 由于模块只能显示两阶的灰度（仅有黑白两色），因此在把图片转换成数组之前，必须转化成单色位图（File → Save as → BMP picture → Monochrome Bitmap）。
   示例程序包中包含一张用于演示的单色位图图片（raspberrypi/python/monocolor.bmp）。
3) 使用 Image2Lcd.exe 软件生成图片所对应的数组（.c 文件）。使用该软件打开图片，设置对应参数：
   - 输出数据类型为：C 语言数组
   - 扫描模式：水平扫描
   - 输出灰度：单色（即两阶）
   - 最大宽度和高度：104 和 212
   - 不勾选“包含图像头数据”
   - 勾选“颜色翻转”（勾选：图片中的白色会转换成 1，黑色会转换成 0）
4) 点击“保存”，就会生成对应的.c 文件。将对应的数组复制到工程中，程序调用这个数组进行显示即可。