2.13inch e-Paper HAT 用户手册

产品概述

- 本品是 2.13 英寸电子墨水屏模块，分辨率为 250x122（逻辑分辨率为 250x128），带有内部控制器，使用 SPI 接口通信，支持局部刷新。
- 具有耗低、视角宽、阳光直射下仍可清晰显示等优点，常用于货架标签、工业仪表等显示应用。
- 本模块基于 SPI 通信协议来对墨水屏进行操作，通过相应函数，就可以完成显示功能，包括几何图形绘制、文字、图片显示，同时本模块支持局部刷新，本产品提供树莓派（基于 BCM2835，Wringpi 和 python），SMT32 以及 Arduino 例程，方便客户进行移植。

规格

- 工作电压：3.3V
- 通信接口： SPI
- 外形尺寸： 65mm × 30.2mm
- 显示尺寸： 23.71mm × 48.55mm
- 点距： 0.194 × 0.194
- 分辨率： 250 × 122
- 显示颜色： 黑、白
- 灰度等级： 2
- 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）

版本：V1.2.2，日期：2017 年 9 月 13 日

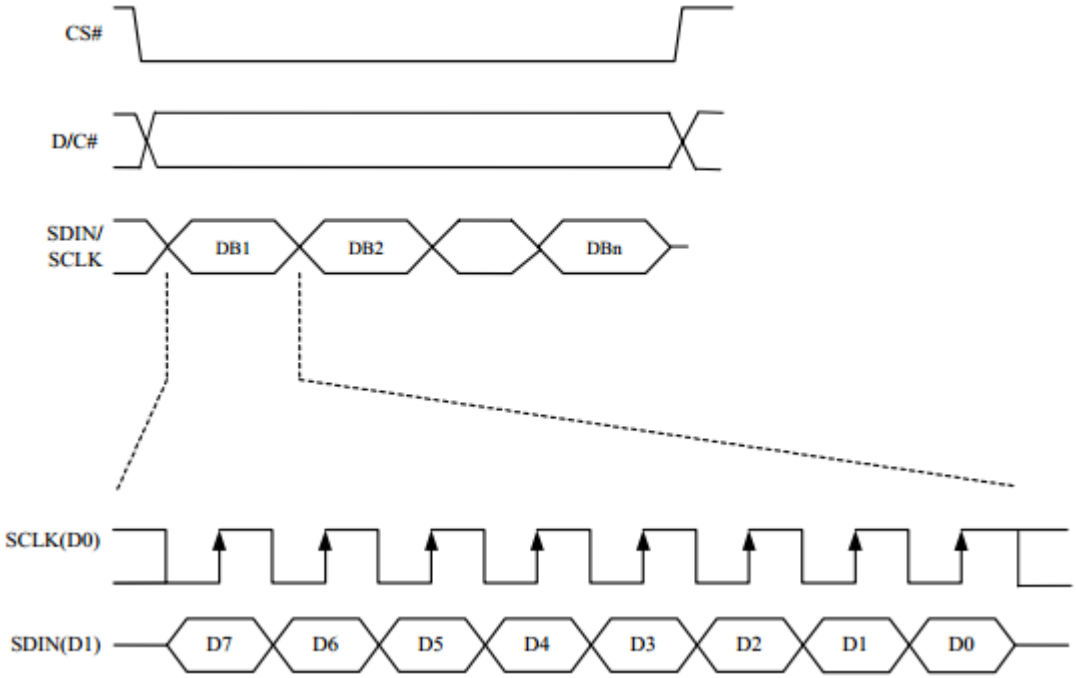
DC	数据/命令控制引脚（高电平表示数据，低电平表示命令）
RST	外部复位引脚（低电平复位）
BUSY	忙状态输出引脚（高电平表示忙）

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



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。

使用方法

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，否则以下的示例程序可能无法正常工作。关于树莓派库的安装详细见：

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

此处详细介绍了 wiringPi、bcm2835、python 函数库的安装。

硬件连接

以下为树莓派 BCM 管脚编码硬件连线（树莓派三代 B）：

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

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
2) 屏幕全部刷新，显示文字和图形。
3) 屏幕局部刷新，显示图片和时间。持续变化的时间字符串展示了局部刷新的功能。

用于 Arduino

硬件连接

硬件连接到开发板 UNO PLUS：

e-Paper	Arduino
3.3V	3V3
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) 屏幕全部刷新，显示文字和图形。
4) 屏幕局部刷新，显示图片和时间。持续变化的时间字符串展示了局部刷新的功能。

用于 STM32 开发板

- 本例程使用的开发板主控为 STM32F103ZE。
- 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

硬件连接到开发板 NUCLEO-F103RB：

e-Paper	NUCLEO-F103RB
3.3V	3V3
GND	GND
NC	--
DIN	PA7
CLK	PA5
CS	PB6
DC	PC7
RST	PA9
BUSY	PA8

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) 屏幕全部刷新，显示文字和图形。
5) 屏幕局部刷新，显示图片和时间。持续变化的时间字符串展示了局部刷新的功能。

代码分析

下面以基于树莓派 WiringPi 的函数库的示例程序为例分析驱动代码。

硬件接口函数

驱动代码中的这些函数：DigitalWrite，DigitalRead，SendCommand，SendData，DelayMs 调用硬件设备所提供的接口函数（epdif.h、epdif.c 或 epdif.cpp），分别实现这些功能：IO 电平控制，IO 电平读取，发送 SPI 命令，发送 SPI 数据，毫秒延迟。用户如需移植示例程序，那么需要根据目标硬件，实现 epdif（e-paper display interface）文件中的所有接口。

值得注意的是，在实现 SPI 数据传输的时候，树莓派使用硬件片选，因此传输数据之前不需要在代码中设置 CS 管脚为低电平，而程序在传输 SPI 数据的时候就会自动置 CS 管脚为低电平。然而，对于 Arduino 和 STM32 等设备，用户需要通过在代码中显式设置 CS 管脚为低电平以启用模块的 SPI 传输。

发送命令（SendCommand）和发送数据（SendData）

SendCommand 和 SendData 分别用于向模块发送命令和发送数据，两者区别在于，发送命令的时候，设置 D/C 管脚为低电平，此时通过 SPI 接口发送到模块的数据会当成命令执行。而发送数据的时候，设置 D/C 管脚为高电平，此时通过 SPI 接口发送到模块的数据只是普通的数据，通常是命令后面跟着的参数或者图像数据。

复位（Reset）

RST 为低电平时，模块会复位。用于模块上电或者进入睡眠模式之后，重启模块。重启模块之后，可能还需要通过初始化函数（Init）对模块进行初始化，模块才能正常工作。

初始化（Init）

Init 有三个作用：1，用于模块上电之后，对模块参数进行配置。2，模块进入睡眠模式之后，唤醒模块。3，把模块设置成全部刷新模式或者局部刷新模式。

模块初始化流程：reset → driver output control → booster soft start control → write VCOM register → set dummy line period → set gate time → data entry mode setting → look-up table setting。

LUT 表设置（SetLut）

LUT 表用于设置模块为全部刷新的模式或者局部刷新的模式。该表由我们提供，可能因不同批次有所不同。如有改动，我们将会尽快更新示例程序。

设置帧内存（SetFrameMemory）

SetFrameMemory 用于给模块的内存写入图像数据，流程如下：

设置图像区域大小（见 SetMemoryArea 函数） → 设置图像起始坐标（见 SetMemoryPointer 函数） → 发送命令 write RAM → 发送图像数据。

- 本模块自带两个内存区域，每次执行完 DisplayFrame 函数之后，再次使用 SetFrameMemory 函数，则会设置另外一个内存区域。如果要设置全部（两个）内存区域的话，那么流程是：SetFrameMemory → DisplayFrame → SetFrameMemory → DisplayFrame。即设置并刷新两次。
- 通过 SPI 接口传入的数据会首先保存在内存中，然后接收到刷新命令才会更新图像。
- 发送的图像：1 byte = 8 pixels，不支持灰度显示（只能显示黑色和白色）。Bit 置位的时候代表该像素为白色，反之则为黑色。

  例如：

  0xC3：8 个像素□□■■■■□□；

  0x00：8 个像素■■■■■■■■；

  0xFF：8 个像素□□□□□□□□；

  0x66：8 个像素■□□■■□□■。

显示一帧图像（DisplayFrame）

DisplayFrame 用于显示帧内存中保存的图像数据。

注意：

- 本模块自带两个内存区域，每次执行完 DisplayFrame 函数之后，再次使用 SetFrameMemory 函数，则会设置另外一个内存区域。如果要设置全部（两个）内存区域的话，那么流程是：SetFrameMemory → DisplayFrame → SetFrameMemory → DisplayFrame。即设置并刷新两次。
- 通过 SPI 接口传入的数据会首先保存在内存中，然后接收到刷新命令才会更新图像。
- 全部刷新的过程中会有闪烁。
- 局部刷新的过程中则不会有闪烁，但是可能出现残影。

睡眠模式（Sleep）

Sleep 可以让模块进入睡眠模式以降低功耗。

进入睡眠模式之后，如需唤醒模块，需要给 RST 复位管脚一个低电平的脉冲。然后可能还需要对重新配置电源参数（不同批次可能有所不同，有些需要重新配置，有些不需要）。因此如需唤醒模块，建议使用 Init 初始化函数，而非 Reset 函数。Init 函数执行过程中会执行 Reset 函数以及相关的初始化命令。

私有函数：设置内存区域（SetMemoryArea）

SetMemoryArea 用于指定帧内存区域，参数是起始、结束的坐标点。由于发送的图像数据 1 byte = 8 pixels，因此参数 x 起始、结束坐标必须是 8 的倍数，否则最后 3 位会被舍去。

私有函数：设置内存指针（SetMemoryPointer）

SetMemoryPointer 用于指定帧内存的起始坐标。由于发送的图像数据 1 byte = 8 pixels，因此参数 x 起始坐标必须是 8 的倍数，否则最后 3 位会被舍去。

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：通过库函数直接读取图片的数据进行解码，并转换成对应的数组发送给模块。关于该方法的实现，请参见树莓派 Python 库示例程序（示例程序包中没有直接显示图片的 C 程序）。

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 128x250。
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：128 和 250
   - 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。