waveshare 4.2inch e-Paper Module (B)

Waveshare 4.2-inch E-Ink Display Module (B) User Manual

Model: 4.2inch e-Paper Module (B)

1. Product Overview

The Waveshare 4.2-inch E-Ink Display Module (B) is a low-power, high-resolution display solution designed for various embedded applications. It features a 400x300 pixel resolution and supports a three-color display (red, black, and white). The module includes an embedded controller and communicates via an SPI interface, making it compatible with popular development boards such as Raspberry Pi, Arduino, and Nucleo.
Key characteristics of this E-Ink display include:

- **Ultra-low power consumption:** Power is primarily required only during display refresh cycles.
- **Wide viewing angle:** Provides clear visibility from various perspectives.
- **Bistable display:** Retains the displayed content indefinitely without continuous power supply, eliminating the need for a backlight.
- **Onboard voltage translator:** Ensures compatibility with both 3.3V and 5V microcontrollers.

Figure 1: The Waveshare 4.2-inch E-Ink Display Module (B) displaying its specifications and capabilities in red, black, and white.

2. Package Contents

Verify that all items listed below are included in your package:

- 1x 4.2inch e-Paper Module (B)
- 1x PH2.0 20cm 8Pin Cable
- Development Resources (accessible via online Wiki)

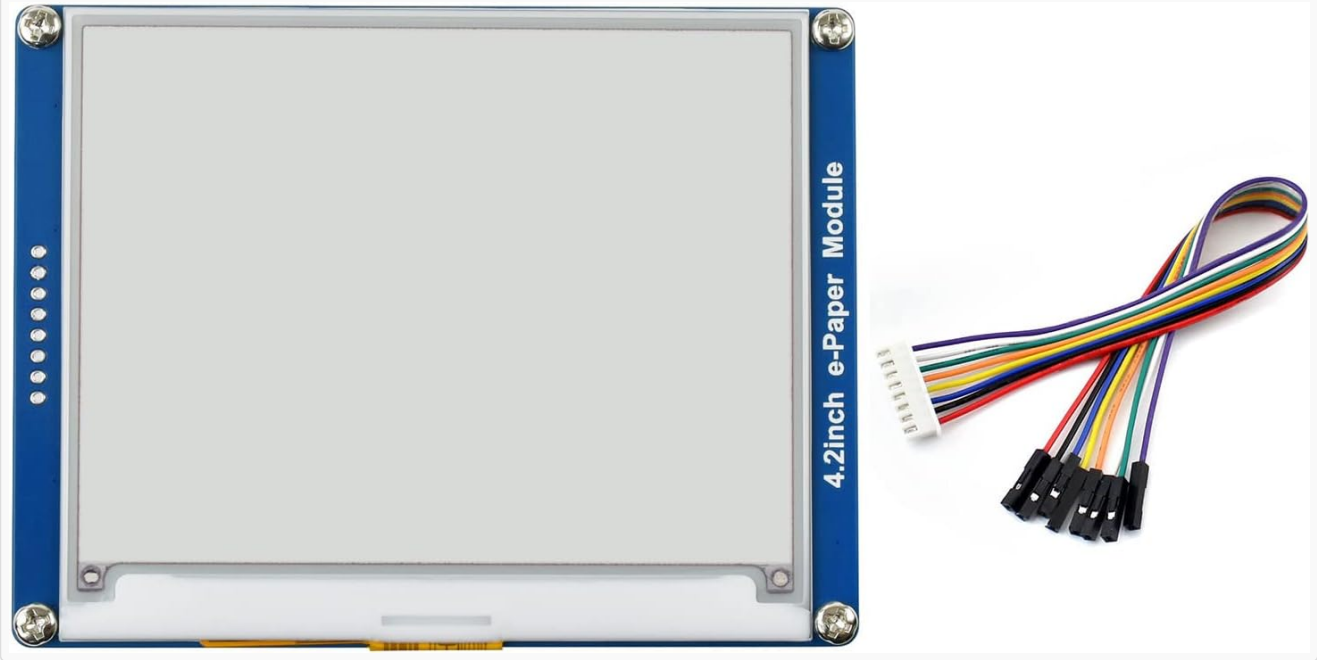


Figure 2: The 4.2-inch E-Ink Display Module (B) alongside the included 8-pin PH2.0 connection cable.

3. Setup Instructions

This section outlines the general steps for connecting and preparing your E-Ink display module. For detailed, platform-specific instructions and code examples, refer to the official Waveshare Wiki.

3.1 Hardware Connection

1. **Identify Pins:** The module uses an SPI interface. Identify the corresponding SPI pins (MOSI, MISO, SCK, CS, DC, RST, BUSY) on your microcontroller (e.g., Raspberry Pi, Arduino).
2. **Connect Power:** Connect the VCC pin of the E-Ink module to a 3.3V or 5V power supply on your microcontroller. Connect the GND pin to the ground. The onboard voltage translator ensures compatibility with both voltage levels.
3. **Connect SPI Interface:** Use the provided PH2.0 8-pin cable to connect the E-Ink module to your microcontroller's SPI pins. Ensure correct pin mapping as per your microcontroller's documentation and the Waveshare Wiki.

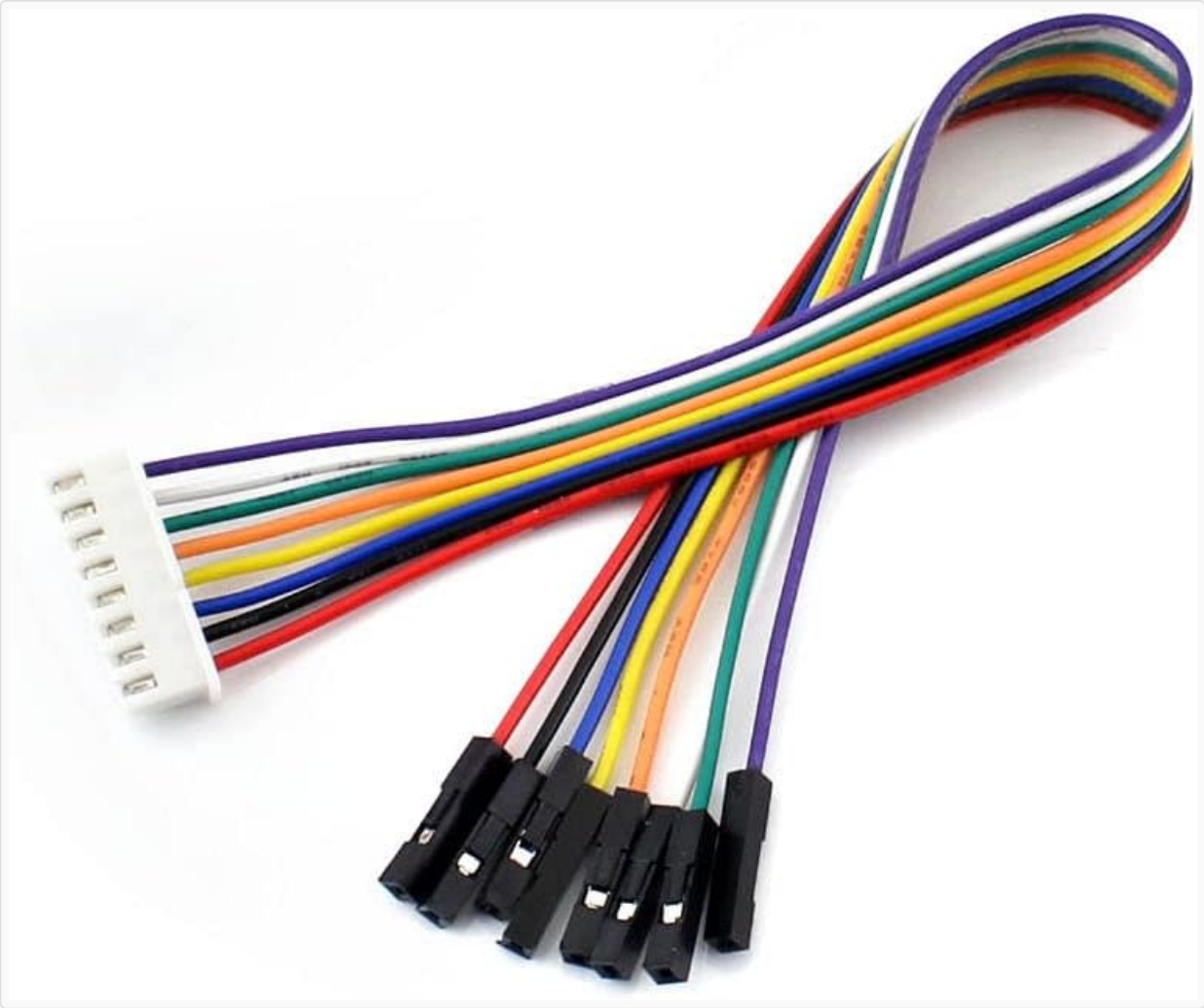


Figure 3: A close-up view of the 8-pin PH2.0 cable, typically used for connecting the display module to a microcontroller.

3.2 Software Setup

The Waveshare Wiki provides comprehensive development resources, including example code for Raspberry Pi, Jetson Nano, Arduino, and STM32. These examples demonstrate how to initialize the display and render content.

- **Download Libraries:** Obtain the necessary E-Ink display libraries from the Waveshare Wiki.
- **Load Examples:** Load the provided example code onto your microcontroller.
- **Configure:** Adjust the example code as needed for your specific application and pin configuration.

4. Operating the Display Module

Once connected and programmed, the E-Ink display operates by receiving data through the SPI interface to update its content. The display supports red, black, and white colors.

4.1 Display Refresh Cycle

E-Ink displays have a distinct refresh cycle. For the 4.2-inch module, a full refresh typically takes approximately 15 seconds. During this time, the display may flash or show intermediate patterns as the pigments are rearranged to form the new image. After the refresh, the image remains stable without consuming power.

4.2 Content Display

You can display various types of content, including text, graphics, and images. The provided software examples illustrate how to prepare and send image data to the display buffer for rendering.

Figure 4: The 4.2-inch E-Ink Display Module (B) displaying a graphic of a sleeping panda, demonstrating its image rendering capability.

5. Maintenance

The E-Ink display module requires minimal maintenance due to its robust design and low power consumption characteristics.

- **Cleaning:** Use a soft, dry, anti-static cloth to gently clean the display surface. Avoid using liquid cleaners or abrasive materials.
- **Environmental Conditions:** Operate and store the module within its specified temperature and humidity ranges to ensure optimal performance and longevity. Avoid exposure to direct sunlight for prolonged periods, as this can affect the display's lifespan.
- **Physical Handling:** Handle the module by its edges. Avoid applying pressure directly to the display area, as this can cause damage.
- **Power Management:** Since the display retains its image without power, you can disconnect power after a refresh if the content does not need to change, further conserving energy.

6. Troubleshooting

If you encounter issues with your E-Ink display module, consider the following troubleshooting steps:

- **No Display/Blank Screen:**
  - Verify all power and SPI connections are secure and correctly wired.
  - Ensure your microcontroller is powered on and the code is running.

    - Check the power supply voltage (3.3V or 5V) to the module.
- **Incorrect/Garbled Display:**
    - Confirm that the SPI communication protocol and pin assignments in your code match your hardware connections.
    - Ensure the display resolution (400x300) is correctly configured in your software.
    - Check for any timing issues or incorrect data formatting in your display refresh routine.
- **Slow Refresh Rate:**
    - The 15-second full refresh time is inherent to this E-Ink technology. This is normal operation.
- **Display Artifacts/Ghosting:**
    - Ensure a full refresh cycle is completed when changing content significantly.
    - Some minor ghosting can occur with partial updates; a full refresh typically clears this.

For more advanced troubleshooting and specific error codes, consult the Waveshare Wiki or community forums.

7. Specifications

Feature	Specification
Operating Voltage	3.3V ~ 5V
Interface	3-wire SPI, 4-wire SPI
Outline Dimension	103.0mm × 78.5mm
Display Size	84.8mm × 63.6mm
Dot Pitch	0.212 × 0.212 mm
Resolution	400 × 300 pixels
Display Color	Red, Black, White
Grey Level	2
Full Refresh Time	Approximately 15 seconds
Refresh Power	26.4mW (typical)
Standby Power	<0.017mW
Viewing Angle	>170°
Item Weight	0.352 ounces
Package Dimensions	4.25 x 4.21 x 1.3 inches

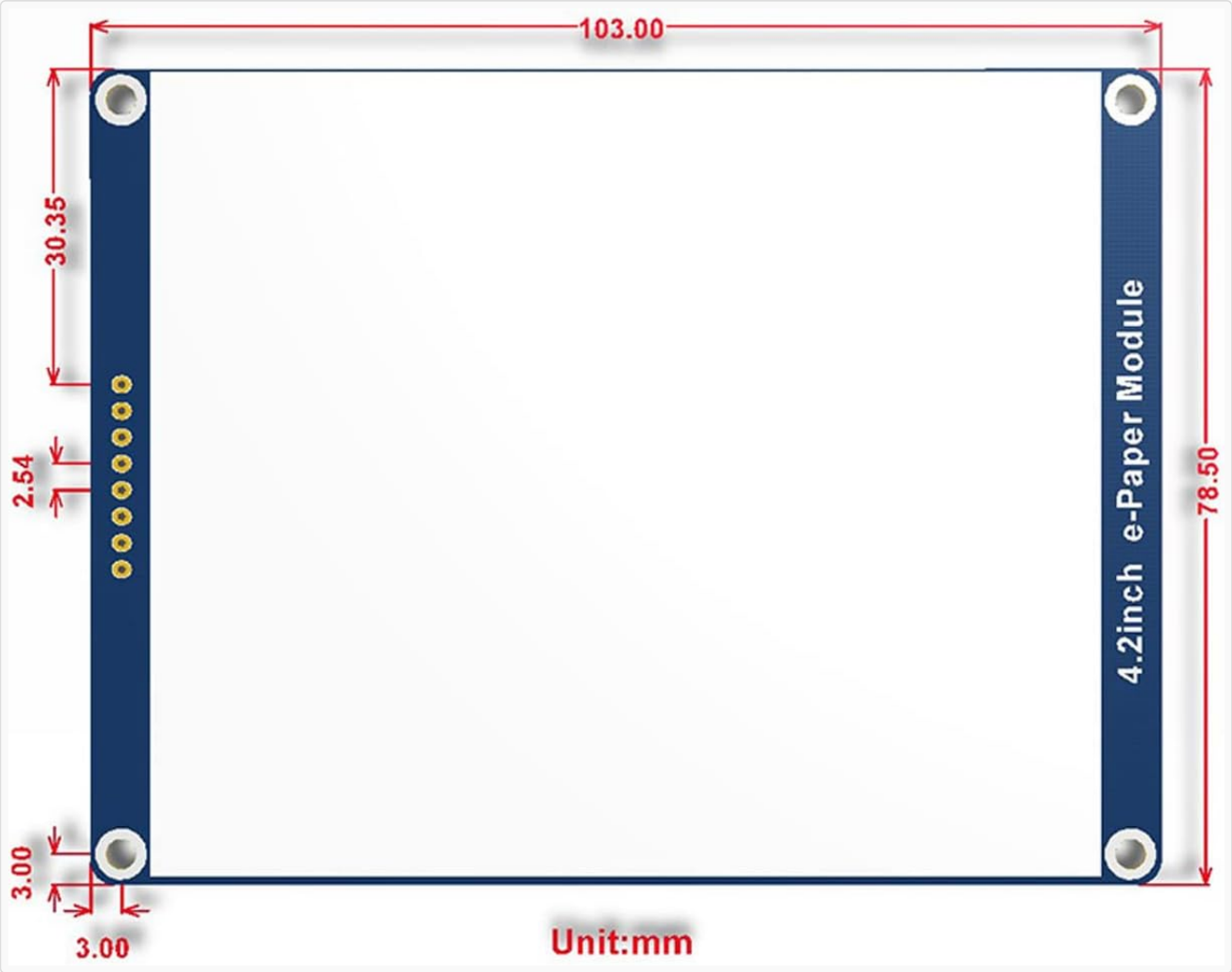


Figure 5: Dimensional drawing of the 4.2-inch E-Ink Display Module (B), with measurements in millimeters.

8. Support and Resources

For the most up-to-date documentation, development resources, and community support, please visit the official Waveshare Wiki:

www.waveshare.com/wiki/4.2inch_e-Paper_Module_(B)

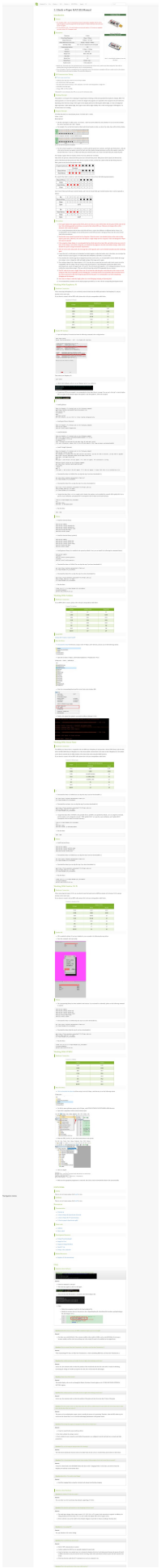
The Wiki includes:

- Detailed hardware connection diagrams.
- Example code for various microcontrollers (Raspberry Pi, Arduino, Jetson Nano, STM32).
- Datasheets and technical information.
- FAQs and troubleshooting guides.

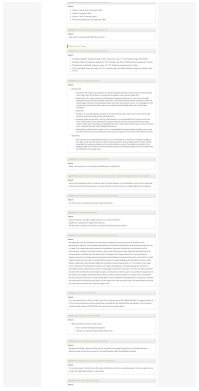
9. Warranty Information

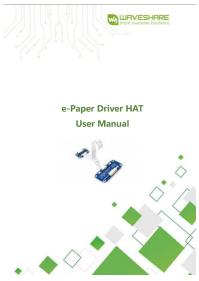
Specific warranty details for this product are not provided in the available information. Please refer to the Waveshare official website or contact their customer support for warranty terms and conditions.

Related Documents - 4.2inch e-Paper Module (B)



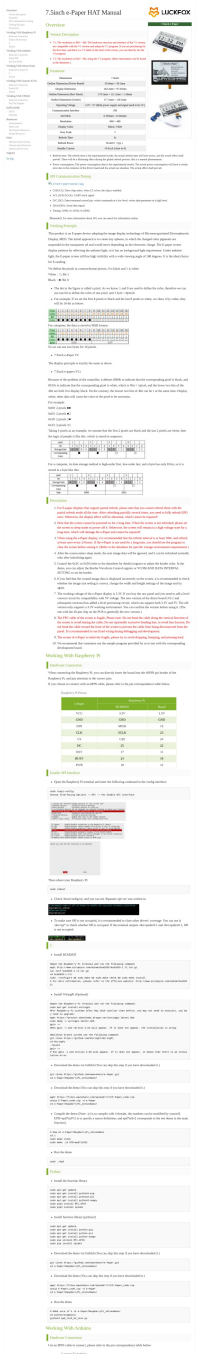
[Waveshare 2.13inch e-Paper HAT (B) User Manual and Technical Guide]
Comprehensive guide for the Waveshare 2.13inch e-Paper HAT (B), covering hardware connections, software setup, programming principles, and troubleshooting for Raspberry Pi, Arduino, Jetson Nano, and STM32.





[Waveshare e-Paper Driver HAT User Manual: Connect SPI E-Paper Displays to Raspberry Pi, Arduino, STM32]

User manual for the Waveshare e-Paper Driver HAT, detailing its features, product parameters, interface specifications, and supported e-Paper models. Includes setup guides for Raspberry Pi, Arduino, and STM32 development boards.



[Waveshare 7.5-inch E-Paper HAT User Manual and Guide]

This comprehensive user manual provides detailed information on the Waveshare 7.5-inch E-Paper HAT (V1/V2), an 800x480 resolution display module utilizing Microencapsulated Electrophoretic Display technology. It covers hardware connections, SPI communication, working principles, and integration with Raspberry Pi, Arduino, Jetson Nano, Sunrise X3 Pi, STM32, ESP32, and ESP8266. Essential precautions, resources, and FAQs are included for optimal use.

every train set up monitoring to prevent heavy rain/snow.

Question: Paper choice: thick/thin?

Answer:
The harder display color use for air through the Border Greenhouse Control engineer or the 3 COLORS AND ILLUSTRATIONAL SETTING original.

Question: When the ink screen is in deep sleep mode, the first time the screen refreshes will be unclear, what can I solve it?

Answer:
The previous B is considering the a paper screen is actually the process of supplying the power again. Therefore, when the EPD wakes up, it must first clear the screen, so as to avoid the afterimage phenomenon in the general screen.

Questions about Hardware

Question: Can dedicate TV drive the ink screen?

Answer:
Yes, now there is a hired customize chip onboard, supporting a TV drive.

Question: Which should be paid attention to when designing the driver board?

Answer:
- The rated input voltage of the e-paper screen is 3.3 V.If 5 V or +5 V system, level conversion is required.In addition, the voltage should not be lower than 2.2V, or as not to affect the display effect of the e-paper screen.
- Device selection can use the model in the schematic diagram we provide or choose according to the data sheet.

Question: Can't see reading SPI

Answer:
Yes, pay attention to the screen's timing.

Question: What is the BUSY pin always busy?

Answer:
- Check if SPI communication is normal.
- Confirm whether the BUSY pin is normally initialized to input mode.
- Jumps for that show it as normal state, try to shorten the duration of the low level during reset (because the internal pull-up resistance is 50K, it will be pulled up to the high level first before being valid, it will cause the driver board to power off and reset the wrong to fail).
- If the busy function could not be GET command, you can try to commment it out.

Question: What is the specification of the e-paper cable interface?

Answer:
- 1.54inch, 4-16inch, 6inch, 6-8inch pitch, 2054w.
- 1.54inch, 6-31inch pitch, 3600w.
- 4.37inch, 7.5inch, 6-64inch pitch, 5600w.
- The rest (unspecified pitch) are 6-8inch pitch, 1600w.

Question: What type of connector does the ink screen use?

Answer:
Cable socket 0.5/clipsx non-flip 0.8H (FPC connector).

Questions about Screens

Question: What is the usage environment of the e-paper screen?

Answer:
- 2.9inch(sg-version): Temperature range: 0~50°C; Humidity range: 45%~85%RH
- 2.9inch(ssd-version): Temperature range: below 35°C; Humidity range: below 85%RH; Maximum storage time: 9 months.
- 2.9inch(interim-version): Temperature range: -25~70°C; Maximum transportation time: 10 days.
- 2.9inbc-repo(tag): Temperature range: 20°C~5°C; Humidity range: 10%~60%RH; Maximum storage time: Available within 72 hours.

Question: Precautions for E-paper screen refresh

Answer:
- Refresh mode
  - Full refresh: The e-paper screen will flicker several times during the refresh process (the number of flicks depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
  - Partial refresh: The e-paper screen has no flickering effect during the refresh process. Users who use the partial loading function note that after refreshing several times, a full refresh operation should be performed to remove the residual image, otherwise the residual image problem will affect the display effect. (Applicable to products with partial refresh function, such as electronic number tags, electronic price labels, etc.)
  - Refresh rate
  - During use, it is recommended that customers set the refresh interval of the e-paper screen to at least 180s, in order to keep the screen in the e-paper life (recommended).
  - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-paper screen to sleep mode, or power off the power supply part of the e-paper screen can be disconnected with an analog switch) to reduce power consumption and prolong the service life of the e-paper screen (the e-paper screen are powered on for a long time, the screen will be damaged.Normal repair.)
  - During the use of the finished e-ink e-paper screen, it is recommended that customers update the display content of the e-paper screen at least once every 24 hours to ensure the same screen, by doing so, the screen lifetime will be 1000 (48 or longer).
- Applications
  - The e-paper screen is not recommended for indoor use. If it is used outdoors, it is necessary to avoid direct sunlight on the e-paper screen, and at the same time, take UV protection measures, otherwise the e-paper screen will be damaged after a long time. (Such as yellowing of the screen, aging and failure to refresh). This situation is irreversible. Unless designing e-paper screen products, customers should pay attention to determining whether the use environment meets the requirements of our e-paper screens.

Question: What is the refresh cycle/lifetime of the e-paper screen?

Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times).

Question: After using for a period of time, the screen retreat, [Gif refresh], but 1 section afterimage (pebble) that cannot be repaired?

Answer:
Because the development board for a long time, after each refresh operation, it is recommended to set to Sleep to shut down to directly power off processing, otherwise, the screen may have not when the screen is in a high-voltage state for a long time.

Question: Why the e-paper screen enters sleep sleep mode, can it be refreshed again?

Answer:
Yes, but you need to re-initialize the ethernet, paper with software.

Question: Why is the busy Pin(busy) always?

Answer:
busy is the PIN pin in a high, resulting in that they try to reduce the SPI rate.
Insufficient or unstable power supply leads to data loss.
The data cable is too long to cause data loss, the transmission cable should not exceed 20cm.

Question: What is the resolution, RGB of the e-paper screen, and what does it do?

Answer:
The display gray scale of electrophoretic electronic paper is determined by the speed (position) of the charged pigment particles moving in the vertical direction. The charged pigment particles will move up and down under the action of voltage. This voltage response time presents the electrophoretic movement of the particles in the driving force of the electronic paper, waveform. The driving waveform is the core part of the electronic paper display and the optimization of the driving waveform will directly affect the display effect of the display. The driving waveform file is used to describe the parameters formed by the combination of a series of voltage pulses. The driving waveform is usually provided to the user in the form of a binary file. The driving waveform is usually written by the IC engineer of the e-paper display manufacturer and matched with the electronic paper of different specifications and models. Different brands of e-paper display panels are usually compatible with multiple different voltage values when driving the display due to materials, manufacturing processes, etc. The selection of the e-paper screen is reflected in the relationship between grayscale, voltage, and temperature. Generally speaking, after each batch of electrophoretic matrix is generated, there will be a corresponding waveform file to the front of it with the film. The film manufacturer will provide the waveform file and corresponding power control board for the electronic paper screen, and the development board of the electronic paper screen manufacturer will burn the waveform file into the FLASH of the main control board. If customers want to change the display effect of the screen, they can modify the waveform file in the customer, if the waveform file does not correspond to the screen, it is likely that the display cannot be displayed or the display effect is unsatisfactory. Generally, the waveform file has OTP built into the driver IC of the ink screen when leaving the factory, and some programs can provide also called external waveform files to drive the e-paper screen.

Question: What does UC and SSD mention?

Answer:
UC is the abbreviation of LOAD VCOMDEL, and OTP is the abbreviation of ONE TIME PROGRAM. The original Comment of UC is to load waveform files, and the waveform files are divided into OTP and REGISTER. Among them, OTP is the built-in waveform storage method, and REGISTER is the external waveform storage method.

Question: What is the process of refreshing a gray waveform?

Answer:
- There are mainly two types of Gray waves.
  - One is waveform, the background image first.
  - The other is an afterimage waveform and data and save data.

Question: How to refresh to 4 different positions of the same 2bit?

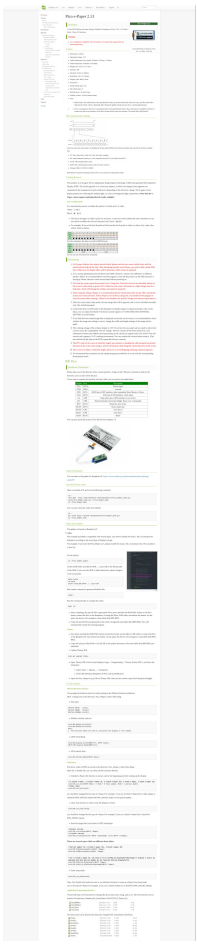
Answer:
Simultaneous binding to different locations needs to be operated via the program design, that is, first binding the data of different locations into the electronic paper IC, and finally doing the Update/Display(display) uniformly.

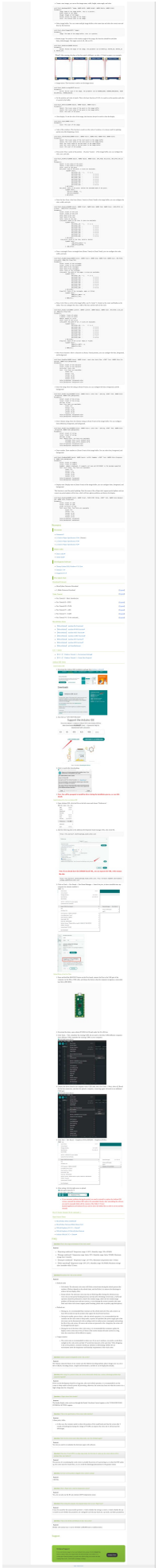
Question: Red font screen clipped with a film?

Answer:
with film.

Question: Does e-paper have a built-in temperature sensor?

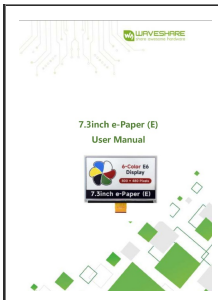
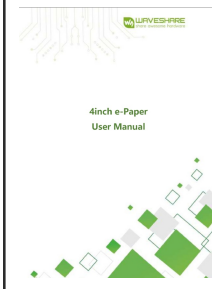
Answer:
At present, all screens have built-in temperature sensors, and you can also use an external GDP3 temperature sensor with IIC pins.



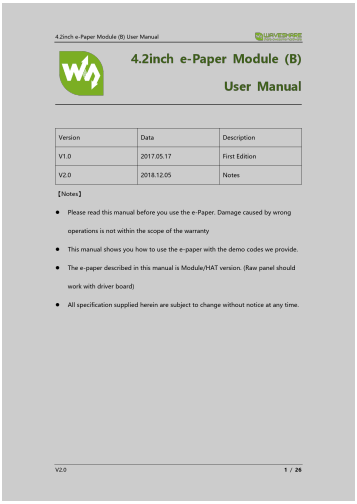


[Waveshare Pico e-Paper 2.13inch EPD Module for Raspberry Pi Pico: Development Guide & API]

Detailed development guide for the Waveshare Pico e-Paper 2.13inch EPD module with Raspberry Pi Pico. Features include 250x122 resolution, SPI interface, C/C++ & MicroPython demo codes, and comprehensive API documentation.

	[Waveshare 7.3inch e-Paper (E) User Manual - Specifications and Guide] Comprehensive user manual for the Waveshare 7.3inch e-Paper (E) display module, detailing specifications, features, pin assignments, electrical and optical characteristics, and handling instructions.
	[Waveshare 4-inch e-Paper Display User Manual] Comprehensive user manual for the Waveshare 4-inch e-Paper display module (EL040EF1), detailing its features, specifications, electrical characteristics, power sequences, optical properties, handling, safety, and reliability tests.

Documents - waveshare – 4.2inch e-Paper Module (B)


4.2inch e-Paper Module (B) User Manual

4.2inch e-Paper Module (B)

User Manual

Version	Data	Description
V1.0	2017.05.17	First Edition
V2.0	2018.12.05	Notes

【Notes】

- Please read this manual before you use the e-Paper. Damage caused by wrong operations is not within the scope of the warranty
- This manual shows you how to use the e-paper with the demo codes we provide.
- The e-paper described in this manual is Module/HAT version. (Raw panel should work with driver board)
- All specification supplied herein are subject to change without notice at any time.

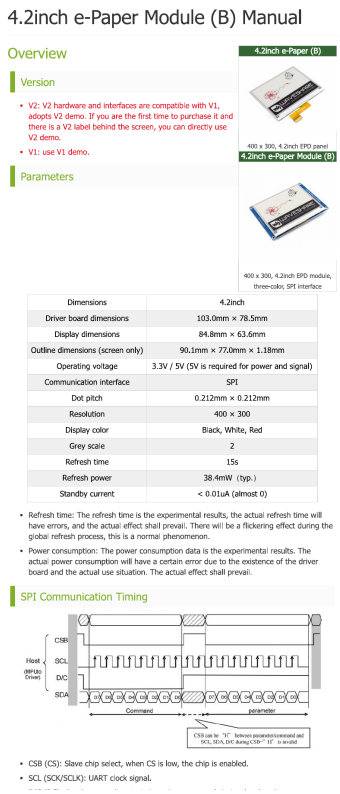
V2.0 1 / 26

[pdf] User Manual Specifications Warranty

User Manual ● This manual shows you how to use the e paper with demo codes we provide The described in this is 4 2inch Paper Module B display ink 4n2in red user wiki amperka ru media products 80 yellow module b en waveshare w upload 2 20

4.2inch e-Paper Module B User Manual 4.2inch e-Paper Module B User Manual Version Data Description V1.0 2017.05.17 First Edition V2.0 2018.12.05 Notes Notes Please read this manual before you use the e-Paper. Damage caused by wrong operations is not within the scope of the warranty ...

lang:en score:39 filesize: 523.25 K page_count: 26 document date: 2018-12-25


4.2inch e-Paper Module (B) Manual

Overview

Version

- V2: V2 hardware and interfaces are compatible with V1, adopts V2 demo. If you are the first time to purchase it and there is a V2 label behind the screen, you can directly use V2 demo.
- V1: use V1 demo.

4.2inch e-Paper (B)

400 x 300, 4.2inch EPD panel

4.2inch e-Paper Module (B)

400 x 300, 4.2inch EPD module, three-color, SPI interface

Parameters

Dimensions	4.2inch
Driver board dimensions	103.0mm × 78.5mm
Display dimensions	84.8mm × 63.6mm
Outline dimensions (screen only)	90.1mm × 77.0mm × 1.18mm
Operating voltage	3.3V / 5V (5V is required for power and signal)
Communication interface	SPI
Dot pitch	0.212mm × 0.212mm
Resolution	400 × 300
Display color	Black, White, Red
Grey scale	2
Refresh time	15s
Refresh power	38.4mW (typ.)
Standby current	< 0.01uA (almost 0)

- Refresh time: The refresh time is the experimental results, the actual refresh time will have errors, and the actual effect shall prevail. There will be a flickering effect during the global refresh process, this is a normal phenomenon.
- Power consumption: The power consumption data is the experimental results. The actual power consumption will have a certain error due to the existence of the driver board and the actual use situation. The actual effect shall prevail.

SPI Communication Timing

- CSB (CS): Slave chip select, when CS is low, the chip is enabled.
- SCL (SCK/SCLK): UART clock signal.

- D/C (DC): data/command control pin, writes commands in low level; writes data/parameter in high level.
- SDA (DIN): serial data signal.
- Timing: CPHL=0, CPOL=0 (SPI0)

【Remarks】For more information about SPI, you can search for information online.

Working Principle

This product is an E-paper device adopting the image display technology of Microencapsulated Electrophoretic Display, MED. The initial approach is to create tiny spheres, in which the charged color pigments are suspended in the transparent oil and would move depending on the electronic charge. The E-paper screen display patterns by reflecting the ambient light, so it has no background light requirement. Under ambient light, the E-paper screen still has high visibility with a wide viewing angle of 180 degrees. It is the ideal choice for E-reading.

Program Principle

- We define the pixels in a monochrome picture, 0 is black and 1 is white.
  - White : □: Bit 1
  - Black : ■: Bit 0
- The dot in the figure is called a pixel. As we know, 1 and 0 are used to define the color, therefore we can use one bit to define the color of one pixel, and 1 byte = 8pixels
- For example, If we set the first 8 pixels to black and the last 8 pixels to white, we show it by codes, they will be 16-bit as below:

Pixel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Color	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□

- For a computer, its data storage method is high-order first, low-order later, and a byte has only 8 bits, so there will be a little change:

Pixel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Bit	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Color	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□
Byte	0x00								0xFF							

- In this way, only 2 bytes are needed to represent 16 pixels.
- For 4.2inch e-paper B, which is red, black, and white, we need to split the picture into two pictures, one black and white picture, one red and white picture, because one register controls black and white display during transmission and one register control Red and white display. 1 byte in the black and white part controls 8 pixels, 1 byte in the red and white part controls 8 pixels
- Suppose there are 8 pixels, the front 4 are red, and the back 4 are black. Then you need to split them into a black and white picture, a red and white picture, these two pictures are 8 pixels, but the front of the black and white picture The four pixels are white, and the last 4 pixels are black, while the first 4 pixels of the red and white picture are red, and the last four pixels are white.

Original picture	■	■	■	■	■	■	■	■
Divided								
Black/White	□	□	□	□	■	■	■	■
Red/White	■	■	■	■	□	□	□	□

- If we stipulate that white is stored as 1, and red or black is stored as 0, then we have the following representation:

Black/White	□	□	□	□	■	■	■	■
Data	1	1	1	1	0	0	0	0
Red/White	■	■	■	■	□	□	□	□
Data	0	0	0	0	1	1	1	1

- And 1 byte of the black and white part controls 8 pixels, and 1 byte of the red and white part controls 8 pixels, then it can be expressed as follows:

Bit	1	2	3	4	5	6	7	8
Black/White	□	□	□	□	■	■	■	■
Data	1	1	1	1	0	0	0	0
Byte	0xF0							
Bit	1	2	3	4	5	6	7	8
Red/White	■	■	■	■	□	□	□	□
Data	0	0	0	0	1	1	1	1
Byte	0x0F							

Precautions

1. For E-paper displays that support partial refresh, please note that you cannot refresh them with the partial refresh mode all the time. After refreshing partially several times, you need to fully refresh EPD once. Otherwise, the display effect will be abnormal, which cannot be repaired!
2. It is a normal phenomenon that the three-color EPD will have a certain color difference in different batches. Hence, It is recommended to use the program to clear all the pictures on the EPD and store it facing up. Please clear the screen several times before powering on.
3. Note that the screen cannot be powered on for a long time. When the screen is not refreshed, please set the screen to sleep mode or power off it. Otherwise, the screen will remain in a high voltage state for a long time, which will damage the e-Paper and cannot be repaired!
4. When using the e-Paper display, it is recommended that the refresh interval be at least 180s, and refresh at least once every 24 hours. If the e-Paper is not used for a long time, you should use the program to clear the screen before storing it. (Refer to the datasheet for specific storage environment requirements.)
5. After the screen enters sleep mode, the sent image data will be ignored, and it can be refreshed normally only after initializing again.
6. Control the 0x3C or 0x50 (refer to the datasheet for details) register to adjust the border color. In the demo, you can adjust the Border Waveform Control register or VCOM AND DATA INTERVAL SETTING to set the border.
7. If you find that the created image data is displayed incorrectly on the screen, it is recommended to check whether the image size setting is correct, change the width and height settings of the image and try again.
8. The working voltage of the e-Paper display is 3.3V. If you buy the raw panel and you need to add a level convert circuit for compatibility with 5V voltage. The new version of the driver board (V2.1 and subsequent versions) has added a level processing circuit, which can support both 3.3V and 5V. The old version only supports a 3.3V working environment. You can confirm the version before using it. (The one with the 20-pin chip on the PCB is generally the new version.)
9. The FPC cable of the screen is fragile, Please note: Do not bend the cable along the vertical direction of the screen to avoid tearing the cable; Do not repeatedly excessive bending line, to avoid line fracture; Do not bend the cable toward the front of the screen to prevent the cable from being disconnected from the panel. It is recommended to use fixed wiring during debugging and development.
10. The screen of e-Paper is relatively fragile, please try to avoid dropping, bumping, and pressing hard.
11. We recommend that customers use the sample program provided by us to test with the corresponding development board.

Working With Raspberry Pi

Hardware Connection

When connecting the Raspberry Pi, you can directly insert the board into the 40PIN pin header of the Raspberry Pi, and pay attention to the correct pins.

If you choose to connect with an 8PIN cable, please refer to the pin correspondence table below:

Raspberry Pi Pinout

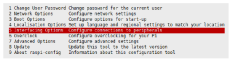
e-Paper	Raspberry Pi	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	MOSI	19

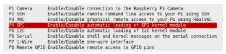
CLK	SCLK	23
CS	CE0	24
DC	25	22
RST	17	11
BUSY	24	18

Enable SPI Interface

- Open the Raspberry Pi terminal and enter the following command in the config interface:

```
sudo raspi-config
Choose Interfacing Options -> SPI -> Yes Enable SPI interface
```



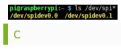


Then reboot your Raspberry Pi:

```
sudo reboot
```

- Check /boot/config.txt, and you can see 'dtparam=spi=on' was written in.

- To make sure SPI is not occupied, it is recommended to close other drivers' coverage. You can use ls /dev/spi* to check whether SPI is occupied. If the terminal outputs /dev/spidev0.1 and /dev/spidev0.1, SPI is not occupied.



C

- Install BCM2835

```
#Open the Raspberry Pi terminal and run the following command
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.71.tar.gz
tar zxvf bcm2835-1.71.tar.gz
cd bcm2835-1.71/
sudo ./configure && sudo make && sudo make check && sudo make install
# For more information, please refer to the official website: http://www.a
irspayce.com/mikem/bcm2835/
```

- Install WiringPi (Optional)

```
#Open the Raspberry Pi terminal and run the following command:
sudo apt-get install wiringpi
#For Raspberry Pi systems after May 2019 (earlier than before, you may not
need to execute), you may need to upgrade:
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
#Run gpio -v and version 2.52 will appear. If it does not appear, the inst
allation is wrong.

#Bullseye branch system use the following command:
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
gpio -v
# Run gpio -v and version 2.60 will appear. If it does not appear, it mean
s that there is an installation error.
```

- Download the demo via GitHub (You can skip this step if you have downloaded it.)

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/
```

- Download the demo (You can skip this step if you have downloaded it.)

```
sudo apt-get install p7zip-full
wget  https://www.waveshare.com/w/upload/3/39/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

- Compile the demo (Note: -j4 is to compile with 4 threads, the numbers can be modified by yourself; EPD=epd4in2bv2 is to specify a macro definition, and epd4in2bV2 corresponds to the test demo in the main function).

```
# Now at e-Paper/RaspberryPi_JetsonNano
cd c
sudo make clean
sudo make -j4 EPD=epd4in2bV2
```

- Run the demo

```
sudo ./epd
```

Python

- Install the function library

```
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install RPi.GPIO
sudo pip3 install spidev
```

- Install function library (python2)

```
sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo apt-get install python-numpy
sudo pip install RPi.GPIO
sudo pip install spidev
```

- Download the demo via GitHub (You can skip this step if you have downloaded it.)

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/
```

- Download the demo (You can skip this step if you have downloaded it.)

```
sudo apt-get install p7zip-full
wget  https://www.waveshare.com/w/upload/3/39/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

- Run the demo

```
# Make sure it's in e-Paper/RaspberryPi_JetsonNano/
cd python/examples/
python3 epd_4in2b_V2_test.py
```

Working With Arduino

Hardware Connection

Use an 8PIN cable to connect, please refer to the pin correspondence table below:

Connect To Arduino

e-Paper	Arduino UNO	Mega2560
VCC	5V	5V
GND	GND	GND
DIN	D11	D51
CLK	D13	D52
CS	D10	D10
DC	D9	D9

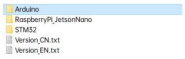
RST	D8	D8
BUSY	D7	D7

Install IDE

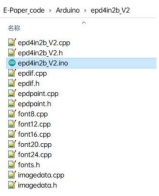
Arduino IDE Windows Install Guide

Run The Demo

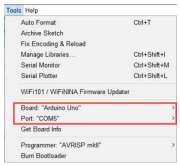
- Download the demo in Resource, unzip it to the "E-Paper_code" directory, and you can see the following content:



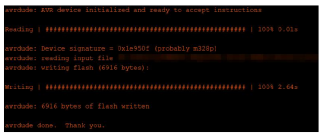
- Open the test demo: E-Paper_code\Arduino\epd4in2b_V2\epd4in2b_V2.ino



- Select the corresponding Board and Port in the Tools in the Arduino IDE.



- Finally, click upload, the upload is successful as follows (Arduino 1.8.13).



Working With Jetson Nano

Hardware Connection

The 40PIN pin of Jetson Nano is compatible with the 40PIN pin of Raspberry Pi and provides a Jetson.GPIO library with the same API as the RPI.GPIO library of Raspberry Pi, so the serial number connected here is the same as that of Raspberry Pi. The module can be directly inserted into the 40Pin headers of the Jetson Nano when using the 40PIN interface.
If you choose to connect with an 8PIN cable, please refer to the pin correspondence table below:

Connect to Jetson nano

e-Paper	Jetson Nano Developer Kit	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	10 (SPI0_MOSI)	19
CLK	11 (SPI0_SCK)	23
CS	8 (SPI0_CS0)	24
DC	25	22
RST	17	11
BUSY	24	18

C

- Download the demo via GitHub (you can skip this step if you have downloaded it.)

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/
```

- Download the test demo: (you can skip this step if you have downloaded it.)

```
sudo apt-get install p7zip-full
wget  https://www.waveshare.com/w/upload/3/39/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

- Compile the demo (Note: JETSON is the specified device, and RPI is not specified by default. -j4 is to compile by 4 threads, and the number can be changed by yourself. "EPD=epd4in2bV2" is to specify a macro definition, and "epd4in2bV2" corresponds to the test demo in the main function.)

```
# Now at e-Paper/RaspberryPi_JetsonNano
cd c
sudo make clean
sudo make JETSON -j4 EPD=epd4in2bV2
```

- Run the demo

```
sudo ./epd
```

Python

- Install function library

```
sudo apt-get update
sudo apt-get install python3-numpy
sudo apt-get install python3-pip
sudo pip3 install Jetson.GPIO
```

- Download the demo via GitHub (you can skip this step if you have downloaded it.)

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/
```

- Download the demo (you can skip this step if you have downloaded it.)

```
sudo apt-get install p7zip-full
wget  https://www.waveshare.com/w/upload/3/39/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

- Run the demo

```
# Make sure it's in e-Paper/RaspberryPi_JetsonNano/
cd python/examples/
python3 epd_4in2b_V2_test.py
```

Working With Sunrise X3 Pi

Hardware Connection

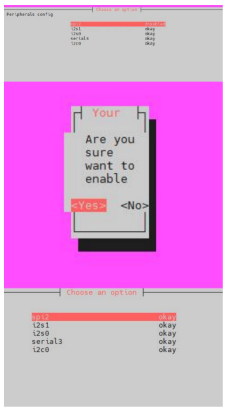
When connecting the Sunrise X3 Pi, you can directly insert the board into the 40PIN pin header of the Sunrise X3 Pi, and pay attention to the correct pins.
If you choose to connect with an 8PIN cable, please refer to the pin correspondence table below:

Connect to Sunrise X3 Pi

e-Paper	Sunrise X3 Pi	
	BCM	Board
VCC	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

Enable SPI

- SPI is enabled by default. If you have disabled it, you can enable it by following the steps below.
- Enter the command: sudo srpi-config



Python

- The corresponding library has been installed in the function. If you uninstall it accidentally, please use the following command to install it.

```
sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo apt-get install python-numpy
sudo pip install Robot.GPIO
sudo pip install spidev
```

- Download the demo via GitHub (skip this step if you have downloaded it).

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/
```

- Download the demo (skip this step if you have downloaded it).

```
sudo apt-get install p7zip-full
wget  https://www.waveshare.com/w/upload/3/39/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

- Run the demo

```
# Make sure you are in e-Paper/RaspberryPi_JetsonNano/
cd python/examples/
python3 epd_4in2b_V2_test.py
```

Working With STM32

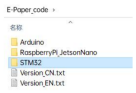
Hardware Connection

Connect to STM32

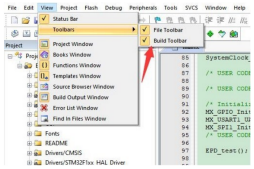
e-Paper	STM32
VCC	3.3V
GND	GND
DIN	PA7
CLK	PA5
CS	PA4
DC	PA2
RST	PA1
BUSY	PA3

Run The Program

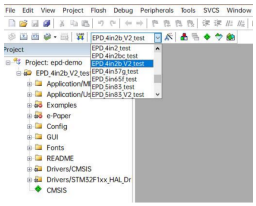
- Click to download the demo#, and then unzip it into the E-Paper_code directory to see the following content.



- Use Keil to open epd-demo.uvprojx in the E-Paper_code\STM32\STM32-F103ZET6\MDK-ARM directory
- Open Keil's compilation toolbar (usually already open).



- Select the EPD_4in2b_V2_test at the location shown in the picture.



- Click to compile.

- Make sure the appropriate programmer is connected, then click LOAD to download the demo to the microcontroller.

ESP32/8266

[pdf] User Manual Dimension Guide Label

4 2inch e Paper Module B Manual OverviewDimensions Driver board dimensions 103 0mm × 78 5mm Display 84 8mm 63 6mm Outline screen onlyA1pTQyAmDRLm media amazon images I A1pTQyAmDRL |||

4.2inch e-Paper Module B Manual Overview 4.2inch e-Paper B Version V2: V2 hardware and interfaces are compatible with V1, adopts V2 demo. If you are the first time to purchase it and there is a V2 label behind the screen, you can directly use V2 demo. V1: use V1 demo. 400 x 300, 4.2inch EPD...

lang:en score:28 filesize: 1.57 M page_count: 1 document date: 2023-03-28

ESP32

There is a lot of content, please click the URL below to view:
https://www.waveshare.com/wiki/E-Paper_ESP32_Driver_Board

ESP8266

There is a lot of content, please click the URL below to view:
https://www.waveshare.com/wiki/E-Paper_ESP8266_Driver_Board

Resources

Documentation

- Schematic
- Datasheets

Demo code

- Demo (E-Paper_code.7z)
- Demo (E-Paper_code.zip)
- Github

Development Resources

- E-Paper Floyd-Steinberg
- Image2Lcd.7z
- Image2Lcd Image Modulo
- Zimo221.7z
- E-Paper_API_Analysis

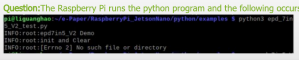
Related Resources

- Raspberry Pi Documentation

FAQ

Question about Software

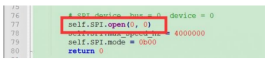
Question: The Raspberry Pi runs the python program and the following occurs?



Answer:

- Enter the command: ls /dev/spi*
- The result may appear as shown in the figure

- This is because the SPI interface is occupied in the /boot/config.txt file.

- Steps:
  - Delete the occupation of spi0-0 in the /boot/config.txt file.
  - Modify the location shown in the picture in the /e-Paper/RaspberryPi_JetsonNano/lib/waveshare_epd/epdconfig.py file and change it to 0,1.
  - 

Question:stm32 drives the ink screen, the MDK compilation display space is not enough?

Answer:
*Our demo uses stm32f103zet6. If the customer modifies other models in MDK, such as stm32f103rbt6, the ram space becomes smaller, and the stack size and heap size in the startup file need to be modified on the original basis.

Question: When to transmit Data Start Transmission 1 and when to use Data Start Transmission 2?

Answer:
When transmitting B/W data, use Data Start Transmission 1; when transmitting RED data, use Data Start Transmission 2.

Question: After multiple positions are brushed, the font is lighter after brushing several times?

Answer:
In this case, the customer needs to reduce the position of the round brush and clear the screen after 5 rounds of refreshing (increasing the voltage of VCOM can improve the color, but it will increase the afterimage).

Question:e-Paper shows black border?

Answer:
The border display color can be set through the Border Waveform Control register or the VCOM AND DATA INTERVAL SETTING register.

Question: After multiple positions are brushed, the font is lighter after brushing several times?

Answer:
In this case, the customer needs to reduce the position of the game and clear the screen after 5 times of the game.

Question: When the ink screen is in deep sleep mode, there will be a problem that the screen refresh will not be clean when it wakes up for the first time. How can I solve it?

Answer:
The process of re-awakening the e-ink screen is actually the process of re-powering. Therefore, when the EPD wakes up, the screen must be cleared first, so as to avoid the afterimage phenomenon to the greatest extent.

Question: When testing the program, the program has been stuck in e-Paper busy?

Answer:
*It may be caused by the unsuccessful spi driver.

1. First check whether the wiring is correct.
2. Check whether the spi is turned on and whether the parameters are configured correctly (spi baud rate, spi mode and other parameters).

Question: Why can't the image be displayed after full refreshing?

Answer:
The full refresh initialization function needs to be added when the ink screen is switched from partial refresh to full refresh.

Question: Why is the printing information normal when running a python program, but the ink screen does not respond?

Answer:
It may be a demo based on the BCM2835 library that has run the C language before. At this time, you need to restart the Raspberry Pi and then run the python demo.

Question: ImportError: No module named Image?

Answer:
*Install the imaging library using the command sudo apt-get install python-imaging

Question about Hardware

Question: Can Arduino 5V drive the ink screen?

Answer:
Yes, now there is a level conversion chip onboard, supporting a 5V drive.

Question: What should be paid attention to when designing the driver board?

Answer:

- The rated input voltage of the ink screen is 2.3~3.6V. If it is a 5V system, level conversion is required. In addition, the voltage should not be lower than 2.5V, so as not to affect the display effect of the ink screen.
- Device selection can use the model in the schematic diagram we provide or choose according to the data sheet.

Question: Can I use analog SPI?

Answer:
Yes, pay attention to the correct timing.

Question: Why is the BUSY pin always busy?

Answer:

- Check if SPI communication is normal.
- Confirm whether the BUSY pin is normally initialized to input mode.
- It may be that there is no normal reset, try to shorten the duration of the low level during reset (because the power-off switch is added to the drive circuit, the reset low level is too long, which will cause the drive board to power off and cause the reset to fail).
- If the busy function sends the 0x71 command, you can try to comment it out.

Question: What is the specification of the screen cable interface?

Answer:

- 1.64inch, 2.36inch, 3inch, 0.5mm pitch, 26Pin.
- 1.02inch, 0.5mm pitch, 30Pin.
- 4.37inch, 7.3inch, 0.5mm pitch, 50Pin.
- The rest (non-parallel ports) are 0.5mm pitch, 24Pin.

Question: What type of connector does the ink screen use?

Answer:
Cable socket 0.5-24pin rear-flip 2.0H (FPC connector).

Question about Screen

Question: What is the usage environment of the e-ink screen?

Answer:

- 【Working conditions】Temperature range: 0~50°C; Humidity range: 35%~65%RH.
- 【Storage conditions】: Temperature range: below 30°C; Humidity range: below 55%RH; Maximum storage time: 6 months.
- 【Transportation conditions】: Temperature range: -25~70°C; Maximum transportation time: 10 days.
- 【After unpacking】: Temperature range: 20°C±5°C; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours.

Question: Precautions for e-ink screen refresh?

Answer:

- refresh mode
    - Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
    - Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full refresh operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial refreshing, please refer to product page description).
    - refresh rate
    - During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except for products that support the local brush function).
    - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode, or power off (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
    - During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).
- Application
    - The e-ink screen is recommended for indoor use. If it is used outdoors, it is necessary to avoid direct sunlight on the e-ink screen, and at the same time, take UV protection measures, because charged particles will dry out under strong light for a long time, resulting in loss of activity and failure to refresh. This situation is irreversible. When designing e-ink screen products, customers should pay attention to determining whether the use environment meets the requirements of an e-ink screen.

Question: What is the refresh rate/lifetime of the e-ink screen?

Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times).

Question: After using for a period of time, the screen refresh (full refresh) has a serious afterimage problem that cannot be repaired?

Answer:
Power on the development board for a long time, after each refresh operation, it is recommended to set the screen to sleep mode or directly power off processing, otherwise, the screen may burn out when the screen is in a high voltage state for a long time.

Question: After the ink screen enters deep sleep mode, can it be refreshed again?

Answer:
Yes, but you need to re-initialize the electronic paper with software.

Question: Why is the image displayed offset?

Answer:
Maybe the SPI rate is too high, resulting in data loss, try to reduce the SPI rate.

Insufficient or unstable power supply leads to data loss.

The data cable is too long to cause data loss, the extension cable should not exceed 20cm.

Question: What is the waveform file of the e-ink screen and what does it do?

Answer:
The display gray scale of electrophoretic electronic paper is determined by the spatial position of the particles in the Microcapsule or Microcup. The electrophoresis phenomenon occurs between black particles and white particles under the action of voltage. This voltage sequence that promotes the electrophoretic movement of the particles is the driving force of the electronic paper. waveform. The driving waveform is the core part of the electronic paper display, and the optimization of the driving waveform will directly affect the display effect of the display. The driving waveform file is used to describe the parameters formed by the voltage sequence that promotes the electrophoretic movement of the particles, and it needs to be called regularly when the electronic paper is refreshed.

Different batches of e-paper diaphragms and electrophoretic matrices require different voltage values when driving the display due to materials, manufacturing processes, etc. The waveform of the e-ink screen is reflected in the relationship between grayscale, voltage, and temperature. Generally speaking, after each batch of electrophoresis matrix is generated, there will be a corresponding waveform file in the form of a .wbf file. The film manufacturer will provide the waveform file and electrophoresis matrix to the manufacturer of the electronic paper screen, and then the manufacturer of the electronic paper screen integrates the protection board, substrate, and driver and then provides it to customers; if the waveform file does not correspond to the screen, it is likely that the display cannot be displayed or the display effect is unsatisfactory. Generally, the waveform file has OTP built into the driver IC of the ink screen when leaving the factory, and some programs we provide also call external waveform files to drive the e-ink screen.

Question: What do LUT and OTP stand for?

Answer:
LUT is the abbreviation of LOOK UP TABLE, and OTP is the abbreviation of ONE TIME PROGRAM. The original intention of LUT is to load waveform files, and the waveform files are divided into OTP and REGISTER. Among them, OTP is the built-in waveform storage method, and REGISTER is the external waveform storage method.

Question: What is the process of partial refreshing?

Answer:
*There are mainly two types of ink screens.

- One is to refresh the background image first.
- The other is to alternately refresh old data and new data.

Question: How do I play in different positions at the same time?

Answer:
Simultaneous brushing in different locations needs to be operated in the program design, that is, first brushing the data of different locations into the electronic paper IC, and finally doing the Update/TurnOnDisplay uniformly.

Question: Does the three-color e-Paper have a red/yellow color difference?

Answer:
Yes, when e-Paper is batched, there will be some color difference, which is a normal phenomenon. Store the e-paper faces up to reduce the reddish/yellowishness to a certain extent.

Question: Are bare screens shipped with a film?

Answer:
with film.

Question: Does e-Paper have a built-in temperature sensor?

Answer:
At present, all screens have built-in temperature sensors, and you can also use an external LM75 temperature sensor with IIC pins.

Support

If you require technical support, please go to the page and open a ticket.

4.2inch e-Paper Module (B)

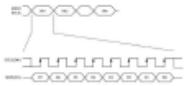
Overview

Version

Features

- Dimension: 4.2inch
- Outline Dimension (Driver board): 105.8mm x 76.5mm
- Display Dimension: 84.8mm x 63.6mm
- Operating Voltage: 3.3V/5V
- Interface: SPI
- Pixel Matrix: 400 × 300
- Display Color: Black, White, Red
- Grey Level: 2
- Full Refresh Time: 15s
- Power Consumption: 15mW(Typ.)
- Standby Current: <0.01uA (almost none)
- Note:
  - Refresh time: The refresh time is the experimental result, the actual refresh time will have errors, and the actual refresh time prevails. There will be a certain error due to the existence of the driver board and the actual use situation. The measured refresh time is
  - Power consumption: The power consumption data is the test result. The actual power consumption will have a certain error due to the existence of the driver board and the actual use situation. The measured refresh time is

SPI Communication Timing



Since the 3-color only needs to be displayed, the data cable (MOSI) sent from the master only needs to send 0 and 1.

- CS: SPI chip select, which CS is low, the chip is enabled
- DC: data/command control pin (high: data, low: command). When DC = 0, write command; DC = 1
- SCK: SPI communication clock
- SDIN: SPI communication data input pin, used to receive
- Timing: CPHA = 0, CPOL = 0 (SPI0)

[Attention] For details, please refer to the SPI protocol, or you can search for the relevant content.

Working principle

This product is an E-paper device adopting the image display technology of Microencapsulated Electrophoretic Display (MED). The internal pigment is to charge tiny spheres, in which the charged color pigments are suspended in the transparent oil. And would move depending on the electronic charge. The E-paper screen display patterns by reflecting the ambient light, so it has no backlight and light requirement. Under ambient light, the E-paper screen still has high visibility with a wide viewing angle of 180 degrees. It is the ideal choice for E-reading. **(Note that the e-Paper cannot support updating directly under sunlight)**

Pixel & Byte

We define the pixels in a monochrome picture. 0 is black and 1 is white.

White : or Bit 1

Black : ac Bit 0

- The dot in the figure is called a pixel. As we know, 1 and 0 are used to define the color, therefore we can use one bit to define the color of one pixel. And 1 byte = 8pixels
- For example, if we set the first 8 pixels to black and the last 8 pixels to white, we show it by codes, they will be 16 bites below:

Pixel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Color	■	■	■	■	■	■	■	■	□	□	□	□	□	□	□	□

- For 4.2inch e-Paper B, it has red and black, we need to split the pictures into 2 pictures, one black and white picture, one red and white picture, because one register controls the black and white display during transmission, and the other register is the control red and white

show: 4.2 1 byte of black and white part controls 8 pixels, 1 byte of red and white part controls 8 pixels. For example, Suppose there are 8 pixels, the first 4 are red, and the last 4 are black:

They need to be split into 1 black and white picture and 1 red and white picture. Both pictures have 8 pixels, but the first four pixels of the black and white picture are white, the first 4 pixels are black and the first 4 pixels of the red and white picture are black, pixels are red, the last four pixels are white

Before	■	■	■	■	■	■	■	■
After Split								
Black White	□	□	□	□	■	■	■	■
Storage	0	0	0	0	1	1	1	1
Red White	■	■	■	■	□	□	□	□
Storage	1	1	1	1	0	0	0	0

If we specify that white is stored as 1 and red or black is stored as 0, then we have the following representation:

Black White	□	□	□	□	■	■	■	■
Storage	1	1	1	1	0	0	0	0
Red White	■	■	■	■	□	□	□	□
Storage	0	0	0	0	1	1	1	1

And 1 byte of black and white part controls 8 pixels, and 1 byte of red and white part controls 8 pixels, then it can be expressed as follows:

Bit	1	2	3	4	5	6	7	8
Black White	□	□	□	□	■	■	■	■
Storage	1	1	1	1	0	0	0	0
Bit of Byte	1	1	1	1	0	0	0	0
Byte	0XF0							
Bit	1	2	3	4	5	6	7	8
Red White	■	■	■	■	□	□	□	□
Storage	0	0	0	0	1	1	1	1
Bit of Byte	0	0	0	0	1	1	1	1
Byte	0X0F							

Precautions

1. For the screen that supports partial update, please note that you cannot refresh the screen with the partial mode all the time. After several partial updating, you need to fully refresh the screen once. Otherwise, the screen display effect will be abnormal, which cannot be repaired!
2. Because of the different batches, some of them have aberrations. Store the e-Paper right side up will fade it. And if the e-Paper didn't be refreshed for a long time, it will become more and more reddish/yellowish. Please use the demo code to refresh the e-paper several times in this case.
3. Note that the screen cannot be powered on for a long time. When the screen is not refreshed, please set the screen to sleep mode, or power off the e-Paper. Otherwise, the screen will remain in a high voltage state for a long time, which will damage the e-Paper and cannot be repaired!
4. When using the e-Paper, it is recommended that the refresh interval be at least 180s. And for the same picture, at most only 24 hours. If the e-Paper is not used for a long time, the ink screen should be brushed and stored. (Refer to the display for specific settings and Environment requirements)
5. After the screen enters sleep mode, the sent image data will be ignored and it can be refreshed normally only after initializing again.
6. Control the BUSY (BS) field to the display for determining whether to adjust the border color. In the routine, you can adjust the Border Waveform Control register or VCOM AND DATA INTERVAL SETTING to set the border.
7. If you find that the created image data is displayed incorrectly on the screen, it is recommended to check whether the image size setting is correct, change the width and height settings of the image and try again.
8. The working voltage of the e-Paper is 3.3V. If you buy the new panel and you need to add a level conversion circuit for compatibility with 5V voltage. The new version of the driver board (V2.1 and subsequent versions) has added a level processing circuit, which can support both 3.3V and 5V working environments. The old version can only support a 3.3V working environment. You can confirm the version before using it. (The one with the 20 pin chip on the PCB is generally the new version)
9. The FPC cable of the screen is relatively fragile, pay attention to bending the cable along the horizontal direction of the screen when using it, and do not bend the cable along the vertical direction of the screen.
10. The screen of e-Paper is relatively fragile, please try to avoid dropping, bumping, and pressing hard.
11. We recommend that customers use the example program provided by us to test with the corresponding development board after they get the screen.

Users Guides of Raspberry Pi

Hardware connection

If the e-Paper you have is the HAT version which has 40pin GPIO, you can directly attach the e-Paper HAT on Raspberry Pi. Otherwise, you can connect your e-Paper to Raspberry Pi by 8pins cable provided.

To connect the e-Paper, you can following the table below

Connect to Raspberry Pi

e-Paper	Raspberry Pi	
	BCM2835	BoArd
VCC	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

Take the 7.5inch HD e-Paper (B) connected to the e-paper Driver HAT as an example, just plug it directly into the Raspberry Pi:

Enable SPI interface

The communication interface of e-Paper is SPI, to use it, we should firstly enable SPI interface

Open terminal of Raspberry Pi and open configuration by the following command

```
sudo raspi-config
```

Choose Interfacing Options -> SPI -> Yes

Restart Raspberry Pi

```
sudo reboot
```

Install libraries

Note: If you use the Bullseye image, please use pip instead of apt-get to install libraries, the Bullseye image could only support python3

Open terminal of Raspberry Pi and run the following command to install corresponding libraries:

- Install BCM2835 libraries

```
#Open the Raspberry Pi terminal and run the following command
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar zxvf bcm2835-1.68.tar.gz
cd bcm2835-1.68/
sudo ./configure && sudo make && sudo make check && sudo make install
# For more information, please refer to the official website: http://www.airspayce.com/mikem/bcm2835/
```

- Install WiringPi libraries

```
#Open the Raspberry Pi terminal and run the following command
sudo apt-get install wiringpi
#For Raspberry Pi systems after May 2019 (earlier than before, you may not need to upgrade), you may need to upgrade:
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# Run gpio -v and version 2.52 will appear. If it does not appear, the installation is wrong
#Bullseye branch system use the following commands:
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
gpio -v
# Run gpio -v and version 2.63 will appear. If it does not appear, it means that there is an installation error
```

- Install Python2 libraries

```
sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo apt-get install python-numpy
sudo pip install RPi.GPIO
sudo pip install spidev
```

- Install Python3 libraries

```
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install RPi.GPIO
sudo pip3 install spidev
```

Download demo codes

Open the Raspberry Pi terminal and execute:

Method 1: Download from GitHub website

```
sudo apt-get install p7zip-full
wget https://www.waveshare.net/w/upload/8/89/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

Method 2: Use GitHub

```
git clone https://github.com/waveshare/e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

Examples

C

Open main.c to select the screen of the corresponding size

```
cd c
sudo make examples/main.c
```

If you are using 2.7inch e-paper, then remove the // in front of the EPD_2IN7_test() function, then press ctrl+c, press the Y key and Enter to save and exit, as shown in the figure: (The picture may not be updated timely subject to the specific demo)

Supporting type:

1.02inch (128×80) :

EPD_1in02d_test() : Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

EPD_1in54_test() : Example for 1.54inch e-paper V1 (Black/White) : This version is stopped production which can be bought before 2019-11-22 ;

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version that can be bought now (2020-07-29). The e-Paper has V2 sticker on the backside.

EPD_1in54b_test() : Example for 1.54inch e-paper B (Black/White/Red) ;

EPD_1in54b_V2_test(): Example for 1.54inch e-Paper B V2 (Black/White/Red);

EPD_1in54c_test() : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

EPD_2in7_test() : Example for 2.7inch e-paper (Black/White) ;

EPD_2in7b_test() : Example for 2.7inch e-paper B (Black/White/Red) ;

EPD_2in7b_V2_test() : Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;

EPD_2in9_V2_test(): Example for 2.9inch e-paper V2 (Black/White) ;

EPD_2in9bc_test() : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2in9b_V3_test() : Example for 2.9inch e-paper B V3 (Black/White/Red) ;

EPD_2in9d_test() : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

EPD_2in13_test() : Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 019-05-13;

EPD_2in13_V2_test() : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;

EPD_2in13_V3_test() : Example for 2.13inch e-paper V3 (Black/White) This is the current version with sticker V3 on the backside (2020-07-29) ;

EPD_2in13b_test() : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;

EPD_2in13b_V3_test() : Example for 2.13inch e-paper B V3 (Black/White/Red)

EPD_2in13d_test() : Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)

EPD_2in66_test(): Example for 2.66inch e-Paper (Black/White)

EPD_2in66b_test(): Example for 2.66inch e-Paper B (Black/White/Red)

3.7inch (280 x 480)

EPD_3in7_test(): Example for 3.7inch e-Paper(Black/White);

4.01inch (640x400)

EPD_4in01_test(): Example for the 4.01inch e-Paper HAT (F);

4.2inch (400×300)

EPD_4in2_test() : Example for 4.2inch e-paper (Black/White) ;
EPD_4in2bc_test() : Example for 4.2inch e-paper B (Black/White/Red) ;
EPD_4in2b_V2_test() : Example for 4.2inch e-paper B V2 (Black/White/Red) ;

5.65inch (600×448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

EPD_5in83_test() : Example for 5.83inch e-paper (Black/White) ;
EPD_5in83_V2_test() : Example for 5.83inch e-paper V2 (Black/White) ;
EPD_5in83bc_test() : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
EPD_5in83b_V2_test() : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

EPD_7in5_test() : Example for 7.5inch e-paper (Black/White) , this version is stopped production and it can be bought before 2019-12-07 ;
EPD_7in5bc_test() : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;
EPD_7in5_V2_test() : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
EPD_7in5bc_V2_test() : Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29)

7.5inch HD

880 × 528)

EPD_7in5_HD_test(): Example for 7.5inch e-Paper HD (Black/White);
EPD_7in5b_HD_test(): Example for 7.5inch e-Paper B HD (Black/White/Red);

*Note: The above time is for reference only, please refer to the screen logo for the specific version

- Recompile, the compilation process may takes few seconds

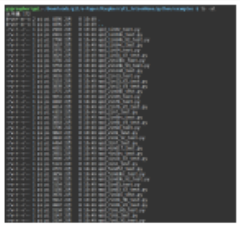
```
Return to the c directory and run the following command
make clean
make
sudo ./epd
```

Python

PS: The subsets of the Bullseye branch only supports Python3.

- Enter the folder of python code

```
cd python/examples
ls -al
```



- Run the corresponding example

```
sudo python epd_xxx_test.py
sudo python3 epd_xxx_test.py
```

Supporting type:

1.02inch (128×80) :

epd_1in02_test.py : Example for 1.02inch e-paper/1.02inch e-paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

epd_1in54_test.py : Example for 1.54inch e-paper V1 (Black/White) ; This version is stopped production which can be bought before 2019-11-22 ;
epd_1in54_V2_test.py : Example for 1.54inch e-paper V2 (Black/White); This is the current version which can be bought now (2020-07-29). The e-Paper has V2 sticker on the backside
epd_1in54b_test.py : Example for 1.54inch e-paper B (Black/White/Red) ;
epd_1in54b_V2_test.py : Example for 1.54inch e-paper B V2 (Black/White/Red) ;
epd_1in54c_test.py : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

epd_2in7_test.py : Example for 2.7inch e-paper (Black/White) ;
epd_2in7b_test.py : Example for 2.7inch e-paper B (Black.White/Red) ;
epd_2in7b_V2_test.py : Example for 2.7inch e-paper B V2 (Black.White/Red) ;

2.9inch (296×128) :

epd_2in9_test.py: Example for 2.9inch e-paper (Black/White) ;
epd_2in9_V2_test.py: Example for 2.9inch e-paper V2 (Black/White) ;
epd_2in9bc_test.py : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;
epd_2in9b_V3_test.py : Example for 2.9inch e-paper B V3 (Black/White/Red)
epd_2in9d_test.py : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

epd_2in13_test.py : Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 019-05-15;
epd_2in13_V2_test.py : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;
epd_2in13bc_test.py : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;
epd_2in13b_V3_test.py : Example for 2.13inch e-paper B V3 (Black/White/Red)
epd_2in13d_test.py : Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)

epd_2in66_test.py: Example for 2.66inch e-Paper (Black/White);
epd_2in66b_test.py: Example for 2.66inch e-Paper (Black/White/Red);

3.7inch (280 x 480)

epd_3in7_test.py: Example for 3.7inch e-Paper (Black/White);

4.01inch (640x400)

epd_4in01f_test.py: Example for 4.01inch e-Paper (SeVen-Color);

4.2inch (400×300)

epd_4in2_test.py : Example for 4.2inch e-paper (Black/White) ;
epd_4in2bc_test.py : Example for 4.2inch e-paper B (Black/White/Red) ;
epd_4in2b_V2_test.py : Example for 4.2inch e-paper B V2 (Black/White/Red) ;

5.65inch (600×448)

epd_5in65f_test.py: Example for 5.65inch e-Paper F (Seven-color);

5.83inch (600×448) :

epd_5in83_test.py : Example for 5.83inch e-paper (Black/White) ;
epd_5in83_V2_test.py : Example for 5.83inch e-paper V2 (Black/White) ;
epd_5in83bc_test.py : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
epd_5in83b_V2_test.py : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

epd_7in5_test.py : Example for 7.5inch e-paper (Black/White) , this version is stopped production and it can be bought before 2019-12-07 ;
epd_7in5_V2_test.py : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
epd_7in5bc_test.py : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stopped production and it can be bought before 2019-12-07;

epd_7in5b_V2_test.py : Example for 7.5inch e-paper B V2 (Black/White/Red) ;
This is the current version with V2 sticker on the backside. (2020-07-29);
7.5inch (HD 880 x 528)

epd_7in5_HD_test.py: Example for 7.5inch e-Paper HD (Black/White);
epd_7in5b_HD_test.py: Example for 7.5inch e-Paper B HD (Black/White/Red);

Note: The above time is for reference only, please refer to the screen logo for the specific version.

Just run the program corresponding to the screen, the program supports python2/3. Take 1.54 V2 as an example:

```
# python2
sudo python2.7 epd_1in54_V2_test.py
# python3
sudo python3 epd_1in54_V2_test.py
```

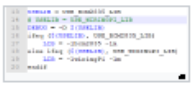
Codes Description(API)

The libraries for Raspberry Pi and Jetson Nano are same. Examples contain three parts, hardware interface, EPD driver and the GUI functions.

C

Hardware interface

We have carried out the underlying encapsulation. All the hardware platforms is different, the interface is various. If you need to know the interface, you can go to the corresponding directory. You can see many definitions in DEV_Config.c(.h). In the directory: RaspberryPi_JetsonNano\c\lib\Config



C language uses two ways to drive: BCM2835 library, WiringPi library. The BCM2835 library is used by default. If you want to use the WiringPi library to drive, you can open RaspberryPi_JetsonNano\c\Makefile and modify line 13 14

- Data type

```
#define UBYTE   uint8_t
#define UWORD   uint16_t
#define UDOUBLE uint32_t
```

- Init and Exit

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

Note: The Init() and Exit() function are used to configure GPIO、SPI and sleep mode after Exit() function is used, and the consumption of e-Paper should be 0 in sleep mode if the PCB is Rev2.1 version.

- GPIO Read/Write

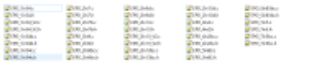
```
void DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE DEV_Digital_Read(UWORD Pin);
```

- SPI transmit data

```
void DEV_SPI_WriteByte(UBYTE Value);
```

EPD driver

The driver files are saved under RaspberryPi&JetsonNano\c\lib\e-Paper



Open .h to see the following functions:

- After the initialization, called when the screen starts working and after exiting sleep mode

```
//1.54inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 
2.13inch e-Paper (D), 2.9inch e-Paper, 2.9inch e-Paper (D)
void EPD_xxx_Init(UBYTE Mode); // Mode = 0 global refresh initialization, Mode 
= 1 local refresh initialization
//other models
void EPD_xxx_Init(void);
```

Where xxx represents the model. If it is 2.13D, the full screen initialization is EPD_2IN13D_Init(0), the partial refresh initialization EPD_2IN13D_Init(1); if it is 1.54 V2, then EPD_1IN54_V2_Init(); if it is 7.5B, it is EPD_7IN5BC_Init(), because 7.5B and 7.5C are common. The driver code, but the color displayed is different.

- Clear the screen (fill with the ink screen to white)

```
void EPD_xxx_Clear(void);
```

Where xxx represents the ink screen model. If it is 2.13D, then it is EPD_2IN13D_Clear(); if it is 7.5B, it is EPD_7IN5_Clear(); because 7.5B and 7.5C share the driver code, but the color displayed is different.

- Transfer a frame of picture data and open the display

```
//Black and white two-color ink screen
void EPD_xxx_Display(UBYTE *Image);
//Black and white red or black and white yellow ink screen
void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);
```

Note that the following are special cases:

```
//For 2.13inch e-paper (B), 2.9inch e-paper (B) two flexible screens, partial r
efresh
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9B_DisplayPart(UBYTE *Image);
```

//For 1.54inch e-paper V2 and 2.13inch e-paper V2, due to the upgrade of the control chip, for partial refresh, it is necessary to call EPD_xxx_DisplayPartBaseImage to display the basic background image, that is to perform partial refresh based on this image, and then call the dynamic EPD_xxx_DisplayPart ()

```
void EPD_1IN54_V2_DisplayPartBaseImage(UBYTE *Image);
void EPD_1IN54_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```

- Enter sleep mode

```
void EPD_xxx_Sleep(void);
```

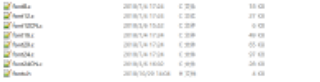
Note that after entering sleep mode, there are only two ways to work again: the first is a hardware reset, and the second is to call the initialization function again.

Where xxx represents the ink screen model. If it is 2.13D, then it is EPD_2IN13D_Sleep(); if it is 7.5B, it is EPD_7IN5BC_Sleep(); because 7.5B and 7.5C share the driver code, but the color displayed is different.

GUI functions

GUI file can be found in RaspberryPi&JetsonNano\c\lib\GUI\GUI_Paint.c(.h) directory

The fonts can be found in the RaspberryPi&JetsonNano\c\lib\Fonts directory



Create an image buffer

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWOR
D Color)
```

- Image: the image buffer
- Width: width of the image
- Height: Height of the image
- Rotate: Rotate Angle
- Color: Color of the image

Select image buffer

Select Image Buffer

```
void Paint_SelectImage(UBYTE *image)
```

- The image buffer, it is a pointer of image buffer's first address

Rotate Image

This function should be used after Paint_SelectImage()

```
void Paint_SetRotate(UWORD Rotate)
```

- Rotate: The angle rotated. It should be ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270
- Note: For different orientation, the position of the first pixel is different. Here we take 1.54inch as example

SPI-epaper-G 90.png SPI-epaper-G 180.png

SPI-epaper-G 270.png

Mirroring

```
void Paint_SetMirroring(UBYTE mirror)
```

- mirror: The type of mirroring. (MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN)

Set Pixel

This function is used to set the position and type of the pixel

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
```

- Xpoint: The x-axis coordination of pixel
- Ypoint: The y-axis coordination of pixel
- Color: The color of the pixel

Clear

This function is used to clear the e-Paper

```
void Paint_Clear(UWORD Color)
```

- Color: The color of the display

Clear window

This function is used to clear a certain area

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
```

- Xstart: The x-axis coordination of the start point
- Ystart: The y-axis coordination of the start point
- Xend: The x-axis coordination of the end point
- Yend: The y-axis coordination of the end point
- Color: The color of the window

Draw point

This function is used to draw point.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
```

- Xpoint: The x-axis coordination of point
- Ypoint: The y-axis coordination of point
- Dot_Pixel: The size of the point

```
typedef enum {
    DOT_PIXEL_1X1  = 1,   // 1 x 1
    DOT_PIXEL_2X2  ,      // 2 x 2
    DOT_PIXEL_3X3  ,      // 3 x 3
    DOT_PIXEL_4X4  ,      // 4 x 4
    DOT_PIXEL_5X5  ,      // 5 x 5
    DOT_PIXEL_6X6  ,      // 6 x 6
    DOT_PIXEL_7X7  ,      // 7 x 7
    DOT_PIXEL_8X8  ,      // 8 x 8
} DOT_PIXEL;
```

- Dot_Style: The style of the point

```
typedef enum {
  DOT_FILL_AROUND  = 1,
  DOT_FILL_RIGHTEN,
} DOT_STYLE;
```

Drawn Line

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
```

This function is used to draw a line

- Xpoint: The point's x-axis coordination of the line
- Ystart: The start y-axis coordination of the line
- Xend: The end x-axis coordination of the line
- Yend: The end y-axis coordination of the line
- Line_width: The width of the line

```
typedef enum {
    DOT_PIXEL_1X1  = 1,   // 1 x 1
    DOT_PIXEL_2X2  ,      // 2 x 2
    DOT_PIXEL_3X3  ,      // 3 x 3
    DOT_PIXEL_4X4  ,      // 4 x 4
    DOT_PIXEL_5X5  ,      // 5 x 5
    DOT_PIXEL_6X6  ,      // 6 x 6
    DOT_PIXEL_7X7  ,      // 7 x 7
    DOT_PIXEL_8X8  ,      // 8 x 8
} DOT_PIXEL;
```

- Line_style: The style of the line

```
typedef enum {
    LINE_STYLE_SOLID = 0,
    LINE_STYLE_DOTTED,
} LINE_STYLE;
```

Draw rectangle

Draw a rectangle from (Xstart, Ystart) to (Xend, Yend).

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
```

- Xstart: Start coordinate of X-axes of the rectangle
- Ystart: Start coordinate of Y-axes of the rectangle
- Xend: End coordinate of X-end of the rectangle
- Yend: End coordinate of Y-end of the rectangle
- Color: color of the rectangle
- Line_width: The width of edges. Its sides are even pixels

```
typedef enum {
    DOT_PIXEL_1X1  = 1,   // 1 x 1
    DOT_PIXEL_2X2  ,      // 2 x 2
    DOT_PIXEL_3X3  ,      // 3 x 3
    DOT_PIXEL_4X4  ,      // 4 x 4
    DOT_PIXEL_5X5  ,      // 5 x 5
    DOT_PIXEL_6X6  ,      // 6 x 6
    DOT_PIXEL_7X7  ,      // 7 x 7
    DOT_PIXEL_8X8  ,      // 8 x 8
} DOT_PIXEL;
```

- Draw_Fill: Fill the rectangle or not (empty)

```
typedef enum {
    DRAW_FILL_EMPTY = 0,
    DRAW_FILL_FULL,
} DRAW_FILL;
```

Draw circle

Draw a circle. In the image cache, with (X_Center, Y_Center) as the center, draw a circle with a radius of Radius. You can choose the color, the width of the line whether to fill the inside of the circle

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
parameter:
 X_Center: X coordinate of the center of the circle
 Y_Center: Y coordinate of the center of the circle
 Radius: the radius of the circle
 Color: fill color
 Line_width: the width of the arc, providing 8 default widths
 typedef enum {
                DOT_PIXEL_1X1 = 1, // 1 x 1
                DOT_PIXEL_2X2 , // 2 X 2
                DOT_PIXEL_3X3 , // 3 X 3
```

```
    DOT_PIXEL_6X8 , // 6 X 8
    DOT_PIXEL_7X9 , // 7 X 9
    DOT_PIXEL_8X9 , // 8 X 9
    DOT_PIXEL_7X7 , // 7 X 7
    DOT_PIXEL_8X8 , // 8 X 8
} DOT_PIXEL;
Draw_Fill: Fill, whether to fill the inside of the circle
    typedef enum {
        DRAW_FILL_EMPTY = 0,
        DRAW_FILL_FULL,
} DRAW_FILL;
```

Draw character (ASCII)

Set(Xstart Ystart) as left top point, draw a ASCII character

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* F
ont, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of the left top pixel of character;
- Ystart: Y coordinate of the left top pixel of character;
- Ascii_Char : Ascii character;
- Font: 5 fonts are available:
  - font12 : 7*12
  - font16 : 11*16
  - font20 : 14*20
  - font24 : 17*24
- Color_Foreground: color of character;
- Color_Background: color of background

Draw String

Set point (Xstart Ystart) as the left top pixel, draw a string.

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFON
T* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of left top pixel of character;
- Ystart: Y coordinate of left top pixel of characters;
- pString : Pointer of string
- Font: 5 fonts are available:
  - font8 : 5*8
  - font12 : 7*12
  - font16 : 11*16
  - font20 : 14*20
  - font24 : 17*24
- Color_Foreground: color of string
- Color_Background: color of the background

Draw Chinese characters

this function is used to draw Chinese fonts based ON GB2312 fonts.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFON
T* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: Coordinate of left top pixel of character;
- Ystart: Coordinate of left top pixel of character;
- pString : Pointer of string;
- Font: GB2312 font :
  - font12CN : 11*21(ascii) , 16*21 (Chinese)
  - font24CN : 24*41(ascii) , 32*41 (Chinese)
- Color_Foreground: color of string
- Color_Background: color of the background

Draw number

Draw a string of numbers (Xstart, Ystart) is the left top pixel

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Number, sFONT* Font, UW
ORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of left top pixel
- Ystart: Y coordinate of left top pixel
- Number: the numbers displayed, the numbers are saved in int format, the maximum is 2147483647
- Font: 5 fonts are available :
  - font8 : 5*8
  - font12 : 7*12
  - font16 : 11*16
  - font20 : 14*20
  - font24 : 17*24
- Color_Foreground: Color of font
- Color_Background: Color of background

Draw Image

Send image data of BMP file to buffer

```
void Paint_SetBitMap(const unsigned char* image_buffer)
```

Parameters:

- image_buffer: address of image data in buffer

Read local bmp picture and write it to buffer

Linux platform like Jetson Nano and Raspberry Pi support to directly operate bmp pictures. Bmp pictures can be read by Raspberry Pi & Jetson Nano :
RaspberryPi&Jetson Nano\c\lib\GUI\GUI_BMPfile.c(.h)

```
UBYTE GUI_ReadBmp(const char *path, UWORD Xstart, UWORD Ystart)
```

Parameters:

- path : The path of BMP pictures
- Xstart: X coordination of left-top of picture, default 0
- Ystart: Y coordination of left-top of picture, default 0

Testing Code

The first thing we should do is introduce the classic Linux three-way code segment, here is a little explanation of the user test code.

For Raspberry Pi and Jetson Nano, in the directory: RaspberryPi_JetsonNano\c\examples, for all test codes, multiple .h files can be read in main.c in this directory; If you need to run the 7 Sixth e-paper test program, you need to remove the 42 line .h file

```
//EPD_5in65f_test();
```

to

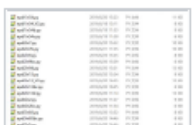
```
EPD_5in65f_test();
```

Then Compile it and run it

```
make clean
make
sudo ./epd
```

Python

For Jetson Nano\Raspberry Pi, based on python2 and python3.
For python, its drivers are not as complicated as C.
For Raspberry Pi and Jetson Nano: RaspberryPi_JetsonNano\c\python\lib\



Bottom Interface

The epdconfig.py file contains the underlying interface

- Initialize module and exit handle :

```
def module_init()
def module_exit()
```

Note:
1. The functions are used to set GPIO before and after driving e-Paper
2. If the board you have is printed with Rev2.1, module enter low-use mode after Module_Init() (as we test, the current is about 0 in this mode);

- GPIO Read/Write :

```
def  digital_write(pin, value)
def  digital_read(pin)
```

- SPI Write Byte:

```
def spi_writebyte(data)
```

Driver Interface

epdXXX.py (XXX means the size. If it is 2.13inch e-Paper, it is epd2in13.py, and so on)

- Initialize e-paper: this function should be used at the beginning. It can also be used to wake up e-Paper from Sleep mode.

```
#For 2.13inch e-Paper, 2.9inch e-Paper
def init(self, update) # Choose int_full_update or int_partial_update
#Other type
def init(self)
```

- Clear e-paper: This function is used to clear e-Paper to white

```
def Clear(self)
def Clear(self, color) # Some types of e-Paper should use this function to clear screen
```

- Convert image to arrays

```
def getbuffer(self, image)
```

- Transmit one frame of image data and display

```
#For two-color e-paper
def display(self, image)

#Because that controllers of 2.13inch e-paper are updated, when partial refresh is, they should first use displayPartBaseImage() to display static background, then use displayPart() to dynamically display.
def displayPartBaseImage(self, image)
def displayPart(self, image)
```

- Enter sleep mode

```
def sleep(self)
```

Testing Function

epd_xxx_test.py (xxx means dimension)python examples are saved in directory :

Raspberry Pi and Jetson Nano - RaspberryPi&JetsonNano\python\examples\

If the python installed in your OS is python2, you should run examples like below :

```
sudo python epd_2in13_V2_test.py
```

If it is python3, the commands should be:

```
sudo python3 epd_2in13_V2_test.py
```

Orientation

To rotate the display, you can use the image function like: blackimage = black.transpose(Image.ROTATE_270)

```
blackimage = blackimage.transpose(Image.ROTATE_270)
redimage = redimage.transpose(Image.ROTATE_270)
#Support ROTATE_90, ROTATE_180, ROTATE_270
```

The rotation effect, taking 1.54B as an example, is 0°, 90°, 180°, 270° in sequence:

GUI

Python has a powerful PIL library, which can be used directly to drawing, flashes. We recommend it for drawing.

- Install the library firstly

```
sudo apt-get install python3-pil
```

Import the library

```
from PIL import Image,ImageDraw,ImageFont
```

Image: BMP/JPG; ImageDraw: drawing shapes; ImageFont: fonts

- Set image black for drawing

```
image = Image.new('1', (epd.width, epd.height), 255)  # 255: clear the frame
```

The first parameter is the depth of color, 1 means 2 grayscale. The second parameter is a tuple of image size. The third parameter is color of the image, 0 is black and 255 is white.

- Create an image object:

```
draw = ImageDraw.Draw(image)
```

- Draw rectangle

```
draw.rectangle((0, 10, 200, 34), fill = 0)
```

The first parameter is a tuple of coordinates. (0, 10) is the top-left point of rectangle, (200, 34) is the right-bottom point. fille = 0 set the filled color to black

- Draw line

```
draw.line((16, 60, 56, 60), fill = 0)
```

The first parameter is a type of coordinates. (16, 60) is the begin point, (200, 34) is the end point, fill=0 set the line to black

- Draw circle

```
draw.arc((90, 40, 150, 120), 0, 360, fill = 0)
```

This function is used to draw a circle of a square. The first parameter is a tuple of coordination of the square, the degree of the circle is 0 to 360 °, fille=0 set the circle to black.
If the figure is not square according to the coordination, you will get an ellipse.
Besides the arc function, you can also use the chord function for drawing solid circle.

```
draw.chord((90, 130, 150, 180), 0, 360, fill = 0)
```

The first parameter is the coordination of the enclosing rectangle. The second

And the parameter 3 is the distance of the bending rectangle. The fourth and fifth parameter are the beginning and degree of the circle. The fourth parameter is the fill color of the circle

- Character

You can directly import ImageFont model for drawing characters:

```
font = ImageFont.truetype(os.path.join(picdir, 'Font.ttc'), 24)
```

You can use the font of Windows or other font with .ttf in the format.

To draw English character, you can directly use the fonts; for Chinese character, you need to add some symbols:

```
draw.text((0, 12), 'hello world', font = font, fill = 255)
draw.text((0, 36), u'微雪电子', font = font, fill = 0)
```

The first parameter is a tuple of coordination of character, the second parameter is the font and the one is to set the color.

- Read local picture

```
image = Image.open(os.path.join(picdir, '125x64.bmp'))
```

The parameter is the path of picture.

- Other functions

For more information about the PIL library, you can search online.

User Guides of Jetson Nano

Hardware connection

Because the 40PIN GPIO of Jetson Nano is compatible with Raspberry Pi and the Jetson GPIO library is compatible with RPi.GPIO, therefore we use the same connection as Raspberry Pi.

To connect the e-Paper, you can following the table below.

Jetson Nano Connection

e-Paper	Jetson Nano Developer Kit	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	10(SPI0_MOSI)	19
CLK	11(SPI0_SCK	23
CS	8(SPI0_CS0)	24
DC	25	22
RST	17	11
BUSY	24	18

Software Setting

Install Libraries

Open the terminal of Jetson Nano and run the following commands to install corresponding libraries:

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install Jetson.GPIO
sudo groupadd -f -r gpio
sudo usermod -a -G gpio your_user_name
sudo cp /opt/nvidia/jetson-gpio/etc/99-gpio.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && sudo udevadm trigger
```

- Install I2C libraries

```
sudo apt-get install python3-smbus
```

- Install PIL library

```
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
```

Download Testing Program

Open a linux terminal and execute:

Method 1: Download it on our official website. It is recommended to use it.

```
sudo apt-get install p7zip-full
sudo wget  https://www.waveshare.net/w/upload/2/28/E-Paper_code.7z
7z x E-Paper_code.7z -O./e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

Method 2: Using the github repository, there may be a delay in the update of github. It is recommended to use method 1

```
sudo git clone https://github.com/waveshare/e-Paper
cd e-Paper/RaspberryPi_JetsonNano/
```

Run the Test Program

Please execute the following commands under RaspberryPi_JetsonNano, otherwise, the directory will not be indexed.

C

- Modify the main.c file for corresponding e-Paper

```
cd c
sudo nano examples/main.c
```

If you are using 2.7inch e-paper, then remove the // in front of the EPD_2IN7_test() function, then press ctrl+c, then press Y and press Enter to save and exit, as shown:

Supporting Type:

1.02inch (128×80) :

EPD_1in02d_test() : Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

EPD_1in54_test() : Example for 1.54inch e-paper V1 (Black/White) : The version is stopped production which can be bought before 2019-11-22 ;

EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version that can be bought now (2020-07-29). The e-Paper has V2 sticker on the back side.

EPD_1in54b_test() : Example for 1.54inch e-paper B (Black/White/Red) ;

EPD_1in54b_V2_test(): Example for 1.54inch e-Paper B V2 (Black/White/Red);

EPD_1in54c_test() : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

EPD_2in7_test() : Example for 2.7inch e-paper (Black/White) ;

EPD_2in7b_test() : Example for 2.7inch e-paper B (Black/White/Red) ;

EPD_2in7b_V2_test() : Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;

EPD_2in9_V2_test(): Example for 2.9inch e-paper V2 (Black/White) ;

EPD_2in9bc_test() : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;

EPD_2in9b_V3_test() : Example for 2.9inch e-paper B V3 (Black/White/Red) ;

EPD_2in9d_test() : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

EPD_2in13_test() : Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 019-05-15;

EPD_2in13_V2_test() : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;

EPD_2in13_V3_test() : Example for 2.13inch e-paper V3 (Black/White) This is the current version with sticker V3 on the backside (2020-07-29) ;

EPD_2in13bc_test() : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;

EPD_2in13b_V3_test() : Example for 2.13inch e-paper B V3 (Black/White/Red)

EPD_2in13d_test() : Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)

EPD_2in66_test(): Example for 2.66inch e-Paper (Black/White)

EPD_2in66b_test(): Example for 2.66inch e-Paper B (Black/White/Red)

3.7inch (280 x 480)

EPD_3in7_test(): Example for 3.7inch e-Paper(Black/White);

4.01inch (640x400)

EPD_4in01_test(): Example for the 4.01inch e-Paper HAT (F);

4.2inch (400x300)

EPD_4in2_test() : Example for 4.2inch e-paper (Black/White) ;
EPD_4in2B...test() : Example for 4.2inch e-paper B (Black/White/Red) ;
EPD_4in2b_V2_test() : Example for 4.2inch e-paper B V2 (Black/White/Red) ;

5.65inch (600x448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-Color);

5.83inch (600x448) :

EPD_5in83_test() : Example for 5.83inch e-paper (Black/White) ;
EPD_5in83_V2_test() : Example for 5.83inch e-paper V2 (Black/White) ;
EPD_5in83B...test() : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
EPD_5in83b_V2...test() : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

EPD_7in5_test() : Example for 7.5inch e-paper (Black/White) , this version is Mapped production and it can be bought before 2019-12-07 ;
EPD_7in5bc_test() : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 Version is Mapp production and it can be bought before 2019-12-07;
EPD_7in5_V2...test() : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
EPD_7in5bc_V2...test() : Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

7.5inch(HD880 x 528)

EPD_7in5_HD_test(): Example for 7.5inch e-Paper HD (Black/White);
EPD_7in5b_HD...test(): Example for 7.5inch e-Paper B HD (Black/White/Red);

*Note: The above time is for reference only, please refer to the screen logo for the type of the version

- Recompile, the compilation process may takes few seconds

```
Return to the c directory and run the following command
make clean
make
sudo ./epd
```

Python

Enter the python program directory and run the command # -*-

```
cd python/examples
ls -al
```

Support Type

1.02inch (128×80) :

epd_1in02_test.py : Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

epd_1in54_test.py : Example for 1.54inch e-paper V1 (Balck/White) : This version is Mapped production which can be bought before 2019-11-22 ;
epd_1in54_V2_test.py : Example for 1.54inch e-paper V2 (Black/White): This is the current version which can be bought now (2020-07-29). The e-Paper has V2 sticker on the backside.
epd_1in54b_test.py : Example for 1.54inch e-paper B (Black/White/Red) ;
epd_1in54b_V2_test.py : Example for 1.54inch e-paper B V2 (Black/White/Red) ;
epd_1in54c_test.py : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

epd_2in7_test.py : Example for 2.7inch e-paper (Black/White) ;
epd_2in7b_test.py : Example for 2.7inch e-paper B (Black/White/Red) ;
epd_2in7b_V2_test.py : Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128) :

epd_2in9_test.py: Example for 2.9inch e-paper (Black/White) ;
epd_2in9_V2_test.py: Example for 2.9inch e-paper V2 (Black/White) ;
epd_2in9bc_test.py : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;
epd_2in9b_V3_test.py : Example for 2.9inch e-paper B V3 (Black/White/Red)
epd_2in9d_test.py : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

epd_2in13_test.py : Example for 2.13inch e-paper V1 (Black/White) , this version is Mapped production and it can be bought before 2019-09-15
epd_2in13_V2_test.py : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;
epd_2in13bc_test.py : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;
epd_2in13b_V3_test.py : Example for 2.13inch e-paper B V3 (Black/White/Red)
epd_2in13d_test.py : Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)

epd_2in66_test.py: Example for 2.66inch e-Paper (Black/White);
epd_2in66b_test.py: Example for 2.66inch e-Paper (Black/White/Red);

3.7inch (280 x 480)

epd_3in7_test.py: Example for 3.7inch e-Paper (Black/White);

4.01inch (640x400)

epd_4in01f_test.py: Example for 4.01inch e-Paper (Seven-color);

4.2inch (400x300)

epd_4in2_test.py : Example for 4.2inch e-paper (Black/White) ;
epd_4in2B...test.py : Example for 4.2inch e-paper B (Black/White/Red) ;
epd_4in2b_V2_test.py : Example for 4.2inch e-paper B V2 (Black/White/Red) ;

5.65inch (600x448)

epd_5in65f_test.py: Example for 5.65inch e-Paper F (Seven-color)

5.83inch (600x448) :

epd_5in83_test.py : Example for 5.83inch e-paper (Black/White) ;
epd_5in83_V2_test.py : Example for 5.83inch e-paper V2 (Black/White) ;
epd_5in83bc_test.py : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
epd_5in83b_V2_test.py : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

epd_7in5_test.py : Example for 7.5inch e-paper (Black/White) , this version is Mapped production and it can be bought before 2019-12-07 ;
epd_7in5_V2_test.py : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
epd_7in5bc_test.py : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 Version is Mapped production and it can be bought before 2019-12-07;
epd_7in5b_V2...test.py : Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29);

7.5inch (HD 880 x 528)

epd_7in5_HD_test.py: Example for 7.5inch e-Paper HD (Black/White);
epd_7in5b_HD...test.py: Example for 7.5inch e-Paper B HD (Black/White/Red);

*Note: The above time is for reference only, please refer to the screen logo for the type of the version

- Just run the program corresponding to the screen, the program supports python2/3. Like 1.54 V2 is an example

```
# python2
sudo python2.7 epd_1in54_V2_test.py
# python3
sudo python3 epd_1in54_V2_test.py
```

Description of codes (API)

The libraries for Raspberry Pi and Jetson Nano are the same. Examples contain these parts, hardware interfaces, EPD drivers and the GUI functions.

C

Hardware Interfaces

Two libraries are used by C example, WiringPi and BCM2835. The codes use wiringPi by default. If you want to use BCM2835, you can modify the RaspberryPi/JetsonNano/c/Makefile file, modify lines 13 and 14. Change it as below:

```
13  #USELIB = USE_WIRINGPI_LIB
14  USELIB = USE_BCM2835_LIB
15  debug:$(TARGET)
16      $(CC) -D $(USELIB) -g -O -Wall -lrt -o $@ $(SRC) $(LIB)
17  release:$(TARGET)
18      $(CC) -D $(USELIB) -O3 -Wall -lrt -o $@ $(SRC) $(LIB)
19  clean:
20      rm -f $(TARGET)
```

- DEV Type

```
#define UBYTE   uint8_t
#define UWORD   uint16_t
#define UDOUBLE uint32_t
```

- Init and Exit

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

Note: The Init() and Exit() function must be used to configure GPIO. EPD enter sleep mode after Exit() function is used, and the consumption of e-Paper should be 0 in sleep mode if the PCB is Rev2.1 version.

- GPIO Read/Write

```
void DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE DEV_Digital_Read(UWORD Pin);
```

- SPI transmit data

```
void DEV_SPI_WriteByte(UBYTE Value);
```

EPD

e-paper driver code file, in the directory: RaspberryPi\JetsonNano\c\lib\e-Paper

As shown below:

Open .h to see the following functions:

- After the initialization, called when the screen is working and after exiting sleep mode

```
//2.13inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 
2.15inch e-Paper (D), 2.9inch e-Paper, 2.9inch e-Paper (D)
void EPD_xxx_Init(UBYTE Mode); // Mode : 0 Global refresh Initialization, Mode 
: 1 Local refresh Initialization
//Other models
void EPD_xxx_Init(void);
```

Where xxx represents the model. If it is 2.13D, the full screen initialization is EPD_2IN13D_Init(0); the partial refresh initialization EPD_2IN13D_Init(1); if it is 1.54 V2, then EPD_1IN54_V2_Init(); if it is 7.5B, it is EPD_7IN5BC_Init(), because 7.5B and 7.5C are common. The driver code but the color displayed is different.

- Clear the screen, that is the ink screen to white

```
void EPD_xxx_Clear(void);
```

Where xxx represents the ink screen model. If it is 2.13D, then it is EPD_2IN13D_Clear(); if it is 7.5B, it is EPD_7IN5_Clear(); because 7.5B and 7.5C share the driver code, but the color displayed is different.

- Transfer a frame of picture data and open the display

```
//Black and white two-color ink screen
void EPD_xxx_Display(UBYTE *Image);
//Black and white and red or black and white yellow ink screen
void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);
```

Note that the following are special cases:

```
//For 2.13inch e-paper (D), 2.9inch e-paper (D) two flexible screens, partial r
efresh
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

//For 1.54inch e-paper V2 and 2.13inch e-paper V2, due to the upgrade of the control chip for partial refresh, it is necessary to call EPD_xxx_DisplayPartBaseImage to display the static background image, that is, to perform partial refresh based on the image, and then call the dynamic EPD_xxx_DisplayPart ()

```
void EPD_1IN54_V2_DisplayPartBaseImage(UBYTE *Image);
void EPD_1IN54_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```

- Enter sleep mode

```
void EPD_xxx_Sleep(void);
```

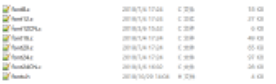
Note that after entering sleep mode, there are only two ways to work again: the first is a hardware reset, and the second is to call the initialization function again.

Where xxx represents the ink screen model. If it is 2.13D, then it is EPD_2IN13D_Sleep(); if it is 7.5B, it is EPD_7IN5BC_Sleep(); because 7.5B and 7.5C share the driver code, but the color displayed is different.

GUI Functions

GUI file can be found in Raspberry Pi\JetsonNano\c\lib\GUI\GUI_Paint.c(.h) directory

The fonts can be found in Raspberry Pi\JetsonNano\c\lib\Fonts directory



Create an image buffer

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWOR
D Color)
```

- Image: the image buffer
- Width: width of the image
- Height: Height of the image
- Rotate: Rotate Angle
- Color: Color of the image

Select image buffer

```
void Paint_SelectImage(UBYTE *image)
```

- The image buffer, it is a pointer of image buffer's first address

Rotate Image

This function should be used after Paint_SelectImage()

```
void Paint_SetRotate(UWORD Rotate)
```

- Rotate: The Angle rotated. It should be ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270
- Note: For different orientation, the position of the first pixel is different, here we take 1.54inch as example

SPI-epaper-C 90.png SPI-epaper-C 180.png

SPI-epaper-C 270.png

Mirroring

```
void Paint_SetMirroring(UBYTE mirror);
```

- mirror: The type of mirroring. (MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN)

Set Pixel

This function is used to set the position and type of the pixel

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color);
```

- Xpoint: The x coordinate of pixel
- Ypoint: The y coordinate of pixel
- Color: The color of the pixel

Clear

This function is used to clear the e-Paper

```
void Paint_Clear(UWORD Color);
```

- Color: The color of the display

Clear window

This function is used to clear a certain area

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend,
UB Color);
```

- Xstart: The x coordinate of the start point
- Ystart: The y coordinate of the start point
- Xend: The x coordinate of the end point
- Yend: The y coordinate of the end point
- Color: The color of the window

Draw point

This function is used to draw points

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pix
el, DOT_STYLE Dot_Style)
```

- Xpoint: The x coordinate of point
- Ypoint: The y coordinate of point
- Dot_Pixel: The size of the point

```
typedef enum {
    DOT_PIXEL_1X1 = 1,   // 1 x 1
    DOT_PIXEL_2X2 ,      // 2 x 2
    DOT_PIXEL_3X3 ,      // 3 x 3
    DOT_PIXEL_4X4 ,      // 4 x 4
    DOT_PIXEL_5X5 ,      // 5 x 5
    DOT_PIXEL_6X6 ,      // 6 x 6
    DOT_PIXEL_7X7 ,      // 7 x 7
    DOT_PIXEL_8X8 ,      // 8 x 8
} DOT_PIXEL;
```

- Dot_Style: The style of the point

```
typedef enum {
  DOT_FILL_AROUND  = 1,
  DOT_FILL_RIGHTED,
} DOT_STYLE;
```

Drawn Line

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD C
olor, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
```

This function is used to draw a line

- Xstart: The start x-axis coordination of the line
- Ystart: The start y-axis coordination of the line
- Xend: The end x-axis coordination of the line
- Yend: The end y-axis coordination of the line
- Line_width: The width of the line

```
typedef enum {
    DOT_PIXEL_1X1 = 1,   // 1 x 1
    DOT_PIXEL_2X2 ,      // 2 x 2
    DOT_PIXEL_3X3 ,      // 3 x 3
    DOT_PIXEL_4X4 ,      // 4 x 4
    DOT_PIXEL_5X5 ,      // 5 x 5
    DOT_PIXEL_6X6 ,      // 6 x 6
    DOT_PIXEL_7X7 ,      // 7 x 7
    DOT_PIXEL_8X8 ,      // 8 x 8
} DOT_PIXEL;
```

- Line_style: The style of the line

```
typedef enum {
    LINE_STYLE_SOLID = 0,
    LINE_STYLE_DOTTED,
} LINE_STYLE;
```

Draw rectangle

Draw a rectangle from (Xstart, Ystart) to (Xend, Yend).

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UW
ORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
```

- Xstart: Start coordinate of X-axes of the rectangle
- Ystart: Start coordinate of Y-axes of the rectangle
- Xend: End coordinate of X-end of the rectangle
- Yend: End coordinate of Y-end of the rectangle
- Color: color of the rectangle
- Line_width: The width of edges. 8 styles are available:

```
typedef enum {
    DOT_PIXEL_1X1 = 1,   // 1 x 1
    DOT_PIXEL_2X2 ,      // 2 x 2
    DOT_PIXEL_3X3 ,      // 3 x 3
    DOT_PIXEL_4X4 ,      // 4 x 4
    DOT_PIXEL_5X5 ,      // 5 x 5
    DOT_PIXEL_6X6 ,      // 6 x 6
    DOT_PIXEL_7X7 ,      // 7 x 7
    DOT_PIXEL_8X8 ,      // 8 x 8
} DOT_PIXEL;
```

- Draw_Fill: Set the rectangle fill or empty

```
typedef enum {
    DRAW_FILL_EMPTY = 0,
    DRAW_FILL_FULL,
} DRAW_FILL;
```

- **Draw a circle:** In the image circle with (X_Center Y_Center) as the center, draw a circle with a radius of Radius, you can choose the color, the width of the line, whether to fill the inside of the circle

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Colo
r, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
parameter:
        X_Center: X coordinate of the center of the circle
        Y_Center: Y coordinate of the center of the circle
        Radius: the radius of the circle
        Color: fill color
        Line_width: the width of the arc, providing 8 default widths
                typedef enum {
                                DOT_PIXEL_1X1 = 1, // 1 x 1
                                DOT_PIXEL_2X2 , // 2 x 2
                                DOT_PIXEL_3X3 , // 3 x 3
                                DOT_PIXEL_4X4 , // 4 x 4
                                DOT_PIXEL_5X5 , // 5 x 5
                                DOT_PIXEL_6X6 , // 6 x 6
                                DOT_PIXEL_7X7 , // 7 x 7
                                DOT_PIXEL_8X8 , // 8 x 8
} DOT_PIXEL;
Draw_Fill: Fill, whether to fill the inside of the circle
                typedef enum {
                                DRAW_FILL_EMPTY = 0,
                                DRAW_FILL_FULL,
} DRAW_FILL;
```

Draw character (ASCII)

Set(Xpoint, Ypoint) as left top point, draw a ASCII character

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* F
ont, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of the left top pixel of character;
- Ystart: Y coordinate of the left top pixel of character;
- Ascii_Char : Ascii character;
- Font: 5 fonts are available:
  font12 : 7*12
  font16 : 11*16

[pdf] Guide

User Guide waveshare 4 2inch E Ink Display Module Compatible with Raspberry Pi 5 4B 3B Zero W 2W Pico WH 400x300 Resolution Three Color SPI Interface Electronics B1rjRAV29OL m media amazon images I |||

4.2inch e-Paper Module B Overview 4.2inch e-Paper B Version As the 4.2inch e-Paper B raw panel was updated to V2, the controller and the driver codes are different and the codes of the two versions are not compatible with each other. Except for the software, the outline of the two versi...

lang:en score:27 filesize: 3.63 M page_count: 1 document date: 2022-06-09

font20 : 14*20

font24 : 17*24

- Color_Foreground: color of character;
- Color_Background: color of background

Draw String

Set point (Xstart, Ystart) as the left top pixel, draw a string

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- Xstart: X coordinate of left top pixel of characters;
- Ystart: Y coordinate of left top pixel of characters;
- pString : Pointer of string
- Font: 5 fonts are available:

  font8 : 5*8

  font12 : 7*12

  font16 : 11*16

  font20 : 14*20

  font24 : 17*24

- Color_Foreground: color of string
- Color_Background: color of the background

Draw Chinese characters

this function is used to draw Chinese font based ON GB2312 font.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: Coordinate of left top pixel of characters;
- Ystart: Coordinate of left top pixel of characters;
- pString : Pointer of string;
- Font: GB2312 font :

  font12CN : 11*21(ascii) . 16*21 (Chinese)

  font24CN : 24*41(ascii) . 32*41 (Chinese)

- Color_Foreground: color of string
- Color_Background: color of the background

Draw number

Draw a string of numbers (Xstart, Ystart) in the left top pixel

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Number, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameter:

- Xstart: X coordinate of left top pixel
- Ystart: Y coordinate of left pixel
- Number: the number display. the number is saved in int form, the maximum is 2147483647
- Font: 5 fonts are available :

  font8 : 5*8

  font12 : 7*12

  font16 : 11*16

  font20 : 14*20

  font24 : 17*24

- Color_Foreground: color of font
- Color_Background: color of background

Draw Image

Send image data of BMP file to buffer

```
void Paint_DrawBitMap(const unsigned char* image_buffer)
```

Parameter:

- image_buffer: address of image data in buffer

Read local bmp picture and write it to buffer

Linux platform like Jetson Nano and Raspberry Pi support to directly operate bmp pictures. Raspberry Pi & Jetson Nano :

RaspberryPi_JetsonNano\c\lib\GUI\GUI_BMPfile.c(.h)

```
UBYTE GUI_ReadBmp(const char *path, UWORD Xstart, UWORD Ystart)
```

Parameters:

- path : The path of BMP pictures
- Xstart: X coordination of left top of picture, default 0
- Ystart: Y coordination of left top of picture, default 0

Testing Code

The first three chapters introduce the CMAKE Linux three-layer code structure, here is a little explanation of the underlying code

For RaspberryPi and Jetson Nano, in the directory: RaspberryPi_JetsonNano\c\examples, for all test codes, multiple shields can be made in main.c in this directory;

If you need to run the 7.5inch e-paper test program, you need to remove the 42 line shield

```
// EPD_7in5_test();
```

change to

```
EPD_7in5_test();
```

Re-execute in linux command mode as follows:

```
make clean
make
sudo ./epd
```

Python

For Jetson Nano\Raspberry Pi, based on python2.7 and python3, python is easy to use the .c codes

RaspberryPi and Jetson Nano : RaspberryPi&JetsonNano\c\python\lib\

Bottom Interface

The epdconfig.py file in the underlying interface

- Initialize module and exit handle :

```
def module_init()
def module_exit()
```

Note:

1. The functions are used to set GPIO before and after driving e-Paper
2. If the board you have is printed with Rev2.1, the module enter low-power mode after Module_Init(), (as we test, the current is about 0 in this mode);

- GPIO Read/Write :

```
def  digital_write(pin, value)
def  digital_read(pin)
```

- SPI Write dMF

```
def spi_writebyte(data)
```

Driver Function

epdXXX.py (XXX means size. If it is 2.13inch e-paper, it is epd2in13.py. And so on)

- InitMethod e-paper: this function should be used at the beginning. It can also be used to wake up e-Paper from Sleep mode.

```
#For 2.13inch e-Paper, 2.9inch e-Paper
def init(self, update):  #  Choose to full update or let_partial_update
#Other type
def init(self)
```

- Clear e-paper: This function is used to clear e-Paper to white

```
def Clear(self)
def Clear(self, color)  # Some types of e-Paper should use this function to clea
r screen
```

- Convert image to arrays

```
def getbuffer(self, image)
```

- Transmit one frame of image data and display

```
#For two-color e-paper
def display(self, image)

#Because that controllers of 2.13inch e-paper are updated, when partial refres
h, they should first use displayPartBaseImage() to display static background, t
hen use displayPart() to dynamically display.
def displayPartBaseImage(self, image)
def displayPart(self, image)
```

- Enter sleep mode

```
def sleep(self)
```

Testing Function

epd_xxx_test.py (xxx means size. If it is 2.13inch e-paper, it is epd_2in13_test.py. And so on)
python is in the following directory:
/RPi and JetsonNano/: /RPi and JetsonNano/python/example/

If your python version is python2, and you need to run the 7.5inch e-paper test program, re-execute it in the linux command mode as follows:

```
sudo python epd_7in5_test.py
```

If your python version is python3, and you need to run the 7.5inch e-paper test program, re-execute it in the linux command mode as follows:

```
sudo python3 epd_7in5_test.py
```

Orientation

To rotate the display, you can use transpose function like blackimage = blackimage.transpose(Image.ROTATE_270):

```
blackimage = blackimage.transpose(Image.ROTATE_270)
redimage = redimage.transpose(Image.ROTATE_270)
#Support ROTATE_90, ROTATE_180, ROTATE_270
```

GUI

Python has a powerful PIL library [4], which can be used directly to drawing figures. Here we use it for drawing

- Install the library firstly

```
sudo apt-get install python3-pil
```

Import the library

```
from PIL import Image,ImageDraw,ImageFont
```

Image: library; ImageDraw: drawing function; ImageFont: font

- Set image buffer for drawing.

```
image = Image.new('1', (epd.width, epd.height), 255)  # 255: clear the frame
```

The first parameter is the depth of color, 1 means 2 grayscale. The second parameter is a tuple of image size. The third parameter is color of the image. 0 is black and 255 is white.

- Create an image object.

```
draw = ImageDraw.Draw(image)
```

- Draw rectangle

```
draw.rectangle((0, 10, 200, 34), fill = 0)
```

The first parameter is a tuple of coordination. 0, 10 is the top-left point of rectangle. 200, 34 is the right-bottom point. fille = 0 set the filled color to black.

- Draw line

```
draw.line((16, 60, 56, 60), fill = 0)
```

The first parameter is a type of coordination. 16, 60 is the beginning point; 200, 34 is the endpoint. fill=0 set the line to black.

- Draw circle

```
draw.arc((90, 40, 150, 100), 0, 360, fill = 0)
```

This function is used to draw a excircle of a square. The first parameter is a tuple of coordination of the square. the degree of the circle is 0 to 360 °. fille=0 set the circle to black.
If the figure is not square according to the coordination. You will get an ellipse .

Besides the arc function, you can also use the chord function for drawing a solid circle.

```
draw.chord((90, 130, 150, 190), 0, 360, fill = 0)
```

The first parameter is the coordination of the enclosing rectangle. The second and third parameters are the beginning and end degree of the circle. The fourth parameter is the fill color of the circle.

- Character

You can directly import ImageFont model for drawing characters:

```
font = ImageFont.truetype(os.path.join(picdir, 'Font.ttc'), 24)
```

You can use the fonts of Windows or other fonts which is in ttc formats.

To draw English character, you can directly use the fonts; for Chinese character, You need to add a Symbol of

```
draw.text((8, 12), 'Hello world', font = font, fill = 255)
draw.text((8, 36), '微雪电子', font = font, fill = 0)
```

The first parameter is a tuple of 2 elements, with (8, 12) as the left vertex, fill is the font color and the fifth parameter fill=255 so it looks like it will not be

displayed. The second sentence shows 亚博电子.

- Read local picture

```
image = Image.open(os.path.join(pic_dir, '1in54.bmp'))
```

The parameter 0 is the path of picture.

- Other functions

For more information about the PIL library, you can search online

User Guides of STM32

Hardware connection

The demo codes we provided are based on STM32F103ZET6, the connection table is also based on STM32F103ZET6. If you want to use other chip, you need to port the codes and change the connection according to actual situation.

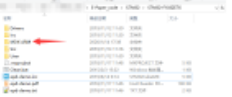
Connect to STM32F103ZET

e-Paper	STM32
Vcc	3.3V
GND	GND
DIN	PA7
CLK	PA5
CS	PA4
DC	PA2
RST	PA1
BUSY	PA3

If the STM32 board you have is STM32F103R8, you can refer to the guides of E-Paper Shield

Software settings

The Codes are based on HAL library. Download the Codes and the project files are saved under the STM32\STM32-F103ZET6\MDK-ARM directory.



Modify epconfig.h, define the line according to the e-paper type and recompile project and download

Supporting types

1.02inch (128×80) :

EPD_1in02d_test() : Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

EPD_1in54_test() : Example for 1.54inch e-paper V1 (Black/White) : This version is stopped production which can be bought before 2019-11-22 ;
EPD_1in54_V2_test(): Example for 1.54inch e-paper V2 (Black/White): This is the current version that can be bought now (2020-07-29). The e-Paper has V2 sticker on the backside
EPD_1in54b_test() : Example for 1.54inch e-paper B (Black/White/Red) ;
EPD_1in54b_V2_test(): Example for 1.54inch e-Paper B V2 (Black/White/Red);
EPD_1in54c_test() : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

EPD_2in7_test() : Example for 2.7inch e-paper (Black/White) ;
EPD_2in7b_test() : Example for 2.7inch e-paper B (Black/White/Red) ;
EPD_2in7b_V2_test() : Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128) :

EPD_2in9_test(): Example for 2.9inch e-paper (Black/White) ;
EPD_2in9_V2_test(): Example for 2.9inch e-paper V2 (Black/White) ;
EPD_2in9bc_test() : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;
EPD_2in9b_V3_test() : Example for 2.9inch e-paper B V3 (Black/White/Red) ;
EPD_2in9d_test() : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

EPD_2in13_test() : Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 019-05-15;
EPD_2in13_V2_test() : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;
EPD_2in13_V3_test() : Example for 2.13inch e-paper V3 (Black/White) This is the current version with sticker V3 on the backside (2020-07-29) ;
EPD_2in13bc_test() : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;
EPD_2in13b_V3_test() : Example for 2.13inch e-paper B V3 (Black/White/Red)
EPD_2in13d_test() : Example for 2.13inch e-paper D (Black/White) ;

2.66inch (152 x 296)

EPD_2in66_test(): Example for 2.66inch e-Paper (Black/White)
EPD_2in66b_test(): Example for 2.66inch e-Paper B (Black/White/Red);

3.7inch (280 x 480)

EPD_3in7_test(): Example for 3.7inch e-Paper(Black/White);

4.01inch (640x400)

EPD_4in01_test(): Example for the 4.01inch e-Paper HAT (F);

4.2inch (400×300)

EPD_4in2_test() : Example for 4.2inch e-paper (Black/White) :
EPD_4in2bc_test() : Example for 4.2inch e-paper B (Black/White/Red) :
EPD_4in2b_V2_test() : Example for 4.2inch e-paper B V2 (Black/White/Red) :

5.65inch (600x448)

EPD_5in65f_test(): Example for for 5.65inch e-Paper F (Seven-Color):

5.83inch (600×448) :

EPD_5in83_test() : Example for 5.83inch e-paper (Black/White) :
EPD_5in83_V2_test() : Example for 5.83inch e-paper V2 (Black/White) :
EPD_5in83bc_test() : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
EPD_5in83b_V2_test() : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

EPD_7in5_test() : Example for 7.5inch e-paper (Black/White) , this Version is stopped production and it can be bought before 2019-12-07 :
EPD_7in5bc_test() : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) . 7.5inch e-paper B V1 version is stop production and it can be bought before 2019-12-07;
EPD_7in5_V2_test() : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
EPD_7in5b_V2_test() : Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside. (2020-07-29)

7.5inch FHD

880 × 528)

EPD_7in5_HD_test(): Example for 7.5inch e-Paper HD (Black/White);
EPD_7in5b_HD_test(): Example for 7.5inch e-Paper B HD (Black/White/Red);

Note: The above time is for reference only, please refer to the screen logo for the specific version

For example, 1.54inch e-Paper Module. Open the epd1in54 folder and run the epd1in54.ino file.

Open the program, select the development board model Arduino UNO

Codes description

Bottom hardware interface

We package the bottom for different hardware platforms.
You can check the DEV_Config.c(.h) file which is located in \STM32\STM32-F103ZET6\User\Config

- DEV Type:

```
#define UBYTE   uint8_t
#define UWORD   uint16_t
#define UDOUBLE uint32_t
```

- Module initialized and exit:

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
```

Note: 1.The functions are used to set GPIO before and after driving e-Paper.
2. If the board is powered with 5V, it is initialized with 5V/3.3V, the module enters low-power mode after DEV_Module_Exit() (as we test, the current is about 0 in this mode);

- GPIO Read/Write :

```
void DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE DEV_Digital_Read(UWORD Pin);
```

- SPI Write data

```
void DEV_SPI_WriteByte(UBYTE Value);
```

Middle EPD driver

The epd driver file is saved in: STM32\STM32-F103ZET6\User\e-Paper

Open .h file, function file data can be read.

- Initialization: It should be used to initialize e-Paper or wake up e-Paper from sleep mode.

```
//2.13inch e-Paper, 1.54inch e-Paper V2, 2.13inch e-Paper, 2.13inch e-Paper V2, 2.13inch e-Paper (B), 2.90inch e-Paper, 2.9inch e-Paper (B)
void EPD_xxx_Init(UBYTE Mode); //Mode = 0 Initialize full refresh; Mode = 1 Initialize partial refresh.
//Other type
void EPD_xxx_Init(void);
```

xxx is the type of e-paper

- Clear display: This function is used to clear the e-paper to white

```
void EPD_xxx_Clear(void);
```

xxx is the type of e-Paper.

- Transmit image data and display

```
//Black/White e-Paper
void EPD_xxx_Display(UBYTE *Image);

Because controllers of 1.54inch e-paper V2 and 2.13inch e-paper V2 have updated, you need to use EPD_xxx_DisplayPartBaseImage to display static image and then use EPD_xxx_DisplayPart() to dynamic display when partial refreshing.
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```

- Enter sleep mode

```
void EPD_xxx_Sleep(void);
```

Note, You should hardware reset or use initialize function to wake up e-Paper from sleep mode
xxx is the type of e-Paper

Application function

GUI drawing function are provided here. You can find them in \STM32\STM32-F103ZET6\User\GUI\GUI_Paint.c(.h)

The fonts are saved in the directory: \STM32\STM32-F103ZET6\User\Fonts

- Create a new image buffer: This function is used to create a new image with width, height, Rotate degree and its color.

```
void Paint_NewImage(UBYTE *Image, UWORD Width, UWORD Height, UWORD Rotate, UWORD Color)
Parameter:
    Image: The buffer of the image, this is a pointer of buffer address;
    Width: Width of the image;
    Height: Height of the image;
    Rotate: Rotate degree;
    Color: Initial color of the image;
```

- Select image buffer: this function is used to select the image buffer. You can create multiple image buffers with the last function, then select the buffer for every image.

```
void Paint_SelectImage(UBYTE *Image)
Parameter:
    Image: The name of image buffer, it is a pointer of buffer address;
```

- Set display orientation: This function is used to set the rotate degree, it is generally be used after Paint_SelectImage(). You can set the rotate degree to 0、90、180、270 degree.

```
void Paint_SetRotate(UWORD Rotate)
Parameter:
    Rotate: Rotate degree, you can choose ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270 which stands for 0, 90, 180, 270 degree respectively.
```

- Image mirroring: This function is used to mirror image.

```
void Paint_SetMirroring(UBYTE mirror)
Parameter:
    mirror: You can set it to MIRROR_NONE, MIRROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN
```

- Set pixel: this function is used to set the position and color of pixels in the buffer. This is the basic function of GUI.

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameter:
    Xpoint: X-axes in the buffer;
    Ypoint: Y-axes in buffer;
    Color: color
```

- Clear: This function is used to clear the screen to certain color.

```
void Paint_Clear(UWORD Color)
Parameter:
    Color:
```

- Clear windows: this function is used to clear a window. It is generally used for time display.

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color)
Parameter:
    Xstart: Start coordinate of X-axes of the window;
    Ystart: Start coordinate of Y-axes of the window;
    Xend: End coordinate of X-axes of the window;
    Yend: End coordinate of Y-axes of the window;
    Color:
```

- Draw point: Draw a point on the position （Xpoint, Ypoint） in buffer

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pixel, DOT_STYLE Dot_Style)
Parameter:
    Xpoint: X coordinate of point;
    Ypoint: Y coordinate of point;
    Color: color of point;
```

```
        Dot_Pixel: the size of point, there are 8 sizes available;
                typedef enum {
                        DOT_PIXEL_1X1  = 1,     // 1 x 1
                        DOT_PIXEL_2X2  ,                // 2 x 2
                        DOT_PIXEL_3X3  ,                // 3 x 3
                        DOT_PIXEL_4X4  ,                // 4 x 4
                        DOT_PIXEL_5X5  ,                // 5 x 5
                        DOT_PIXEL_6X6  ,                // 6 x 6
                        DOT_PIXEL_7X7  ,                // 7 x 7
                        DOT_PIXEL_8X8  ,                // 8 x 8
                } DOT_PIXEL;
        Dot_Style: style of point.
                typedef enum {
                  DOT_FILL_AROUND  = 1,
                  DOT_FILL_RIGHTED,
                } DOT_STYLE;
```

- Drew line: draw a line for (Xstart, Ystart) to (Xend, Yend)

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD C
olor, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
Parameter:
        Xstart: Start coordinate of X-axes of line;
        Ystart: Start coordinate of Y-axes of line;
        Xend: End coordinate of X-axes of line;
        Yend: End coordinate of Y-axes of line
        Color:
        Line_width: the width of line, 8 sizes are available;
                typedef enum {
                        DOT_PIXEL_1X1  = 1,     // 1 x 1
                        DOT_PIXEL_2X2  ,                // 2 x 2
                        DOT_PIXEL_3X3  ,                // 3 x 3
                        DOT_PIXEL_4X4  ,                // 4 x 4
                        DOT_PIXEL_5X5  ,                // 5 x 5
                        DOT_PIXEL_6X6  ,                // 6 x 6
                        DOT_PIXEL_7X7  ,                // 7 x 7
                        DOT_PIXEL_8X8  ,                // 8 x 8
                } DOT_PIXEL;
         Line_Style: style of the line;
                typedef enum {
                        LINE_STYLE_SOLID = 0,
                        LINE_STYLE_DOTTED,
                } LINE_STYLE;
```

- Drew rectangle: Draw a rectangle from (Xstart, Ystart) to (Xend, Yend)

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UW
ORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameter:
        Xstart: Start coordinate of X-axes of rectangle
        Ystart: Start coordinate of Y-axes of rectangle
        Xend: End coordinate of X-and of rectangle
        Yend: End coordinate of Y-and of rectangle
        Color: color of rectangle
        Line_width: The width of edges, 8 sizes are available;
                typedef enum {
                        DOT_PIXEL_1X1  = 1,     // 1 x 1
                        DOT_PIXEL_2X2  ,                // 2 x 2
                        DOT_PIXEL_3X3  ,                // 3 x 3
                        DOT_PIXEL_4X4  ,                // 4 x 4
                        DOT_PIXEL_5X5  ,                // 5 x 5
                        DOT_PIXEL_6X6  ,                // 6 x 6
                        DOT_PIXEL_7X7  ,                // 7 x 7
                        DOT_PIXEL_8X8  ,                // 8 x 8
                } DOT_PIXEL;
        Draw_Fill: set the rectangle full or empty.
                typedef enum {
                        DRAW_FILL_EMPTY = 0,
                        DRAW_FILL_FULL,
                } DRAW_FILL;
```

- Drew circle:Draw a circle, use (X_Center Y_Center) as center;

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Colo
r, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameter:
        X_Center: X coordinate of center
        Y_Center: Y coordinate of center
        Radius :Radius of circle
        Color: color of circle
        Line_width: width of circle, 8 sizes are available
                typedef enum {
                        DOT_PIXEL_1X1  = 1,     // 1 x 1
                        DOT_PIXEL_2X2  ,                // 2 x 2
                        DOT_PIXEL_3X3  ,                // 3 x 3
                        DOT_PIXEL_4X4  ,                // 4 x 4
                        DOT_PIXEL_5X5  ,                // 5 x 5
                        DOT_PIXEL_6X6  ,                // 6 x 6
                        DOT_PIXEL_7X7  ,                // 7 x 7
                        DOT_PIXEL_8X8  ,                // 8 x 8
                } DOT_PIXEL;
        Draw_Fill: style of circle
                typedef enum {
                        DRAW_FILL_EMPTY = 0,
                        DRAW_FILL_FULL,
                } DRAW_FILL;
```

- Drew character (ASCII): Set(Xstart Ystart) as left top point, draw a ASCII character

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* F
ont, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: X coordinate of left-top pixel of character;
        Ystart: Y coordinate of left-top pixel of character;
        Ascii_Char: Ascii character;
        Font: 5 fonts are available:
                &Font: 8*8
                &Font12: 7*12
                &Font16: 11*16
                &Font20: 14*20
                &Font24: 17*24
        Color_Foreground: color of character;
        Color_Background: color of background;
```

- Drew String: Set point (Xstart Ystart) as the left-top pixel, draw a string.

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFON
T* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: X coordinate of left-top pixel of characters;
        Ystart: Y coordinate of left-top pixel of characters;
        pString: Pointer of string
        Font: 5 fonts are available:
                &Font: 8*8
                &Font12: 7*12
                &Font16: 11*16
                &Font20: 14*20
                &Font24: 17*24
        Color_Foreground: color of string
        Color_Background: color of background
```

- Drew Chinese characters: this function is used to drew Chinese fonts based ON GB2312 font.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, sFON
T* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: Coordinate of left-top pixel of characters;
        Ystart: Coordinate of left-top pixel of characters;
        pString: Pointer of string
        Font: GB2312 fonts:
                &Font12CN: 12*21(ascii), 16*21 (Chinese)
                &Font24CN: 24*41(ascii), 32*41 (Chinese)
        Color_Foreground: color of string
        Color_Background: color of background
```

- Drew number: Drew a string of numbers, (Xstart, Ystart) is the left-top pixel.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Number, sFONT* Font, UW
ORD Color_Foreground, UWORD Color_Background)
Parameter:
        Xstart: X coordinate of left-top pixel;
        Ystart: Y coordinate of left-to pixel;
        Number: the numbers displayed. the numbers are saved in int format, th
e maximum is 2147483647;
        Font: 5 fonts are available:
                &Font: 8*8
                &Font12: 7*12
                &Font16: 11*16
                &Font20: 14*20
                &Font24: 17*24
        Color_Foreground: color of font;
        Color_Background: color of background;
```

- Display time:Display time, (Xstart, Ystart) is the left-top pixel. This function is used for e-Paper which supports partial refresh

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, SCREEN_TIME *pTime, sFONT* Font,
UWORD Color_Background, UWORD Color_Foreground)
Parameter:
        Xstart: X coordinate of left-top pixel of character;
        Ystart: Y coordinate of left-top pixel of character;
        pTime: Pointer of time displayed
        Font: 5 fonts are available:
                &Font: 8*8
                &Font12: 7*12
                &Font16: 11*16
                &Font20: 14*20
                &Font24: 17*24
        Color_Foreground: color of fonts
        Color_Background: color of background
```

- Drew image:send image data of bmp file to buffer

```
void Paint_DrawBitMap(const unsigned char* image_buffer)
Parameter:
        image_buffer: address of image data in buffer
```

User Guides of Arduino

Hardware connection

The demo codes provided are based on Arduino UNO. If you want to use another Arduino board you may need to change the connection and port the codes by yourself.

The RAM of Arduino UNO is too small to realize drawing function of e-Paper. In this case, the most of the e-Paper we only realize the basic display function and

this case, no matter of the e-paper, we only need the image display function and save the image to flash.

We recommend you to use E-Paper Shield if you use Arduino UNO. The seven-color e-Paper is not compatible with the e-Paper Shield, please note it.

Connect to Arduino UNO

e-Paper	Arduino
Vcc	5V
GND	GND
DIN	D11
CLK	D13
CS	D10
DC	D9
RST	D8
BUSY	D7

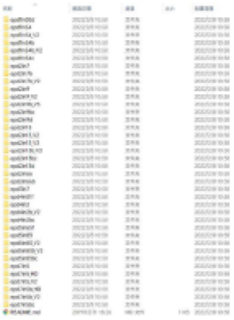
Install Arduino IDE

Arduino IDE Install Guide

Run the Program

Download the program on the product encyclopedia interface, and then unzip it. Arduino program file is stored in Arduino...

Please select the corresponding program to open according to the ink screen model



Supporting types

1.02inch (128×80) :

epd_1in02 : Example for 1.02inch e-Paper/1.02inch e-Paper Module

1.54inch (1.54inch e-paper c : 152×152 , others : 200×200) :

epd_1in54 : Example for 1.54inch e-paper V1 (Black/White) : This version is stopped production which can be bought before 2019-11-22 ;
epd_1in54_V2 : Example for 1.54inch e-paper V2 (Black/White): This is the current version which can be bought now (2020-07-29). The e-Paper has V2 sticker on the backside.
epd_1in54b : Example for 1.54inch e-paper B (Black/White/Red) ;
epd_1in54b_V2 : Example for 1.54inch e-paper B V2 (Black/White/Red) ;
epd_1in54c : Example for 1.54inch e-paper C (Black/White/Red) ;

2.7inch (264×176) :

epd_2in7 : Example for 2.7inch e-paper (Black/White) ;
epd_2in7b : Example for 2.7inch e-paper B (Black/White/Red) ;
epd_2in7b_V2 : Example for 2.7inch e-paper B V2 (Black/White/Red) ;

2.9inch (296×128) :

epd_2in9 Example for 2.9inch e-Paper (Black/White) ;
epd_2in9_V2: Example for 2.9inch e-paper V2 (Black/White) ;
epd_2in9bc : Example for 2.9inch e-paper B (Black/White/Red) and 2.9inch e-paper C (Black/White/Yellow) ;
epd_2in9b_V3 : Example for 2.9inch e-paper B V3 (Black/White/Red;
epd_2in9d : Example for 2.9inch e-paper D (Black/White) ;

2.13inch (2.13inch e-Paper : 250×122 , others : 212×104) :

epd_2in13 : Example for 2.13inch e-paper V1 (Black/White) , this version is stopped production and it can be bought before 2019-05-15;
epd_2in13_V2r : Example for 2.13inch e-paper V2 (Black/White) This is the current version with sticker V2 on the backside (2020-07-29) ;
epd_2in13bc : Example for 2.13inch e-paper B (Black/White/Red) and 2.13inch e-paper C (Black/White/Yellow) ;
epd_2in13b_V3 : Example for 2.13inch e-paper B V3 (Black/White/Red)
epd_2in13d : Example for 2.13inch e-Paper D (Black/White) ;

2.66inch (152 x 296)

epd_2in66 Example for 2.66inch e-Paper (Black/White);
epd_2in66b Example for 2.66inch e-Paper (Black/White/Red);

3.7inch (280 x 480)

epd_3in7 Example for 3.7inch e-Paper (Black/White);

4.01inch (640x400)

epd_4in01f Example for 4.01inch e-Paper (Seven color);

4.2inch (400×300)

epd_4in2 : Example for 4.2inch e-paper (Black/White) ;
epd_4in2bc : Example for 4.2inch e-paper B (Black/White/Red) ;
epd_4in2b_V2 : Example for 4.2inch e-paper B V2 (Black/White/Red) ;

5.65inch (600x448)

epd_5in65f Example for 5.65inch e-Paper F (Seven-color)

5.83inch (600×448) :

epd_5in83 : Example for 5.83inch e-paper (Black/White) ;
epd_5in83_V2 : Example for 5.83inch e-paper V2 (Black/White) ;
epd_5in83bc : Example for 5.83inch e-paper B (Black/White/Red) and 5.83inch e-paper C (Black/White/Yellow) ;
epd_5in83b_V2 : Example for 5.83inch e-paper B V2 (Black/White/Red)

7.5inch (V1 : 640×384 , V2 : 800×480) :

epd_7in5 : Example for 7.5inch e-paper (Black/White) , this version is stopped production and it can be bought before 2019-12-07 ;
epd_7in5_V2 : Example for 7.5inch e-paper V2 (Black/White) , This is the current version with V2 sticker on the backside (2020-07-29)
epd_7in5bc : Example for 7.5inch e-paper B (Black/White/Red) and 7.5inch e-paper C (Black/White/Yellow) , 7.5inch e-paper B V1 version is stopped production and it can be bought before 2019-12-07;
epd_7in5b_V2 : Example for 7.5inch e-paper B V2 (Black/White/Red) ; This is the current version with V2 sticker on the backside (2020-07-29);

7.5inch (HD 880 x 528)

epd_7in5_HD: Example for 7.5inch e-Paper HD (Black/White);
epd_7in5b_HD: Example for 7.5inch e-Paper B HD (Black/White/Red);

*Note: The above time is for reference only, please refer to the screen logo for the specific version

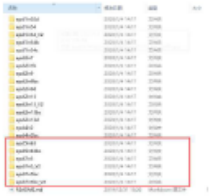
For example, 1.54inch e-Paper Module. Open the epd1in54 folder and run the epd1in54.ino file.

Open the program, select the development board model Arduino UNO

Select the COM corresponding COM port

Then click compile and download

Since the flash of arduino UNO is very small, several large-size screens use MEGA2560 for better running



Code Description

Files

Take the 1.54-inch ink screen controlled by Arduino UNO as an example, open the epd1in54 directory:

In which:

epd1in54.ino: open with Arduino IDE

epd1in54.cpp(.h): is the driver of the ink screen;

epdif.cpp(.h): is the hardware interface definition, which defines the GPIO and write pin level, SPI data transmission, and pin initialization;

font8.cpp, font12.cpp, font16.cpp, font20.cpp, font24.cpp, font.h: are the models of characters of different sizes;

imagedata.cpp(.h): is the image data, which can convert the 2-bit depth BMP image into an array through Img2Lcd (downloadable in the development data).

The program is divided into the bottom hardware interface, the middle layer ink screen driver, and the upper layer application:

Bottom Hardware Interface

hardware interfaces are defined in epdif.cpp(.h) file

Write GPIO

```
void DigitalWrite(int pin, int value)
```

The first parameter is GPIO, and second parameter is level

Read GPIO

```
int DigitalRead(int pin)
```

The parameter is GPIO, and return value is level

Delay

```
Delay(ms(unsigned int delaytime)
```

Delay time, unit is ms

SPI transmit data

```
SpiTransfer(unsigned char data)
```

Type of parameter is char

Hardware initialize

```
int IfInit(void)
```

The initialize function of SPI, input/output are packaged here.

Middle EPD driver

Instantiate e-Paper class

The Arduino code are based on C++, should instantiate e-Paper class is necessary.

```
Epd epd;
```

Initialize e-Paper, it should be used to initialize e-Paper or wakeup e-Paper from sleep mode.

- 2.13inch e-Paper / 2.9inch e-Paper

```
epd.Init(lut_full_update); //Fully update
epd.Init(lut_partial_update); //Partial update
```

- 4.2inch e-Paper

```
epd.Init();
```

Clear, clear the e-Paper to white

```
epd.Clear();
```

In some of projects, the operation is divided to two part, they work in the same way

```
epd.ClearFrameMemory(0xFF);
epd.DisplayFrame();//Display it
```

Transmit one frame of image and display

```
void DisplayFrame(const unsigned char* frame_buffer);
void DisplayFrame(const unsigned char* frame_buffer_black, const unsigned char* frame_buffer_red); //Three color e-Paper
```

Sleep

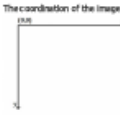
```
epd.Sleep();
```

Set the e-Paper enter sleep mode. The consumption of the e-Paper will be reduced. However, you still need to update the display periodically to avoid ghost problem.

Application functions

The drawing functions are defined in this part.

The coordination of the image buffer：



The functions are defined in epdpaint.h file

```
class Paint {
  public:
    Paint(unsigned char* image, int width, int height);
    ~Paint();
    void Clear(int colored);
    int  GetWidth(void);
    void SetWidth(int width);
    int  GetHeight(void);
    void SetHeight(int height);
    int  GetRotate(void);
    void SetRotate(int rotate);
    unsigned char* GetImage(void);
    void DrawAbsolutePixel(int x, int y, int colored);
    void DrawPixel(int x, int y, int colored);
    void DrawCharAt(int x, int y, char ascii_char, sFONT* font, int colored);
    void DrawStringAt(int x, int y, const char* text, sFONT* font, int colored);
    void DrawLine(int x0, int y0, int x1, int y1, int colored);
    void DrawHorizontalLine(int x, int y, int width, int colored);
    void DrawVerticalLine(int x, int y, int height, int colored);
    void DrawRectangle(int x0, int y0, int x1, int y1, int colored);
    void DrawFilledRectangle(int x0, int y0, int x1, int y1, int colored);
    void DrawCircle(int x, int y, int radius, int colored);
    void DrawFilledCircle(int x, int y, int radius, int colored);

  private:
    unsigned char* image;
    int width;
    int height;
    int rotate;
};
```

Initalize image buffer

```
Paint(unsigned char* image, int width, int height);
```

The first parameter is image buffer, the second one is the width of the picture, and the third one is the height.

```
Paint paint(image, 0, 0);    // width should be the multiple of 8
```

The second and third parameters are set to 0, you can reconfigure them with functions below:

Set the width, height, rotate degree.

```
int  GetWidth(void); //Get the width
void SetWidth(int width);//Set the width
int  GetHeight(void);//Get the height
void SetHeight(int height);//Set the height
int  GetRotate(void);//Get the degree
void SetRotate(int rotate);//Set the rotate degree
```

Get the image data

```
unsigned char* GetImage(void);
```

Draw circle

```
void DrawPixel(int x, int y, int colored);
```

Coordination (x,y)

Draw character

```
void DrawCharAt(int x, int y, char ascii_char, sFONT* font, int colored);
```

Set (x,y) as the start point, draw character with size of font, color is colored

Draw string

```
void DrawStringAt(int x, int y, const char* text, sFONT* font, int colored);
```

Set (x,y) as the start point, draw the string with font in font, color is colored

Draw line

```
void DrawLine(int x0, int y0, int x1, int y1, int colored);
```

Like (x0,y0) as start point, (x1,y1) as end point;

Draw cross line

```
void DrawHorizontalLine(int x, int y, int width, int colored);
```

Set (x0,y0) as start point, draw a line, the width is width, and color is colored</br>

Draw a vertical line

```
void DrawVerticalLine(int x, int y, int height, int colored);
```

Like (x0,y0) as start point, draw a vertical line, width is height, and color is colored / br >

Draw a empty rectangle

```
void DrawRectangle(int x0, int y0, int x1, int y1, int colored);
```

Like (x0,y0) as start point, (x1,y1) is end point, draw a rectangle, color of edges are colored.

Draw a full rectangle

```
void DrawFilledRectangle(int x0, int y0, int x1, int y1, int colored);
```

Like (x0,y0) as start point, (x1,y1) is end point, draw a rectangle, filled it with color: colored

Draw an empty circle

```
void DrawCircle(int x, int y, int radius, int colored);
```

Like (x,y) as center, draw a empty circle with radius, color is colored

Draw a full circle

```
void DrawFilledCircle(int x, int y, int radius, int colored);
```

Like (x,y) as center, draw a circle, radius is radius, filled with color: colored

Resources

Documentation

- User Manual
- INSTRUCTION About make new font
- Make BMP file for e-Paper
- Schematic

Demo code

- Github

Datasheets

- Datasheets

Related Resources

Notice:

The project is developed and maintained by the project owners. We are not responsible for project other than updates.

- Waveshare e-Paper display with SPI

This is a post in Arduino Forum about our SPI e-Paper, thanks to ZinggJM, maybe you want to refer to.

- Inkycal Project

This is the Inkycal project for reference.

FAQ

Question about Software

Question: stm32 drive the ink screen, the MDK compilation display space is not enough?

Answer:

Our demo uses stm32f103ret6. If the customer modifies other models in MDK, such as stm32f103rbt6, the ram space becomes smaller, and the stack size and heap size in the startup file need to be modified on the original basis.

Question:ImportError: No module named 'spidev'

Answer:
*It is not the Raspberry Pi using the command "sudo apt-get install python-spidev"

Question:What is the difference between DRIVER_SET_TWO_PASS_MODE 1 and when to use DRIVER_SET_TWO_PASS_MODE 2. If it is not clear, what is "OLD" and what is "NEW"?

Answer:
If it is a three-color screen, when you need to transmit B/W data, use Data Start Transmission 1, when transmitting RED data, use Data Start Transmission 2 because the 2.13inch e-Paper HAT (D) can only display black and white. Therefore, when working in BW mode DRIVER_SET_TWO_PASS_MODE 1, the BW data is sent, that is, the so-called old, and 0x00 is directly sent in the program. When DRIVER_SET_TWO_PASS_MODE 2, the driver IC needs to be refreshed. the so-called "NEW" is next step.

Question:Can't display Chinese on the e-ink screen?

Answer:
The Chinese character library of our routine uses GB2312 encoding method, please change your xxx.txt file to GB2312 encoding format, compile and download it, and it will display normally

Question:The e-Paper display is too dark or too light?

Answer:
You can adjust the value of Vcom in the program to change the display contrast, and the screen improvement effect with local brushing is particularly obvious.

Question:E-Paper how to set its border?

Answer:
The border display color can be set through the Border Waveform Control register of the VCOM AND DATA INTERVAL SETTING register.

Question:After multiple positions are brushed, the font is lighter after brushing several times?

Answer:
In this case, the customer needs to reduce the position of the round brush and clear the screen after 5 rounds of brushing (increasing the voltage of VCOM can improve the point, but it will increase the afterimage)

Question:When the ink screen is in deep sleep mode, the first time the screen refreshes will be unclean. How can I solve it?

Answer:
The process of re-awakening the e-ink screen is actually the process of re-powering on the power. Therefore, when the EPD wakes up, it must first clear the screen twice to avoid the afterimage phenomenon to the greatest extent.

Question:When testing the program, the program has been stuck in e-Paper busy?

Answer:
*It may be caused by the unsuccessful spi driver

1. First check whether the wiring is correct
2. Check whether the spi is turned on and whether the parameters are configured correctly (spi baud rate, spi mode and other parameters)

Question about Hardware

Question:Can it drive 3.3V drive pin ink screen?

Answer:
Yes, now there is a level conversion chip onboard, supporting 3.3V drive

Question:What should be pay attention to when designing the driver board?

Answer:
The rated input voltage of the ink screen is 2.3~3.6V. If it is a 5V system, level conversion is required. In addition, the voltage should not be lower than 2.5V so as not to affect the display effect of the ink screen.

Device selection can use the model in the schematic diagram we provide or choose according to the data sheet

Question:Can I use analog SPI?

Answer:
Yes, pay attention to the correct timing

Question:Why is the BUSY pin always busy?

Answer:
Check if SPI communication is normal

Confirm whether the BUSY pin is normally initialized to input mode

It may be that there is no normal reset, try to shorten the duration of the low level during reset (because the power-off switch is added to the drive circuit, the reset low level is too long, which will cause the drive board to power off and cause the reset to fail) If the busy function sends the 0x71 command, you can try to comment it out

Question about Screen

Question:What is the usage environment of the e-ink screen?

Answer:
* [Working Condition] Temperature range: 0~50℃ Humidity range: 35%~65%RH

- [Storage condition] : Temperature range: below 30℃ Humidity range: below 55%RH; Maximum storage time: 6 months
- [Transportation conditions] : Temperature range: -25~70℃; Maximum transportation time: 10 days
- [After unpacking] : Temperature range: 20℃±5℃; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours

Question:Precautions for e-ink screen refresh

Answer:
*Refresh mode

- - Full brush: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect
  - Partial brush: The electronic ink screen has no flicking effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full brush operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial brushing, please refer to product page description)
  - refresh rate
  - During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except products that support the local brush function)
  - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode or power off operation (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
  - During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24

hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair)

- Application
  - The e-ink screen is recommended for indoor use. If it is used outdoors, it is necessary to avoid direct sunlight on the e-ink screen, and at the same time, the UV protection material, because charged particles will dry out under strong light for a long time, resulting in loss of activity and failure to refresh. This situation is irreversible. When designing e-ink screen products, customers should pay attention to determine whether the use environment meets the requirements of e-ink screen

Question:What is the refresh/rewrite type of the e-ink screen?

Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times)

Question:After using for a period of time, the screen remains (but not refresh) has a serious afterimage problem that cannot be repaired?

Answer:
Power on the development board for a long time without refresh operation. It is recommended to let the screen to sleep mode or directly power off processing, otherwise the screen may burn out when the screen is in a high voltage state for a long time.

Question:After the ink screen enters deep sleep mode, can it be refreshed again?

Answer:
Yes, but you need to re-initialize the electronic paper with software

Question:Why is the image displayed offset

Answer:
Maybe the SPI rate is too high, resulting in data loss, try to reduce the SPI rate

Insufficient or unstable power supply leads to data loss

The data cable is too long to cause data loss, the extension cable should not exceed 20cm

Question:What is the waveform file of the e-ink screen and what does it do?

Answer:
The display gray scale of electrophoretic electronic paper is determined by the specific position of the particles in the certain state of a pixel. The electrophoresis phenomenon occurs between black particles and white particles under the action of voltage. The voltage-driven time formats the electrophoretic movement of the particles is the driving force of the electronic paper. waveform. The driving waveform is the core part of the electronic paper display, and the optimization of the driving waveform will directly affect the display effect of the display. The driving waveform file is used to describe the parameters formed by the voltage sequence that promotes the electrophoretic movement of the particles, and it needs to be called regularly when the electronic paper is refreshed.

Different batches of e-paper display inks and electrophoretic particles require different voltage values when driving the display due to materials, manufacturing processes, etc. The waveform of the e-ink screen is reflected in the relationship between grayscale, voltage and temperature. Generally speaking, after each batch of electrophoretic particle is generated, there will be a corresponding waveform file in the form of a .wbf file. The film manufacturer will provide the waveform file and electrophoretic particle to the manufacturer of the electronic paper screen, and then the The manufacturer of electronic paper screen integrates the protection board, substrate and driver board then provides it to customers; if the waveform file does not correspond to the screen, it is likely that the display cannot be displayed or the display effect is unsatisfactory. Generally, the waveform file has OTP built into the driver IC of the ink screen when leaving the factory. Hence programs we provide also call external waveform files to drive the e-ink screen.

Question: What do LUT and OTP stand for?

Answer:
LUT is the abbreviation of LOOK UP TABLE, and OTP is the abbreviation of ONE TIME PROGRAM. The original intention of LUT is to load waveform files, and the waveform files are divided into OTP and REGISTER. Among them, OTP is the built-in waveform storage method, and REGISTER is the external waveform storage method.

Question:What is the process of brushing e-paper?

Answer:
There are mainly two types of ink screen

One is to brush the background image first

The other is to directly refresh old data and new data

Question:How do I play in different positions at the same time?

Answer:
Simultaneous brushing in different positions needs to be operated in the program design, that is, first brushing the data of different locations into the electronic paper IC and finally doing the UpdateDisplay unified refresh

Question: Does the three-color e-paper have a red/yellow color difference?

Answer:
Yes, when e-paper is refreshed, there will be some color difference, which is normal phenomenon. Store the e-paper face up to reduce the red/black/yellow three-color difference.

Others

Question: Are the screen shipped with a film?

Answer:
with film

Question: What is the specification of the screen cable interface?

Answer:
0.5mm pitch 24Pin

Question: What type of connector does the ink screen use?

Answer:
Offset Reflow 0.5-24pin FPC Flip Type 2.0H (FPC Connector)

Question: Does e-paper have a built-in temperature sensor?

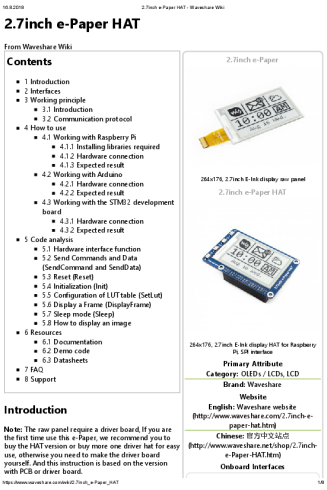
Answer:
All of the ink screens have built-in temperature sensors. And also the LM75 temperature sensors can also be used with IIC pins.

Question:Why can't the image be displayed after full brushing?

Answer:
The full brush initialization function needs to be added when the ink screen is switched from partial brush to full brush

Question: Why is the prompt information normal when running the python program, but the ink screen does not respond?

Answer:
It may be the demo board on the BCM2835 library that has run the C language before. At this time, you need to restart the Raspberry Pi and then run the python demo



2.7inch e-Paper HAT

From Waveshare Wiki

Contents

- 1 Introduction
- 2 Interfaces
- 3 Working principle
  - 3.1 Introduction
  - 3.2 Communication protocol
- 4 How to use
  - 4.1 Working with Raspberry Pi
    - 4.1.1 Installing libraries required
    - 4.1.2 Hardware connection
    - 4.1.3 Expected result
  - 4.2 Working with Arduino
    - 4.2.1 Hardware connection
    - 4.2.2 Expected result
  - 4.3 Working with the STM32 development board
    - 4.3.1 Hardware connection
    - 4.3.2 Expected result
- 5 Code analysis
  - 5.1 Hardware interface function
  - 5.2 Send Commands and Data (SendCommand and SendData)
  - 5.3 Reset (Reset)
  - 5.4 Initialization (Init)
  - 5.5 Configuration of LUT table (SetLut)
  - 5.6 Display a Frame (DisplayFrame)
  - 5.7 Sleep mode (Sleep)
  - 5.8 How to display an image
- 6 Resources
  - 6.1 Documentation
  - 6.2 Demo code
  - 6.3 Datasheets
- 7 FAQ
- 8 Support

Introduction

Note: The raw panel require a driver board, If you are the first time use this e-Paper, we recommend you to buy the HAT version or buy more one driver hat for easy use, otherwise you need to make the driver board yourself. And this instruction is based on the version with PCB or driver board.

2.7inch e-Paper

264x176, 2.7inch E-Ink display raw panel

2.7inch e-Paper HAT

264x176, 2.7inch E-Ink display HAT for Raspberry Pi, SPI interface

Primary Attribute

Category: OLEDs / LCDs, LCD

Brand: Waveshare

Website

English: Waveshare website (http://www.waveshare.com/2.7inch-e-paper-hat.htm)

Chinese: 官方中文站点 (http://www.waveshare.net/shop/2.7inch-e-Paper-HAT.htm)

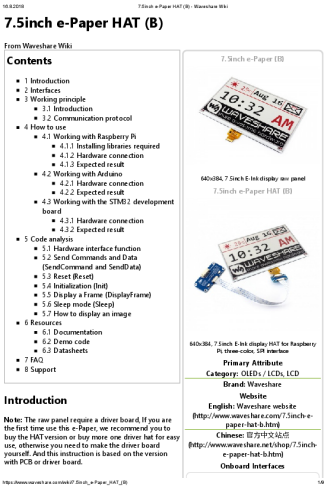
Onboard Interfaces

[pdf] Frequently Asked Questions Datasheet Documentation

51741840 cdn petec ch documents2 0 4 8

16.8.2018 2.7inch e-Paper HAT - Waveshare Wiki 2.7inch e-Paper HAT From Waveshare Wiki Contents 2.7inch e-Paper 1 Introduction 2 Interfaces 3 Working principle 3.1 Introduction 3.2 Communication protocol 4 How to use 4.1 Working with Raspberry Pi 4.1.1 Installing libraries required 4.1.2 Hardwa...

lang:en score:20 filesize: 384.43 K page_count: 8 document date: 2018-08-16



7.5inch e-Paper HAT (B)

From Waveshare Wiki

Contents

- 1 Introduction
- 2 Interfaces
- 3 Working principle
  - 3.1 Introduction
  - 3.2 Communication protocol
- 4 How to use
  - 4.1 Working with Raspberry Pi
    - 4.1.1 Installing libraries required
    - 4.1.2 Hardware connection
    - 4.1.3 Expected result
  - 4.2 Working with Arduino
    - 4.2.1 Hardware connection
    - 4.2.2 Expected result
  - 4.3 Working with the STM32 development board
    - 4.3.1 Hardware connection
    - 4.3.2 Expected result
- 5 Code analysis
  - 5.1 Hardware interface function
  - 5.2 Send Commands and Data (SendCommand and SendData)
  - 5.3 Reset (Reset)
  - 5.4 Initialization (Init)
  - 5.5 Display a Frame (DisplayFrame)
  - 5.6 Sleep mode (Sleep)
  - 5.7 How to display an image
- 6 Resources
  - 6.1 Documentation
  - 6.2 Demo code
  - 6.3 Datasheets
- 7 FAQ
- 8 Support

Introduction

Note: The raw panel require a driver board, If you are the first time use this e-Paper, we recommend you to buy the HAT version or buy more one driver hat for easy use, otherwise you need to make the driver board yourself. And this instruction is based on the version with PCB or driver board.

7.5inch e-Paper (B)

640x384, 7.5inch E-Ink display raw panel

7.5inch e-Paper HAT (B)

640x384, 7.5inch E-Ink display HAT for Raspberry Pi, three-color, SPI interface

Primary Attribute

Category: OLEDs / LCDs, LCD

Brand: Waveshare

Website

English: Waveshare website (http://www.waveshare.com/7.5inch-e-paper-hat-b.htm)

Chinese: 官方中文站点 (http://www.waveshare.net/shop/7.5inch-e-paper-hat-b.htm)

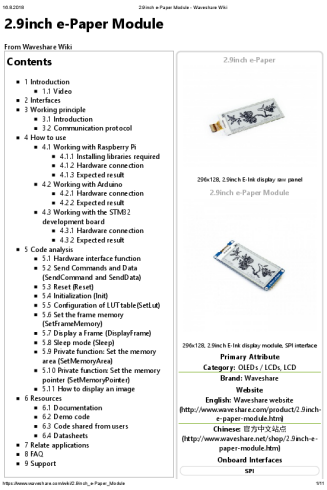
Onboard Interfaces

[pdf] Frequently Asked Questions Datasheet Documentation

51742032 cdn petec ch documents2 2 3 0 |||

16.8.2018 7.5inch e-Paper HAT B - Waveshare Wiki 7.5inch e-Paper HAT B From Waveshare Wiki Contents 7.5inch e-Paper B 1 Introduction 2 Interfaces 3 Working principle 3.1 Introduction 3.2 Communication protocol 4 How to use 4.1 Working with Raspberry Pi 4.1.1 Installing libraries required ...

lang:en score:20 filesize: 365.18 K page_count: 9 document date: 2018-08-16



2.9inch e-Paper Module

From Waveshare Wiki

Contents

- 1 Introduction
  - 1.1 Video
- 2 Interfaces
- 3 Working principle
  - 3.1 Introduction
  - 3.2 Communication protocol
- 4 How to use
  - 4.1 Working with Raspberry Pi
    - 4.1.1 Installing libraries required
    - 4.1.2 Hardware connection
    - 4.1.3 Expected result
  - 4.2 Working with Arduino
    - 4.2.1 Hardware connection
    - 4.2.2 Expected result
  - 4.3 Working with the STM32 development board
    - 4.3.1 Hardware connection
    - 4.3.2 Expected result
- 5 Code analysis
  - 5.1 Hardware interface function
  - 5.2 Send Commands and Data (SendCommand and SendData)
  - 5.3 Reset (Reset)
  - 5.4 Initialization (Init)
  - 5.5 Configuration of LUT table(SetLut)
  - 5.6 Set the frame memory (SetFrameMemory)
  - 5.7 Display a Frame (DisplayFrame)
  - 5.8 Sleep mode (Sleep)
  - 5.9 Private function: Set the memory area (SetMemoryArea)
  - 5.10 Private function: Set the memory pointer (SetMemoryPointer)
  - 5.11 How to display an image
- 6 Resources
  - 6.1 Documentation
  - 6.2 Demo code
  - 6.3 Code shared from users
  - 6.4 Datasheets
- 7 Relate applications
- 8 FAQ
- 9 Support

2.9inch e-Paper

296x128, 2.9inch E-Ink display raw panel

2.9inch e-Paper Module

296x128, 2.9inch E-Ink display module, SPI interface

Primary Attribute

Category: OLEDs / LCDs, LCD

Brand: Waveshare

Website

English: Waveshare website (http://www.waveshare.com/product/2.9inch-e-paper-module.htm)

Chinese: 官方中文站点 (http://www.waveshare.net/shop/2.9inch-e-paper-module.htm)

Onboard Interfaces

SPI

[pdf] Frequently Asked Questions Datasheet Documentation

51742165 cdn petec ch documents2 5 6 1 |||

16.8.2018 2.9inch e-Paper Module - Waveshare Wiki 2.9inch e-Paper Module From Waveshare Wiki Contents 1 Introduction 1.1 Video 2 Interfaces 3 Working principle 3.1 Introduction 3.2 Communication protocol 4 How to use 4.1 Working with Raspberry Pi 4.1.1 Installing libraries required 4.1.2 Hardware...

lang:en score:20 filesize: 542.67 K page_count: 11 document date: 2018-08-16