# 7.5inch e-Paper HAT (B)

# 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.

800x480, 7.5inch E-Ink display HAT for Raspberry Pi, three-color, SPI interface

【Version update instructions】：

* This product has been updated to V2 version with higher resolution, supporting 800x480 resolution (V1 version has 640x384 resolution)
* The hardware structure and interface of V2 version of this product are compatible with V1 version, but the software needs to be updated

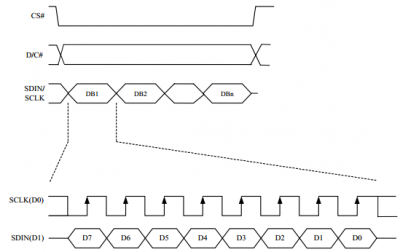
# Interfaces

|  |  |
| --- | --- |
| VCC | 3.3V |
| GND | GND |
| DIN | SPI MOSI |
| CLK | SPI SCK |
| CS | SPI chip select (Low active) |
| DC | Data/Command control pin (High for data, and low for command) |
| RST | External reset pin (Low for reset) |
| BUSY | Busy state output pin (Low for busy) |

# 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 suspending 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. (**Note that the e-Paper cannot support updating directly under sunlight**)

**Communication protocol**

[](https://www.waveshare.com/wiki/File:1.54inch-e-paper-manual-1.png)

Note: Different from the traditional SPI protocol, the data line from the slave to the master is hidden since the device only has display requirement.

* CS is slave chip select, when CS is low, the chip is enabled.
* DC is data/command control pin, when DC = 0, write command, when DC = 1, write data.
* SCLK is the SPI communication clock.
* SDIN is the data line from the master to the slave in SPI communication.

SPI communication has data transfer timing, which is combined by CPHA and CPOL.

1. CPOL determines the level of the serial synchronous clock at idle state. When CPOL = 0, the level is Low. However, CPOL has little effect to the transmission.
2. CPHA determines whether data is collected at the first clock edge or at the second clock edge of serial synchronous clock; when CPHL = 0, data is collected at the first clock edge.

* There are 4 SPI communication modes. SPI0 is commonly used, in which CPHL = 0, CPOL = 0.

As you can see from the figure above, data transmission starts at the first falling edge of SCLK, and 8 bits of data are transferred in one clock cycle. In here, SPI0 is in used, and data is transferred by bits, MSB first.

# Hardware/Software setup

We provide examples for four popular platforms: Arduino UNO, Jetson Nano， Raspberry Pi, and STM32. You can refer to the corresponding part according to the platform you use(This is a common template will be used by all types of the e-Paper, please refer to the corresponding type you have)

# **Arduino UNO**

The example we provide for Arduino platform is based on Waveshare UNO PLUS (it is compatible with official Arduino UNO R3). If you use other Arduino board which is not compatible with UNO, you may need to change the wring.

**Hardware connection**

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

**Running examples**

Download demo codes from Resources, unzip it to get projects. Arduino example is located in the directory ~/Arduino UNO/…  
Open project according to the type. For example, if the e-Paper you have is 1.54inch e-Paper Module, please open the epd1in54 folder and run project epd1in54.ino.  
Open project, choose the correct Board and Port, then compile and upload it to board.  
Note: Because of the small RAM of Arduino, it cannot support drawing function, therefore, we only provide image display function. The image data are stored in flash. Or you can think about using Waveshare [e-Paper Shield](https://www.banggood.com/Waveshare-e-Paper-Electronic-Paper-ink-Screen-Driver-Board-Expansion-Board-compatible-with-NUCLEO-For-Bare-Screen-p-1754771.html) for Arduino board

# **Raspberry Pi**

**Hardware connection**

If the board you get is the HAT version like 2.13inch e-Paper HAT, you can directly attach it on the 40PIN GPIO of Raspberry Pi. Or you can wire it to Raspberry Pi with 8PIN cable.

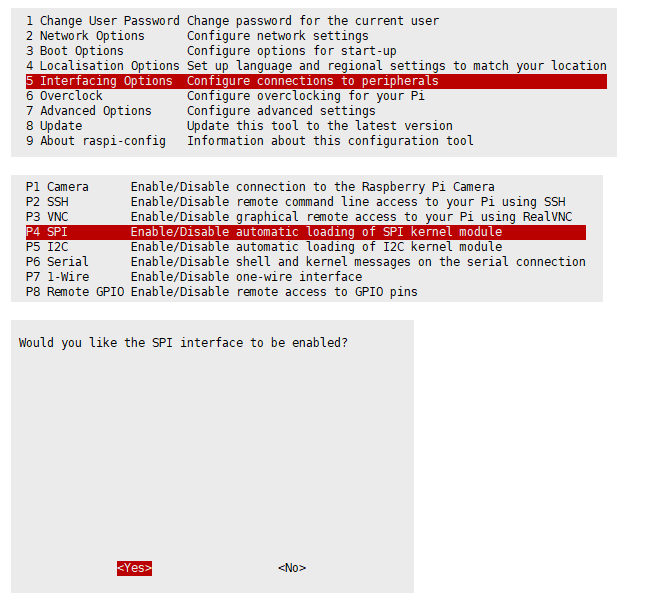
|  |  |  |
| --- | --- | --- |
| Connect to Raspberry Pi | | |
| e-Paper | Raspberry Pi | |
| BCM2835 | Board |
| VCC | 3.3V | 3.3V |
| GND | GND | GND |
| DIN | MOSI | 19 |
| CLK | SCLK | 23 |
| CS | CE0 | 24 |
| DC | 25 | 22 |
| RST | 17 | 11 |
| BUSY | 24 | 18 |

## Enable SPI interface

* Open terminal, use command to enter the configuration page

sudo raspi-config

Choose Interfacing Options -> SPI -> Yes to enable SPI interface

[](https://www.waveshare.com/wiki/File:RPI_open_spi.png)  
Reboot Raspberry Pi：

sudo reboot

Please make sure that SPI interface was not used by other devices

**Libraries Installation**

* Install BCM2835 libraries

wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.60.tar.gz

tar zxvf bcm2835-1.60.tar.gz

cd bcm2835-1.60/

sudo ./configure

sudo make

sudo make check

sudo make install

#For more details, please refer to http://www.airspayce.com/mikem/bcm2835/

* Install wiringPi libraries

sudo apt-get install wiringpi

#For Pi 4, you need to update it：

cd /tmp

wget https://project-downloads.drogon.net/wiringpi-latest.deb

sudo dpkg -i wiringpi-latest.deb

gpio -v

#You will get 2.52 information if you install it correctly

* Install Python libraries

#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

#python3

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

Note: if you install on Raspian Lite, the git package is not istalled by default.

* Install git

sudo apt-get install git -y

**Download examples**

Open terminal and execute command to download demo codes

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

cd e-Paper/RaspberryPi\&JetsonNano/

**Running examples**

* C codes

Find the main.c file, uncomment the definition of e-Paper types, then compile and run the codes.

cd c

make clean

make

sudo ./epd

* python

Run examples, xxx is the name of the e-Paper. For example, if you want to run codes of 1.54inch e-Paper Module, you xxx should be epd\_1in54

cd python/examples

# python2

sudo python xxx.py

# python3

sudo python3 xxx.py

# **Jetson nano Developer Kit**

The example for Jetson Nano use software SPI, speed of sfotware SPI is a little slow

**Hardware connection**

Jetson Nano's 40PIN GPIO is compatible with Raspberry PI, and API of Jetson.GPIo is same as RPi.GPIO, therefore, the pin numbers of Jetson nano are same as Raspberry Pi's

|  |  |  |
| --- | --- | --- |
| 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 |

**Software setting**

* Open terminal, and install GPIO 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

Note： your\_user\_name is the user name of your Jetson, for example:waveshare

* Install I2C libraries

sudo apt-get install python-smbus

* Install PIL libraries

sudo apt-get install python3-pil

sudo apt-get install python3-numpy

**Download examples**

Open terminal and execute commands:

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

cd e-Paper/RaspberryPi\&JetsonNano/

**Running examples**

* C codes

Find main.c file, Open it and uncommend the e-Paper which you use, compile and run it

cd c

make clear

make

sudo ./epd

* python

Run examples, xxx is the name of e-Paper. For example, if you want to run examples of 1.54inch e-Paper Module, xxx should be epd\_1in54

cd python/examples

# python2

sudo python xxx.py

# python3

sudo python3 xxx.py

# **STM32**

**Hardware connection**

The examples we provide are based on Wavshare Open103Z, the connecting method provide is based on STM32F13ZET6 as well. For other board, please port it by yourself.

|  |  |
| --- | --- |
| Connect to STM32F103ZET6 | |
| e-Paper | STM32F103ZET6 |
| Vcc | 3.3V |
| GND | GND |
| DIN | PA7 |
| CLK | PA5 |
| CS | PA3 |
| DC | PA2 |
| RST | PA1 |
| BUSY | PA3 |

**Running examples**

Enter the directly of STM32 examples, open project by Keil5 software. Set Board and programmer, then compile and download it to board

# Codes description

# About the codes

We provide examples for four popular hardware platforms: Arduino UNO, Jetson UNO, Raspberry Pi, and STM32. (This is common Template for all e-Paper, some of the description/function may not be used by the e-Paper you have)  
Every project is divided into hardware interface, EPD driver and the application function;  
The programming languages are C\C++\python：

* Arduino UNO：C++
* Jetson Nano：C and python
* Raspberry Pi：C and python
* STM32：C

Note:  
The EPD driver of C codes of Jetson Nano, Raspberry Pi, and STM32 are compatible. Except for the hardware interface, the codes are same;

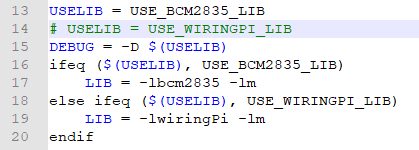
# C (Used for Jetson Nano、Raspberry Pi、STM32)

**Hardware interface**

Because of multiple hardware platforms, we package the bottom, for details of how it realizes, you go to related directory for certain codes

In file DEV\_Config.c(.h):  
For Raspberry Pi, the files are located in: RaspberryPi&JetsonNano\c\lib\Config

Here we use two libraries: bcm2835 and wiringPi

WiringPi library is used by default, if you want to use bcm2835 libraries, you just need to modify RaspberryPi&JetsonNano\c\Makefile file, change the lines 13 and 14 as below:：[](https://www.waveshare.com/wiki/File:E-paper_Driver_HAT_RPI_Makefile.png)

For Jetson Nano, the files are located in RaspberryPi&JetsonNano\c\lib\Config  
For STM32, the files are located in STM32\STM32-F103ZET6\User\Config

* Data type：

#define UBYTE uint8\_t

#define UWORD uint16\_t

#define UDOUBLE uint32\_t

* Module Init and Exit handle:

void DEV\_Module\_Init(void);void DEV\_Module\_Exit(void);

Note：

1.The functions are used to set GPIP before and after driving e-Paper.

2.If the board you have is printed with Rev2.1, module enter low-ultra mode after DEV\_Module\_Exit(). (as we test, the current is about 0 in this mode);

* GPIO Read/Write：

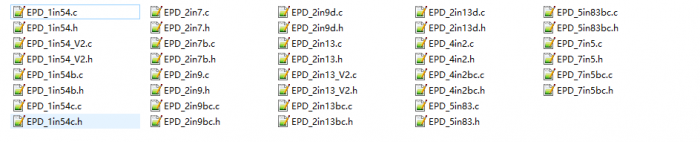
void DEV\_Digital\_Write(UWORD Pin, UBYTE Value);

UBYTE DEV\_Digital\_Read(UWORD Pin);

* SPI Write data

void DEV\_SPI\_WriteByte(UBYTE Value);

**EPD driver**

For Raspberry Pi and Jetson Nano, epd driver are saved in:RaspberryPi&JetsonNano\c\lib\e-Paper  
For STM32, the epd driver are saved in: STM32\STM32-F103ZET6\User\e-Paper  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_HAT_RPI_epd.png)  
Open .h file, functions are declarated here

* Initialization: It should be used to initialize e-Paper or wakeup e-Paper from 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 Initialize full refresh; Mode = 1 Initilize partial refresh //Other types*void EPD\_xxx\_Init(void);

xxx is the type of e-paper, for example, if the e-paper you have is 2inch e-Paper (D), then it should be EPD\_2IN13D\_Init(0) or EPD\_2IN13D\_Init(1); If it is 7.5inch e-Paper (B), the function should be EPD\_7IN5BC\_Init(). B type and C type of 7.5inch e-Paper use the same codes.

* 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. For example, if the e-Paper you have is 4.2inch e-Paper, it should be EPD-4IN2\_Clear()

* Transmit a frame of image and display

*//Black/White e-Paper*void EPD\_xxx\_Display(UBYTE \*Image);*//Three colors e-Paper*void EPD\_xxx\_Display(const UBYTE \*blackimage, const UBYTE \*ryimage);

**There are some exceptions：**

*//To partial refresh 2.13inch e-paper (D)、2.9inch e-paper (D), you should use*

void EPD\_2IN13D\_DisplayPart(UBYTE \*Image);

void EPD\_2IN9D\_DisplayPart(UBYTE \*Image);

*//Because controllers of 1.54inch e-Paper V2 and 2.13inch e-Paper V2 were updated, you need to use EPD\_xxx\_DisplayPartBaseImage to display static image and ten use EPD\_xxx\_displayPart() to dymatic display when partial refreshing.*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);

*//Because STM32103ZET5 has no enough RAM for image, therefore 7.5B、7.5C、5.83B、5.83C can only display half of the screen:'''*void EPD\_7IN5BC\_DisplayHalfScreen(const UBYTE \*blackimage, const UBYTE \*ryimage);void EPD\_5IN83BC\_DisplayHalfScreen(const UBYTE \*blackimage, const UBYTE \*ryimage);

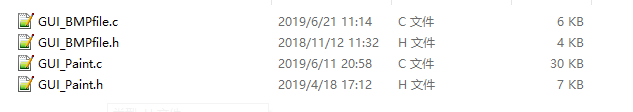
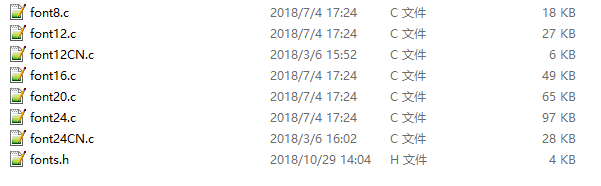
xxx is the type of e-Paper

* 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**

Basic drawing functions are provided here. You can find then in:  
Raspbian Pi & Jetson Nano: RaspberryPi&JetsonNano\c\lib\GUI\GUI\_Paint.c(.h)  
STM32: STM32\STM32-F103ZET6\User\GUI\GUI\_Paint.c(.h)  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_HAT_GUI.png)  
The fonts are saved in the directory:  
Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\c\lib\Fonts  
STM32: STM32\STM32-F103ZET6\User\Fonts  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_HAT_Fonts.png)

* 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)

Paratemeters:

image : The buffer of image, this is an 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 buffer with last function, then select the buffer for every image.

void Paint\_SelectImage(UBYTE \*image)

Parameters：

image: The name of image buffer, it is a pointer of buffer address;

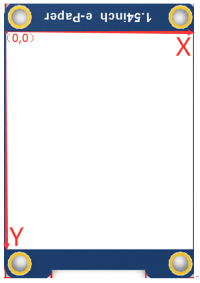
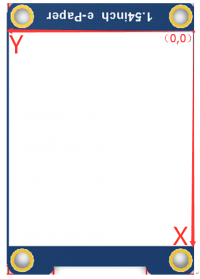
* Set display orientation: This function is used to set the rotate degree, it is generally be used after Paint\_SelectImage(). You can set the rotate degree to 0、90、180、270 degree.

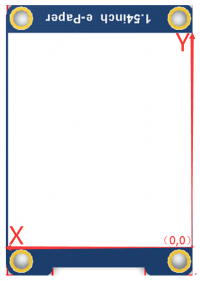
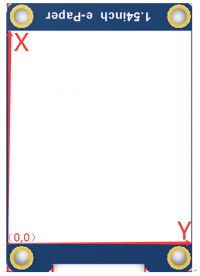
void Paint\_SetRotate(UWORD Rotate)

Parameters：

Rotate: Rotate degree, you can choose ROTATE\_0、ROTATE\_90、ROTATE\_180、ROTATE\_270 which stands for 0、90、180、270 degree repetitively.

【Note】Three figures below shows the display effect in differen degree. (0°, 90°, 180°, 270°)

[](https://www.waveshare.com/wiki/File:SPI-epaper-C-0.png)[](https://www.waveshare.com/wiki/File:SPI-epaper-C-90.png)

[](https://www.waveshare.com/wiki/File:SPI-epaper-C-180.png)[](https://www.waveshare.com/wiki/File:SPI-epaper-C-270.png)

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

void Paint\_SetMirroring(UBYTE mirror)

Paramters：

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)

Parameters：

Xpoint: X-axes in 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)

Parameters：

Xstart: Start coordinate of X-axes of window;

Ystart: Start coordinate of Y-axes of window;

Xend: End coordinate of X-axes of window;

Yend: End coordinate of Y-axes of 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\_RIGHTUP,

} DOT\_STYLE;

* Draw line: draw a line for (Xstart, Ystart) to (Xend, Yend)

void Paint\_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, 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: color of line

Line\_width: the width of line, 8 sizes are avalilable;

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;

* 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)

Parameter：

Xstart: Start coordinate of X-axes of rectangle

Ystart: Start coordinate of Y-axes of rectangle

Xend: End coordinate of X-end of rectangle

Yend: End coordinate of Y-end of rectangle

Color: color of rectangle

Line\_width: The width of edges, 8 sides 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;

* Draw circle:Draw a circle, use (X\_Center Y\_Center) as center;

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 center

Y\_Center: Y coordinate of center

Radius：Radius of circle

Color: color of circle

Line\_width: width of circle, 8 sizes are avalilable

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;

* Draw character (ASCII): Set(Xstart Ystart) as letf-top point, draw a ASCII character.

void Paint\_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii\_Char, sFONT\* 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 character;

Ascii\_Char：Ascii character;

Font: 5 fonts are available;

font8：5\*8

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, 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 background

* Draw Chinese charactgers: this function is used to draw Chinese fonts based ON GB2312 fonts.

void Paint\_DrawString\_CN(UWORD Xstart, UWORD Ystart, const char \* pString, cFONT\* 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：11\*21(ascii)，16\*21 (Chinese)

font24CN：24\*41(ascii)，32\*41 (Chinese)

Color\_Foreground: color of string

Color\_Background: color of background

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

void Paint\_DrawNum(UWORD Xpoint, UWORD Ypoint, int32\_t Nummber, sFONT\* Font, UWORD Color\_Foreground, UWORD Color\_Background)

Parameter：

Xstart: X coordinate of left-top pixel;

Ystart: Y coordicate of left-to pixel;

Nummber：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: volor 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, PAINT\_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;

font8：5\*8

font12：7\*12

font16：11\*16

font20：14\*20

font24：17\*24

Color\_Foreground: color of fonts

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: adrress 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)

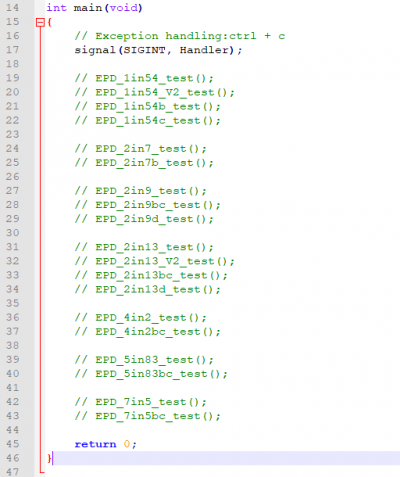
Parameter：

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**

In the above part, we describe about the tree structures of linux codes, here we talk about the testing code for user.  
Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\c\examples;  
The codes in exampleas are testing code, you can modify the definition in main.c file for different types of e-Paper.  
[](https://www.waveshare.com/wiki/File:E-Paper_Shield_c_test.png)  
For example, if you want to test 7.5inch e-paper, you need to delete the "//" symbol on line 42.

// EPD\_7in5\_test();

change it to

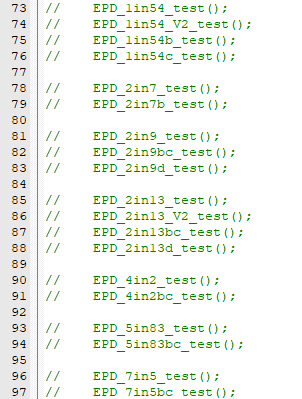
EPD\_7in5\_test();

Then compile it again and run

make clean

make

sudo ./epd

STM32:STM32\STM32-F103ZET6\User\Examples;  
testing codes are saved in this folder, open project, and then modify the definition stentences in main.c file;  
Open project：STM32\STM32-F103ZET6\MDK-ARM\epd-demo.uvprojx  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_stm32_main.png)  
For example, if you want to test 7.5inch e-paper, you should delete the "//" symble of on line 96

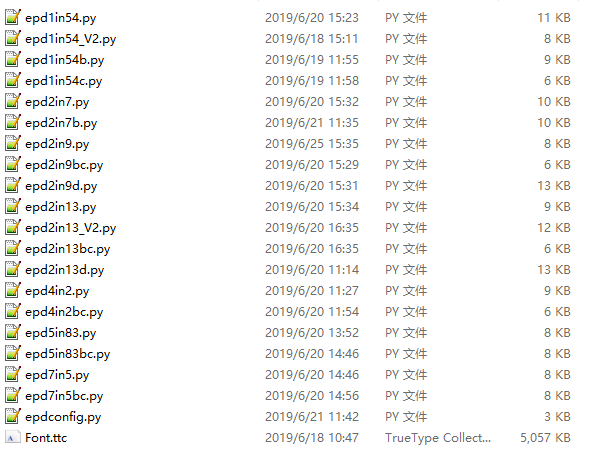
// EPD\_7in5\_test();

Change it to

EPD\_7in5\_test();

Then re-compile project and donwload it

**Python(Used for Jetson Nano\Raspberry Pi)**

Supports python2.7 and python3  
python is easy to use than c codes  
Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\python\lib\  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_python_lib.png)

**epdconfig.py**

* Initialize module and exit handle：

def module\_init()

def module\_exit()

Note：

1.The functions are used to set GPIP before and after driving e-Paper.

2.If the board you have is printed with Rev2.1, module enter low-ultra mode after Module\_Exit(). (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 data

def spi\_writebyte(data)

**epdxxx.py(xxx is the type of the e-Paper)**

* 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 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)

def init(self, update) # update should be lut\_full\_update or lut\_partial\_update

Other types:

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 imgae data and display

#For two-color e-paper

def display(self, image)#For three-color e-Paper

def display(self, blackimage, redimage)

There are several excepation：<br />

For flexible e-Paper 2.13inch e-paper (D)、2.9inch e-paper (D), the partial refresh should use

def DisplayPartial(self, image)#Because that controllers of 1.54inch e-paper V2、2.13inch e-paper V2 are updated, when partial refresh, they should first use displayPartBaseImage() to display static background, then use displayPart() to dynamaticlly display.

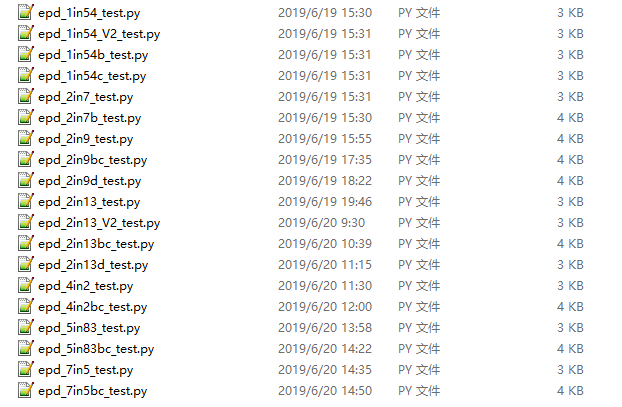
def displayPartBaseImage(self, image)

def displayPart(self, image)

* Enter sleep mode

def sleep(self)

**epd\_xxx\_test.py(xxx is type of e-paper)**

python examples are saved in directory：  
Raspberry Pi & Jetson Nano：RaspberryPi&JetsonNano\python\examples\  
[](https://www.waveshare.com/wiki/File:E-paper_Driver_python_examples.png)  
If the python installed in your OS is python2, you should run examples like below：

sudo python epd\_7in5\_test.py

If it is python3, the commands should be:

sudo python3 epd\_7in5\_test.py

Note: You can change epd\_7inch5\_test.py to the certain type you use.

**Orientation**

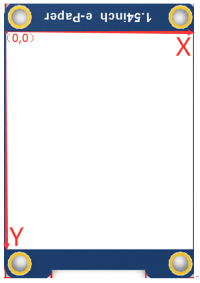
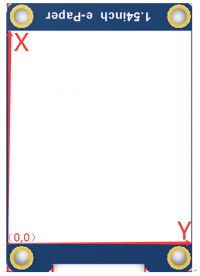
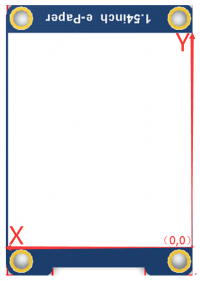
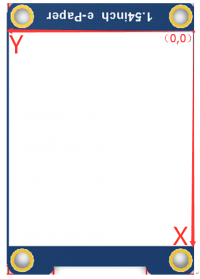
To rotate the display, you can use transpose function

blackimage = blackimage.transpose(Image.ROTATE\_270)

redimage = redimage.transpose(Image.ROTATE\_270)

#Supports OTATE\_90, ROTATE\_180, ROTATE\_270

【Note】Three figures below shows the display effect in different degree. (0°, 90°, 180°, 270°)

[](https://www.waveshare.com/wiki/File:SPI-epaper-Python-0.png)[](https://www.waveshare.com/wiki/File:SPI-epaper-Python-90.png)[](https://www.waveshare.com/wiki/File:SPI-epaper-Python-180.png)[](https://www.waveshare.com/wiki/File:SPI-epaper-Python-270.png)

**Arduino**

Because Arduino doesn't have full RAM for display dynamatic image, we don't provide other functions for it. If you want to use Arduino, we recommend you to use Waveshare [e-paper Sheild](https://www.banggood.com/Waveshare-e-Paper-Electronic-Paper-ink-Screen-Driver-Board-Expansion-Board-compatible-with-NUCLEO-For-Bare-Screen-p-1754771.html).

# Resources

# Documentation

* [Instruction about make new font](https://wavesharejfs.blogspot.com/2018/08/make-new-larger-font-for-waveshare-spi.html" \t "https://www.waveshare.com/wiki/_blank)
* [Make BMP file for e-Paper](https://www.waveshare.com/wiki/E-Paper_Floyd-Steinberg" \t "https://www.waveshare.com/wiki/_blank)
* [Schematic](https://www.waveshare.com/wiki/File:E-Paper-Driver-HAT-Schematic.pdf" \o "File:E-Paper-Driver-HAT-Schematic.pdf)

# Demo code

* [Github](https://github.com/waveshare/e-Paper" \t "https://www.waveshare.com/wiki/_blank)

# Datasheets

* [Datasheet](https://www.waveshare.com/wiki/File:7.5inch_e-paper-b-Specification.pdf" \o "File:7.5inch e-paper-b-Specification.pdf)
* [7.5inch e-Paper B V2 Specification](https://www.waveshare.com/w/upload/4/44/7.5inch_e-Paper_B_V2_Specification.pdf" \t "https://www.waveshare.com/wiki/_blank)
* [7.5inch e-Paper B V3 Specification](https://www.waveshare.com/w/upload/8/8c/7.5inch-e-paper-b-v3-specification.pdf" \t "https://www.waveshare.com/wiki/_blank)

## Related Resources

Notice:  
The projects listed are all made and shared by the project owners, Waveshare isn't responsible for project either the update.

* [Waveshare e-Paper display with SPI](https://forum.arduino.cc/index.php?topic=487007.0" \t "https://www.waveshare.com/wiki/_blank)

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

* [Inkycal Project](https://github.com/aceisace/Inkycal" \t "https://www.waveshare.com/wiki/_blank)

This is the Inkycal project for reference.

* [E-Paper Calendar with iCal sync and live weather (shared by user)](https://github.com/aceisace/E-Paper-Calendar-with-iCal-sync-and-live-weather" \t "https://www.waveshare.com/wiki/_blank)

# FAQ

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 1:** | | Working requirements of e-Paper? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 1:** | | * Two-color B/W e-paper * 【Working】Temperature: 0~50°C; Humidity: 35%~65%RH * 【Storage】Temperature: ≤30°C; Humidity: ≤55%RH; Max storage time: 6 months * 【Transport】Temperature: -25~70°C; Max transport time: 10 days * 【Unpack】Temperature: 20°C±5°C; Humidity: 50%RH±5%RH; Max storage time: Should be assembled in 72h * Three-Color e-Paper * 【Working】Temperature: 0~40°C; Humidity: 35%~65%RH * 【Storage】Temperature: ≤30°C; Humidity: ≤55%RH; Max storage time: 3 months * 【Transport】Temperature: -25~60°C; Max transport time: 10 days * 【Unpack】Temperature: 20°C±5°C; Humidity: 50%RH±5%RH; Max storage time: Should be assembled in 72h   **When store three-color e-Paper, please refresh it to white, and keep the screen upward. Note that you need to update it at least every three months.** | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 2:** | | What do you need to note about e-Paper refreshing? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 2:** | | * Refresh mode   + Full refresh: e-Paper flicker when full refreshing to remove ghost image for best display.   + Partial refresh: It don't flicker if you use partial refresh (only some of the two-color e-paper support partial refresh). Note that you cannot use Partial refresh all the time, you should full refresh e-paper regularly, otherwise, ghost problem will get worse even damage. * Refresh rate   + When using, you should set the update interval at least 180s.(except Partial supportable types)   + Please set the e-Paper to sleep mode or power off it directly, otherwise, the e-Paper is damaged because of working in high voltage for long time.   + You need to update the content of three-color e-Paper at least one time every 24h to avoid from burn-in problem. * Working place * We recommend you to use the e-Paper indoor. If you need to set the e-paper outdoor, Please place the e-paper in shadow and protect it from UV. When you designed you e-paper product, you should take care about the working situation like temperature, humidity, etc. | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 3:** | | How much could the flexible e-paper be bended | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 3:** | | * The IC part of e-Paper cannot be bended, you can bend the display area in degree larger than 60°C | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 4:** | | Why the e-Paper cant work with Arduino? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 4:** | | The I/O level of Arduino is 5V, and the e-Paper should be driven with 3V3. If your Arduino cant drive the e-Paper successfully, please try to convert the level to 3.3V  You can also try to connect the Vcc pin to the 5V of Arduino to see whether the e-Paper works, but we recommend you not to use 5V for a long time. | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 5:** | | Why does the color of e-Paper look a little black or grey? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 5:** | | You can try to change the value of Vcom on demo codes. | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 6:** | | Three-color e-paper looks more red/yellow than the picture on website? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 6:** | | Because of different batch, some of them have aberration. Store the e-Paper right side up will reduce it. And if the e-Paper didn't be refreshed for long time, it will become more and more red/yellow. Please use the demo code to refresh the e-paper for several times in this case. | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 7:** | | Why my e-paper has ghosting problem after working for some days | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 7:** | | Please set the e-paper to sleep mode or disconnect it if you needn't refresh the e-paper but need to power on your development board or Raspberry Pi for long time.Otherwise, the voltage of panel keeps high and it will damage the panel | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Question 8:** | | Why the FPC of the e-Paper is broken after using for some times? | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Answer 8:** | | Please make sure you have used it in correct way.  [IMG_256](https://www.waveshare.com/wiki/File:E-Paper_FAQ_FPC.jpg) | |