2.7inch 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.

264x176, 2.7inch E-Ink display HAT for Raspberry Pi, three-color

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)

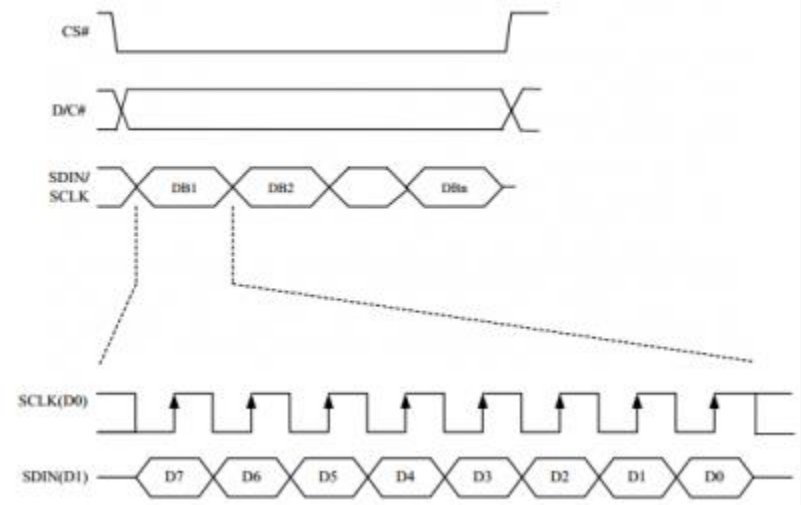
Working principle

Introduction

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 sunshine, the E-paper screen still has high visibility with a wide viewing angle of 180 degree. It is the ideal choice for E-reading.

Communication protocol



SPI timing

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.

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 for Arduino board

Raspberry Pi

Hardware connection

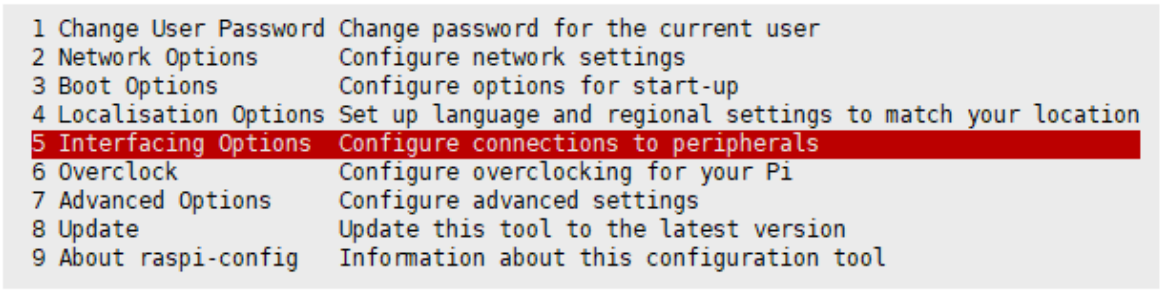
If the board you get is the HAT version like 2.13inch e-Paper HAT, you can directly attach it on the 40PIN GPIO of Raspberry Pi. Or you can wire ti to Raspberry Pi with 8PIN cable.

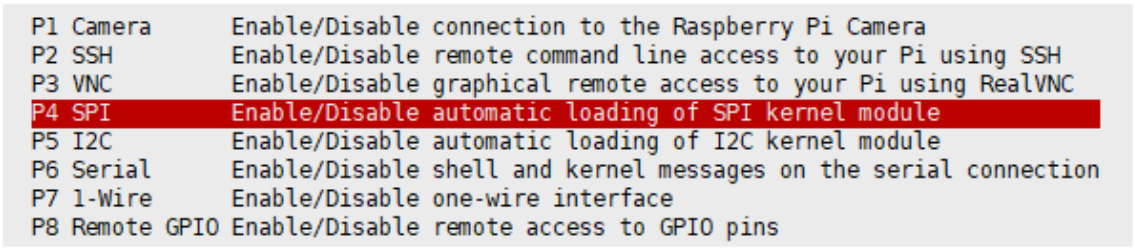
Connect to Raspberry Pi		
e-Paper	Raspberry Pi	
	BCM2835	Board
VCC	3.3V	3.3V
GND	GND	GND
DIN	MOSI	19
CLK	SCLK	23
CS	CE0	24

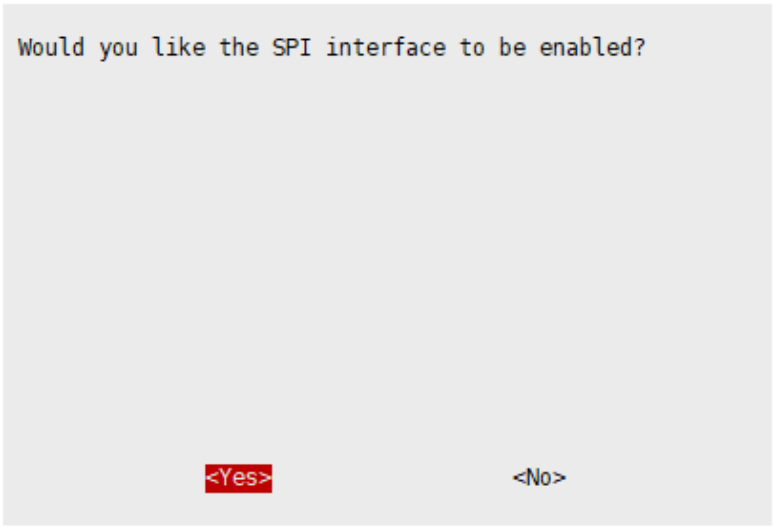
DC	25	22
RST	17	11
BUSY	24	18

- Open terminal, use command to enter the configuration page

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







Reboot Raspberry Pi：

```
sudo reboot
```

Please makre sure that SPI interface was not used by other device

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
```

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 bo

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 the hardware interface, the codes are same;

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

Hardware interface

Because of multiple hardware platforms, we packge 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:：
```

```
USELIB = USE_BCM2835_LIB
# USELIB = USE_WIRINGPI_LIB
DEBUG = -D $(USELIB)
ifeq ($(USELIB), USE_BCM2835_LIB)
    LIB = -lbcm2835 -lm
else ifeq ($(USELIB), USE_WIRINGPI_LIB)
    LIB = -lwiringPi -lm
endif
```

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：

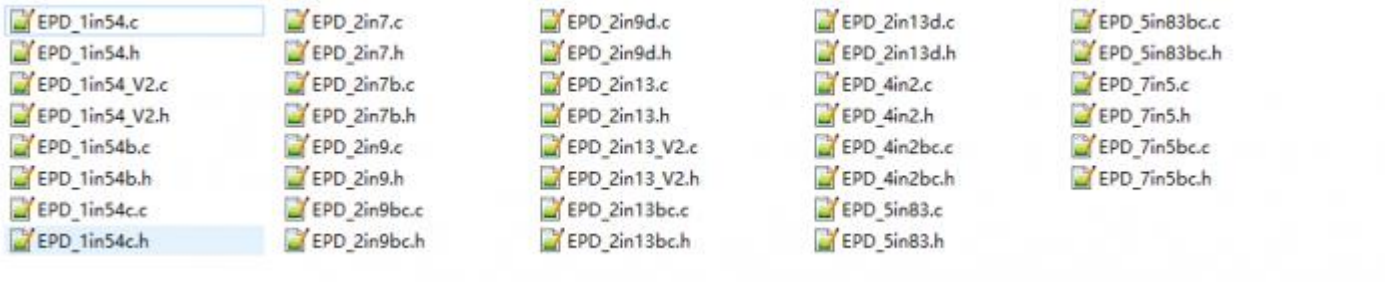
```
void DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE DEV_Digital_Read(UWORD Pin);
```

- 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-F1037ET6\User\e-Paper



Open .h file, functions are declared 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

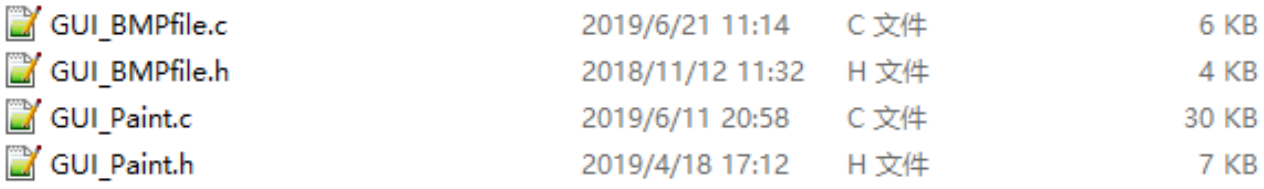
- Enter sleep mode

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

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



The fonts are saved in the directory:
Raspberry Pi & Jetson Nano: RaspberryPi&JetsonNano\c\lib\Fonts

STM32: STM32\STM32-F103ZET6\User\Fonts

font8.c	2018/7/4 17:24	C 文件	18 KB
font12.c	2018/7/4 17:24	C 文件	27 KB
font12CN.c	2018/3/6 15:52	C 文件	6 KB
font16.c	2018/7/4 17:24	C 文件	49 KB
font20.c	2018/7/4 17:24	C 文件	65 KB
font24.c	2018/7/4 17:24	C 文件	97 KB
font24CN.c	2018/3/6 16:02	C 文件	28 KB
fonts.h	2018/10/29 14:04	H 文件	4 KB

- 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;
```

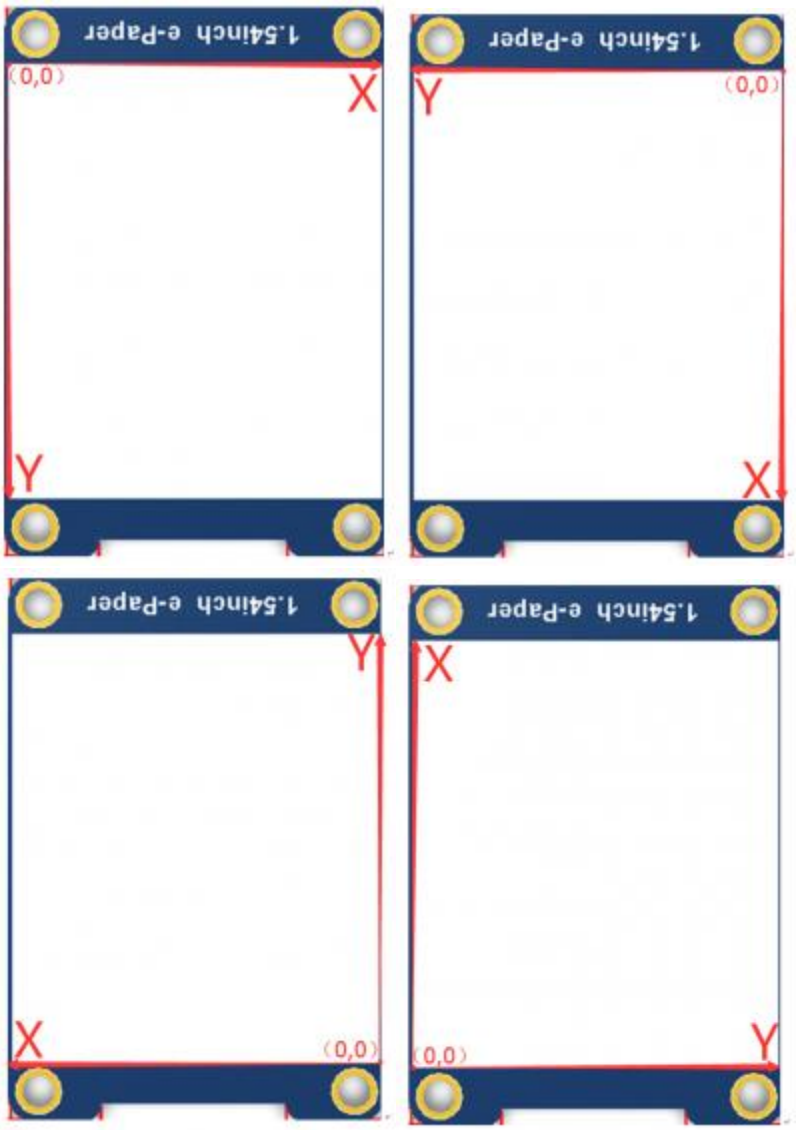
- Select image buffer: this function is used to select the image buffer. You can create multiple image buffer with last function, then select the buffer for every image.

```
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°)



- 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 availilable;
        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

```
14  int main(void)
15  {
16      // Exception handling:ctrl + c
17      signal(SIGINT, Handler);
18
19      // EPD_1in54_test();
20      // EPD_1in54_V2_test();
21      // EPD_1in54b_test();
22      // EPD_1in54c_test();
23
24      // EPD_2in7_test();
25      // EPD_2in7b_test();
26
27      // EPD_2in9_test();
28      // EPD_2in9bc_test();
29      // EPD_2in9d_test();
30
31      // EPD_2in13_test();
32      // EPD_2in13_V2_test();
33      // EPD_2in13bc_test();
34      // EPD_2in13d_test();
35
36      // EPD_4in2_test();
37      // EPD_4in2bc_test();
38
39      // EPD_5in83_test();
40      // EPD_5in83bc_test();
41
42      // EPD_7in5_test();
43      // EPD_7in5bc_test();
44
45      return 0;
46  }
47
```

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

```
73  //    EPD_1in54_test();
74  //    EPD_1in54_V2_test();
75  //    EPD_1in54b_test();
76  //    EPD_1in54c_test();
77
78  //    EPD_2in7_test();
79  //    EPD_2in7b_test();
80
81  //    EPD_2in9_test();
82  //    EPD_2in9bc_test();
83  //    EPD_2in9d_test();
84
85  //    EPD_2in13_test();
86  //    EPD_2in13_V2_test();
87  //    EPD_2in13bc_test();
88  //    EPD_2in13d_test();
89
90  //    EPD_4in2_test();
91  //    EPD_4in2bc_test();
92
93  //    EPD_5in83_test();
94  //    EPD_5in83bc_test();
95
96  //    EPD_7in5_test();
97  //    EPD_7in5bc_test();
```

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\

epd1in54.py	2019/6/20 15:23	PY 文件	11 KB
epd1in54_V2.py	2019/6/18 15:11	PY 文件	8 KB
epd1in54b.py	2019/6/19 11:55	PY 文件	9 KB
epd1in54c.py	2019/6/19 11:58	PY 文件	6 KB
epd2in7.py	2019/6/20 15:32	PY 文件	10 KB
epd2in7b.py	2019/6/21 11:35	PY 文件	10 KB
epd2in9.py	2019/6/25 15:35	PY 文件	8 KB
epd2in9bc.py	2019/6/20 15:29	PY 文件	6 KB
epd2in9d.py	2019/6/20 15:31	PY 文件	13 KB
epd2in13.py	2019/6/20 15:34	PY 文件	9 KB
epd2in13_V2.py	2019/6/20 16:35	PY 文件	12 KB
epd2in13bc.py	2019/6/20 16:35	PY 文件	6 KB
epd2in13d.py	2019/6/20 11:14	PY 文件	13 KB
epd4in2.py	2019/6/20 11:27	PY 文件	9 KB
epd4in2bc.py	2019/6/20 11:54	PY 文件	6 KB
epd5in83.py	2019/6/20 13:52	PY 文件	8 KB
epd5in83bc.py	2019/6/20 14:46	PY 文件	8 KB
epd7in5.py	2019/6/20 14:46	PY 文件	8 KB
epd7in5bc.py	2019/6/20 14:56	PY 文件	8 KB
epdconfig.py	2019/6/21 11:42	PY 文件	3 KB
Font.ttc	2019/6/18 10:47	TrueType Collect...	5,057 KB

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 type of the e-Paper)

- Initailize e-paper: this function should be used at the begining. 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),
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在 examples are saved in directory：
Raspberry Pi & Jetson Nano：RaspberryPi&JetsonNano\python\examples\

epd_1in54_test.py	2019/6/19 15:30	PY 文件	3 KB
epd_1in54_V2_test.py	2019/6/19 15:31	PY 文件	3 KB
epd_1in54b_test.py	2019/6/19 15:31	PY 文件	3 KB
epd_1in54c_test.py	2019/6/19 15:31	PY 文件	3 KB
epd_2in7_test.py	2019/6/19 15:31	PY 文件	3 KB
epd_2in7b_test.py	2019/6/19 15:30	PY 文件	4 KB
epd_2in9_test.py	2019/6/19 15:55	PY 文件	4 KB
epd_2in9bc_test.py	2019/6/19 17:35	PY 文件	4 KB
epd_2in9d_test.py	2019/6/19 18:22	PY 文件	4 KB
epd_2in13_test.py	2019/6/19 19:46	PY 文件	3 KB
epd_2in13_V2_test.py	2019/6/20 9:30	PY 文件	3 KB
epd_2in13bc_test.py	2019/6/20 10:39	PY 文件	4 KB
epd_2in13d_test.py	2019/6/20 11:15	PY 文件	3 KB
epd_4in2_test.py	2019/6/20 11:30	PY 文件	3 KB
epd_4in2bc_test.py	2019/6/20 12:00	PY 文件	4 KB
epd_5in83_test.py	2019/6/20 13:58	PY 文件	3 KB
epd_5in83bc_test.py	2019/6/20 14:22	PY 文件	4 KB
epd_7in5_test.py	2019/6/20 14:35	PY 文件	3 KB
epd_7in5bc_test.py	2019/6/20 14:50	PY 文件	4 KB

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
```

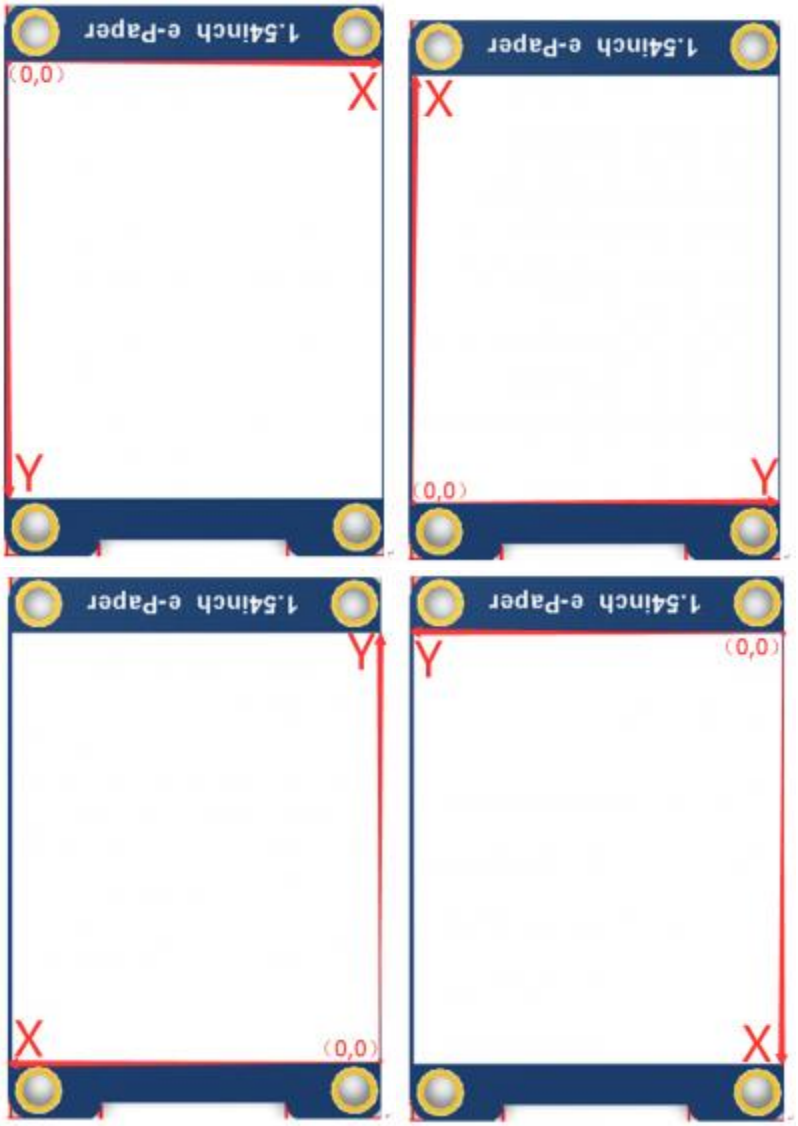
Note: You can change epd_7inch5_test.py to the certain type you use.

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 differen degree. (0°, 90°, 180°, 270°)



Documentation

- User Manual
- Instruction about make new font
- Schematic

Demo code

- Github

Datasheets

- Datasheets

Related Resources

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

- Waveshare e-Paper display with SPI

Related applications

- Raspberry Pi E-paper Display demo