E-Paper API Analysis

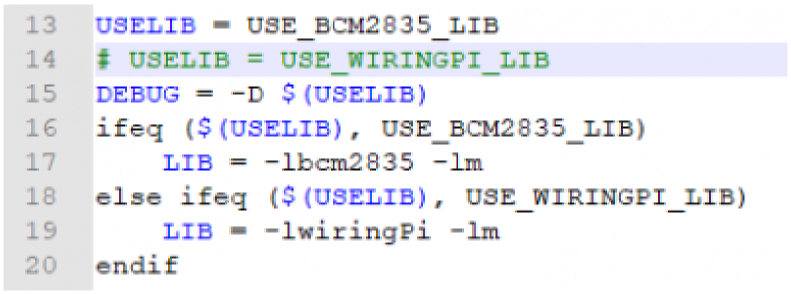
From Waveshare Wiki
Jump to: navigation, search

Raspberry Pi and Jetson Nano use the same program. As they are both embedded systems, their compatibilities are high.
Examples contain three parts, hardware interface, EPD driver and the GUI functions.

C

Hardware interface

We have carried out the underlying package. As the hardware platform is different, the internal is various. If you need to know the internal, 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, and WiringPi library. The BCM2835 library is used by default. If you need to use the WiringPi library to drive, you can open RaspberryPi_JetsonNano\c\Makefile and modify lines 13-14.



```
13 USELIB = USE_BCM2835_LIB
14 # USELIB = USE_WIRINGPI_LIB
15 DEBUG = -D $(USELIB)
16 ifeq ($(USELIB), USE_BCM2835_LIB)
17     LIB = -lbcm2835 -lm
18 else ifeq ($(USELIB), USE_WIRINGPI_LIB)
19     LIB = -lwiringPi -lm
20 endif
```

(/wiki/File:E-paper_Driver_HAT_RPI_Makefile.png)

- 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() functions are used to configure GPIOs. EPD enters 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

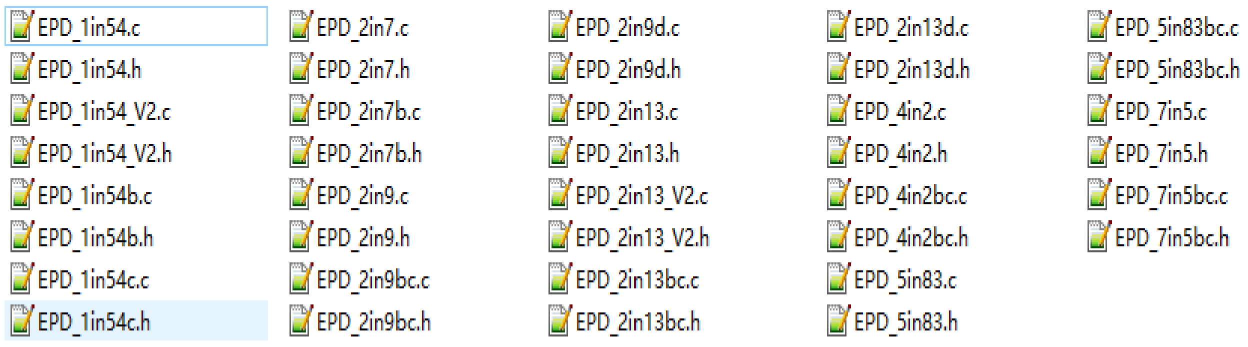
```
void DEV_Digital_Write(UWORD Pin, UBYTE Value);
UBYTE DEV_Digital_Read(UWORD Pin);
```

- SPI transmits data.

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

EPD Driver

The driver file is saved under RaspberryPi&JetsonNano\c\lib\e-Paper.



EPD_1in54.c
EPD_1in54.h
EPD_1in54_V2.c
EPD_1in54_V2.h
EPD_1in54b.c
EPD_1in54b.h
EPD_1in54c.c
EPD_1in54c.h
EPD_2in7.c
EPD_2in7.h
EPD_2in7b.c
EPD_2in7b.h
EPD_2in9.c
EPD_2in9.h
EPD_2in9bc.c
EPD_2in9bc.h
EPD_2in9d.c
EPD_2in9d.h
EPD_2in13.c
EPD_2in13.h
EPD_2in13_V2.c
EPD_2in13_V2.h
EPD_2in13bc.c
EPD_2in13bc.h
EPD_2in13d.c
EPD_2in13d.h
EPD_4in2.c
EPD_4in2.h
EPD_4in2bc.c
EPD_4in2bc.h
EPD_5in83.c
EPD_5in83.h
EPD_5in83bc.c
EPD_5in83bc.h
EPD_7in5.c
EPD_7in5.h
EPD_7in5bc.c
EPD_7in5bc.h

(/wiki/File:E-paper_Driver_HAT_RPI_epd.png)

Open .h to see the following functions

- EPD initialization: please use it when the EPD starts to work or exits the 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-Pape
r, 2.9inch e-Paper (D )
void EPD_xxx_Init(UBYTE Mode); // Mode = 0 full refresh initialization, Mode = 1 partial 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(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

- Clear the screen, and refresh 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_2IN9D_Clear(); if it is 7.5B, it is EPD_7IN5_Clear(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

- 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 (D), 2.9inch e-paper (D) two flexible screens, partial refresh
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 a partial refresh, it is necessary to call EPD_xxx_DisplayPartBaseImage to display the static 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);
```

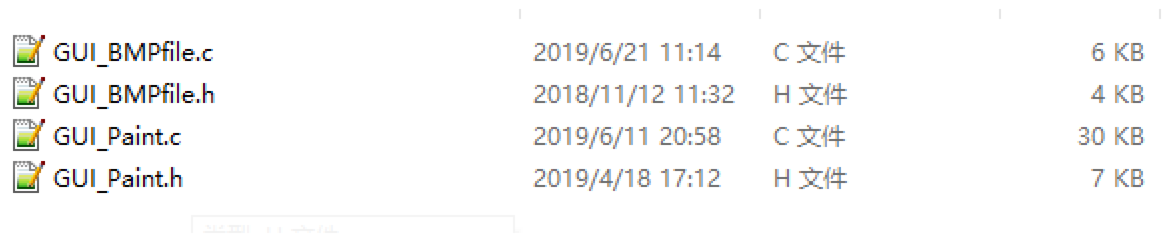
- Enter sleep mode

```
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(). Although the display colors are different, the driver code of 7.5B and 7.5C is the same.

GUI functions

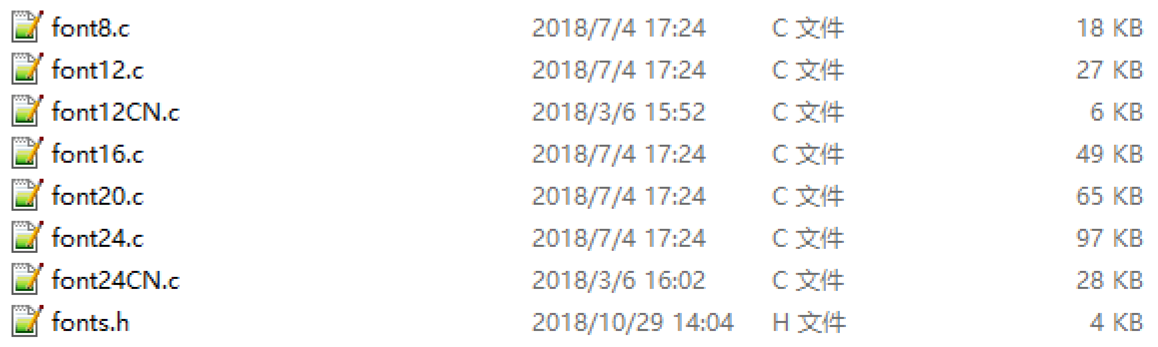
GUI files can be found in RaspberryPi & JetsonNano\c\lib\GUI\GUI_Paint.c(.h) directory.



GUI_BMPfile.c	2019/6/21 11:14	C 文件	6 KB
GUI_BMPfile.h	2018/11/12 11:32	H 文件	4 KB
GUI_Paint.c	2019/6/11 20:58	C 文件	30 KB
GUI_Paint.h	2019/4/18 17:12	H 文件	7 KB

(/wiki/File:E-paper_Driver_HAT_GUI.png)

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



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

(/wiki/File:E-paper_Driver_HAT_Fonts.png)

Create an image buffer

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

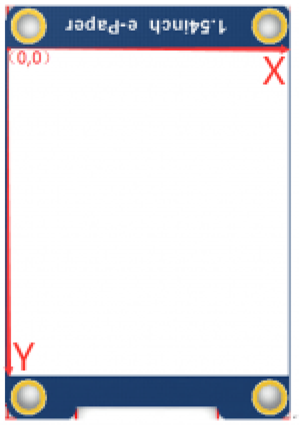
- The image buffer is a pointer of the image buffer's first address.

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 orientations, the position of the first pixel is different, here we take 1.54inch as an example.



(/wiki/File:SPI-epaper-C-0.png)SPI-epaper-C-90.png (/wiki/File:SPI-epaper-C-90.png)
SPI-epaper-C-180.png (/wiki/File:SPI-epaper-C-180.png)SPI-epaper-C-270.png (/wiki/File: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 types 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 partial 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 windows

Draw point

This function is used to draw points.

```
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_RIGHTUP,
                 } DOT_STYLE;
```

Draw Line

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD Color, LINE_STYLE Line_Style , LINE_STY
LE 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, 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: the color of the 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 as full or empty.

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

Draw a circle

In the image cache, with (X_Center Y_Center) as the center, draw a circle with a Radius, you can choose the color, the width of the line, and 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_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(Xstart Ystart) as the left-top point, and draw an ASCII character.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* Font, UWORD Color_Foreground, UWORD Col
or_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: the color of character;
- Color_Background: the color of background;

Draw String

Set point (Xstart Ystart) as the left-top pixel, and 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 fonts based ON GB2312 fonts.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, cFONT* font, UWORD Color_Foreground, UWORD Color_Background)
```

Parameters:

- 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 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, 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: the color of the font;
- Color_Background: the color of the background;

Draw image

Send image data of BMP file to buffer

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

Parameters:

- image_buffer: the first address of image data in the buffer.

Read the local BMP picture and write it to buffer.

Linux platforms like Jetson Nano and Raspberry Pi support directly operating 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

Test code.png (/wiki/File:Test_code.png)

The first three chapters introduce the classic Linux three-layer code structure, 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 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_5in65f_test();
```

to

```
EPD_5in65f_test();
```

Then compile it again and run.

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

Python

For Jetson Nano\Raspberry Pi, based on python2.7 and python3.
For python, its calls are not as complicated as C.
Raspberry Pi and Jetson Nano: RaspberryPi_JetsonNano\python\lib\

Epd.png (/wiki/File:Epd.png)

Bottom Interface

The epdconfig.py file package the underlying interface.

- 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, the module enters 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)
```

Driver Interface

epdxxx.py (xxx means 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 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-Pa
per, 2.9inch e-Paper (D)
def init(self, update) # choose lut_full_update or lut_partial_update
other models
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 the screen
```

- Convert image to arrays

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

- Transmit one frame of image data and display

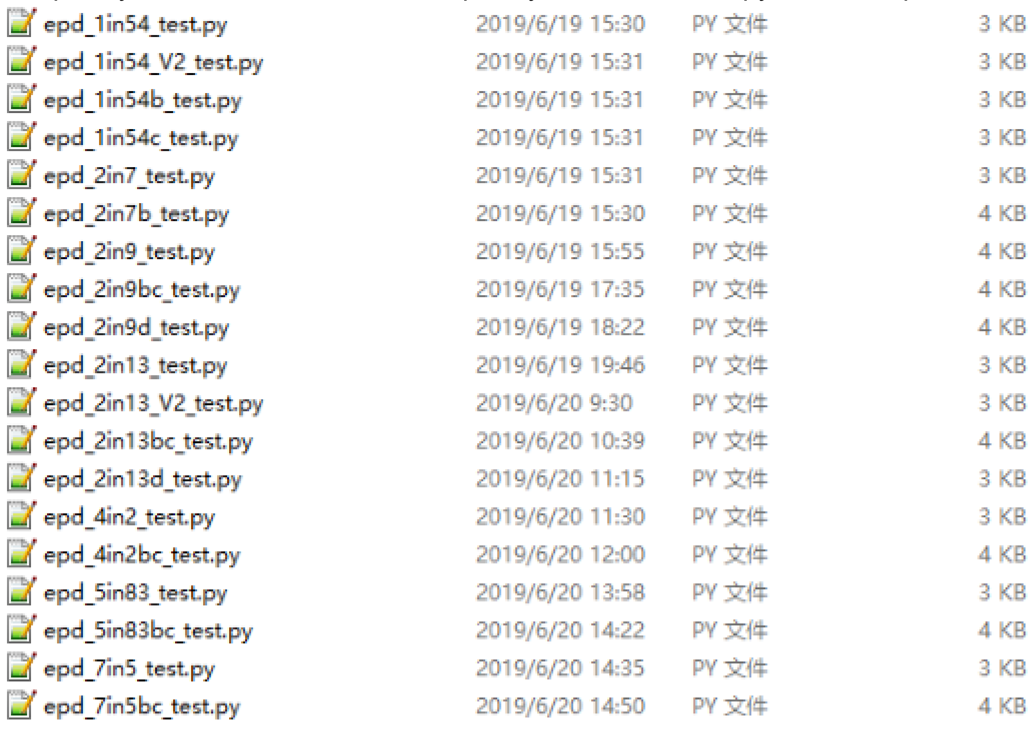
```
#For two-color e-paper
def display(self, image)
#For Black, White, Red Or Black, White, Yellow e-paper
def display(self, blackimage, redimage)

#Note that the following are special cases:
#For 2.13inch e-paper (D) and 2.9inch e-paper (D) these two flexible screens, partially refresh them
def DisplayPartial(self, image)

#For 1.54inch e-paper V2 and 2.13inch e-paper V2, you need to use "displayPartBaseImage()" to display the still backg
round picture for a partial refresh, that is, use the dynamic "displayPart()" to refresh partially based on this stil
l picture.
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 the directory：
Raspberry Pi and 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

(/wiki/File:Testing_function.png)

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

```
sudo python epd_2in13_V2_test.py
```

If it is python3, the commands should be:

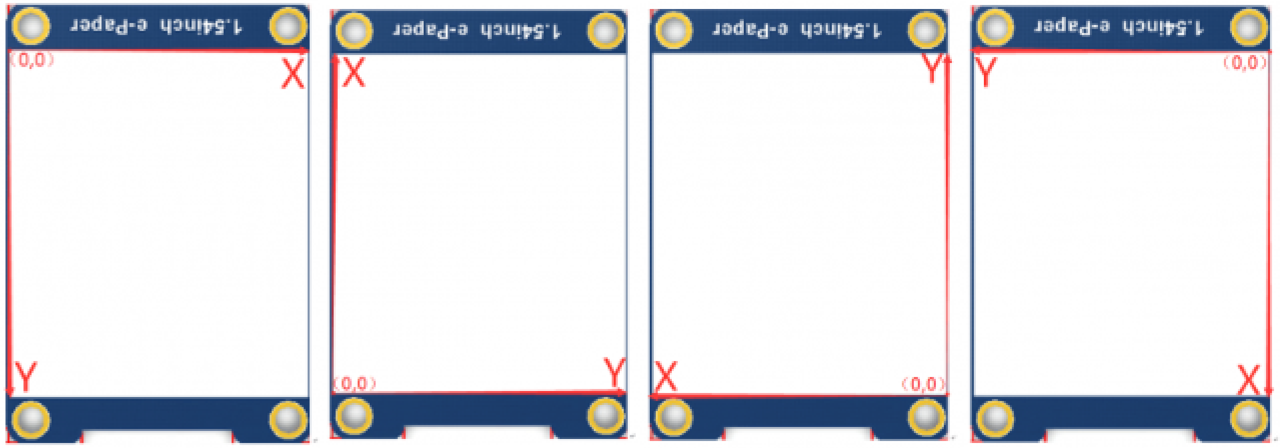
```
sudo python3 epd_2in13-V2_test.py
```

Orientation

To rotate the display, you can use 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
```

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



(/wiki/File:Orientation.png)

GUI

Python has a powerful PIL library (http://effbot.org/imagingbook), 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: fonts

- 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 the 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 the rectangle, and 200, 34) is the right-bottom point. fille = 0 set the fill 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, and 200, 34 is the endpoint. fill=0 set the line to black.

- Draw circle

```
draw.arc((90, 60, 150, 120), 0, 360, fill = 0)
```

This function is used to draw a encircle 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 degrees of the circle. The fourth parameter is the fill color of the circle.

- Character

You can directly import the 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 format.

To draw English character, you can directly use the fonts; for Chinese characters, you need to add a symbol u:

```
draw.text((8, 12), 'hello world', font = font, fill = 255)
draw.text((8, 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 las one is set the color.

- Read local picture

```
image = Image.open(os.path.join(picdir, 'lin54.bmp'))
```

The parameter is the path of picture.

- Other functions.

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

Retrieved from "https://www.waveshare.com/w/index.php?title=E-Paper_API_Analysis&oldid=48137 (https://www.waveshare.com/w/index.php?title=E-Paper_API_Analysis&oldid=48137)"