



2.9inch e-Paper Module

User Manual

Version	Date	Description
V1.0	2017.05.17	Factory
V2.0	2018.12.05	Notes

【Notes】

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

CONTENT

Type Description	4
Notes	5
Overview.....	7
Specification	7
Interfaces	8
Working Principle.....	9
4-wire SPI.....	9
Demo codes	11
Download	11
Raspberry Pi.....	12
Hardware connection.....	12
Copy examples to Raspberry Pi.....	12
Libraries Installation	13
Running demo codes.....	15
Expected result.....	16
Arduino.....	17
Hardware connection.....	17
Running examples.....	17
Expected result.....	18
STM32.....	19
Hardware connection.....	19

Running example.....	19
Expected result.....	19
Codes Description	20
Version	20
C codes.....	20
Python codes	23
Image Data Array	26

TYPE DESCRIPTION

Generally, there are two version for every type of e-paper, one is raw panel, and another is Module/HAT version.

Raw panel should work with driver board, if you are the first time to use Waveshare e-Paper, we recommend you the Module/HAT version or purchase a driver board separately.

Module/HAT version has PCB which integrates driver circuit. You can connect 8Pin connector to MCU directly.

Raw Panel (2.9inch e-Paper):



Module Version (2.9inch e-Paper Module):



NOTES

1. For those e-Paper which support partial refresh, you cannot use partial refresh all the time. A full refresh should be done to clear screen after several times(partial refresh), otherwise, e-Paper will be damaged and cannot fixed.
2. Three-color e-Paper is normal to be a little 'color' . You can refresh it to white and keep it upward for storage.
3. The e-Paper cannot be powered on for long time, you must set e-Paper to sleep mode or power off when it needn' t refresh, otherwise, e-Paper keeps in high voltage status for long time, which will damage e-Paper and cannot be fixed.
4. We suggest you update e-Paper once every 24 hours or at least 10 days¹ to update again. Otherwise, ghost of the last content may cannot be cleared.
5. e-Paper ignores the data sent when it is in sleep mode, you need to initialize it for properly refreshing.
6. You can adjust border color by controlling 0x3C register. In some of demo codes, you can adjust Border Waveform Control or VCOM AND DATA INERTVAL SETTING registers.
7. If you find that the image data you made cannot be properly displayed on e-Paper, please check the size of image, or change its width and height and try again.

¹ For details of operating requirement, please refer to datasheet

8. The e-Paper cannot refresh directly under sunlight². The refresh steps should be done indoor
9. For raw panel, its working voltage is 3.3V, note that when you designed you own driver board, level convert circuit is required if the working voltage of your board is 5V. Module/HAT version can support 5V voltage if the Module you buy is the new version which has been integrated convert circuit.
10. The FPC of the panel is fragile, please make sure that your bend it in correct way.
11. The glass raw panel is fragile, please do not falling, crashing or pressing hard.
12. We recommend you test the e-Paper with our demo code when you first time receive it.

² e-Paper don't have UV film, cannot work under sunlight directly

OVERVIEW

- This is an E-Ink display module, 2.9inch, 296x128 resolution, with embedded controller, communicating via SPI interface, support partial refresh.
- Due to the advantages like ultra-low power consumption, wide viewing angle, clear display without electricity, it is an ideal choice for applications such as shelf label, industrial instrument, and so on.
- We provide demo codes for you (Arduino, Raspberry Pi and STM32)

SPECIFICATION

- Operating Voltage: 3.3V/5V³
- Working Temperature: 0~50°C
- Interface: 3-wire SPI/4-wire SPI
- Dimension:
 - PCB: 89.5mm × 38mm
 - Raw Panel: 79.0mm × 36.7mm
- Display area: 66.89mm × 29.05mm
- Dot pitch: 0.138 × 0.138
- Resolution: 296 × 128
- Display color: Black, White
- Refresh time⁴:
 - Partial refresh: ≈0.3s

³ The voltage of every pins should be same 3.3V or 5V

⁴ These are experimental data; actual data may be different according to working situation

- Full refresh: 2s
- Consumption⁵:
 - Refresh consumption: 26.4mW(typ.)
 - Standby consumption: <0.017mW
- Viewing angle: >170°

INTERFACES

VCC : 3.3V/5V

GND : GND

DIN : SPI MOSI pin

CLK : SPI SCK pin

CS : SPI chip selection, low active

DC : Data/Command Selection (High: Data; Low: Command)

RST : Reset (Low active)

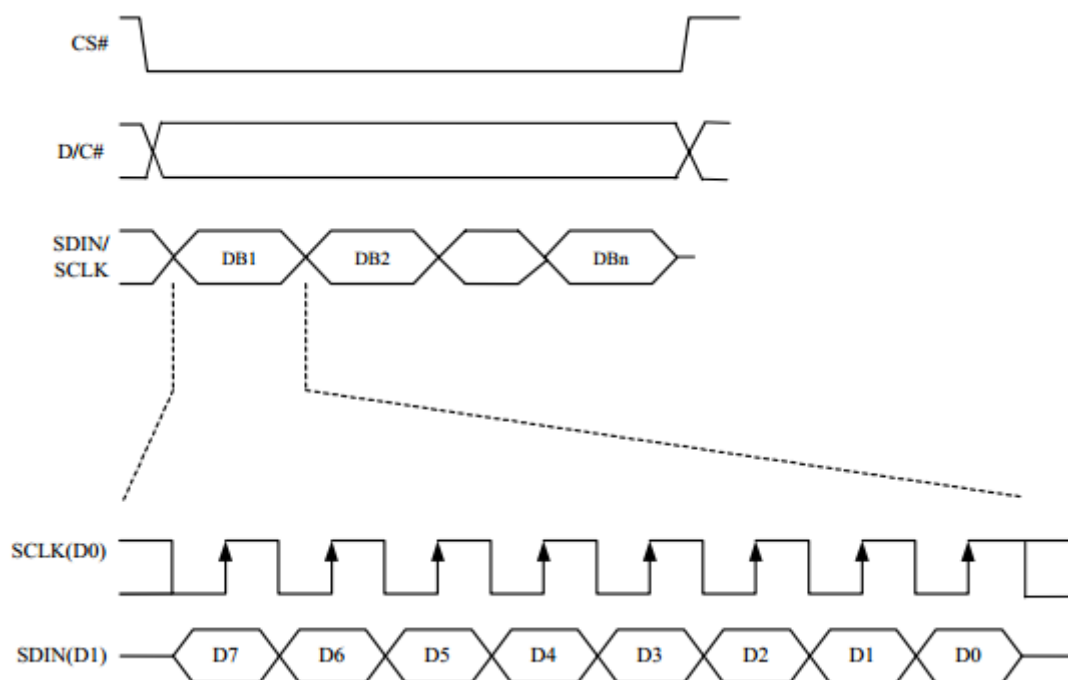
BUSY : Busy (High active)

⁵ These are experimental data; actual data may be different according to working situation

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 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.

4-WIRE SPI



Different from the traditional SPI protocol⁶, 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.

⁶ You can search online for more information about traditional SPI protocol

- 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.

DEMO CODES

DOWNLOAD

Please visit Waveshare Wiki, search with key words "1.54inch e-Paper module.

Download the demo code from [wiki](#),

Resources

Documentation

- [Instruction about make new font](#)⁷
- [Schematic](#)

Demo code

- [Demo code](#)

Extract it and you can get these folders:

	Arduino	2018/11/26 19:18	文件夹
	RaspberryPi	2018/11/24 17:27	文件夹
	STM32	2018/11/26 19:18	文件夹

Arduino⁷: Demo code for Arduino UNO;

RaspberryPi: Demo codes for Raspberry Pi. (BCM2835, wiringPi and python)

STM32: Demo code for STM32, control board is STM32F103ZET6

⁷ If you use other Arduino board, you should check if it is compatible with Arduino Uno, or modification is required.

RASPBERRY PI

There are four examples for Raspberry Pi, bcm2835, wiringPi, python2 and python3.

HARDWARE CONNECTION

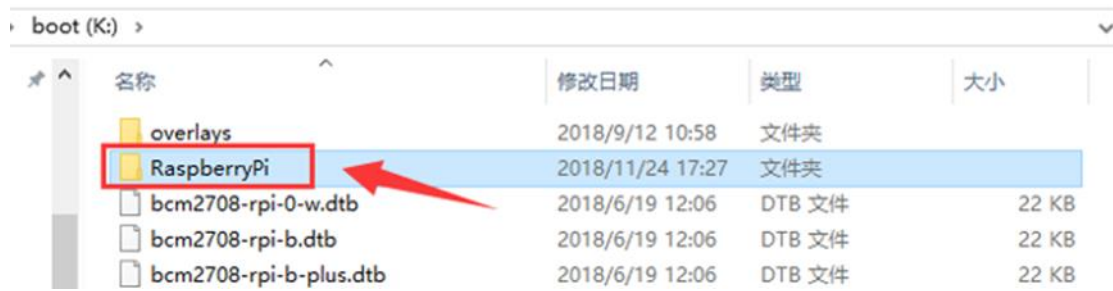
e-Paper	Raspberry Pi	
	BCM2835 No.	Board No.
3.3V	3.3V	3.3V
GND	GND	GND
DIN	MOSI	19
CLK	SCLK	23
CS	CE0	24
DC	25	22
RST	17	11
BUSY	24	18

COPY EXAMPLES TO RASPBERRY PI

1. Insert SD card which has Raspbian installed to your PC



2. Copy RaspberryPi extracted to root directory (BOOT) of SD card



3. Power on your Raspberry Pi and open Terminal, you can find that the examples is listed in boot directory

```
pi@raspberrypi:~$ ls /boot/
bcm2708-rpi-0-w.dtb  bcm2710-rpi-3-b.dtb  config.txt  fixup_x.dat  kernel.img  start_cd.elf
bcm2708-rpi-b.dtb   bcm2710-rpi-3-b-plus.dtb  COPYING.Linux  FSCK0000.REC  LICENCE.broadcom  start_db.elf
bcm2708-rpi-b-plus.dtb  bcm2710-rpi-cm3.dtb  fixup_cd.dat  FSCK0001.REC  LICENSE.oracle  start.elf
bcm2708-rpi-cm.dtb    bootcode.bin        fixup.dat     issue.txt    overlays      start_x.elf
bcm2709-rpi-2-b.dtb   cmdline.txt          fixup_db.dat  kernel7.img  RaspberryPi  System Volume Information
```

4. Copy the RaspberryPi folder to /home/pi and change its execute permission.

```
sudo cp -r /boot/RaspberryPi/ ./
```

```
sudo chmod 777 -R RaspberryPi/
```

```
pi@raspberrypi:~$ sudo cp -r /boot/RaspberryPi/ ./
pi@raspberrypi:~$ ls
code  libcode  RaspberryPi  RPiLib  ubuntu  usbdisk
pi@raspberrypi:~$ sudo chmod 777 -R RaspberryPi/
pi@raspberrypi:~$ ls
code  libcode  RaspberryPi  RPiLib  ubuntu  usbdisk
```

LIBRARIES INSTALLATION

To use demo codes, libraries should be installed first.

BCM2835 LIBRARY

Download link of the bcm2835 library: <http://www.airspayce.com/mikem/bcm2835/>

Download the library and copy it to raspberry pi without extracting. Open terminal of raspberry pi and install:

```
sudo tar zxvf bcm2835-1.xx.tar.gz

cd bcm2835-1.xx

sudo ./configure

make

sudo make check

sudo make install
```

Note: xx is version of the library you download. For example, if the version is bcm2835-1.52, you should complete the command to: sudo tar zxvf bcm2835-1.52.tar.gz

WIRINGPI LIBRARY

Open Terminal and install wiringPi with commands below:

```
sudo apt-get install git

sudo git clone git://git.drogon.net/wiringPi

cd wiringPi

sudo ./build
```

PYTHON2 LIBRARY

Open Terminal and execute commands to install:

```
sudo apt-get install python-pip

sudo apt-get install python-pip

sudo apt-get install python-imaging

sudo pip install spidev
```

```
sudo pip install RPi.GPIO
```

PYTHON3 LIBRARY

Open Terminal and install library with commands:

```
sudo apt-get install python3-pip  
  
sudo apt-get install python-imaging  
  
sudo pip3 install spidev  
  
sudo pip3 install RPi.GPIO  
  
sudo pip3 install Pillow
```

If you get the error while installing Pillow: **ImportError: libopenjp2.so.7: cannot open shared object file: No such file or directory.** Please install libopenjp2-7-dev with the command: **sudo apt-get install libopenjp2-7-dev** then try again.

RUNNING DEMO CODES

Enter folder of examples

Running bcm2835 example:

```
cd ~/RaspberryPi/bcm2835    #enter the example directory  
  
make                        #compile codes  
  
sudo ./epd                  #running
```

Running wiringPi example:

```
cd ~/RaspberryPi/wiringpi   #enter the example directory  
  
make                         #Compile  
  
sudo ./epd                   #running
```

Running python2 example:

```
cd ~/RaspberryPi/python2    #enter example directory  
  
sudo python main.py         #running
```

Running python3 example:

```
cd ~/RaspberryPi/python3    #enter example directory  
  
sudo python3 main.py        #running
```

EXPECTED RESULT

After running example:

- 1) The screen is refresh to white
- 2) Display a picture
- 3) Draw circles and lines, text is displayed
- 4) Display time (Partial refresh)

You can press ctrl and c at the same time to stop code.

ARDUINO

Note: If the driver board you bought is e-Paper shield, you can use the demo codes provided for e-Paper shield separately which can be downloaded in corresponding wiki.

HARDWARE CONNECTION

If you use other Arduino board instead of Arduino UNO, you should check if its interfaces are compatible with UNO.

e-Paper	Arduino UNO
3.3V	3V3
GND	GND
DIN	D11
CLK	D13
CS	D10
DC	D9
RST	D8
BUSY	D7

RUNNING EXAMPLES

1. Make sure you have installed Arduino IDE in your PC
2. Open Arduino project
3. Select correct Board and Port
4. Compile and upload to Arduino board

EXPECTED RESULT

- 1) e-Paper is refreshed to white
- 2) Draw circles and lines, text is displayed
- 3) Display time (Partial refresh)

STM32

The development board used is Open103z, project is based on HAL library.

HARDWARE CONNECTION

e-Paper	Open103Z
3.3V	3V3
GND	GND
DIN	PA7
CLK	PA5
CS	PA4
DC	PA2
RST	PA1
BUSY	PA3

RUNNING EXAMPLE

1. Open STM32 project with keil V5
2. Compile and download the demo code to STM32 board

EXPECTED RESULT

- 1) e-paper is refreshed to white
- 2) Display a picture
- 3) Draw circles and lines, display characters
- 4) Display time (Partial refresh)

CODES DESCRIPTION

VERSION

The newest demo codes are V2.0 version, updated functions:

- 1) read pictures
- 2) python3 examples
- 3) update display function
- 4) fix bug that e-Paper cannot enter sleep mode

C CODES

Take bcm2835 codes as example:

1. Initialize pins and SPI interface:

```
DEV_ModuleInit () ;
```

2. Initialize e-Paper and clear it:

```
...if (EPD_Init (lut_full_update) != 0) {
...    printf ("e-Paper init failed\r\n");
...}
...EPD_Clear ();
...DEV_Delay_ms (500);
```

3. Create an image buffer

```
...//Create a new image cache
...UBYTE *BlackImage;
...UWORD Imagesize = ((EPD_WIDTH % 8 == 0) ? (EPD_WIDTH / 8) : (EPD_WIDTH / 8 + 1)) * EPD_HEIGHT;
...if ((BlackImage = (UBYTE *) malloc (Imagesize)) == NULL) {
...    printf ("Failed to apply for black memory...\r\n");
...    exit (0);
...}
...printf ("Paint_NewImage\r\n");
...Paint_NewImage (BlackImage, EPD_WIDTH, EPD_HEIGHT, 270, WHITE);
...Paint_SelectImage (BlackImage);
...Paint_Clear (WHITE);
```

Create an image buffer and set its size as: Imagesize = EPD width/8 * EPD height.

Paint_newImage: Create a paint for new image, parameter 1 is image buffer,

parameter 2 and 3 is width and height of image, parameter 4 is color if image

Paint_SelectImage: Select image

4. Read image and display

```

... Paint_Clear(WHITE);
... Paint_SelectImage(BlackImage);
... Paint_Clear(WHITE);
...
... GUI_ReadBmp("./pic/100x100.bmp", 50, 10);
... EPD_Display(BlackImage);
... DEV_Delay_ms(500);
...
... Paint_Clear(WHITE);
... GUI_ReadBmp("./pic/2in9.bmp", 0, 0);
... EPD_Display(BlackImage);
... DEV_Delay_ms(2000);
#endif

```

Paint_SelectImage: Select image;

Paint_Clear(WHITE): Set image color to white

GUI_ReadBmp: read bmp for certain path. Parameter 1: the address of BMP

picture locates, Parameter 2 and 3 is the display position x and y of BCM display.

Note that if the picture read is bigger than the image buffer, the rest part cannot be displayed.

EPD_Display: Send image data to e-paper and display

5. Read image data from array

```

... Paint_SelectImage(BlackImage);
... Paint_Clear(WHITE);
... Paint_DrawBitMap(gImage_2in9);
...
... EPD_Display(BlackImage);
... DEV_Delay_ms(2000);
#endif

```

Paint_DrawBitBmp: send image data to e-paper. Because STM32 and Arduino

cannot read picture directly, we need to first convert picture to data on PC

6. Drawing point, lines, rectangles, circles and characters

```
#if 1 // Drawing on the image
... // 1. Select Image
... printf("SelectImage:BlackImage\r\n");
... Paint_SelectImage(BlackImage);
... Paint_Clear(WHITE);

... // 2. Drawing on the image
... printf("Drawing:BlackImage\r\n");
... Paint_DrawPoint(10, 80, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
... Paint_DrawPoint(10, 90, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
... Paint_DrawPoint(10, 100, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
... Paint_DrawLine(20, 70, 70, 120, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
... Paint_DrawLine(70, 70, 20, 120, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
... Paint_DrawRectangle(20, 70, 70, 120, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
... Paint_DrawRectangle(80, 70, 130, 120, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
... Paint_DrawCircle(45, 95, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
... Paint_DrawCircle(105, 95, 20, WHITE, DRAW_FILL_FULL, DOT_PIXEL_1X1);
... Paint_DrawLine(85, 95, 125, 95, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
... Paint_DrawLine(105, 75, 105, 115, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
... Paint_DrawString_EN(10, 0, "waveshare", &Font16, BLACK, WHITE);
... Paint_DrawString_EN(10, 20, "hello world", &Font12, WHITE, BLACK);
... Paint_DrawNum(10, 33, 123456789, &Font12, BLACK, WHITE);
... Paint_DrawNum(10, 50, 987654321, &Font16, WHITE, BLACK);
... Paint_DrawString_CN(130, 0, "你好abc树莓派", &Font12CN, BLACK, WHITE);
... Paint_DrawString_CN(130, 20, "微雪电子", &Font24CN, WHITE, BLACK);

... printf("EPD_Display\r\n");
... EPD_Display(BlackImage);
... DEV_Delay_ms(2000);
#endif
```

Paint_DrawPoint: Drawing points, parameter 1 and 2 are position of the point,

parameter 3 are color, parameter 4 are size, parameter 5 are style

Paint_DrawLine: Drawing line, parameter 1 and 2 are begin position of line,

parameter 3 and 4 are end position of line, parameter 5 is color, parameter 6 is

style, parameter 7 is width of line

Paint_DrawRectangle: Drawing rectangle, parameter 1 and 2 are begin position of

rectangle, parameter 3 and 4 are end position of rectangle, parameter 5 is fill

color, parameter 6 and 7 are style

Paint_DrawCircle: Drawing circle, parameter 1 and 2 are position of center,

parameter 3 is radius of circle, parameter 4 and 5 are style

Paint_DrawString_EN: Display English characters, parameter 1 and 2 are position, parameter 3 is string, parameter 4 is font size, parameter 5 and 6 are background and font colors

Paint_DrawString_CN: Display Chinese characters

Paint_DrawNum: Display number, parameter 1 and 2 are position, parameter 3 is number (int), parameter 4 is size, parameter 5 and 6 are background and font colors.

7. Partial refresh

```
#if 1
...printf("show time...\r\n");
...if (EPD_Init(lut_partial_update) != 0) {
...    printf("e-Paper init failed\r\n");
...}
...Paint_SelectImage(BlackImage);
...PAINT_TIME sPaint_time;
...struct tm *t;
...time_t tt;
...for(;;){
...    time(&tt);
...    t = localtime(&tt);
...    sPaint_time.Hour = t->tm_hour;
...    sPaint_time.Min = t->tm_min;
...    sPaint_time.Sec = t->tm_sec;
...    Paint_ClearWindows(150, 70, 150 + Font24.Width * 7, 70 + Font24.Height, WHITE);
...    Paint_DrawTime(150, 70, &sPaint_time, &Font24, WHITE, BLACK);
...    EPD_Display(BlackImage);
...}
```

First initialize EPD as partial refresh mode. Then display time.

8. Sleep mode:

```
...printf("Goto Sleep mode...\r\n");
...EPD_Sleep();
...free(BlackImage);
...BlackImage = NULL;
```

It is recommended to add after refreshing.

PYTHON CODES

1. Initialize e-Paper

```
try:
    epd = epd2in9.EPD()
    epd.init(epd.lut_full_update)
    epd.Clear(0xFF)
```

2. Create image buffer using python image library

```
..
..# Drawing on the Horizontal image
Himage = Image.new('1', (epd2in9.EPD_HEIGHT, epd2in9.EPD_WIDTH), 255) ..# 255: clear the frame
..# Drawing on the Vertical image
Limage = Image.new('1', (epd2in9.EPD_WIDTH, epd2in9.EPD_HEIGHT), 255) ..# 255: clear the frame
..
```

3. Drawing with python imageDraw library

```
..# Horizontal
..print "Drawing"
..draw = ImageDraw.Draw(Himage)
..font24 = ImageFont.truetype('/usr/share/fonts/truetype/wqy/wqy-microhei.ttc', 24)
..draw.text((10, 0), 'hello world', font = font24, fill = 0)
..draw.text((10, 20), '2.9inch e-Paper', font = font24, fill = 0)
..draw.text((150, 0), 'u'微雪电子', font = font24, fill = 0) ...
..draw.line((20, 50, 70, 100), fill = 0)
..draw.line((70, 50, 20, 100), fill = 0)
..draw.rectangle((20, 50, 70, 100), outline = 0)
..draw.line((165, 50, 165, 100), fill = 0)
..draw.line((140, 75, 190, 75), fill = 0)
..draw.arc((140, 50, 190, 100), 0, 360, fill = 0)
..draw.rectangle((80, 50, 130, 100), fill = 0)
..draw.chord((200, 50, 250, 100), 0, 360, fill = 0)
..epd.display(epd.getbuffer(Himage))
..time.sleep(2)
..
..# Vertical
..draw = ImageDraw.Draw(Limage)
..font18 = ImageFont.truetype('/usr/share/fonts/truetype/wqy/wqy-microhei.ttc', 18)
..draw.text((2, 0), 'hello world', font = font18, fill = 0)
..draw.text((2, 20), '2.9inch epd', font = font18, fill = 0)
..draw.text((20, 50), 'u'微雪电子', font = font18, fill = 0)
..draw.line((10, 90, 60, 140), fill = 0)
..draw.line((60, 90, 10, 140), fill = 0)
..draw.rectangle((10, 90, 60, 140), outline = 0)
..draw.line((95, 90, 95, 140), fill = 0)
..draw.line((70, 115, 120, 115), fill = 0)
..draw.arc((70, 90, 120, 140), 0, 360, fill = 0)
..draw.rectangle((10, 150, 60, 200), fill = 0)
..draw.chord((70, 150, 120, 200), 0, 360, fill = 0)
..epd.display(epd.getbuffer(Limage))
..time.sleep(2)
```


4. Read picture and display

```

    .
    .print "read.bmp file"
    .Himage := Image.open('2in9.bmp')
    .epd.display(epd.getbuffer(Himage))
    .time.sleep(2)
    .
    .print "read.bmp file on window"
    .Himage2 := Image.new('1', (epd2in9.EPD_HEIGHT, epd2in9.EPD_WIDTH), 255) .# 255: clear the frame
    .bmp := Image.open('100x100.bmp')
    .Himage2.paste(bmp, (50,10))
    .epd.display(epd.getbuffer(Himage2))
    .time.sleep(2)
    .
  
```

5. Partial refresh

```

    .
    .# partial update
    .epd.init(epd.lut_partial_update) . . .
    .epd.Clear(0xFF)
    .time_image := Image.new('1', (epd2in9.EPD_HEIGHT, epd2in9.EPD_WIDTH), 255)
    .time_draw := ImageDraw.Draw(time_image)
    .while (True):
    . . . .time_draw.rectangle((10, 10, 120, 50), fill=255)
    . . . .time_draw.text((10, 10), time.strftime('%H:%M:%S'), font=font24, fill=0)
    . . . .newimage := time_image.crop([10, 10, 120, 50])
    . . . .time_image.paste(newimage, (10,10)) . .
    . . . .epd.display(epd.getbuffer(time_image))
  
```

6. Sleep mode

```

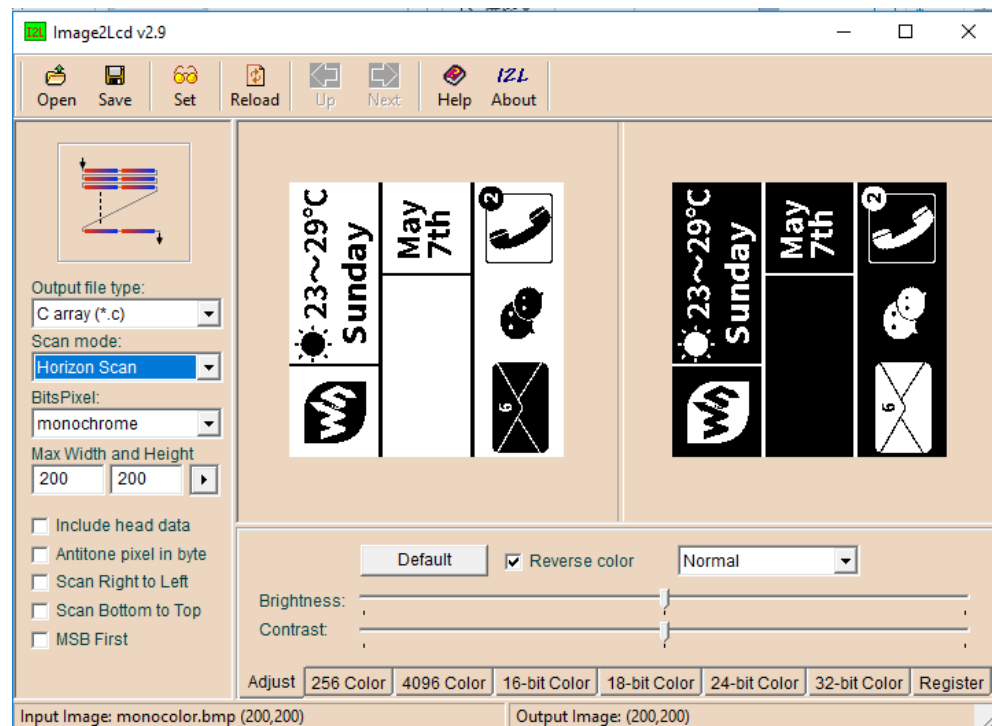
    .
    . . . .epd.sleep()
    . . . .
  
```

sleep mode is recommended to add after refreshing

IMAGE DATA ARRAY

1.54inch⁸ e-paper can only display black and white color.

- 1) Open a picture and set it as monochrome picture with Paint software of Windows
- 2) Resize the picture to supported resolution (according to e-paper)
- 3) Use Image2Lcd.exe software to convert picture to C array (.c file)



Open the picture in Image2Lcd.exe, configure:

Output file type: C array (*.c)

Scan mode: Horizon Scan

BitsPixel: monochrome

Max Width and Height: 200 200

Check Reverse color and click Save to generate the array. Copy to demo code for using.

⁸ Here take 1.54inch as example to show you how to create image data array