2.13inch e-Paper HAT (D) User Manual

OVERVIRE

- This is a flexible E-Ink display HAT for Raspberry Pi, 2.13inch, 212x104 resolution, with embedded controller, communicating via SPI interface, supports partial refresh.
- Due to the advantages like ultra-low power consumption, wide viewing angle, it is an ideal choice for applications such as shelf label, industrial instrument, and so on.

FEATURES

- No backlight, keeps displaying last content for a long time even when power down
- Supports partial refresh, partial refresh time is about 0.55s
- Ultra-low power consumption, basically power is only required for refreshing
- Could be driven with e-Paper HAT and e-Paper Shield
- Comes with development resources and manual (examples for Raspberry Pi/Arduino/STM32)

SPECIFICATIONS

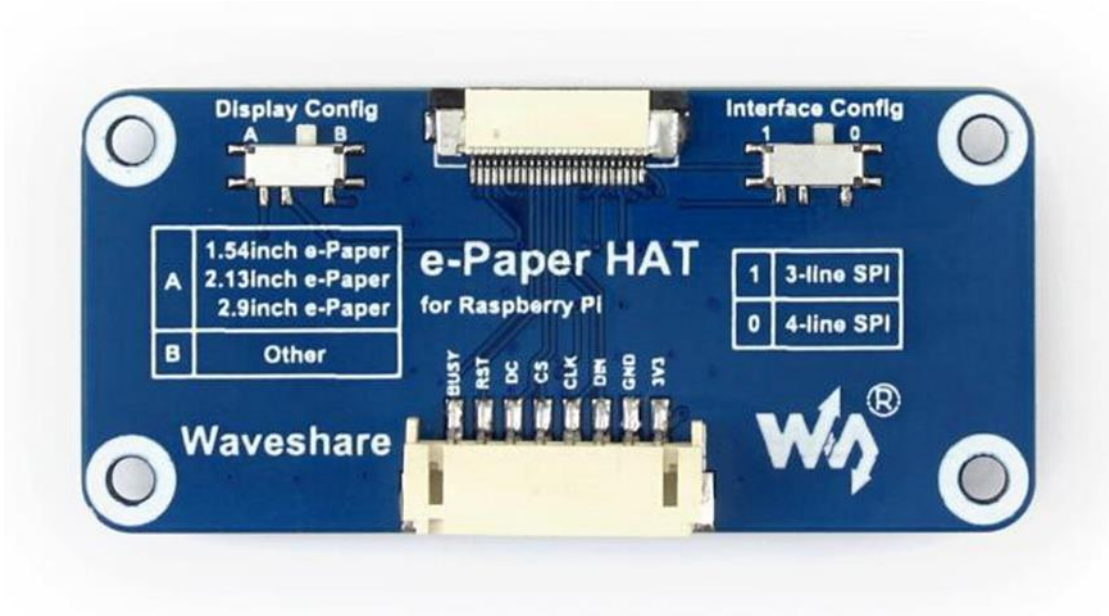
Operating voltage	: 3.3V
Interface	: SPI
Outline dimension	: 59.2(H) × 29.20(V) × 0.3(D)mm
Display size	: 48.55(H) × 23.71(V)mm
Dot pitch	: 0.229 × 0.228
Resolution	: 212(H) × 104(V)
Display color	: black, white
Grey level	: 2
Full refresh time	: 2.3s
Refresh power	: 26.4mW(typ.)
Standby power	: <0.017mW
Viewing angle	: >170°

PINS

VCC:	3.3V
GND:	Ground
DIN:	SPI MOSI pin
CLK:	SPI SCK pin
CS:	SPI chip selection, low active
DC:	Data/Command selection (high for data, low for command)
RST:	External reset, low active

SWITCHES

There are two switches on e-Paper Driver HAT: Display Config and Interface Config as Figure1:



1 E-Paper Driver HAT

DISPLAY CONFIG

The Display Config switch is used for various type of Waveshare e-paper raw panels. It is A and B sides. For different e-paper, it should be set as below:

B (With these e-Paper, should set to B)	**A** (with these types, should set to A)
1.54inch e-Paper (B)	1.54inch e-Paper
1.54inch e-Paper (C)	2.13inch e-Paper
2.13inch e-Paper (B)	2.13inch e-Paper (D)
2.13inch e-Paper (C)	2.9inch e-Paper
2.7inch e-Paper	
2.7inch e-Paper (B)	

2.9inch e-Paper (B)	
2.9inch e-Paper (C)	
4.2inch e-Paper	
4.2inch e-Paper (B)	
4.2inch e-Paper (C)	
7.5inch e-Paper	
7.5inch e-Paper (B)	
7.5inch e-Paper (C)	

INTERFACE CONFIG

Setting the Interface Config switch, you can choose the communicating interface.

If you set the switch into 1, the module will work with 3-line SPI;

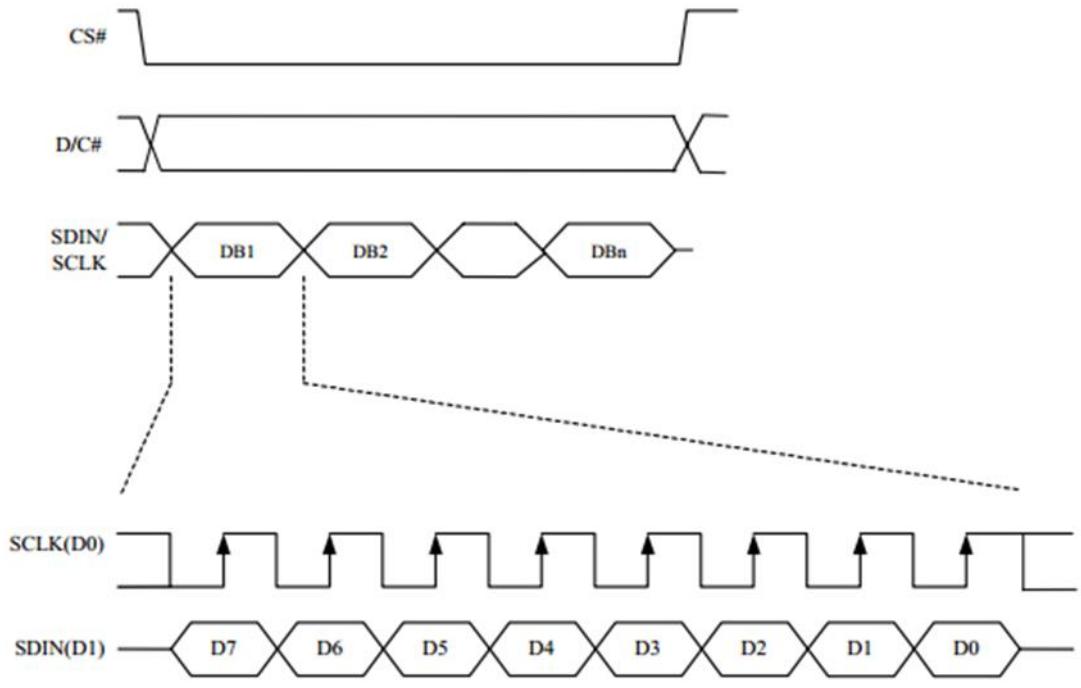
if you set the switch into 0, the module will work with 4-lin SPI;

The demo code we provide use 4-lin SPI.

HARDWARE DESCRIPTION

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



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.

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.

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.

DISPLAY CONTROL

There are two important registers for display, R10H and R13H refer to datasheet.

(9) Data Start Transmission 1 (DTM1) (R10H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Starting data transmission	0	0	0	0	0	1	0	0	0	0
	0	1	Pixel1	Pixel2	Pixel3	Pixel4	Pixel5	Pixel6	Pixel7	Pixel8
	0	1	..	..	..	..	..	..	..	..
	0	1	Pixel(n-7)	Pixel(n-6)	Pixel(n-5)	Pixel(n-4)	Pixel(n-3)	Pixel(n-2)	Pixel(n-1)	Pixel(n)

This command starts transmitting data and write them into SRAM.

In KW mode, this command writes "OLD" data to SRAM.

In KWR mode, this command writes "B/W" data to SRAM.

In Program mode, this command writes "OTP" data to SRAM for programming.

(12) Data Start Transmission 2(DTM2) (R13H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Starting data transmission	0	0	0	0	0	1	0	0	1	1
	0	1	Pixel1	Pixel2	Pixel3	Pixel4	Pixel5	Pixel6	Pixel7	Pixel8
	0	1	..	..	..	..	..	..	..	..
	0	1	Pixel(n-7)	Pixel(n-6)	Pixel(n-5)	Pixel(n-4)	Pixel(n-3)	Pixel(n-2)	Pixel(n-1)	Pixel(n)

This command starts transmitting data and write them into SRAM.

In KW mode, this command writes "NEW" data to SRAM.

In KWR mode, this command writes "RED" data to SRAM.

This e-Paper could only support black and white, that is KW mode. It requires to write old data to R10H and new data to R13H. As it is full refresh, you just need to change the old data to refresh white.

This e-Paper could also support partial refresh, there are R91H and R90H which is used to set the e-paper to partial refresh mode and set the display area. Then you can control R10G and R13H.

(36) Partial Window (PTL) (R90H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Set Partial Window	0	0	1	0	0	1	0	0	0	0
	0	1	HRST[7:3]					0	0	0
	0	1	HRED[7:3]					1	1	1
	0	1	-	-	-	-	-	-	-	VRST[8]
	0	1	VRST[7:0]							
	0	1	-	-	-	-	-	-	-	VRED[8]
	0	1	VRED[7:0]							
	0	1	-	-	-	-	-	-	-	PT_SCAN

This command sets partial window.

HRST[7:3]: Horizontal start channel bank. (value 00h~13h)

HRED[7:3]: Horizontal end channel bank. (value 00h~13h). HRED must be greater than HRST.

VRST[8:0]: Vertical start line. (value 000h~127h)

VRED[8:0]: Vertical end line. (value 000h~127h). VRED must be greater than VRST.

PT_SCAN: 0: Gates scan only inside of the partial window.

1: Gates scan both inside and outside of the partial window. (default)

(37) Partial In (PTIN) (R91H)

Action	W/R	C/D	D7	D6	D5	D4	D3	D2	D1	D0
Partial In	0	0	1	0	0	1	0	0	0	1

This command makes the display enter partial mode.

HOW TO USE

If you just have the raw panel, you need to buy one more driver board, otherwise it couldn’ t work. e-Paper HAT and e-Paper Shield are both OK.

If you want to use Arduino or NUCLEO, we recommend you buy the e-Paper Shield. And e-Paper HAT is more suitable for Raspberry Pi

E-PAPER HAT

WORKIGN WITH RASPBERRY PI

LIBRARIES INSTALLING

To use the demo code, you need to install necessary libraries first, otherwise the examples could not work properly. About how to install libraries (wiringPi, bcm2835 and python), please visit the page:

https://www.waveshare.com/wiki/Libraries_Installation_for_RPi

HARDWARE CONNECTION

The pins connection is based on Raspberry Pi 3B/3B+

e-Paper	Raspberry Pi
3.3V	3.3V
GND	GND
DIN	MOSI
CLK	SCLK

CS	CE0
DC	PIN22/P25
RST	PIN11/P17
BUSY	PIN18/P24

EXPECTED RESULT

1. After installing libraries, copy the example codes (you need to first download it for Wiki) to Raspbian, and enter to the directory:

   a) **bcm2835:** Execute command make to compile code and then command sudo ./epd to run the code

   b) **wiringpi:** Execute command make to compile code and then command sudo ./epd to run the code

   c) **python:** Execute command sudo python main.py to run the code

2. The e-Paper will display image as the code

Note: It will flicker during refresh, it is normal effect.

WORKING WITH ARDUINO

HARDWARE CONNECTION

This connection is based on UNO PLUS/Arduino UNO

e-Paper	Arduino
3.3V	3.3V
GND	GND

DIN	D11
CLK	D13
CS	D10
DC	D9
RST	D8
BUSY	D7

EXPECTED RESULT

1. Copy the files from the directory Arduino/libraries of the demo package to directory .../Arduino/libraries, which is under the installation directory of IDE.
2. Click the button Upload to compile and upload the program to your Arduino board.
3. The screen displays as the code.

Note:

- It will flicker during refresh, it is normal effect.
- The RAM of Arduino UNO is only 2k. It requires 2756 byte for every frame refreshing, the code itself will also cost some memory. Thus, if you use e-Paper HAT, it could provide demo code for static image display. For other function, you can use e-Paper Shield.

WORKING WITH STM32

- The demo code is based on STM32F103RB (XNUCLEOF103R)

- The demo code is based on HAL libraries, users could port the code to other STM chip with STM32CubeMX
- The demo code is compiled in Keil v5

HARDWARE CONNECTION

The connection is based on XNUCLEOF103R

e-Paper	STM32F103ZE
3.3V	3.3V
GND	GND
DIN	PA7(MOSI)
CLK	PA5(SCK)
CS	PB6
BUSY	PA8
DC	PC7
RST	PA9

EXPECTED RESULT

1. Open the Keil project (MDK-ARM/epd-demo.uvprojx)
2. Click Build to compile the project.
3. Click Download to download the program to the target board.
4. The screen displays after reset the board.

Note: It will flicker during refreshing, it is normal effect.

E-PAPER SHIELD

HARDWARE CONFIGURATION

- If your Arduino has ICSP interface, set the SPI Config switch of Shield into ICSP option (default)
- If your Arduino has not ICSP interface, set the SPI Config switch of Shield into SCLK\D12, MISO\D12, MOSI\D11 separately

DEMO CODES

We provide examples for Arduino UNO and XNUCLEO-F103RB

ARDUINO UNO

Open the Arduino Uno directory, the materials are as below:

You need to copy the Waveshare_ePaper folder which is in libraries directory to libraries which is located in the installation directory of IDE, which path is generally:

C:\Program Files (x86)\Arduino\libraries

Display pictures from SD card

1. Initialized e-Paper, and clear its buffer

```
GUIPAINT paint;
EPD2IN13D epd;
EPD_SDCARD SD;

Dev->System_Init();
Serial.print("2.13inch e-Paper D demo\n");
//Initialize e-Paper
if(epd.EPD_Init() != 0) {
    Serial.print("e-Paper init failed");
    return;
}
epd.EPD_Clear();
```

2. Initialize SD card. Errors occurs without SD card

```
//1.Initialize the SD card
SD.SDCard_Init();
```

3. Create a buffer for image and set it to Black/White. Define height and width of image, rotate and invert parameters. Clear buffer.

```
//2.Create a new image cache named IMAGE_BW and fill it with white
paint.Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
paint.Paint_Clear(WHITE);
```

4. Read image and save to external RAM

```
//3.Read BMP images into RAM
SD.SDCard_ReadBMP("f2in13.bmp", 0, 0);
```

5. Refresh the data of external RAM to buffer of e-paper and display

```
//4.Refresh the picture in RAM to e-Paper
epd.EPD_Display();
Dev->Dev_Delay_ms(2000);
```

Display pictures from arrays

1. Create a buffer for image and set it to Black/White. Define height and width of image, rotate and invert parameters. Clear buffer.

2. Save data from array to external RAM

```
//2. show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
paint.Paint_DrawBitMap(IMAGE_DATA);
```

3. Refresh the data of external RAM to buffer of e-paper and display

Draw figures

1. Initialize e-paper, and clear the buffer

2. create an image buffer, set it to Black/White. Define the width and length. Set the rotate and invert parameters. Clear buffer

3. Draw point, set its position, color, size and full/empty

```
paint.Paint_DrawPoint(10, 70, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
paint.Paint_DrawPoint(10, 80, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
paint.Paint_DrawPoint(10, 90, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
```

4. Draw line, set its begin position, color, soild/dotted and width

```
paint.Paint_DrawLine(20, 70, 70, 100, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(70, 70, 20, 100, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
paint.Paint_DrawLine(180, 70, 180, 100, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
paint.Paint_DrawLine(165, 85, 195, 85, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
```

5. Draw rectangle, set the begin position of diagonal line, color, full/empty and width of line

```
paint.Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

6. Draw circle, set center position, radius, color, full/empty and width of line

```
paint.Paint_DrawCircle(180, 85, 15, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
paint.Paint_DrawCircle(180, 45, 15, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);
```

7. Display string, set the begin position, contents, font size, background color and the color of font

```
paint.Paint_DrawString_EN(10, 2, "Waveshare Electronic", &Font12, BLACK, WHITE);
paint.Paint_DrawString_EN(10, 20, "hello world", &Font12, WHITE, BLACK);
```

8. Display number, set the begin position, number, font size, background color and the color of font

```
paint.Paint_DrawNum(10, 50, 987654321, &Font16, WHITE, BLACK);
```

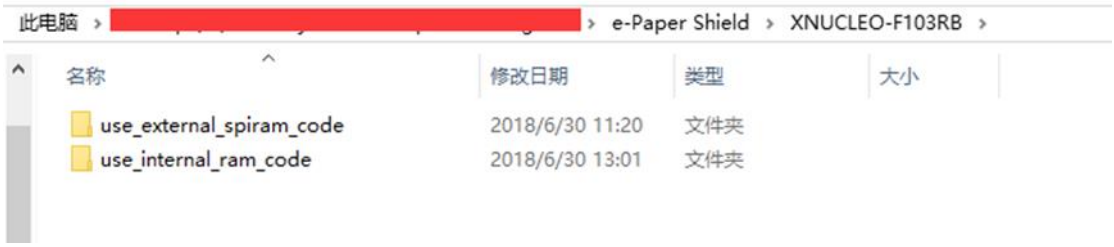
9. Display time, set the begin position, structure of time, font size, background color and the color of font (partial refresh)

```
Serial.print("Show time\r\n");
PAINT_TIME nowtime;
nowtime.Hour = 23;
nowtime.Min = 59;
nowtime.Sec = 50;

sFONT font = Font24;
UWORD Xstart = 40;
UWORD Ystart = 40;
for (;;) {
    paint.Paint_DrawTime(Xstart, Ystart, &nowtime, &font, WHITE, BLACK);
    nowtime.Sec = nowtime.Sec + 1;
    if (nowtime.Sec == 60) {
        nowtime.Min = nowtime.Min + 1;
        nowtime.Sec = 0;
        if (nowtime.Min == 60) {
            nowtime.Hour =  nowtime.Hour + 1;
            nowtime.Min = 0;
            if (nowtime.Hour == 24) {
                nowtime.Hour = 0;
                nowtime.Min = 0;
                nowtime.Sec = 0;
            }
        }
    }

    //Brushing the entire screen is not as obvious as painting a window.
    epd.EPD_DisplayPartial(0, 0, Paint_Image.Image_Width, Paint_Image.Image_Height);
}
```

NUCLEO-F103RB

- This example is based on XNUCLEO-F103RB
- The code is based on HAL libraries, you can port it to other STM32 chip with STM32CubeMX
- The code is compiled in Keil v5

There are two examples: one is use internal RAM (some big size of e-paper cannot work), another is use external RAM



Display pictures from SD card

1. Initialize e-paper, can clear the its buffer
2. Initialize SD card, make sure you have inserted SD card, or it will error.
3. Create an image buffer and set it to black/white image, define width, height, parameters of rotate and color invert. Clear the buffer
4. Read image, and save the data to external RAM

5. Refresh the data of external RAM to buffer of e-Paper, display it

```
    if(EPD_Init() != 0) {
        printf("e-Paper init failed\r\n");
    } else {
        printf("e-Paper init Successful\r\n");
    }
    EPD_Clear();
    printf("e-Paper clear over...\r\n");
    DEV_Delay_ms(500);
#if 1
    /*show sd card pic*/
    printf("/**************/\r\nshow image for SD card\r\n");
    //1.Initialize the SD card
    SDCard_Init();

    //2.Create a new image cache named IMAGE_BW and fill it with white
    printf("Initialize the cached related properties...\r\n");
    GUI_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
    GUI_Clear(WHITE);

    //3.Read BMP pictures into RAM
    printf("Read the image into the cache...\r\n");
    SDCard_ReadBMP("f2in13.bmp", 0, 0);

    //4.Refresh the picture in RAM to e-Paper
    printf("Refresh the picture in RAM to e-Paper...\r\n/**************/\r\n");
    EPD_DisplayFull();
#endif
```

Display image from arrays

1. Initialize e-Paper, and clear its buffer

2. Create an image buffer and set it to black/white image, define width, height, parameters of rotate and color invert. Clear the buffer

3. Save the data of Image array to external RAM

4. Refresh the data of external RAM to buffer of e-Paper, display it

```
    /*show image for array*/
    printf("show image for array\r\n");
    //1.Create a new image cache named IMAGE_BW and fill it with white
    Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_0, IMAGE_COLOR_POSITIVE);
    Paint_Clear(WHITE);

    printf("Paint_DrawBitMap\r\n");
    //2.show image for array, IMAGE_ROTATE_0 and IMAGE_COLOR_POSITIVE will not affect reading
    Paint_DrawBitMap(IMAGE_DATA);

    printf("EPD_Display\r\n");
    //3.Refresh the picture in RAM to e-Paper
    EPD_Display();
    Dev_Delay_ms(2000);
```

Drawing

1. Initialize e-paper, and clear the buffer
2. create an image buffer, set it to Black/White. Define the width and length. Set the rotate and invert parameters. Clear buffer
3. Draw point, set its position, color, size and full/empty
4. Draw line, set its begin position, color, soild/dotted and width
5. Draw rectangle, set the begin position of diagonal line, color, full/empty and width of line
6. Draw circle, set center position, radius, color, full/empty and width of line
7. Display string, set the begin position, contents, font size, background color and the color of font
8. Display number, set the begin position, number, font size, background color and the color of font
9. Display time, set the begin position, structure of time, font size, background color and the color of font (partial refresh)

```
//1.Create a new image cache named IMAGE_BW and fill it with white
Paint_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_90, IMAGE_COLOR_POSITIVE);
Paint_Clear(WHITE);

//2.Drawing on the image
Paint_DrawPoint(5, 10, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
Paint_DrawPoint(5, 25, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
Paint_DrawPoint(5, 40, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
Paint_DrawPoint(5, 55, BLACK, DOT_PIXEL_4X4, DOT_STYLE_DFT);

Paint_DrawLine(20, 10, 70, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(70, 10, 20, 60, BLACK, LINE_STYLE_SOLID, DOT_PIXEL_1X1);
Paint_DrawLine(170, 15, 170, 55, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);
Paint_DrawLine(150, 35, 190, 35, BLACK, LINE_STYLE_DOTTED, DOT_PIXEL_1X1);

Paint_DrawRectangle(20, 10, 70, 60, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawRectangle(85, 10, 130, 60, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawCircle(170, 35, 20, BLACK, DRAW_FILL_EMPTY, DOT_PIXEL_1X1);
Paint_DrawCircle(170, 85, 20, BLACK, DRAW_FILL_FULL, DOT_PIXEL_1X1);

Paint_DrawString_EN(5, 70, "hello world", &Font16, WHITE, BLACK);
Paint_DrawString_EN(5, 90, "waveshare", &Font20, BLACK, WHITE);

Paint_DrawNum(5, 120, 123456789, &Font20, BLACK, WHITE);

//3.Refresh the picture in RAM to e-Paper
EPD_Display();
Dev_Delay_ms(2000);
```

Partial refresh

1. Initialize e-paper and set it to partial refresh mode

2. Define the structure of time

3. clear data of window

4. Display time, set its begin position, the structure, font size, background color and the font color

5. Transmit data from external RAM to e-Paper.

```
#if 1
  printf("/**************/\r\nPartial display routine\r\n");
    GUI_NewImage(IMAGE_BW, EPD_WIDTH, EPD_HEIGHT, IMAGE_ROTATE_90, IMAGE_COLOR_POSITIVE);
    GUI_Clear(WHITE);
    EPD_DisplayPartial(0, 0, GUI_Image.Image_Width, GUI_Image.Image_Height);
    DEV_Delay_ms(500);

    printf("Show time\r\n");
    GUI_TIME nowtime;
    nowtime.Hour = 23;
    nowtime.Min = 59;
    nowtime.Sec = 50;

    sFONT font = Font24;
    UWORD Xstart = 40;
    UWORD Ystart = 40;
    for (;;) {
        GUI_DrawTime(Xstart, Ystart, &nowtime, &font, WHITE, BLACK);
        nowtime.Sec = nowtime.Sec + 1;
        if (nowtime.Sec == 60) {
            nowtime.Min = nowtime.Min + 1;
            nowtime.Sec = 0;
            if (nowtime.Min == 60) {
                nowtime.Hour =  nowtime.Hour + 1;
                nowtime.Min = 0;
                if (nowtime.Hour == 24) {
                    nowtime.Hour = 0;
                    nowtime.Min = 0;
                    nowtime.Sec = 0;
                }
            }
        }

        //Brushing the entire screen is not as obvious as painting a window.
        EPD_DisplayPartial(0, 0, GUI_Image.Image_Width, GUI_Image.Image_Height);
        DEV_Delay_ms(450);//Analog clock 1s
    }
#endif
```

IMAGE DATA

There are two ways to display pictures. One is display directly and other is indirectly.

Display directly: Read the data of pictures with library functions and decode. Then convert it to arrays and send to module. The demo code of Raspberry Pi and e-Paper Shield realize this function

Display indirectly: Converting pictures to relative arrays on PC and save as c file. Then you can use the c file on your project. This chapter we will talk about how to convert a picture to array.

1. Open a picture with drawing tool comes with Windows system, create a new image, and set the pixel to 104x212

2. Because this module can only display two gray level (Only black and white), we need to convert picture to monochrome bitmap before converting it to array. That is, File --> BMP picture --> Monochrome Bitmap.

   There is a monochrome bitmap on examples pack for demonstration (raspberrypi/python/monocolor.bmp).

3. Use Image2Lcd.exe software to generate corresponding array for picture (.c file).

   Open picture with this software, set the parameters:

   - Output data type: C language array
   - Scanning mode: vertical scanning
   - Output gray: single color (gray level of two)
   - Maximum width and height: 104 and 212
   - Include the data of Image Header: Don’ t check
   - Inverse color: Check (Check: the white on image will be converted to 1, and black is converted to 0)

4. Click Save, to generate .c file. Copy the corresponding array into your project, and you can display picture by calling this array.