waveshare Pico-ePaper-2.13

Waveshare 2.13-inch E-Paper Display Module for Raspberry Pi Pico User Manual

Model: Pico-ePaper-2.13

Introduction Features Specifications Setup Operation Maintenance Troubleshooting Support

1. Introduction

This manual provides detailed instructions for the Waveshare 2.13-inch E-Paper E-Ink Display Module (Model: Pico-ePaper-2.13). This module is designed for use with the Raspberry Pi Pico, offering a low-power, partial refresh display solution. E-paper displays are known for their paper-like effect and ability to retain content without continuous power.

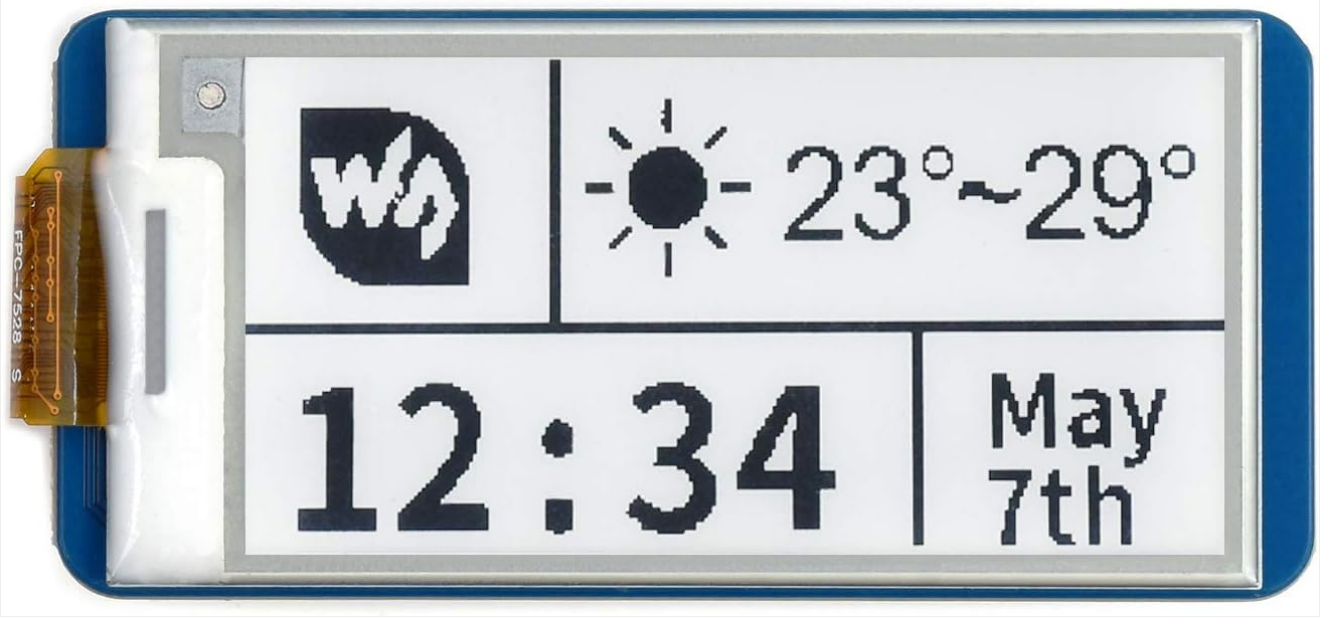


Figure 1: Waveshare 2.13-inch E-Paper Display Module displaying example content.

2. Key Features

- **No Backlight:** The display retains its content indefinitely without power, making it highly energy-efficient.
- **Ultra-Low Power Consumption:** Power is primarily required only during content refresh cycles.
- **SPI Interface:** Utilizes a Serial Peripheral Interface, requiring minimal I/O pins for communication.

- **Development Resources:** Includes comprehensive development resources and a manual with Raspberry Pi Pico C/C++ and MicroPython examples.
- **Direct Raspberry Pi Pico Attachment:** Features an onboard female pin header for direct connection to a Raspberry Pi Pico.

Figure 2: Overview of the 2.13-inch E-Paper Module's features and specifications.

3. Technical Specifications

Feature	Value
Operating Voltage	3.3V
Grey Scale	2
Interface	3-wire SPI, 4-wire SPI
Partial Refresh Time	0.3s
Outline Dimensions	65.00 × 30.50mm
Full Refresh Time	2s
Display Size	48.55 × 23.70mm
Refresh Power	26.4mW (typ.)
Dot Pitch	0.194 × 0.194mm

Feature	Value
Standby Current	<0.01uA (almost none)
Resolution	250×122 pixels
Viewing Angle	>170°
Display Color	Black, White

Figure 3: Detailed technical specifications of the E-Paper module.

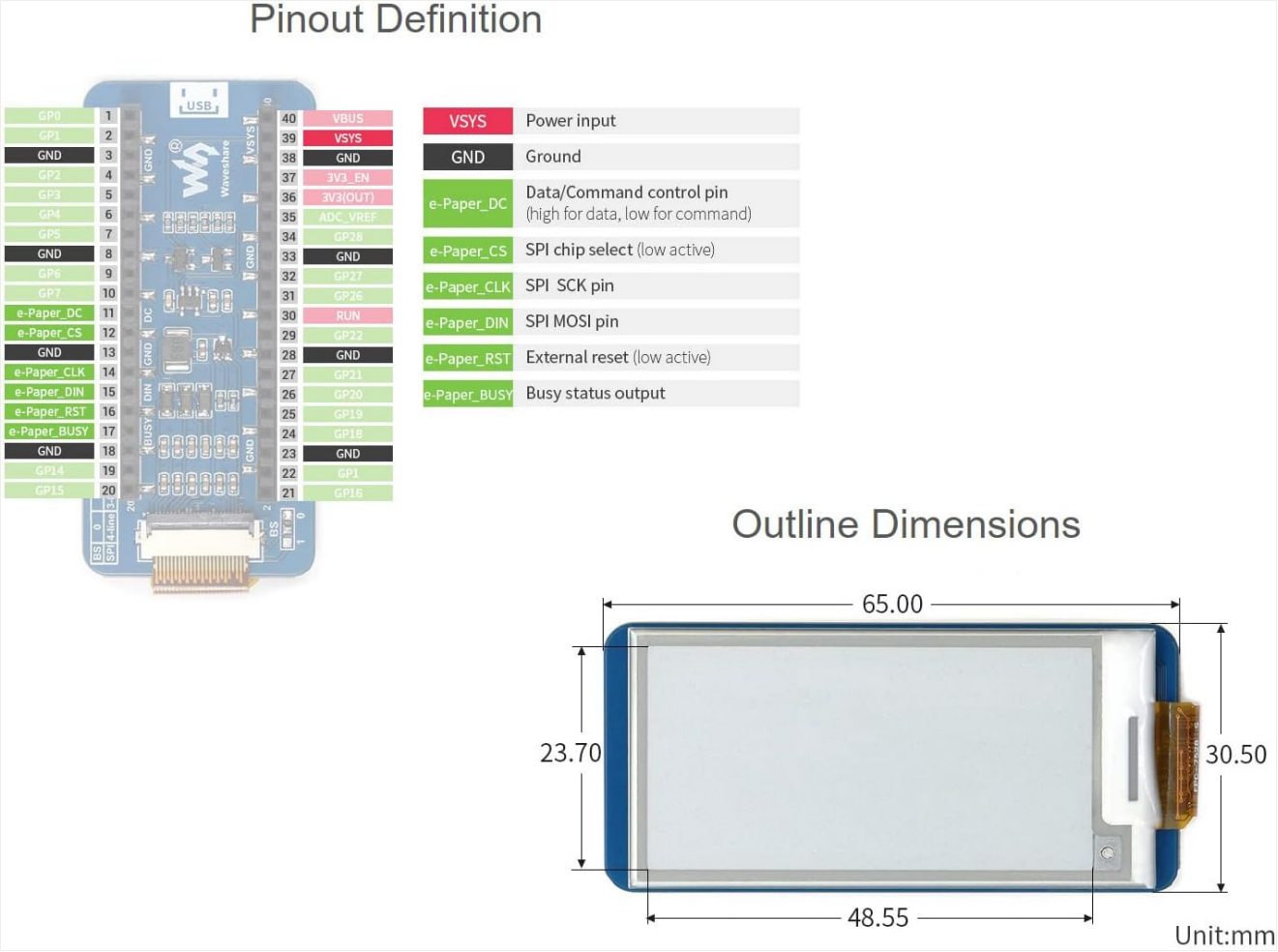


Figure 4: Outline dimensions of the 2.13-inch E-Paper module in millimeters.

4. Setup and Connection

The Waveshare Pico-ePaper-2.13 module is designed for direct attachment to a Raspberry Pi Pico. Follow these steps for proper setup:

1. **Prepare Raspberry Pi Pico:** Ensure your Raspberry Pi Pico is ready for use, with necessary firmware or operating system installed if required for your project.
2. **Align and Connect:** Carefully align the female pin header on the E-Paper module with the male pins on your Raspberry Pi Pico. Gently press them together until fully seated. Ensure correct orientation to prevent damage.
3. **Power Supply:** Once connected, power the Raspberry Pi Pico via its USB port or other designated power input. The E-Paper module draws power directly from the Pico.
4. **Software Setup:** Refer to the provided development resources for specific C/C++ or MicroPython examples to initialize and control the display. This typically involves configuring the SPI interface on the Pico.

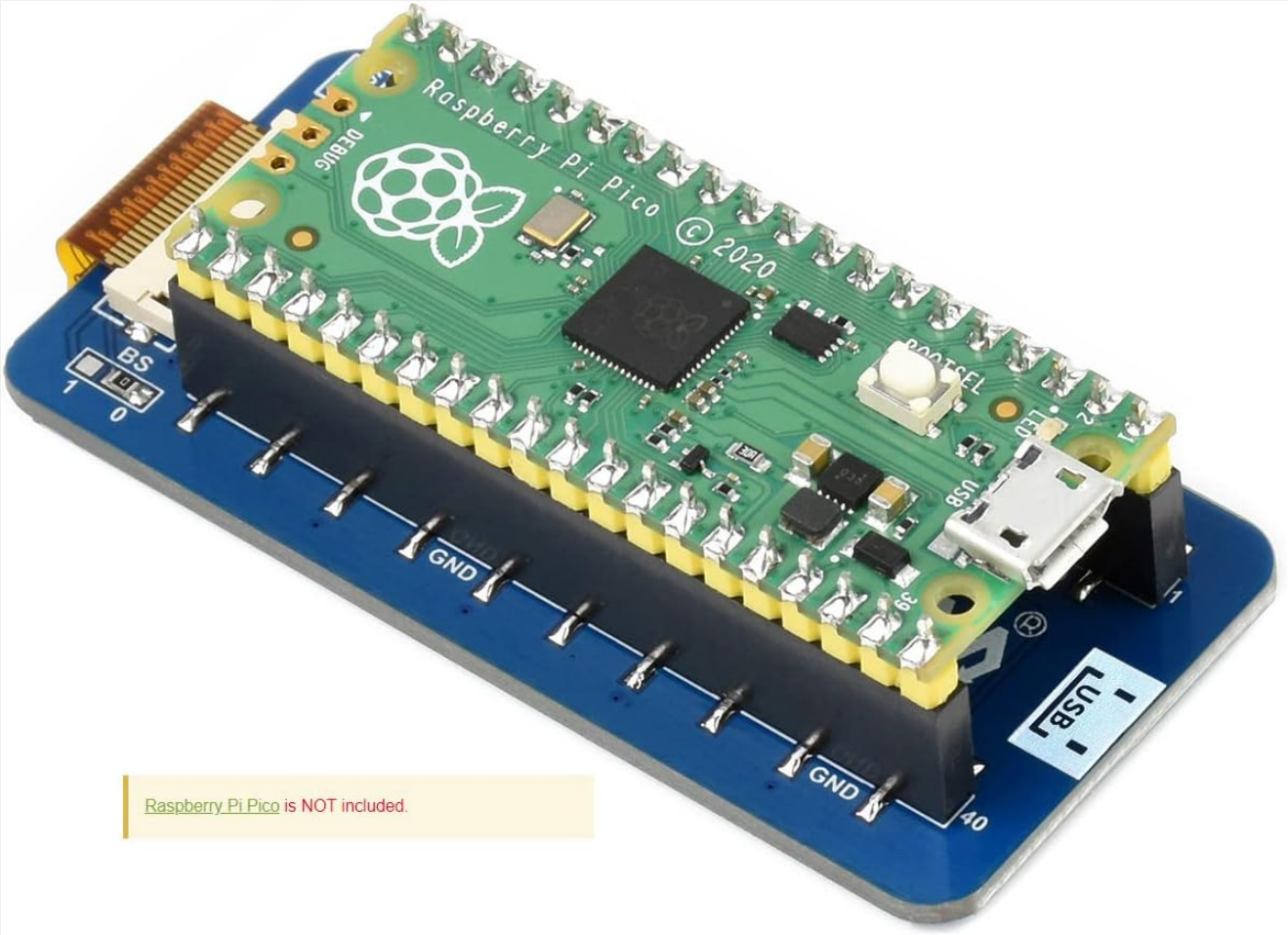


Figure 5: Raspberry Pi Pico securely connected to the E-Paper module. Note: Raspberry Pi Pico is not included with the display module.

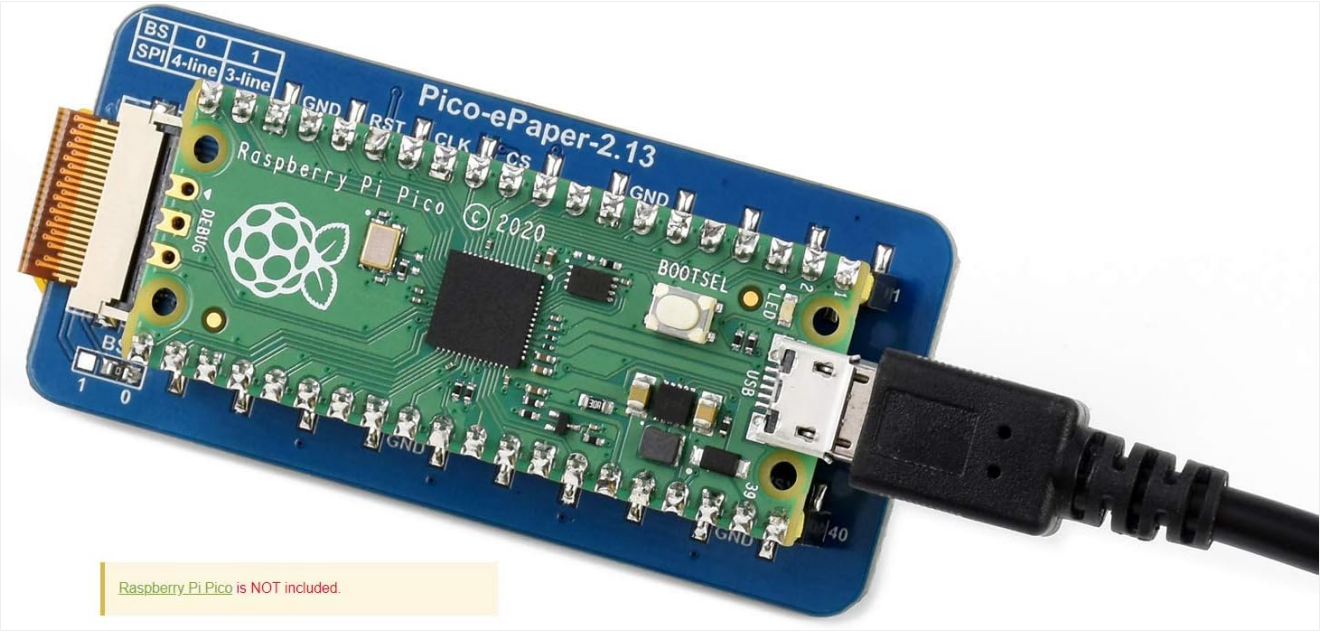


Figure 6: Raspberry Pi Pico with E-Paper module connected and powered via USB. Note: Raspberry Pi Pico is not included with the display module.

4.1 Pinout Definition

Understanding the pinout is crucial for advanced configurations or troubleshooting. The module uses an SPI interface.

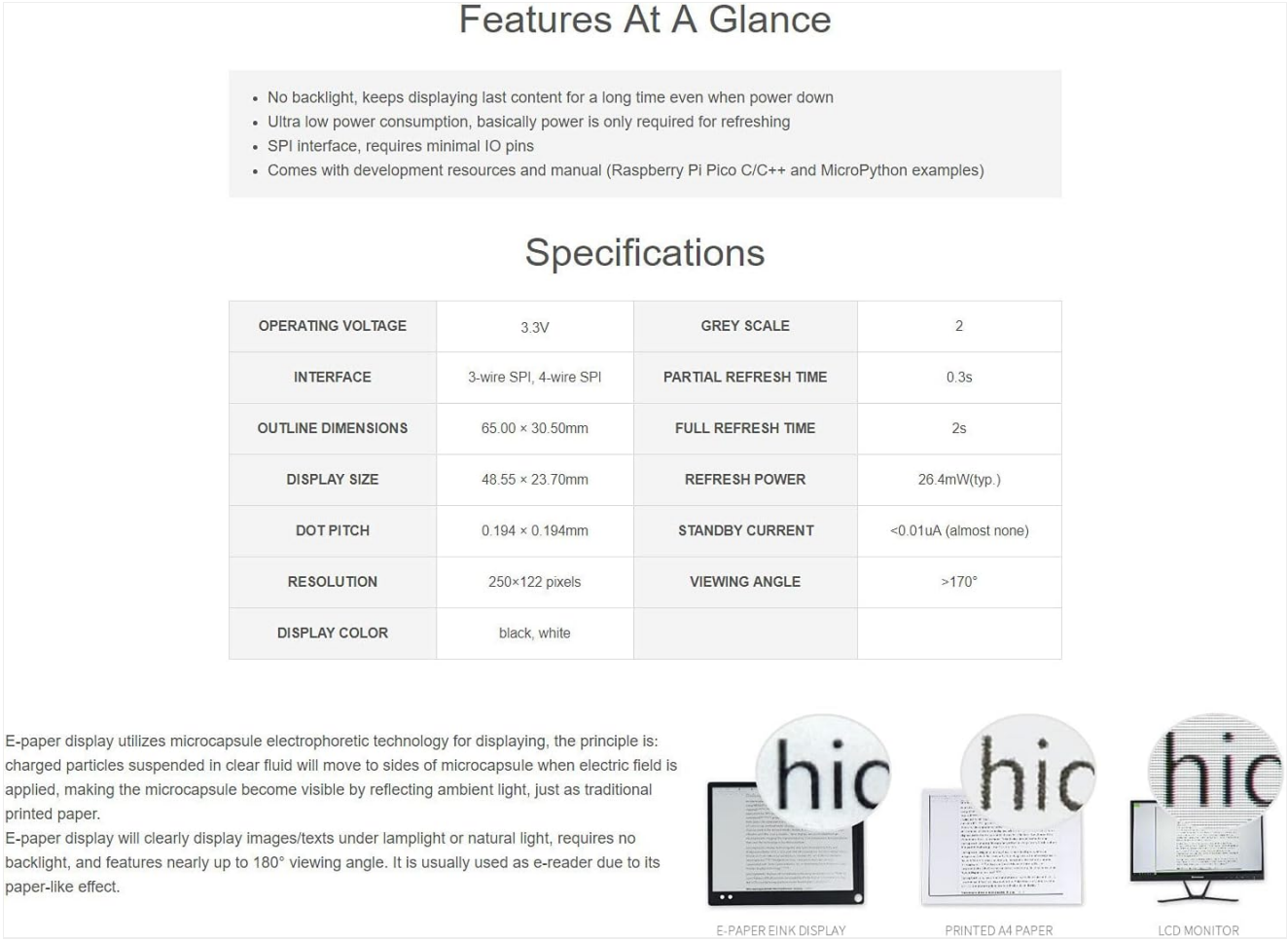


Figure 7: Pinout definition for the Waveshare Pico-ePaper-2.13 module, showing connections for SPI, power, and control pins.

Pin Name	Description
VSYS	Power input
GND	Ground
e-Paper_DC	Data/Command control pin (high for data, low for command)
e-Paper_CS	SPI chip select (low active)
e-Paper_CLK	SPI SCK pin
e-Paper_DIN	SPI MOSI pin
e-Paper_RST	External reset (low active)
e-Paper_BUSY	Busy status output

5. Operation Guide

Operating the Waveshare 2.13-inch E-Paper module involves programming your Raspberry Pi Pico to send display data via the SPI interface. Waveshare provides example code and libraries to facilitate this process.

5.1 Programming the Display

- **Access Development Resources:** Visit the official Waveshare product page or documentation portal for the Pico-ePaper-2.13 module. Here you will find example code in C/C++ and MicroPython.
- **Install Libraries:** Download and install any necessary libraries or drivers for your chosen programming language

(e.g., MicroPython libraries for e-paper displays).

- **Load Example Code:** Upload the provided example code to your Raspberry Pi Pico. These examples typically demonstrate basic functions like displaying text, images, or performing partial/full refreshes.
- **Customize Content:** Modify the example code to display your desired content. Pay attention to the display resolution (250x122 pixels) and color depth (black/white, 2 grey scales).
- **Refresh Cycles:** Understand the difference between full refresh (approx. 2 seconds) and partial refresh (approx. 0.3 seconds). Partial refresh is faster and consumes less power but may leave ghosting artifacts over time, requiring an occasional full refresh.

5.2 Application Examples

The E-Paper module is versatile and suitable for various applications due to its low power consumption and persistent display capabilities.

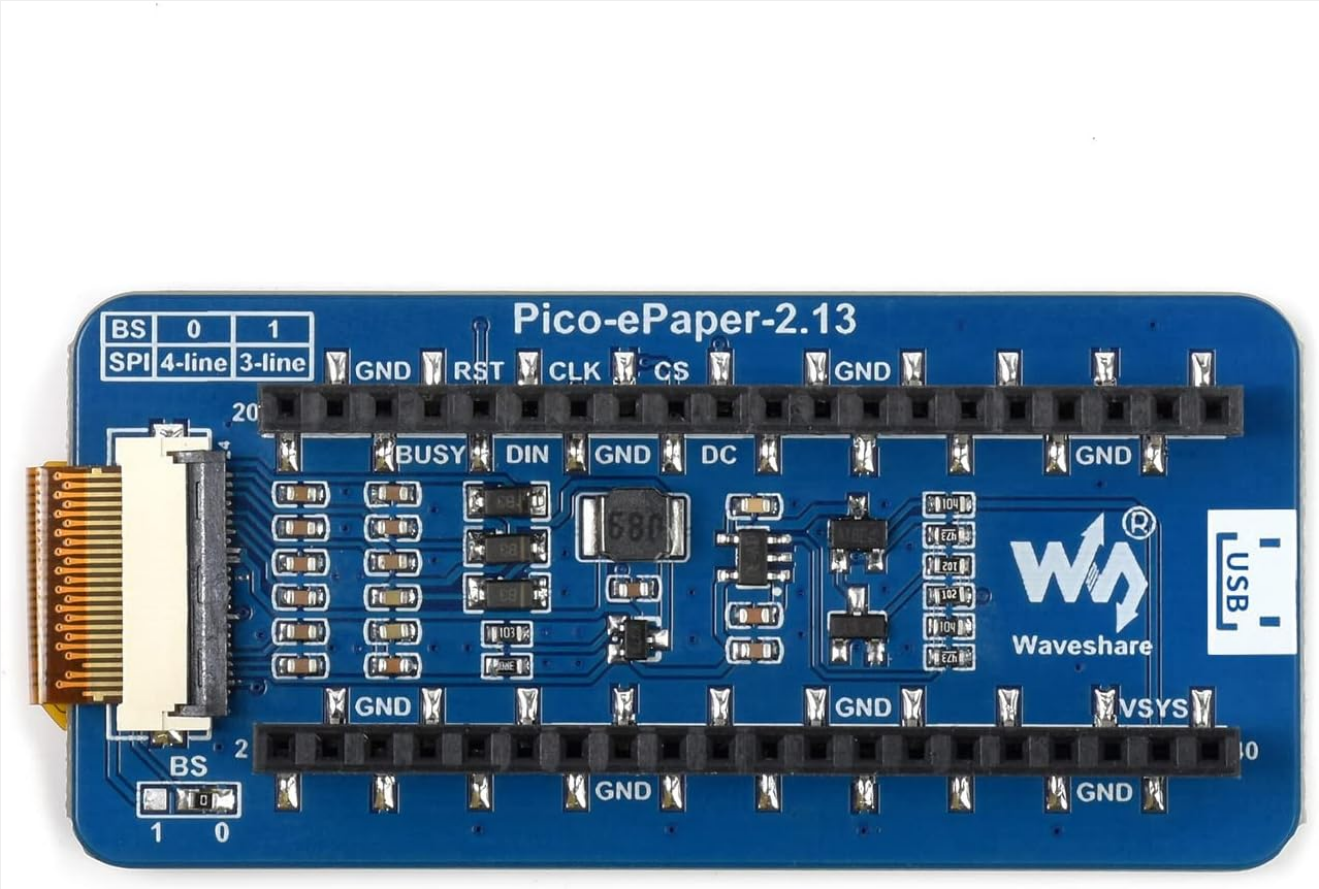


Figure 8: Examples of potential applications for the E-Paper module, such as price tags, shelf labels, and information displays in various environments.

6. Care and Maintenance

To ensure the longevity and optimal performance of your Waveshare E-Paper display module, follow these maintenance guidelines:

- **Handle with Care:** E-paper displays are delicate. Avoid applying excessive pressure or bending the module, especially the flexible FPC cable.
- **Cleaning:** Use a soft, dry, lint-free cloth to gently wipe the display surface. Do not use liquid cleaners, solvents, or abrasive materials, as these can damage the screen.
- **Environmental Conditions:** Operate and store the module within its specified temperature and humidity ranges. Avoid extreme temperatures, direct sunlight for prolonged periods, and high humidity.
- **Power Management:** While the display retains content without power, ensure proper power-down sequences for

the Raspberry Pi Pico to prevent data corruption or unexpected behavior.

- **Static Electricity:** Take precautions against electrostatic discharge (ESD) when handling the module, as electronic components can be sensitive to static.

7. Troubleshooting

If you encounter issues with your E-Paper display module, consider the following troubleshooting steps:

- **Display Not Refreshing/Blank:**
  - Check all connections between the E-Paper module and the Raspberry Pi Pico. Ensure they are secure and correctly oriented.
  - Verify the power supply to the Raspberry Pi Pico.
  - Review your code for display initialization and refresh commands. Ensure the SPI interface is correctly configured.
  - Confirm that the correct libraries for the Pico-ePaper-2.13 are installed and imported in your code.
- **Ghosting or Image Retention:**
  - Ghosting is common with partial refreshes. Perform a full refresh periodically to clear the display completely.
  - Ensure your refresh cycles are correctly implemented in your code.
- **Incorrect Display Orientation:**
  - The display typically operates in portrait mode by default. If you require landscape orientation, check if your chosen library or example code supports rotation and how to implement it.
- **Module Not Detected by Pico:**
  - Double-check the pin connections against the pinout diagram (Figure 7).
  - Ensure the Raspberry Pi Pico is functioning correctly independently.

For further assistance, refer to the official Waveshare documentation and community forums.

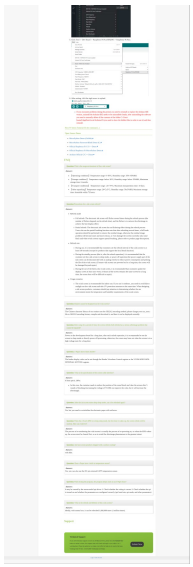
8. Support and Warranty

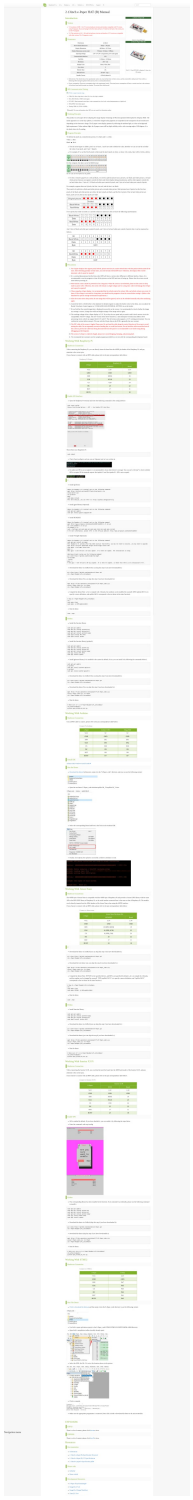
For technical support, additional resources, and the latest documentation, please visit the official Waveshare website. You can typically find detailed product wikis, example code, and community forums there.
Manufacturer: Waveshare
Website: www.waveshare.com
Information regarding product warranty is typically available on the Waveshare website or included with your purchase documentation. Please retain your proof of purchase for warranty claims.

Related Documents - Pico-ePaper-2.13

[Waveshare Pico e-Paper 2.13inch EPD Module for Raspberry Pi Pico: Development Guide & API]

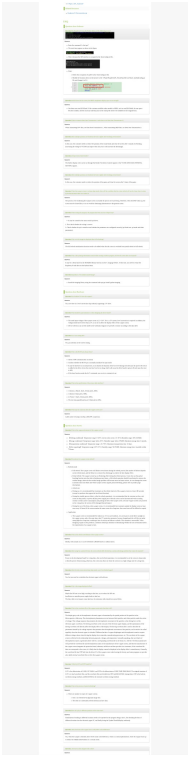
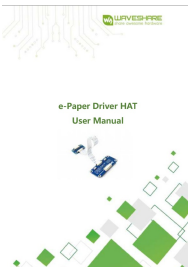
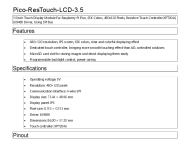
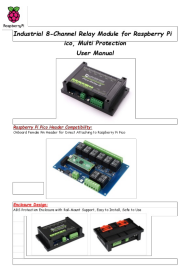
Detailed development guide for the Waveshare Pico e-Paper 2.13inch EPD module with Raspberry Pi Pico. Features include 250x122 resolution, SPI interface, C/C++ & MicroPython demo codes, and comprehensive API documentation.





[Waveshare 2.13inch e-Paper HAT (B) User Manual and Technical Guide]

Comprehensive guide for the Waveshare 2.13inch e-Paper HAT (B), covering hardware connections, software setup, programming principles, and troubleshooting for Raspberry Pi, Arduino, Jetson Nano, and STM32.

	
	[Waveshare e-Paper Driver HAT User Manual: Connect SPI E-Paper Displays to Raspberry Pi, Arduino, STM32] User manual for the Waveshare e-Paper Driver HAT, detailing its features, product parameters, interface specifications, and supported e-Paper models. Includes setup guides for Raspberry Pi, Arduino, and STM32 development boards.
	[Waveshare Pico-ResTouch-LCD-3.5: 3.5-inch SPI Touch Display Module for Raspberry Pi Pico] Detailed specifications, features, pinout, and hardware connection guide for the Waveshare Pico-ResTouch-LCD-3.5, a 3.5-inch IPS touch display module with XPT2046 controller and ILI9488 driver for Raspberry Pi Pico.
	[Waveshare Industrial 8-Channel Relay Module for Raspberry Pi Pico User Manual] User manual for the Waveshare Industrial 8-Channel Relay Module for Raspberry Pi Pico (Pico-Relay-B). Details features, compatibility, enclosure, and pinout for industrial control applications.
	[Waveshare 7.3inch e-Paper (E) User Manual - Specifications and Guide] Comprehensive user manual for the Waveshare 7.3inch e-Paper (E) display module, detailing specifications, features, pin assignments, electrical and optical characteristics, and handling instructions.

Pico e-Paper 2.9 (B)

Overview

2.9inch EPD Module for Raspberry Pi Pico,
296 × 128, Black / White /Red, SPI

2.9inch EPD (Electronic Paper Display) Module For Raspberry Pi Pico, 296 × 128 Pixels, Black / White / Red, SPI Interface.

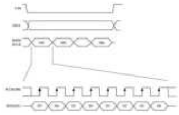
Specification

- Size: 2.9inch
- Outline dimensions(raw panel): 79.0mm × 36.7mm × 1.05mm
- Outline dimension(driver board): 82.0mm × 38.0mm
- Display size: 66.89mm × 29.05mm
- Operating voltage: 3.3V/5V
- Interface: SPI
- Dot pitch: 0.138 × 0.138
- Resolution: 296 × 128
- Display color: Black, White, Red
- Greyscale: 2
- full refresh time: 15s
- Refresh power: 26.4mW (typ.)
- Standby current: <0.01uA (almost none)
- Note:

1. Refresh time: The refresh time is the experimental results, the actual refresh time will have errors, and the actual effect shall prevail. There will be a flickering effect during the global refresh process, this is a normal phenomenon.

2. Power consumption: The power consumption data is the experimental results. The actual power consumption will have a certain error due to the existence of the driver board and the actual use situation. The actual effect shall prevail.

SPI Communication Timing



Since the ink screen only needs to be displayed, the data cable (MISO) sent from the machine and received by the host is hidden here.

- CS: Slave chip select, when CS is low, the chip is enabled.
- DC: data/command control pin, write command when DC=0; write data when DC=1.
- SCLK: SPI communication clock.
- SDIN: SPI communication master sends, the slave receives.
- Timing: CPHL=0, CPOL=0 (SPI0)

【Remarks】 For specific information about SPI, you can search for information online.

Working Protocol

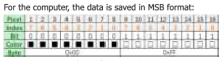
This product is an E-paper device adopting the image display technology of Microencapsulated Electrophoretic Display, MED. The initial approach is to create tiny spheres, in which the charged color pigments are suspended in the transparent oil and would move depending on the electronic charge. The E-paper screen displays patterns by reflecting the ambient light, so it has no background light requirement. (**Note that the e-Paper cannot support updating directly under sunlight**).

How to define pixels

In a monochrome picture we define the pixels, 0 is black and 1 is white.
White : □: Bit 1
Black : ■ : Bit 0

- The dot in the figure is called a pixel. As we know, 1 and 0 are used to define the color, therefore we can use one bit to define the color of one pixel, and 1 byte = 8pixels
- For example, If we set the first 8 pixels to black and the last 8 pixels to white, we show it by codes, they will be 16-bit as below:

For the computer, the data is saved in MSB format:



So we can use two bytes for 16 pixels.

For 2.13inch e-paper B, the display colors are red, black, and white. We need to split the picture into 2 pictures, one is a black and white picture, and another is a red and white picture. When transmitting, because one register controls a black or white pixel, one controls a Red or white display. The black and white part of 2.13 use 1 byte to control 8 pixels, and the red and white part uses 1 byte to control 8 pixels.
For example, suppose there are 8 pixels, the first 4 are red, and the back 4 are black:
They need to be disassembled into a black and white picture and a red and white picture. Both pictures have 8 pixels, but the first four pixels of the black and white picture are white, the last 4 pixels are black, and the first 4 pixels of the red and white picture One pixel is red, and the last four pixels are white.

Original picture	■	■	■	■	□	□	□	□
Divided								
Black/White	□	□	□	□	■	■	■	■
Red/White	■	■	■	■	□	□	□	□

If you define that the data of the white pixel is 1 and the black is 0, then we can get:

Black/White	□	□	□	□	■	■	■	■
Data	1	1	1	1	0	0	0	0
Red/White	■	■	■	■	□	□	□	□
Data	0	0	0	0	1	1	1	1

So that we can use 1 byte to control every eight pixels.

Bit	1	2	3	4	5	6	7	8
Black/White	□	□	□	□	■	■	■	■
Data	1	1	1	1	0	0	0	0
Byte	0xF0							
Bit	1	2	3	4	5	6	7	8
Red/White	■	■	■	■	□	□	□	□
Data	0	0	0	0	1	1	1	1
Byte	0x0F							

Precautions

1. For the screen that supports partial update, please note that you cannot refresh the screen with the partial mode all the time. After several partial updating, you need to fully refresh the screen once. Otherwise, the screen display effect will be abnormal, which cannot be repaired!
2. Because of the different batches, some of them have aberrations. Store the e-Paper right side up will reduce it. And if the e-Paper didn't be refreshed for a long time, it will become more and more reddish/yellowish. Please use the demo code to refresh the e-paper several times in this case.
3. Note that the screen cannot be powered on for a long time. When the screen is not refreshed, please set the screen to sleep mode, or power off the e-Paper. Otherwise, the screen will remain in a high voltage state for a long time, which will damage the e-Paper and cannot be repaired!
4. When using the e-Paper, it is recommended that the refresh interval be at least 180s, and refresh at least once every 24 hours. If the e-Paper is not used for a long time, the ink screen should be brushed and stored. (Refer to the datasheet for specific storage environment requirements)
5. After the screen enters sleep mode, the sent image data will be ignored, and it can be refreshed normally only after initializing again.
6. Control the 0x3C or 0x50 (refer to the datasheet for details) register to adjust the border color. In the routine, you can adjust the Border Waveform Control register or VCOM AND DATA INTERVAL SETTING to set the border.
7. If you find that the created image data is displayed incorrectly on the screen, it is recommended to check whether the image size setting is correct, change the width and height settings of the image and try again.
8. The working voltage of the e-Paper is 3.3V. If you buy the raw panel and you need to add a level convert circuit for compatibility with 5V voltage. The new version of the driver board (V2.1 and subsequent versions) has added a level processing circuit, which can support both 3.3V and 5V working environments. The old version can only support a 3.3V working environment. You can confirm the version before using it. (The one with the 20-pin chip on the PCB is generally the new version)
9. The FPC cable of the screen is relatively fragile, pay attention to bending the cable along the horizontal direction of the screen when using it, and do not bend the cable along the vertical direction of the screen
10. The screen of e-Paper is relatively fragile, please try to avoid dropping,

bumping, and pressing hard.

11. We recommend that customers use the sample program provided by us to test with the corresponding development board after they get the screen.

RPi Pico

Hardware Connection

Please take care of the direction when connecting Pico. A logo of the USB port is printed to indicate the directory, you can also check the pins.
If you want to connect the board by an 8-pin cable, you can refer to the table below:

e-Paper	Pico	Description
VCC	VSYS	Power input
GND	GND	Ground
DIN	GP11	MOSI pin of SPI interface, data transmitted from Master to Slave.
CLK	GP10	SCK pin of SPI interface, clock input
CS	GP9	Chip select pin of SPI interface, Low Active
DC	GP8	Data/Command control pin (High: Data; Low: Command)
RST	GP12	Reset pin, low active
BUSY	GP13	Busy output pin
KEY0	GP2	User key 0
KEY1	GP3	User key 1
RUN	RUN	Reset

You can just attach the board to Pico like the Pico-ePaper-7.5.

Setup Environment

You can refer to the guides for Raspberry PI:
https://www.raspberrypi.org/documentation/pico/getting-started/

Download Demo codes

Open a terminal of Pi and run the following command:

```
cd ~
sudo wget  https://files.waveshare.com/upload/2/27/Pico_ePaper_Code.zip
unzip Pico_ePaper_Code.zip -d Pico_ePaper_Code
cd ~/Pico_ePaper_Code
```

You can also clone the codes from Github.

```
cd ~
git clone https://github.com/waveshare/Pico_ePaper_Code.git
cd ~/Pico_ePaper_Code
```

About the examples

The guides are based on Raspberry Pi.

C codes

The example provided is compatible with several types, you need to modify the main.c file, uncomment the definition according to the actual type of display you get.
For example, if you have the Pico-ePaper-2.13, please modify the main.c file, uncomment line 18 (or maybe it is line 19).

Set the project:

```
cd ~/Pico_ePaper_Code/c
```

Create build folder and add the SDK.
../../pico-sdk is the default path of the SDK, if you save the SDK to other directories, please change it to the actual path.

```
mkdir build
cd build
export PICO_SDK_PATH=../../pico-sdk
```

Run cmake command to generate Makefile file.

```
cmake ..
```

Run the command make to compile the codes.

```
make -j9
```

- After compiling, the epd.uf2 file is generated. Next, press and hold the BOOTSEL button on the Pico board, connect the Pico to the Raspberry Pi using the Micro USB cable, and release the button. At this point, the device will recognize a removable disk (RPI-RP2).
- Copy the epd.uf2 file just generated to the newly recognized removable disk (RPI-RP2), Pico will automatically restart the running program.

Python

- First press and hold the BOOTSEL button on the Pico board, use the Micro USB cable to connect the Pico to the Raspberry Pi, then release the button. At this point, the device will recognize a removable disk (RPI-RP2).
- Copy the rp2-pico-20210418-v1.15.uf2 file in the python directory to the removable disk (RPI-RP2) just identified.
- Update Thonny IDE.

```
sudo apt upgrade thonny
```

- Open Thonny IDE (click on the Raspberry logo -> Programming -> Thonny Python IDE), and select the interpreter:
  - Select Tools -> Options... -> Interpreter.
  - Select MicroPython (Raspberry Pi Pico and ttyACM0 port).
- Open the Pico_ePaper-xxx.py file in Thonny IDE, then run the current script (click the green triangle).

C Code Analysis

Bottom Hardware Interface

We package the hardware layer for easily porting to the different hardware platforms.
DEV_Config.c(.h) in the directory: Pico_ePaper_Code\c\lib\Config.

- Data type:

```
#define UBYTE   uint8_t
#define UWORD   uint16_t
#define UDOUBLE uint32_t
```

- Module initialize and exit:

```
void DEV_Module_Init(void);
void DEV_Module_Exit(void);
Note:
1. The functions above are used to initialize the display or exit handle.
```

- GPIO Write/Read:

```
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 codes of EPD are saved in the directory: Pico_ePaper_Code\c\lib\e-Paper
Open the .h header file, you can check all the functions defined.

- Initialize e-Paper, this function is always used at the beginning and after waking up the display.

```
//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 fully update, Mode = 1 partial updat
e
//Other types
void EPD_xxx_Init(void);
```

xxx should be changed by the type of e-Paper, For example, if you use 2.13inch e-Paper (D), to fully update, it should be EPD_2IN13D_Init(0) and EPD_2IN13D_Init(1) for the partial update;

- Clear: this function is used to clear the display to white.

```
void EPD_xxx_Clear(void);
```

xxx should be changed by the type of e-Paper, For example, if you use 2.9inch e-Paper (D), it should be EPD_2IN9D_Clear();

- Send the image data (one frame) to EPD and display

```
//Bicolor version
void EPD_xxx_Display(UBYTE *Image);
//Tricolor version
void EPD_xxx_Display(const UBYTE *blackimage, const UBYTE *ryimage);
```

There are several types which are different from others

```
//Partial update for 2.13inch e-paper (D), 2.9inch e-paper (D)
void EPD_2IN13D_DisplayPart(UBYTE *Image);
void EPD_2IN9D_DisplayPart(UBYTE *Image);
```

```
//For 2.13inch e-paper V2, you need to first useEPD_xxx_DisplayPartBaseImage to
display a static background and then partial update by the function EPD_xxx_Dis
playPart()
void EPD_2IN13_V2_DisplayPart(UBYTE *Image);
void EPD_2IN13_V2_DisplayPartBaseImage(UBYTE *Image);
```

- Enter sleep mode

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

Note, You should only hardware reset or use initialize function to wake up e-Paper from sleep mode
xxx is the type of e-Paper, for example, if you use 2.13inch e-Paper D, it should be EPD_2IN13D_Sleep().

Application Programming Interface

We provide basic GUI functions for testing, like draw point, line, string, and so on. The GUI function can be found in the directory: RaspberryPi_JetsonNano\c\lib\GUI\GUI_Paint.c(.h).

The fonts used can be found in the directory: RaspberryPi_JetsonNano\c\lib\Fonts.

- Create a new image, you can set the image name, width, height, rotate angle, and color.

```
void Paint_NewImage(UBYTE *image, UWORD Width, UWORD Height, UWORD Rotate, UWOR
D Color)
Parameters:
        image: Name of the image buffer, this is a pointer;
        Width: Width of the image;
        Height: Height of the image;
        Rotate: Rotate the angle of the Image;
        Color: The initial color of the image;
```

- Select image buffer: You can create multiple image buffers at the same time and select the certain one and draw by this function.

```
void Paint_SelectImage(UBYTE *image)
Parameters:
        image: The name of the image buffer, this is a pointer;
```

- Rotate image: You need to set the rotation angle of the image, this function should be used after Paint_SelectImage(). The angle can be 0, 90, 180, or 270.

```
void Paint_SetRotate(UWORD Rotate)
Parameters:
        Rotate: Rotate the angle of the image, the parameter can be ROTATE_0, R
OTATE_90, ROTATE_180, ROTATE_270.
```

【Note】After rotating, the place of the first pixel is different, we take a 1.54-inch e-paper as an example.

- Image mirror: This function is used to set the image mirror.

```
void Paint_SetMirroring(UBYTE mirror)
Parameters:
        mirror: Mirror type if the image, the parameter can be MIRROR_NONE, MIR
ROR_HORIZONTAL, MIRROR_VERTICAL, MIRROR_ORIGIN.
```

- Set the position and color of pixels: This is the basic function of GUI, it is used to set the position and color of a pixel in the buffer.

```
void Paint_SetPixel(UWORD Xpoint, UWORD Ypoint, UWORD Color)
Parameters:
        Xpoint: The X-axis value of the point in the image buffer
        Ypoint: The Y-axis value of the point in the image buffer
        Color: The color of the point
```

- Clear display: To set the color of the image, this function always be used to clear the display.

```
void Paint_Clear(UWORD Color)
Parameters:
        Color: The color of the image
```

- Color of the windows: This function is used to set the color of windows, it is always used for updating partial areas like displaying a clock.

```
void Paint_ClearWindows(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWO
RD Color)
Parameters:
        Xpoint: The X-axis value of the start point in the image buffer
        Ypoint: The Y-axis value of the start point in the image buffer
        Xend: The X-axis value of the end point in the image buffer
        Yend: The Y-axis value of the end point in the image buffer
        Color: The color of the windows
```

- Draw point: Draw a point at the position （X point, Y point） of the image buffer, you can configure the color, size, and style.

```
void Paint_DrawPoint(UWORD Xpoint, UWORD Ypoint, UWORD Color, DOT_PIXEL Dot_Pix
el, DOT_STYLE Dot_Style)
Parameters:
        Xpoint: X-axis value of the point.
        Ypoint: Y-axis value of the point.
        Color: Color of the point
        Dot_Pixel: Size of the point, 8 sizes 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;
        Dot_Style: Style of the point, define the extended mode of the point.
                typedef enum {
                  DOT_FILL_AROUND  = 1,
                  DOT_FILL_RIGHTUP,
                } DOT_STYLE;
```

- Draw the line: Draw a line from (Xstart, Ystart) to (Xend, Yend) in the image buffer, you can configure the color, width, and style.

```
void Paint_DrawLine(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UWORD C
olor, LINE_STYLE Line_Style , LINE_STYLE Line_Style)
Parameters:
        Xstart: Xstart of the line
        Ystart: Ystart of the line
        Xend: Xend of the line
        Yend: Yend of the line
        Color: Color of the line
        Line_width: Width of the line, 8 sizes 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;
         Line_Style: Style of the line, Solid or Dotted.
```

```
    typedef enum {
        LINE_STYLE_SOLID = 0,
        LINE_STYLE_DOTTED,
    } LINE_STYLE;
```

- Draw a rectangle: Draw a rectangle from (Xstart, Ystart) to (Xend, Yend), you can configure the color, width, and style.

```
void Paint_DrawRectangle(UWORD Xstart, UWORD Ystart, UWORD Xend, UWORD Yend, UW
ORD Color, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
    Xstart: Xstart of the rectangle.
    Ystart: Ystart of the rectangle.
    Xend: Xend of the rectangle.
    Yend: Yend of the rectangle.
    Color: Color of the rectangle
    Line_width: The width of the edges. 8 sizes 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: Style of the rectangle, empty or filled.
        typedef enum {
            DRAW_FILL_EMPTY = 0,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Draw circle: Draw a circle in the image buffer, use (X_Center Y_Center) as the center and Radius as the radius. You can configure the color, width of the line, and the style of the circle.

```
void Paint_DrawCircle(UWORD X_Center, UWORD Y_Center, UWORD Radius, UWORD Colo
r, DOT_PIXEL Line_width, DRAW_FILL Draw_Fill)
Parameters:
    X_Center: X-axis of center
    Y_Center: Y-axis of center
    Radius: Radius of circle
    Color: Color of the circle
    Line_width: The width of arc, 8 sizes 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: Style of the circle: empty or filled.
        typedef enum {
            DRAW_FILL_EMPTY = 0,
            DRAW_FILL_FULL,
        } DRAW_FILL;
```

- Show Ascii character: Show a character in (Xstart, Ystart) position, you can configure the font, foreground, and background.

```
void Paint_DrawChar(UWORD Xstart, UWORD Ystart, const char Ascii_Char, sFONT* F
ont, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: Xstart of the character
    Ystart: Ystart of the character
    Ascii_Char: Ascii char
    Font:  five fonts are available :
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: foreground color
    Color_Background: background color
```

- Draw the string: Draw the string at (Xstart Ystart), you can configure the fonts, foreground, and the background

```
void Paint_DrawString_EN(UWORD Xstart, UWORD Ystart, const char * pString, sFON
T* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: Xstart of the string
    Ystart: Ystart of the string
    pString: String
    Font: five fonts are available:
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: foreground color
    Color_Background: background color
```

- Draw Chinese string: Draw the Chinese string at (Xstart Ystart) of the image buffer. You can configure fonts (GB2312), foreground, and background.

```
void Paint_DrawString_CN(UWORD Xstart, UWORD Ystart, const char * pString, sFON
T* font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: Xstart of string
    Ystart: Ystart of string
    pString: string
    Font: GB2312 fonts, two fonts are available
        font12CN: ascii 11*21，Chinese 16*21
        font24CN: ascii 24*41，Chinese 32*41
    Color_Foreground: Foreground color
    Color_Background: Background color
```

- Draw number: Draw numbers at (Xstart Ystart) of the image buffer. You can select font, foreground, and background.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummber, sFONT* Font, UW
ORD Color_Foreground, UWORD Color_Background)
Parameters:
    Xstart: Xstart of numbers
    Ystart: Ystart of numbers
    Nummber: numbers displayed. It supports int type and 2147483647 is the
maximum supported
    Font: Ascii fonts, five fonts are available:
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: foreground
    Color_Background: background
```

- Display time: Display time at (Xstart Ystart) of the image buffer, you can configure fonts, foreground, and background.

This function is used for partial updating. Note that some of the e-Paper don't support partial updates and you cannot use partial updates all the time, which will have ghosts problems and destroy the display.

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font,
UWORD Color_Background, UWORD Color_Foreground)
Parameters:
    Xstart: Xstart of time
    Ystart: Ystart of time
    pTime: Structure of time
    Font: Ascii font, five fonts are available
        font8: 5*8
        font12: 7*12
        font16: 11*16
        font20: 14*20
        font24: 17*24
    Color_Foreground: foreground
    Color_Background: background
```

Resource

Document

- Schematic
- 2.9inch e-Paper (B) Specification

Demo codes

- Demo codes
- Github link

Development Software

- Thonny Python IDE (Windows V3.3.3)
- Zimo221.7z
- Image2Lcd.7z

Pico Quick Start

Download Firmware

- MicroPython Firmware Download
- C_Blink Firmware Download [Expand]

Video Tutorial [Expand]

- Pico Tutorial I - Basic Introduction
- Pico Tutorial II - GPIO [Expand]
- Pico Tutorial III - PWM [Expand]
- Pico Tutorial IV - ADC [Expand]
- Pico Tutorial V - UART [Expand]
- Pico Tutorial VI - To be continued... [Expand]

MicroPython Series

- 【MicroPython】 machine.Pin Function
- 【MicroPython】 machine.PWM Function
- 【MicroPython】 machine.ADC Function

[pdf] Specifications Dimension Guide Guide

User Guide Waveshare 2 9inch E Paper Ink Display Module B for Raspberry Pi Pico 296×128 Pixels Red Black White Electronics A1qdm5qHEBL m media amazon images I |||

Pico e-Paper 2.9 B Overview Pico e-Paper 2.9 B 2.9inch EPD Electronic Paper Display Module ... omment the definition according to the actual type of display you get. For example, if you have the Pico-ePaper-2.13, please modify the main.c file, uncomment line 18 or maybe it is line 19 . Set the...

lang:en score:29 filesize: 1.56 M page_count: 1 document date: 2023-10-23

- 【MicroPython】 machine.UART Function
- 【MicroPython】 machine.I2C Function
- 【MicroPython】 machine.SPI Function
- 【MicroPython】 rp2.StateMachine

C/C++ Series

- 【C/C++】 Windows Tutorial 1 - Environment Setting
- 【C/C++】 Windows Tutorial 1 - Create New Project

Arduino IDE Series

Install Arduino IDE

1. Download the Arduino IDE installation package from Arduino website.

2. Just click on "JUST DOWNLOAD".

3. Click to install after downloading.

4. **Note: You will be prompted to install the driver during the installation process, we can click Install.**

Install Arduino-Pico Core on Arduino IDE

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".

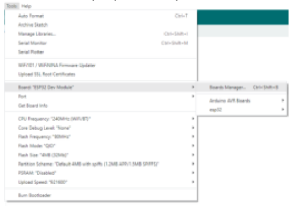
2. Add the following link in the additional development board manager URL, then click OK.

```
https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
```

Note: If you already have the ESP8266 board URL, you can separate the URLs with commas like this:

```
https://dl.espressif.com/dl/package_esp32_index.json,https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
```

3. Click on Tools -> Dev Board -> Dev Board Manager -> Search for pico, it shows installed since my computer has already installed it.

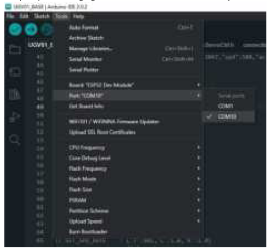


Upload Demo At the First Time

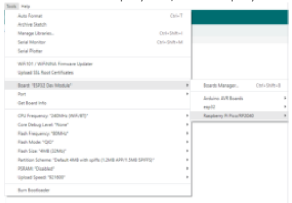
1. Press and hold the BOOTSET button on the Pico board, connect the Pico to the USB port of the computer via the Micro USB cable, and release the button when the computer recognizes a removable hard drive (RPI-RP2).

2. Download the demo, open arduino\PWM\D1-LED path under the D1-LED.ino.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).

4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.



5. Click Tool -> Dev Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.



6. After setting, click the right arrow to upload.

   - If you encounter problems during the period, you need to reinstall or replace the Arduino IDE version, uninstall the Arduino IDE needs to be uninstalled cleanly, after uninstalling the software you need to manually delete all the contents of the folder C:\Users\[name]\AppData\Local\Arduino15 (you need to show the hidden files in order to see it) and then reinstall.

Pico-W Series Tutorial (To be continued...)

Open Source Demo

- MicroPython Demo (GitHub)
- MicroPython Firmware/Blink Demo (C)
- Official Raspberry Pi C/C++ Demo
- Official Raspberry Pi MicroPython Demo
- Arduino Official C/C++ Demo

FAQ

Question: What is the usage environment of the e-ink screen?

Answer:

- 【Operating conditions】Temperature range: 0~50°C; Humidity range: 35%~65%RH.
- 【Storage conditions】 Temperature range: below 30°C; Humidity range: below 55%RH; Maximum storage time: 6 months.
- 【Transport conditions】 Temperature range: -25~70°C; Maximum transportation time: 10 days.
- 【After unpacking】Temperature range: 20°C±5°C; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours.

Question: Precautions for e-ink screen refresh?

Answer:

- Refresh mode
  - Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
  - Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full brush operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial brushing, please refer to product page description).
- Refresh rate
  - During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except for products that support the local brush function)
  - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode, or power off operation (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
  - During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).
- Usage scenarios
  - The e-ink screen is recommended for indoor use. If you use it outdoors, you need to avoid direct sunlight on the e-ink screen and take UV protection measures at the same time. When designing e-ink screen products, customers should pay attention to determining whether the use environment meets the temperature and humidity requirements of the e-ink screen.

Question: Chinese cannot be displayed on the e-ink screen?

Answer:
The Chinese character library of our routine uses the GB2312 encoding method, please change your xxx_test.c file to GB2312 encoding format, compile and download it, and then it can be displayed normally.

Question: After using for a period of time, the screen refresh (full refresh) has a serious afterimage problem that cannot be repaired?

Answer:
Power on the development board for a long time, after each refresh operation, it is recommended to set the screen to sleep mode or directly power off processing, otherwise, the screen may burn out when the screen is in a high voltage state for a long time.

Question: e-Paper shows black border?

Answer:
The border display color can be set through the Border Waveform Control register or the VCOM AND DATA INTERVAL SETTING register.

Question: What is the specification of the screen cable interface?

Answer:
0.5mm pitch, 24Pin.

- In this case, the customer needs to reduce the position of the round brush and clear the screen after 5 rounds of brushing (increasing the voltage of VCOM can improve the color, but it will increase the afterimage).

Question: After the ink screen enters deep sleep mode, can it be refreshed again?

Answer:
Yes, but you need to re-initialize the electronic paper with software.

Question: When the 2.9-inch EPD is in deep sleep mode, the first time it wakes up, the screen refresh will be unclean. How can I solve it?

Answer:
The process of re-awakening the e-ink screen is actually the process of re-powering on, so when the EPD wakes up, the screen must be cleared first, so as to avoid the afterimage phenomenon to the greatest extent.

Question: Are bare screen products shipped with a surface coating?

Answer:
with film.

Question: Does e-Paper have a built-in temperature sensor?

Answer:
Yes, you can also use the IIC pin external LM75 temperature sensor.

Question: When testing the program, the program keeps stuck on an e-Paper busy?

Answer:
It may be caused by the unsuccessful spi driver 1. Check whether the wiring is correct 2. Check whether the spi is turned on and whether the parameters are configured correctly (spi baud rate, spi mode, and other parameters).

Question: What is the refresh rate/lifetime of this e-ink screen?

Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times).

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 AM GMT+8 (Monday to Friday)

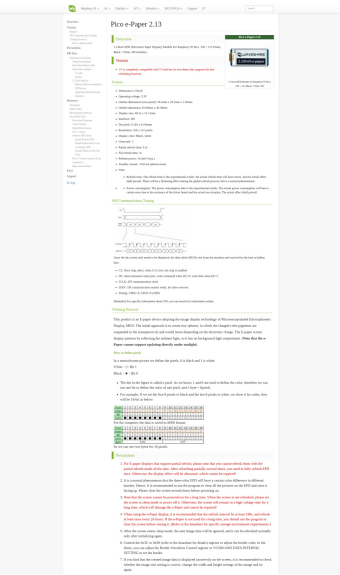
Submit Now

[pdf] Datasheet

Datenblatt zu 2 13 Zoll 212×104 ePaper Display Modul für Raspberry Pi Pico rot schwarz weiß BerryBaseEigenschaften Keine Hintergrundbeleuchtung zeigt den letzten Inhalt auch im ausgeschalteten Zustand noch lange Zeit anWS 19588berrybase de fr product datasheet 019234a4c0e07394ab4cad8b4afcee4e create srsltid AfmBOooza1n6Ql wip0NZpCtdJco DdO7o vMcYAhzT1ayztHFeQkr6 |||

Datenblatt zu 2,13 Zoll 212104 ePaper Display Modul fr Raspberry Pi Pico, rot/schwarz/wei 2,13 Zoll ... mm STANDBY-STROM 0,01uA fast keine Auflsung 212104 Pixel ANSICHTSWINKEL 170 Lieferumfang Pico-ePaper-2.13-B x1 Dokumentation / Downloads https://www.waveshare.com/wiki/Pico-ePaper-2.13-B P...

lang:de score:26 filesize: 4.44 M page_count: 2 document date: 2025-09-24



[Waveshare Pico e-Paper 2.13inch EPD Module for Raspberry Pi Pico: Development Guide & API]

Detailed development guide for the Waveshare Pico e-Paper 2.13inch EPD module with Raspberry Pi Pico. Features include 250x122 resolution, SPI interface, C/C++ & MicroPython demo codes, and comprehensive API documentation.

background.

```
void Paint_DrawNum(UWORD Xpoint, UWORD Ypoint, int32_t Nummber, sFONT* Font, UWORD Color_Foreground, UWORD Color_Background)
Parameters:
        Xstart: Xstart of numbers
        Ystart: Ystart of numbers
        Nummber: numbers displayed. It supports int type and 2147483647 is the maximum supported
        Font: Ascii fonts, five fonts are available:
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Color_Foreground: foreground
        Color_Background: background
```

- Display time: Display time at (Xstart Ystart) of the image buffer, you can configure fonts, foreground, and background.

This function is used for partial updating. Note that some of the e-Paper don't support partial updates and you cannot use partial updates all the time, which will have ghosts problems and destroy the display.

```
void Paint_DrawTime(UWORD Xstart, UWORD Ystart, PAINT_TIME *pTime, sFONT* Font, UWORD Color_Background, UWORD Color_Foreground)
Parameters:
        Xstart: Xstart of time
        Ystart: Ystart of time
        pTime: Structure of time
        Font: Ascii font, five fonts are available
                font8: 5*8
                font12: 7*12
                font16: 11*16
                font20: 14*20
                font24: 17*24
        Color_Foreground: foreground
        Color_Background: background
```

Resource

Document

- Schematic
- 2.13inch e-Paper Specification V4 (Newest)
- 2.13inch e-Paper Specification V3
- 2.13inch e-Paper Specification V2

Demo codes

- Demo codes
- Github link

Development Software

- Thonny Python IDE (Windows V3.3.3)
- Zimo221.7z
- Image2Lcd.7z

Pico Quick Start

Download Firmware

- MicroPython Firmware Download
- C_Blink Firmware Download [Expand]

Video Tutorial [Expand]

- Pico Tutorial I - Basic Introduction
- Pico Tutorial II - GPIO [Expand]
- Pico Tutorial III - PWM [Expand]
- Pico Tutorial IV - ADC [Expand]
- Pico Tutorial V - UART [Expand]
- Pico Tutorial VI - To be continued... [Expand]

MicroPython Series

- 【MicroPython】 machine.Pin Function
- 【MicroPython】 machine.PWM Function
- 【MicroPython】 machine.ADC Function
- 【MicroPython】 machine.UART Function
- 【MicroPython】 machine.I2C Function
- 【MicroPython】 machine.SPI Function
- 【MicroPython】 rp2.StateMachine

C/C++ Series

- 【C/C++】 Windows Tutorial 1 - Environment Setting
- 【C/C++】 Windows Tutorial 1 - Create New Project

Arduino IDE Series

Install Arduino IDE

1. Download the Arduino IDE installation package from Arduino website.

2. Just click on "JUST DOWNLOAD".

3. Click to install after downloading.

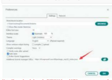
4. **Note: You will be prompted to install the driver during the installation process, we can click Install.**

Install Arduino-Pico Core on Arduino IDE

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".

2. Add the following link in the additional development board manager URL, then click OK.

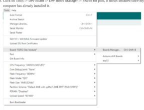
```
https://dl.espressif.com/dl/package_esp32_index.json
```

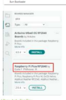


Note: If you already have the ESP8266 board URL, you can separate the URLs with commas like this:

```
https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

3. Click on Tools -> Dev Board -> Dev Board Manager -> Search for pico, it shows installed since my computer has already installed it.



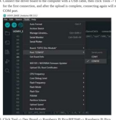


Upload Demo At the First Time

1. Press and hold the BOOTSET button on the Pico board, connect the Pico to the USB port of the computer via the Micro USB cable, and release the button when the computer recognizes a removable hard drive (RPI-RP2).

2. Download the demo, open arduino\PWM\D1-LED path under the D1-LED.ino.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).

4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.



5. Click Tool -> Dev Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.

6. After setting, click the right arrow to upload.

   - If you encounter problems during the period, you need to reinstall or replace the Arduino IDE version, uninstall the Arduino IDE needs to be uninstalled cleanly, after uninstalling the software you need to manually delete all the contents of the folder C:\Users\[name]\AppData\Local\Arduino15 (you need to show the hidden files in order to see it) and then reinstall.

Pico-W Series Tutorial (To be continued...)

Open Source Demo

- MicroPython Demo (GitHub)
- MicroPython Firmware/Blink Demo (C)
- Official Raspberry Pi C/C++ Demo
- Official Raspberry Pi MicroPython Demo
- Arduino Official C/C++ Demo

FAQ

Question: What is the usage environment of the e-ink screen?

Answer:

- 【Operating conditions】Temperature range: 0~50°C; Humidity range: 35%~65%RH.
- 【Storage conditions】 Temperature range: below 30°C; Humidity range: below 55%RH; Maximum storage time: 6 months.
- 【Transport conditions】 Temperature range: -25~70°C; Maximum transportation time: 10 days.
- 【After unpacking】Temperature range: 20°C±5°C; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours.

Question: Precautions for e-ink screen refresh?

Answer:

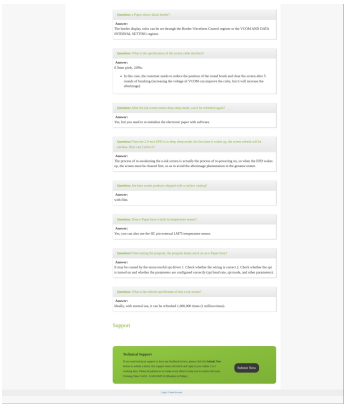
- Refresh mode
  - Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
  - Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full brush operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial brushing, please refer to product page description).
- Refresh rate
  - During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except for products that support the local brush function)
  - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode, or power off operation (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
  - During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).
- Usage scenarios
  - The e-ink screen is recommended for indoor use. If you use it outdoors, you need to avoid direct sunlight on the e-ink screen and take UV protection measures at the same time. When designing e-ink screen products, customers should pay attention to determining whether the use environment meets the temperature and humidity requirements of the e-ink screen.

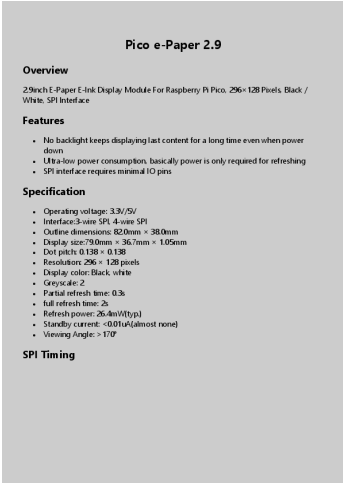
Question: Chinese cannot be displayed on the e-ink screen?

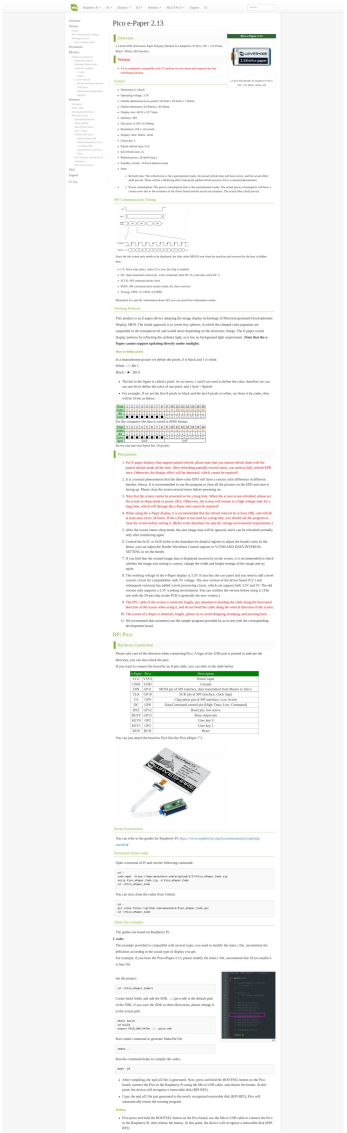
Answer:
The Chinese character library of our routine uses the GB2312 encoding method, please change your xxx_test.c file to GB2312 encoding format, compile and download it, and then it can be displayed normally.

Question: After using for a period of time, the screen refresh (full refresh) has a serious afterimage problem that cannot be repaired?

Answer:
Power on the development board for a long time, after each refresh operation, it is recommended to set the screen to sleep mode or directly power off processing, otherwise, the screen may burn out when the screen is in a high voltage state for a long time.





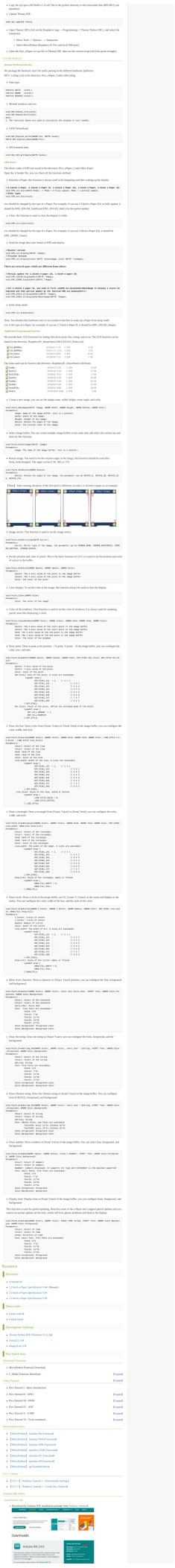


[pdf] Specifications Dimension Guide

1 Дуйсенбаев DOC009844715 static chipdip ru lib 844

Pico e-Paper 2.9 Overview 2.9inch E-Paper E-Ink Display Module For Raspberry Pi Pico, 296128 Pixels, ... Data; Low: Command Reset pin, low active Busy pin You can just attach the board to Pico like the Pico-ePaper-2.13 Setup Environment You can refer to the guides of Raspberry Pi: https://www.raspberr...

lang:en score:19 filesize: 461.81 K page_count: 15 document date: 2021-06-09



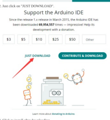
[pdf] Installation Guide Frequently Asked Questions Dimension Guide

Pico e Paper 2 13Dimension 13inch Operating voltage 3 3V Outline dimensions raw panel 59 2mm x 29 1 05mm dimension 65 00mm 30 50mmA1qZJy QlnLm media amazon images I A1qZJy QlnL ref dp product quick view |||

Raspberry Pi AI Displays IoT Robotics MCU/FPGA Support IC Search Overview Version Feature SP ... omment the definition according to the actual type of display you get. For example, if you have the Pico-ePaper-2.13, please modify the main.c file, uncomment line 18 or maybe it is line 19 . Set the...

lang:en score:13 filesize: 1.52 M page_count: 1 document date: 2023-08-21

2. Just click on "JUST DOWNLOAD".



3. Click to install after downloading.

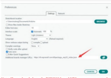
4. **Note: You will be prompted to install the driver during the installation process, we can click Install.**

Install Arduino-Pico Core on Arduino IDE

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".

2. Add the following link in the additional development board manager URL, then click OK.

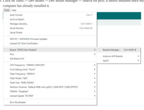
```
https://dl.espressif.com/dl/package_esp32_index.json
```

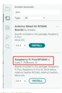


Note: If you already have the ESP8266 board URL, you can separate the URLs with commas like this:

```
https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

3. Click on Tools -> Dev Board -> Dev Board Manager -> Search for pico, it shows installed since my computer has already installed it.

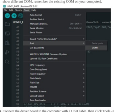




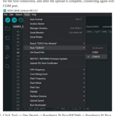
Upload Demo At the First Time

1. Press and hold the BOOTSET button on the Pico board, connect the Pico to the USB port of the computer via the Micro USB cable, and release the button when the computer recognizes a removable hard drive (RPI-RP2).

2. Download the demo, open arduino\PWM\D1-LED path under the D1-LED.ino.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).



4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.



5. Click Tool -> Dev Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.

6. After setting, click the right arrow to upload.

- If you encounter problems during the period, you need to reinstall or replace the Arduino IDE version, uninstall the Arduino IDE needs to be uninstalled cleanly, after uninstalling the software you need to manually delete all the contents of the folder C:\Users\[name]\AppData\Local\Arduino15 (you need to show the hidden files in order to see it) and then reinstall.

Pico-W Series Tutorial (To be continued...)

Open Source Demo

- MicroPython Demo (GitHub)
- MicroPython Firmware/Blink Demo (C)
- Official Raspberry Pi C/C++ Demo
- Official Raspberry Pi MicroPython Demo
- Arduino Official C/C++ Demo

FAQ

Question: What is the usage environment of the e-ink screen?

Answer:

- 【Operating conditions】Temperature range: 0~50°C; Humidity range: 35%~65%RH.
- 【Storage conditions】 Temperature range: below 30°C; Humidity range: below 55%RH; Maximum storage time: 6 months.
- 【Transport conditions】 Temperature range: -25~70°C; Maximum transportation time: 10 days.
- 【After unpacking】Temperature range: 20°C±5°C; Humidity range: 50±5%RH; Maximum storage time: Assemble within 72 hours.

Question: Precautions for e-ink screen refresh?

Answer:

- Refresh mode
  - Full refresh: The electronic ink screen will flicker several times during the refresh process (the number of flickers depends on the refresh time), and the flicker is to remove the afterimage to achieve the best display effect.
  - Partial refresh: The electronic ink screen has no flickering effect during the refresh process. Users who use the partial brushing function note that after refreshing several times, a full brush operation should be performed to remove the residual image, otherwise the residual image problem will become more and more serious, or even damage the screen (currently only some black and white e-ink screens support partial brushing, please refer to product page description).
- Refresh rate
  - During use, it is recommended that customers set the refresh interval of the e-ink screen to at least 180 seconds (except for products that support the local brush function)
  - During the standby process (that is, after the refresh operation), it is recommended that the customer set the e-ink screen to sleep mode, or power off operation (the power supply part of the ink screen can be disconnected with an analog switch) to reduce power consumption and prolong the life of the e-ink screen. (If some e-ink screens are powered on for a long time, the screen will be damaged beyond repair.)
  - During the use of the three-color e-ink screen, it is recommended that customers update the display screen at least once every 24 hours (if the screen remains the same screen for a long time, the screen burn will be difficult to repair).
- Usage scenarios
  - The e-ink screen is recommended for indoor use. If you use it outdoors, you need to avoid direct sunlight on the e-ink screen and take UV protection measures at the same time. When designing e-ink screen products, customers should pay attention to determining whether the use environment meets the temperature and humidity requirements of the e-ink screen.

Question: Chinese cannot be displayed on the e-ink screen?

Answer:
The Chinese character library of our routine uses the GB2312 encoding method, please change your xxx_test.c file to GB2312 encoding format, compile and download it, and then it can be displayed normally.

Question: After using for a period of time, the screen refresh (full refresh) has a serious afterimage problem that cannot be repaired?

Answer:
Power on the development board for a long time, after each refresh operation, it is recommended to set the screen to sleep mode or directly power off processing, otherwise, the screen may burn out when the screen is in a high voltage state for a long time.

Question: e-Paper shows black border?

Answer:
The border display color can be set through the Border Waveform Control register or the VCOM AND DATA INTERVAL SETTING register.

Question: What is the specification of the screen cable interface?

Answer:
0.5mm pitch, 24Pin.

- In this case, the customer needs to reduce the position of the round brush and clear the screen after 5 rounds of brushing (increasing the voltage of VCOM can improve the color, but it will increase the afterimage).

Question: After the ink screen enters deep sleep mode, can it be refreshed again?

Answer:
Yes, but you need to re-initialize the electronic paper with software.

Question: When the 2.9-inch EPD is in deep sleep mode, the first time it wakes up, the screen refresh will be unclean. How can I solve it?

Answer:
The process of re-awakening the e-ink screen is actually the process of re-powering on, so when the EPD wakes up, the screen must be cleared first, so as to avoid the afterimage phenomenon to the greatest extent.

Question: Are bare screen products shipped with a surface coating?

Answer:
with film.

Question: Does e-Paper have a built-in temperature sensor?

Answer:
Yes, you can also use the IIC pin external LM75 temperature sensor.

Question: When testing the program, the program keeps stuck on an e-Paper busy?

Answer:
It may be caused by the unsuccessful spi driver 1. Check whether the wiring is correct 2. Check whether the spi is turned on and whether the parameters are configured correctly (spi baud rate, spi mode, and other parameters).

Question: What is the refresh rate/lifetime of this e-ink screen?

Answer:
Ideally, with normal use, it can be refreshed 1,000,000 times (1 million times).

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 AM GMT+8 (Monday to Friday)

Submit Now