Bare E-Ink Displays Crash Course

Created by Liz Clark

https://learn.adafruit.com/bare-e-ink-displays-crash-course

Last updated on 2025-11-11 10:20:34 PM EST

Table of Contents

Overview

What's the deal with all of these eInk displays? Why is this example code that I found showing a bunch of jumbled pixels instead of a beautiful, dithered Blinka? If these are your questions, then you are in the right place. This guide will show you everything you need to get going with your chosen Adafruit bare eInk display.

Each display in the shop has its own dedicated page here with usage examples for Arduino, CircuitPython or Python. You'll be able to find your display's page and use the example code.

Steps to Success

To ensure that you have a painless and beautifully dithered experience, first make your way over to the [Hardware Setup page](https://adafru.it/1atv) (https://adafru.it/1atv). This page walks through your hardware options for interfacing with a bare eInk display.

Bare E-Ink Displays Crash Course
By Liz Clark
Hardware Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/hardware-setup

If you're using **CircuitPython with displayio**, then you can go directly to the page for your chosen display. You'll be able to download the Project Bundle, which includes the driver in the **/lib** folder that you'll need to interface with your display.

However, if you know you're going to use **Arduino or Python**, you'll want to go to either the Arduino Setup (https://adafru.it/1atw) or Python Setup (https://adafru.it/1atx) page(s). These pages list the steps necessary for installing libraries and dependencies.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

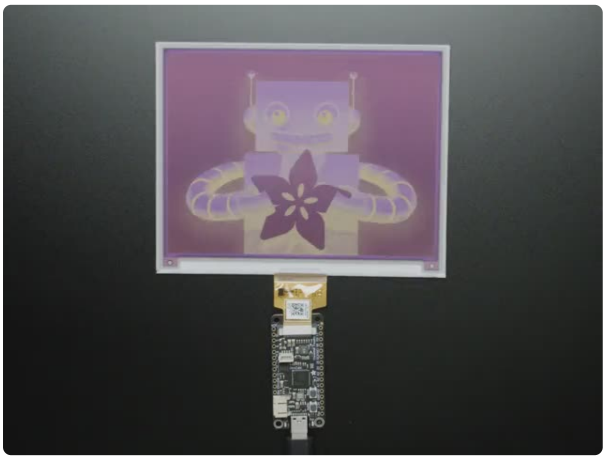
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Hardware Setup

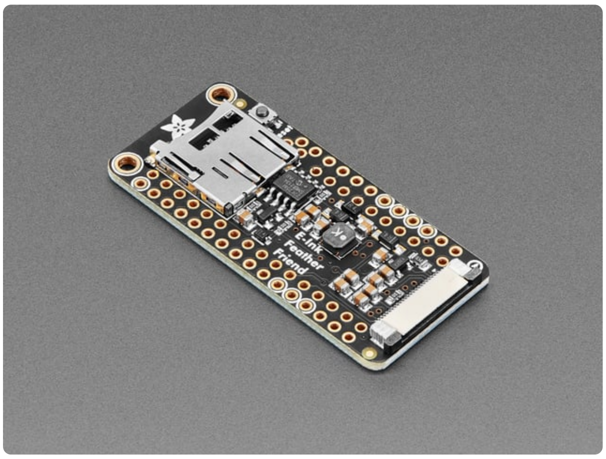
You'll need a board or breakout with a 24-pin connector to use a bare eInk display. There are a few options in the Adafruit shop:



Adafruit RP2040 Feather ThinkInk for 24-pin E-Paper Displays

Easy e-paper and RP2040 finally come to your Feather with this Adafruit RP2040 Feather Think Ink that's designed to make it a breeze to add...

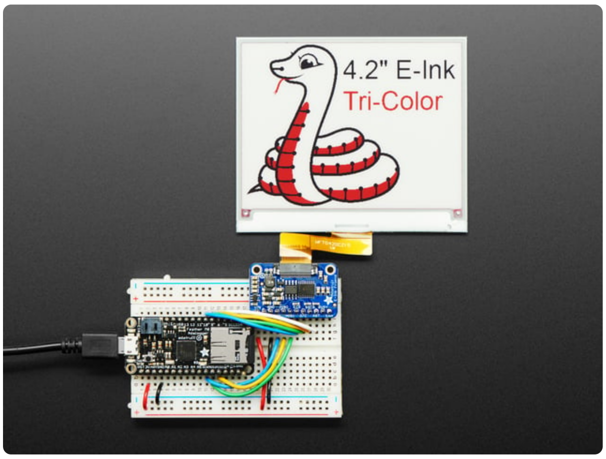
https://www.adafruit.com/product/5727



Adafruit eInk Feather Friend with 32KB SRAM

Easy e-paper finally comes to your Feather with this Adafruit EInk Feather Friend that's designed to make it a breeze to add an eInk display. Chances are...

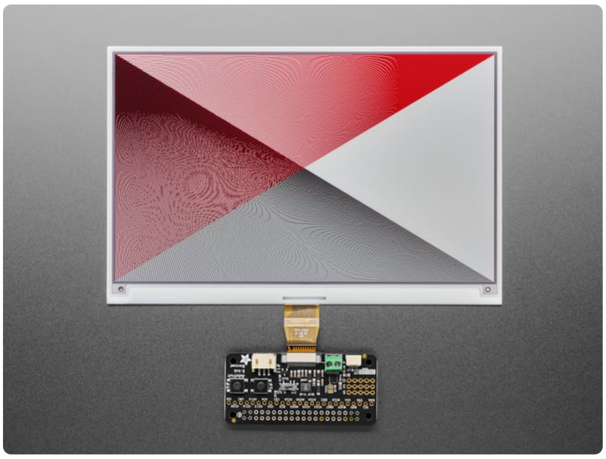
https://www.adafruit.com/product/4446



Adafruit eInk Breakout Friend with 32KB SRAM

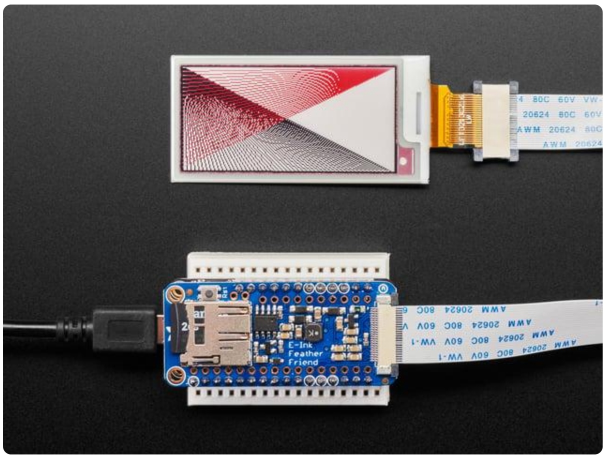
Add some bi-stability to your life with an Adafruit EInk Breakout Friend - your fun pal who's happy to lend a hand when you want to work with EInk/E-Paper...

https://www.adafruit.com/product/4224

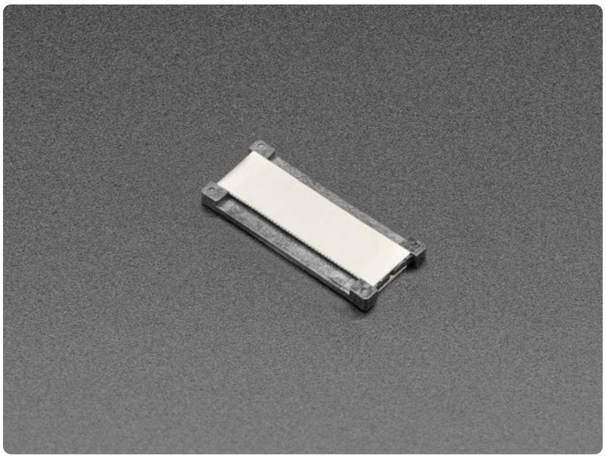


[Adafruit E-Ink Bonnet for Raspberry Pi - 24-pin E-Paper Displays](https://www.adafruit.com/product/6418)
Easy e-paper finally comes to Raspberry Pi, with this bonnet that's designed to make it a breeze to add any 'standard' pinout 24-pin flex-connector E-Ink display. This...
https://www.adafruit.com/product/6418

You can also add a cable extender if you want to give your display cable some extra slack:



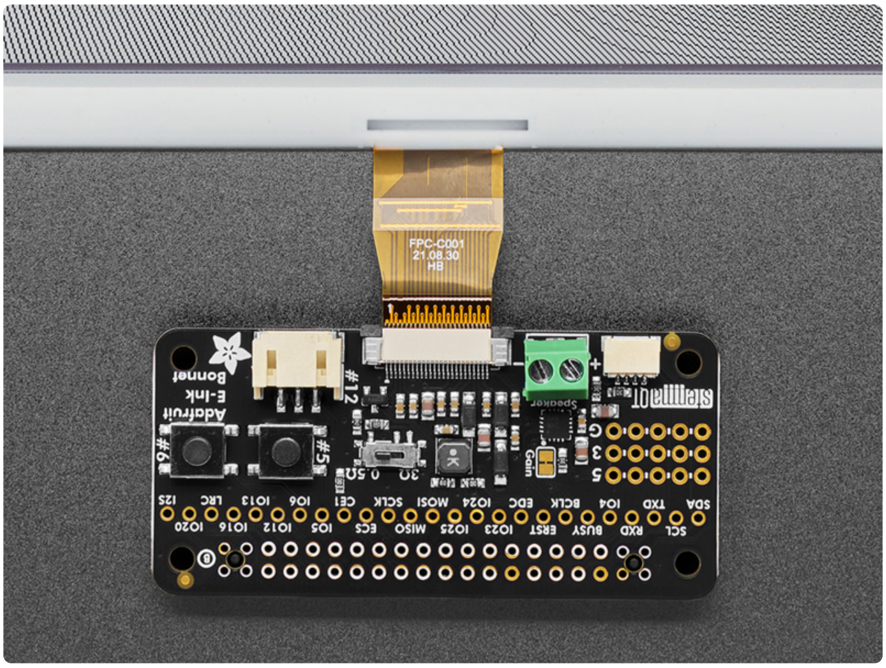
[24-pin eInk / ePaper Extension Cable 0.5mm Pitch - 25cm Long](https://www.adafruit.com/product/4230)
When you need more distance between you and your eInk display, use this 24-pin 0.5mm pitch FPC cable with an extender adapter to...
https://www.adafruit.com/product/4230



[24-pin 0.5mm FFC / FPC Extender](https://www.adafruit.com/product/4524)
This 24-pin FFC / FPC Extender is very simple. It lets you connect two 24-pin 0.5mm pitch FFC (Flexible Flat Cable) / FPC (Flexible Printed Circuit) cables...
https://www.adafruit.com/product/4524

Hardware Setup

The display ribbon cables plug into the connectors by sliding out the black tabs on the connector, inserting the cable and then pushing the black tabs back in to secure the cable.



Pinouts

Your pinout will vary depending on the breakout or dev board that you choose.

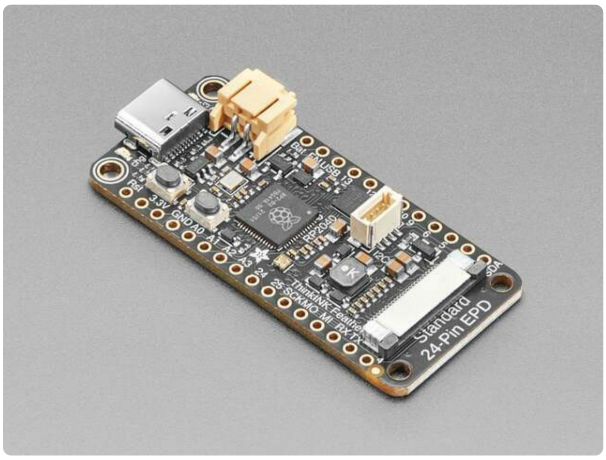
Feather RP2040 ThinkInk

The Feather RP2040 ThinkInk has the onboard 24-pin connector. It uses these specific pins included as a part of its board definition in both Arduino and CircuitPython:

- **EPD_BUSY/GPIO16** - The Busy Detect pin for e-Ink displays. It is available as `EPD_BUSY` in CircuitPython and `PIN_EPD_BUSY` in Arduino.
- **EPD_RESET/GPIO17** - The Reset pin for e-Ink displays. It is available as `EPD_RESET` in CircuitPython and `PIN_EPD_RESET` in Arduino.
- **EPD_DC/GPIO18** - The Data Command pin for e-Ink displays. It is available as `EPD_DC` in CircuitPython and `PIN_EPD_DC` in Arduino.
- **EPD_CS/GPIO19** - The Chip Select pin for e-Ink displays. It is available as `EPD_CS` in CircuitPython and `PIN_EPD_CS` in Arduino.
- **EPD_SCK/GPIO22** - The SPI SCK pin for e-Ink displays. It is available as `EPD_SCK` in CircuitPython and `PIN_EPD_SCK` in Arduino.
- **EPD_MOSI/GPIO23** - The SPI MOSI pin for e-Ink displays. It is available as `EPD_MOSI` in CircuitPython and `PIN_EPD_MOSI` in Arduino.

You'll use `busio` to initialize the SPI bus for the 24-pin connector:

- `spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)`



Adafruit Feather RP2040 ThinkInk
By Liz Clark
  Pinouts

https://learn.adafruit.com/adafruit-rp2040-feather-thinkink/pinouts

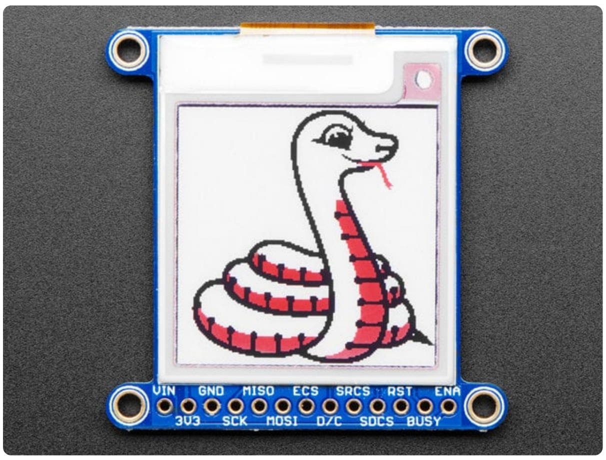
eInk Feather Friend FeatherWing

The eInk Feather Friend is a FeatherWing that has the included 24-pin connector. It uses the following pins when connected to a Feather board:

- **SPI MOSI/MISO/SCK**
- **SD CS** to Pin **D5**
- **SRAM CS** to Pin **D6**
- **eInk CS** to Pin **D9**
- **eInk DC** to Pin **D10**

eInk Breakout Friend

The eInk Breakout Friend is a breakout board that can be connected to any dev board. It has an onboard 24-pin connector and pins broken out along the edge to interface with a connected display.

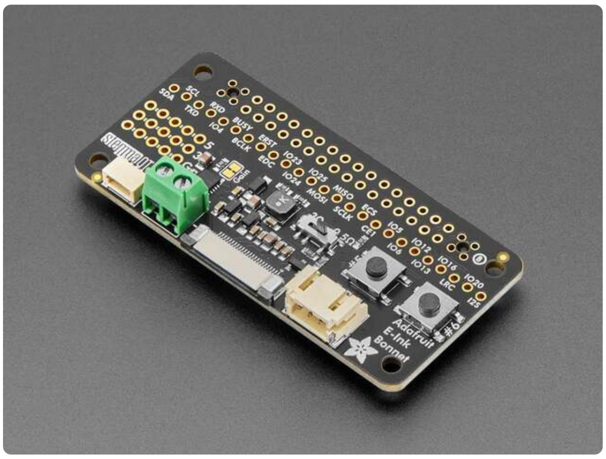


Adafruit eInk Display Breakouts and FeatherWings
By lady ada
  Pinouts

https://learn.adafruit.com/adafruit-eink-display-breakouts/pinouts eInk Bonnet for Raspberry Pi

The eInk Bonnet plugs directly into the 2x20 header on your Raspberry Pi, letting you connect a bare eInk display easily via the 24-pin connector. The following pins are used for interfacing with a display:

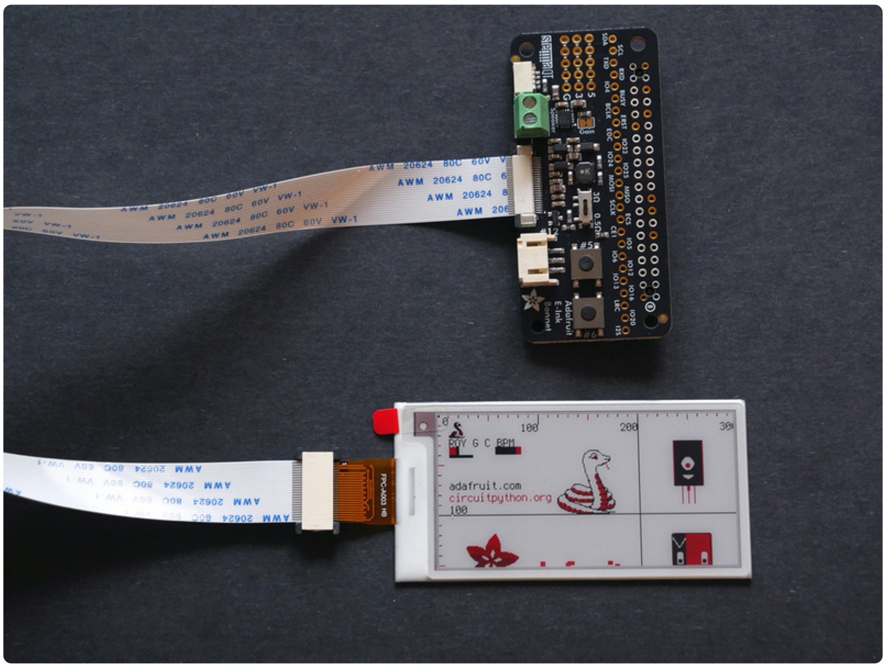
- **MOSI/MISO/SCK** - SPI pins are connected to the default **SPI** port on the Raspberry Pi.
- **CS** on Raspberry Pi **CE 0.**
- **DC** on Raspberry Pi **GPIO 22.**
- **Reset** on Raspberry Pi **GPIO 27.**
- **Busy** on Raspberry Pi **GPIO 17**



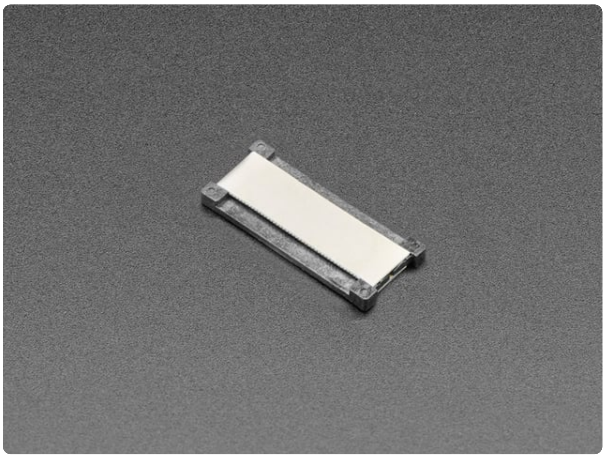
Adafruit E-Ink Bonnet for Raspberry Pi
By Liz Clark
Pinouts

https://learn.adafruit.com/adafruit-e-ink-bonnet-for-raspberry-pi/pinouts

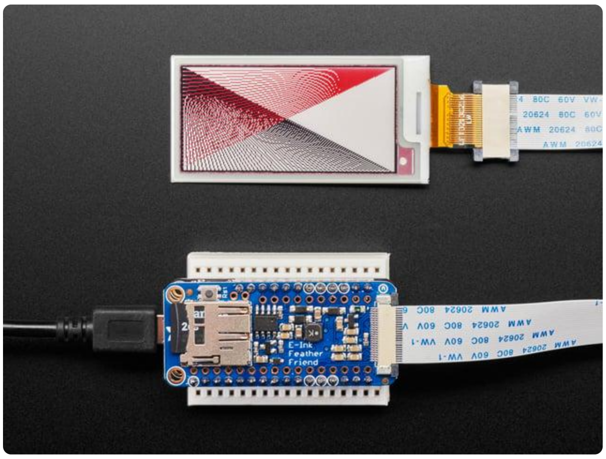
Using an Extension Cable



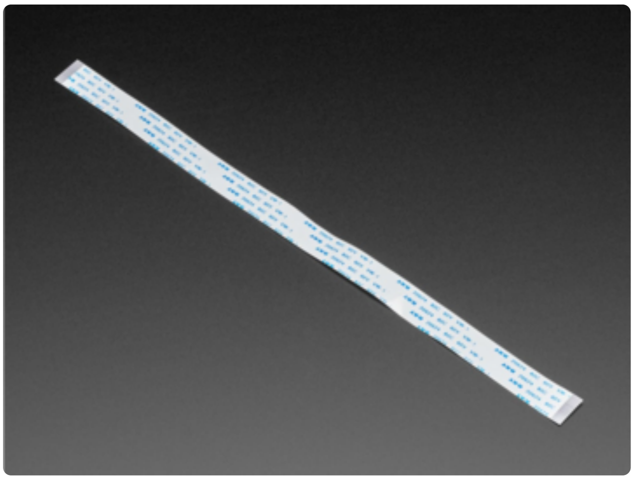
You can use a 24-pin eInk extension cable with an FPC extender to give your eInk display and hardware some personal space.

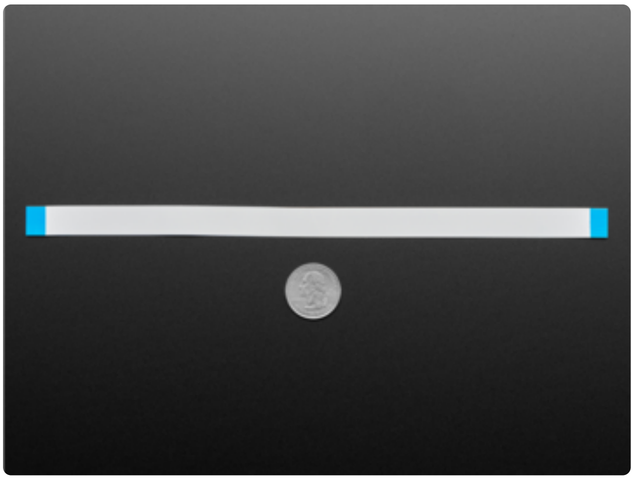


[24-pin 0.5mm FFC / FPC Extender](https://www.adafruit.com/product/4524)
This 24-pin FFC / FPC Extender is very simple. It lets you connect two 24-pin 0.5mm pitch FFC (Flexible Flat Cable) / FPC (Flexible Printed Circuit) cables...
https://www.adafruit.com/product/4524



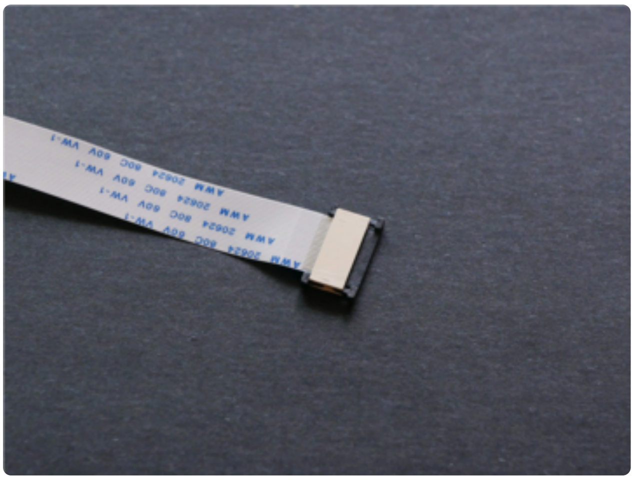
[24-pin eInk / ePaper Extension Cable 0.5mm Pitch - 25cm Long](https://www.adafruit.com/product/4230)
When you need more distance between you and your eInk display, use this 24-pin 0.5mm pitch FPC cable with an extender adapter to...
https://www.adafruit.com/product/4230



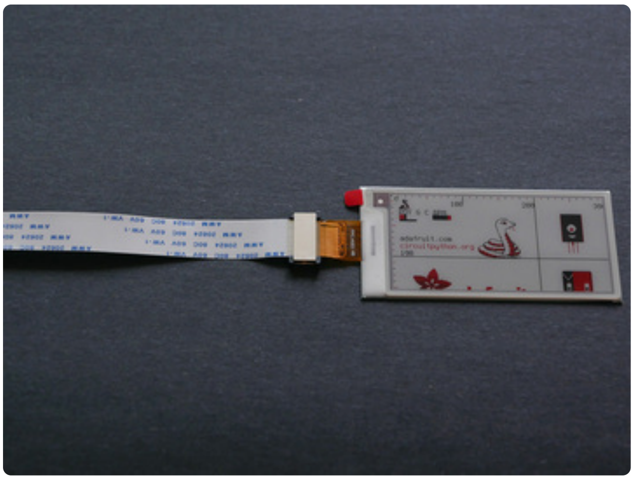


Each extension cable has **blue stripes** to denote the back of the cable. On the front of the cable, opposite the blue stripes, are the connector pins that make contact with the FPC connector pins on the extender or breakout.

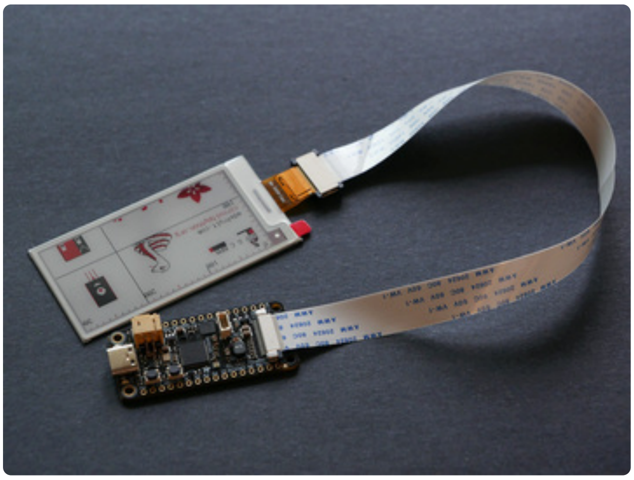
Making Connections



You'll use an FPC extender with the extension cable. Gently pull the FPC extender black latch out towards you and insert the cable into the slot with the front side facing you (blue stripe facing away from you). Push the FPC extender latch in to secure the cable in place.



Take your display's cable and insert it into the opposite side of the FPC extender. The contacts on the display cable should be facing up, same as the extension cable.



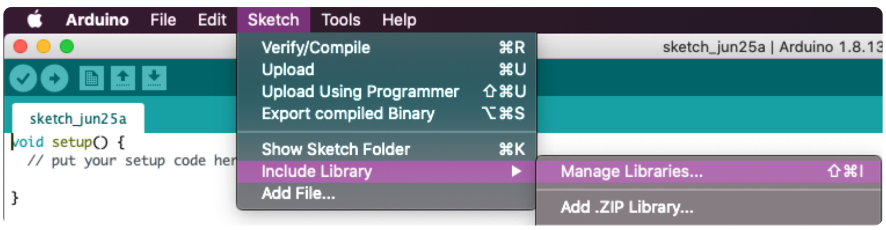
Then, insert the other end of the extension cable into your board or breakout 24-pin connector. Do this by gently pulling out the latch, inserting the cable with its contacts facing up and then pushing the latch back in to secure the cable.

Arduino Setup

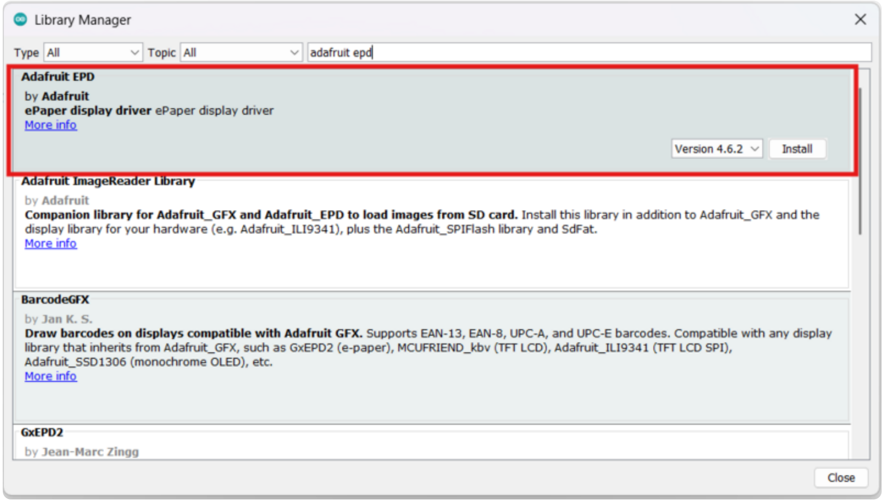
Using the eInk displays with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the [Adafruit_EPD](https://adafru.it/BRK) (https://adafru.it/BRK) library, and running the provided example code.

Library Installation

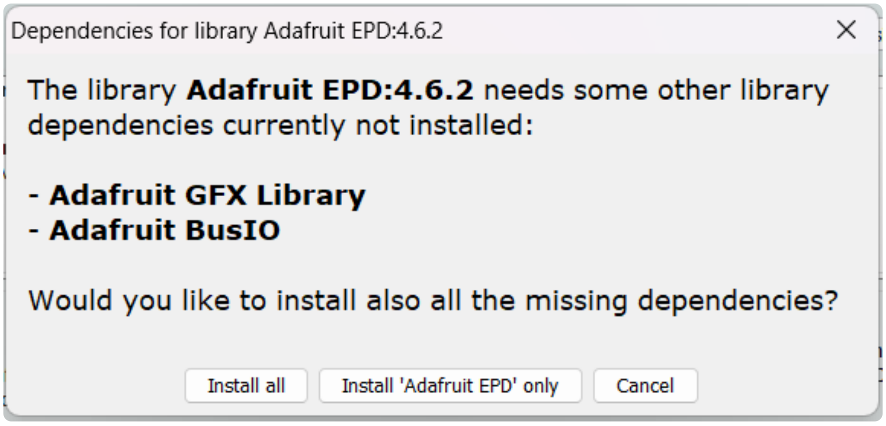
You can install the **Adafruit_EPD** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_EPD**, and select the **Adafruit EPD** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

e dependencies are already installed, you must make sure you update
n through the Arduino Library Manager before loading the example!

Example Code

Each display will have Arduino example code for you to use. After following the library installation steps, go to the page for your display and upload the code to your board.

Python Setup

It's easy to use eInk breakouts with Python and the Adafruit CircuitPython EPD (https://adafru.it/BTd) library. This library allows you to easily write Python code to control the display.

Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

e this is not a kernel driver that will let you have the console appear on the
. However, this is handy when you want to use the eInk display purely
ı 'user Python' code!

can only use this technique with Linux/computer devices that have
lware SPI support, and not all single board computers have an SPI device,
heck before continuing

Setup Virtual Environment

If you are installing on the Bookworm version of Raspberry Pi OS or later, you will need to install your python modules in a virtual environment. You can find more information in the Python Virtual Environment Usage on Raspberry Pi (https://adafru.it/19a5) guide. To Install and activate the virtual environment, use the following commands:

```
sudo apt install python3-venv
python -m venv env --system-site-packages
```

To activate the virtual environment:

```
source env/bin/activate
```

Install Adafruit_Blinka

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling SPI on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready](https://adafru.it/BSN) (https://adafru.it/BSN)!

Python Installation of EPD Library

Once that's done, from your command line run the following command:

```
pip3 install adafruit-circuitpython-epd
```

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

If that complains about pip3 not being installed, then run this first to install it:

```
sudo apt-get install python3-pip
```

Download font5x8.bin

This library also requires a font file to run! You can download it below. Before continuing, make sure the folder you are running scripts from contains the **font5x8.bin** file.

https://adafru.it/Xbr

Alternatively, you can use `wget` to directly download the file to your pi:

```
wget https://github.com/adafruit/Adafruit_CircuitPython_framebuf/raw/main/examples/
font5x8.bin
```

DejaVu TTF Font

Raspberry Pi usually comes with the DejaVu font already installed, but in case it didn't, you can run the following to install it:

```
sudo apt-get install fonts-dejavu
```

This package was previously calls **ttf-dejavu**, so if you are running an older version of Raspberry Pi OS, it may be called that.

Pillow Library

Some of the examples also use PIL, the Python Imaging Library, to allow graphics and using text with custom fonts. There are several system libraries that PIL relies on, so installing via a package manager is the easiest way to bring in everything:

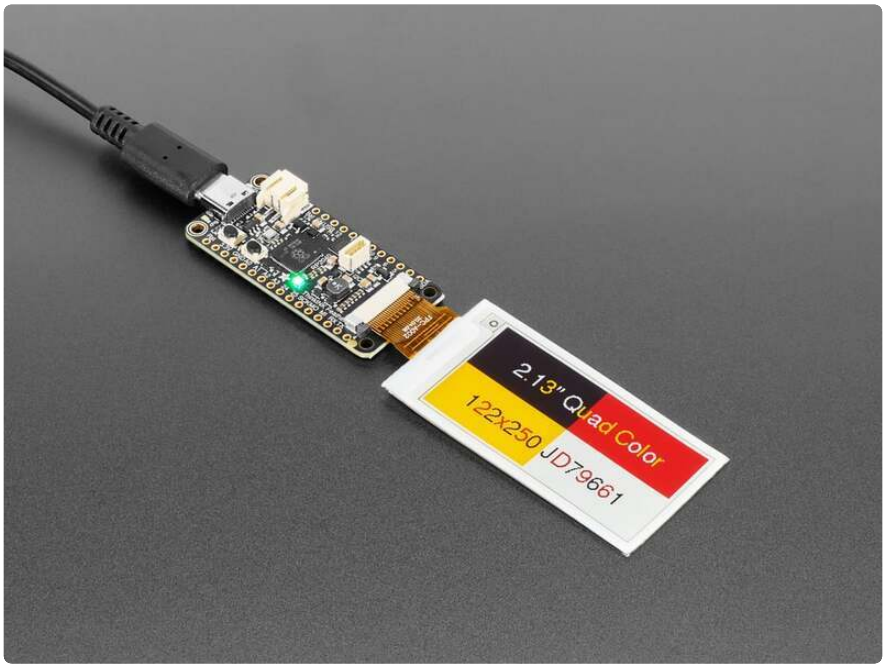
```
sudo apt-get install python3-pil
```

Chip Enable Lines

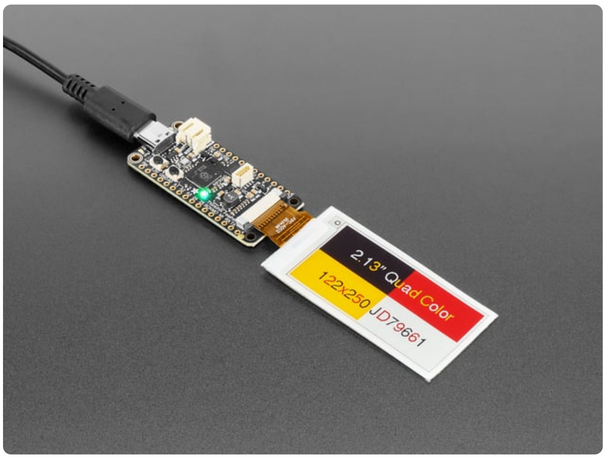
Follow these instructions (https://adafru.it/19fg) for dealing with SPI chip enable line issues.

That's it. You should be ready to go.

2.13" 250x122 Quad-Color eInk JD79661



This page will get you up and running with the 2.13" 250x122 Quad-Color eInk display. This display uses the JD79661 driver.



2.13" 250x122 Quad-Color eInk / ePaper - Bare Display
What's better than our tri-color E-Paper displays? Quad-color of course! This is a 2.13" 250x122...
https://www.adafruit.com/product/6373

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.13" Quadcolor EPD with JD79661 chipset
ThinkInk_213_Quadcolor_AJHE5 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
                                     EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/yellow/black/white");
  display.begin(THINKINK_QUADCOLOR);
}

void loop() {
```

```
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  String text = "QuadColor";
  uint16_t colors[] = {EPD_BLACK, EPD_RED, EPD_YELLOW};

  for (int i = 0; i < text.length(); i++) {
    // Change color for every character (0: BLACK, 1: RED, 2: YELLOW, 3: BLACK,
etc.)
    display.setTextColor(colors[i % 3]);
    display.print(text.charAt(i));
  }
  display.display();

  delay(15000);

  Serial.println("Color quadrant demo");
  display.clearBuffer();
  // Top-left quadrant - EPD_BLACK
  display.fillRect(0, 0, display.width() / 2, display.height() / 2, EPD_BLACK);
  // Top-right quadrant - EPD_RED
  display.fillRect(display.width() / 2, 0, display.width() / 2, display.height() /
2, EPD_RED);
  // Bottom-left quadrant - EPD_YELLOW
  display.fillRect(0, display.height() / 2, display.width() / 2, display.height() /
2, EPD_YELLOW);
  // Bottom-right quadrant - assume you have a 4th color like EPD_WHITE or another
color
  display.fillRect(display.width() / 2, display.height() / 2, display.width() / 2,
display.height() / 2, EPD_WHITE);

  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }
  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  for (int16_t i = 0; i < display.width(); i += 4) {
     display.drawLine(display.width()/2, display.height()-1, i, 0,
                      EPD_YELLOW);
  }

  display.display();

  delay(15000);
}
```

```
void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.

CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_JD79661** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_jd79661.mpy**

Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 2.13" Quad Color Display"""
```

```python
import time

import board
import displayio
from fourwire import FourWire

import adafruit_jd79661

displayio.release_displays()

spi = board.SPI()
epd_cs = board.D9
epd_dc = board.D10
epd_reset = board.D6
epd_busy = board.D5

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_jd79661.JD79661(
    display_bus,
    width=250,
    height=122,
    busy_pin=epd_busy,
    rotation=270,
    colstart=0,
    highlight_color=0x00FF00,
    highlight_color2=0xFF0000,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
# Always refresh a little longer. It's not a problem to refresh
# a few seconds more, but it's terrible to refresh too early
# (the display will throw an exception when if the refresh
# is too soon)
print("waited correct time")


# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

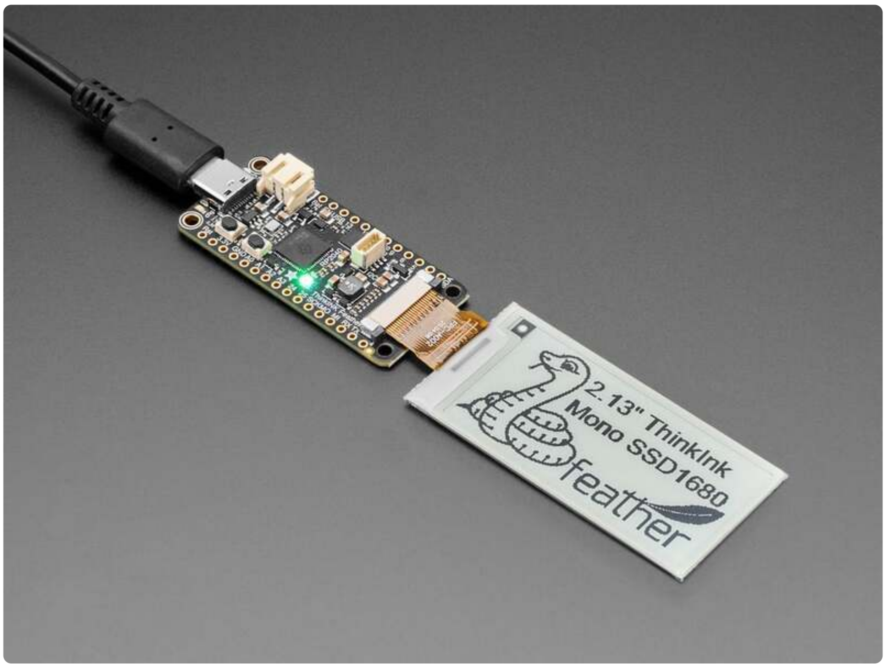
`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT
"""Blinka EPD Demo for the Quad Color eInk"""
import board
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.epd import Adafruit_EPD
from adafruit_epd.jd79661 import Adafruit_JD79661

# create the spi device and pins we will need
spi = board.SPI()
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D25)
srcs = None
rst = digitalio.DigitalInOut(board.D27)  # can be None to not use this pin
busy = digitalio.DigitalInOut(board.D17)  # can be None to not use this pin

display = Adafruit_JD79661(122, 250, spi,
                          cs_pin=ecs, dc_pin=dc, sramcs_pin=srcs,
                          rst_pin=rst, busy_pin=busy)

display.rotation = 3
width = display.width
height = display.height
image = Image.new("RGB", (width, height))
```

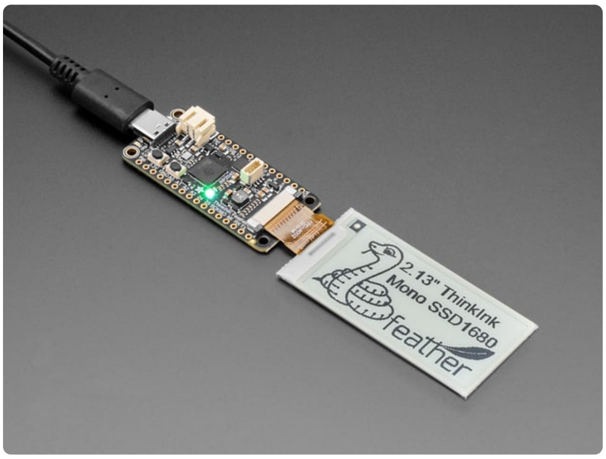
```
WHITE = (0xFF, 0xFF, 0xFF)
YELLOW = (0xFF, 0xFF, 0x00)
RED = (0xFF, 0x00, 0x00)
BLACK = (0x00, 0x00, 0x00)

# clear the buffer
display.fill(Adafruit_EPD.WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 5
shape_width = 30
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=YELLOW, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=RED, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=RED,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=YELLOW)
draw.line((x, top, x + shape_width, bottom), fill=YELLOW)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 20)

draw.text((x, top), "Hello", font=font, fill=YELLOW)
draw.text((x, top + 20), "World!", font=font, fill=YELLOW)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

2.13" 250x122 Monochrome or Gray eInk SSD1680Z



This page will get you up and running with the 2.13" 250x122 monochrome eInk display. This display uses the SSD1680Z driver.



2.13" 250x122 Monochrome or Gray eInk / ePaper - Bare Display
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6383

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.13" Monochrome displays with 250x122 pixels and SSD1680Z chipset
ThinkInk_213_Mono_GDEY0213B74 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("2.13 Monochrome EPD with SSD1680Z chipset");
  display.begin(THINKINK_MONO);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.

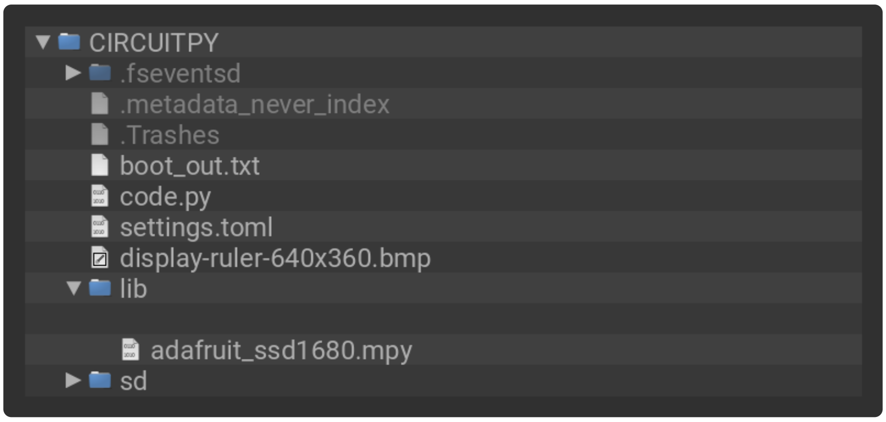
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_SSD1680** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_ssd1680.mpy**



Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: MIT

"""Simple test script for 2.13" 250x122 monochrome display."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_ssd1680

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
```

```
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_ssd1680.SSD1680(
    display_bus,
    width=250,
    height=122,
    busy_pin=epd_busy,
    highlight_color=0xFF0000,
    rotation=270,
    colstart=0,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

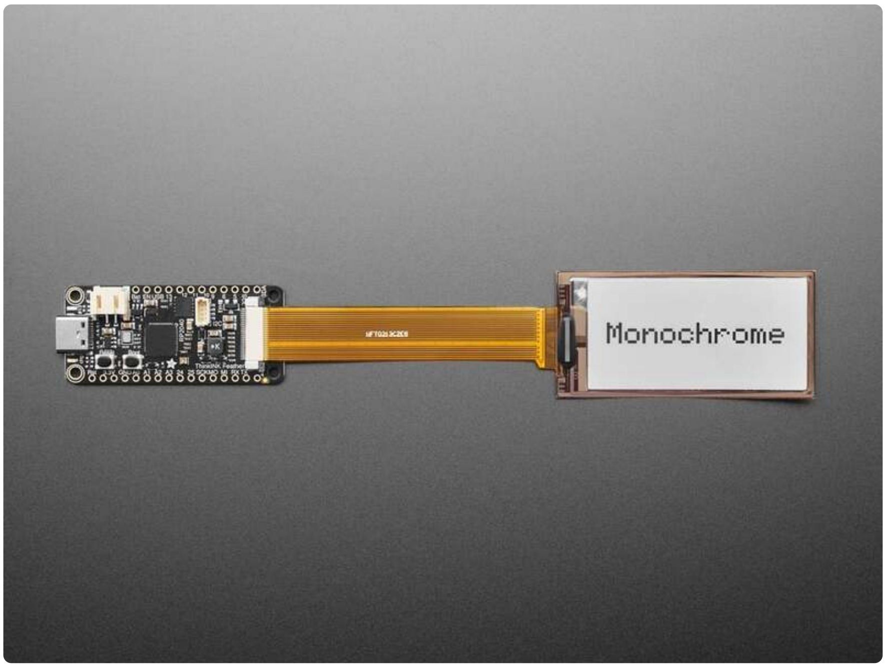
`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
2.13" 250x122 Monochrome
https://www.adafruit.com/product/6383
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.ssd1680 import Adafruit_SSD1680

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
```

```
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_SSD1680(122,
    250,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 3
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 5
shape_width = 30
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 20)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 20), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

2.13" Flexible 212x104 Monochrome eInk IL0373



This page will get you up and running with the 213" flexible 212x104 monochrome eInk display. This display uses the IL0373 driver.

2.13" 212x104 Flexible Monochrome / 4-Gray eInk ePaper Display
Woah, the cyber-future is here! Flexible E-Ink has been demo'd at high-tech events for years but now you can actually get your paws on it. This display is true E-Ink / E-Paper,...
https://www.adafruit.com/product/4243

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

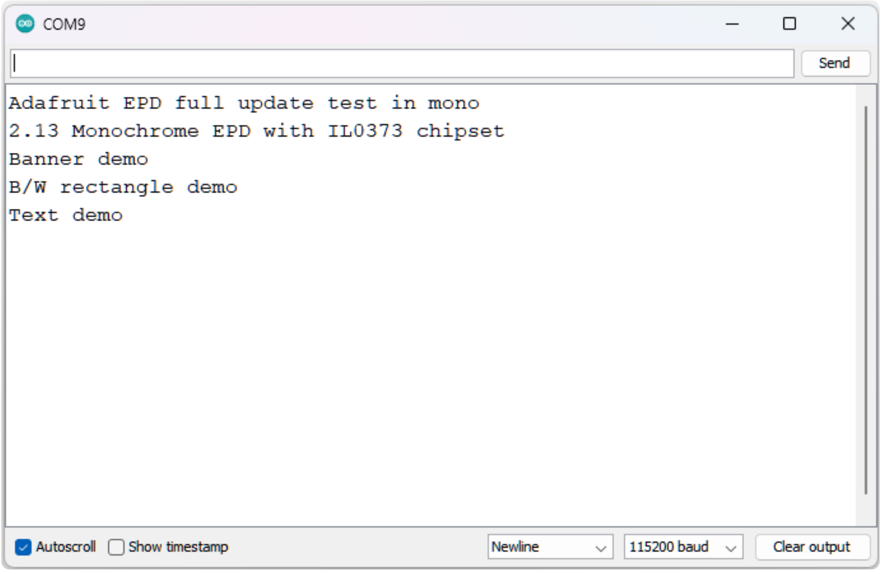
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.13" 212x104 Flexible Monochrome and IL0373 chipset
ThinkInk_213_Grayscale4_T5 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("2.13 Monochrome EPD with IL0373 chipset");
  display.begin(THINKINK_MONO);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



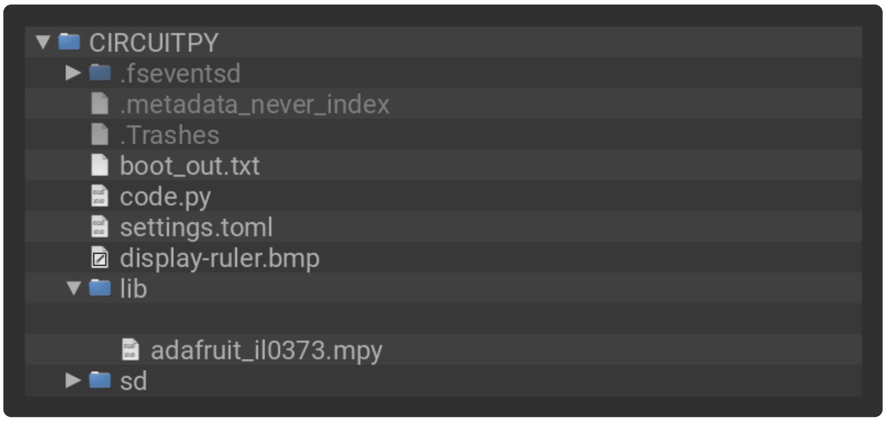
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_IL0373** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_il0373.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: MIT

"""Simple test for 2.13" 212x104 Flexible Monochrome eInk display."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_il0373

displayio.release_displays()

if "EPD_MOSI" in dir(board):  # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_il0373.IL0373(
    display_bus, width=212, height=104, rotation=90, busy_pin=epd_busy,
    black_bits_inverted=False, color_bits_inverted=True, swap_rams=True
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

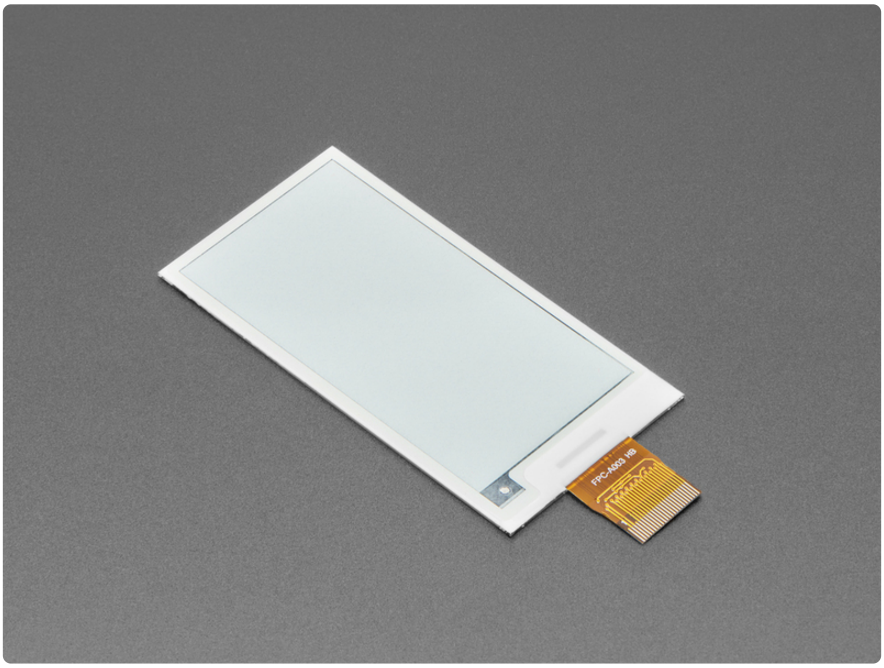
```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
2.13" 212x104 Flexible Monochrome
https://www.adafruit.com/product/4243
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.il0373 import Adafruit_IL0373_213_Flex_Mono

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK
```

```python
# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

# give them all to our driver
display = Adafruit_IL0373_213_Flex_Mono(104, 212,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 3
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 5
shape_width = 30
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 20)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 20), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)
```

```
display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

2.66" 296x152 Monochrome or Gray eInk SSD1680



This page will get you up and running with the 2.66" 296x152 monochrome eInk display. This display uses the SSD1680 driver.

[2.66" 296x152 Monochrome or Gray eInk / ePaper - Bare Display](https://www.adafruit.com/product/6392)
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6392

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

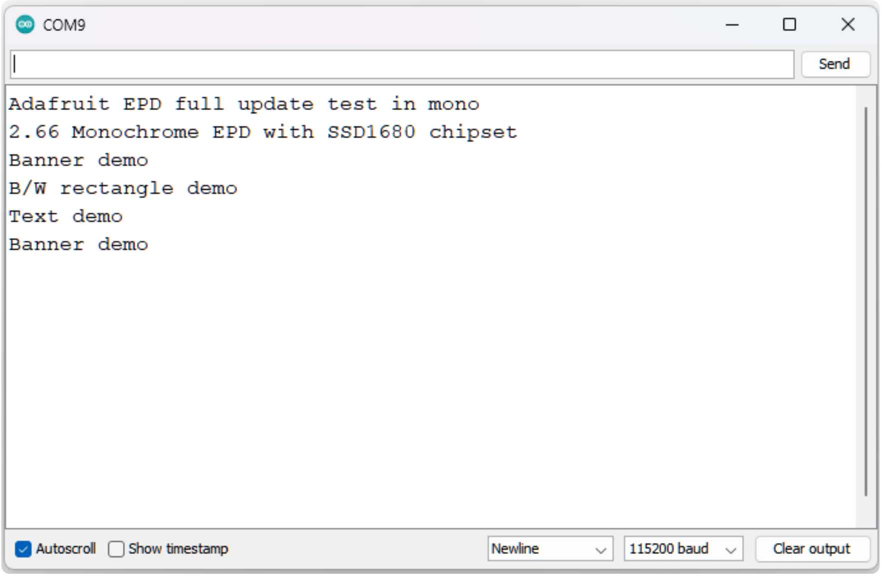
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.66" Monochrome display with 296x152 pixels and SSD1680 chipset
ThinkInk_266_Grayscale4_MFGN display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
                                      EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
```

```
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("2.66 Monochrome EPD with SSD1680 chipset");
  display.begin(THINKINK_MONO);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_SSD1680** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_ssd1680.mpy**

Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: MIT

"""Simple test script for 2.66" 296x152 monochrome display."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_ssd1680

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_ssd1680.SSD1680(
    display_bus,
    width=296,
    height=152,
    busy_pin=epd_busy,
    highlight_color=0xFF0000,
    rotation=270,
    colstart=0,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

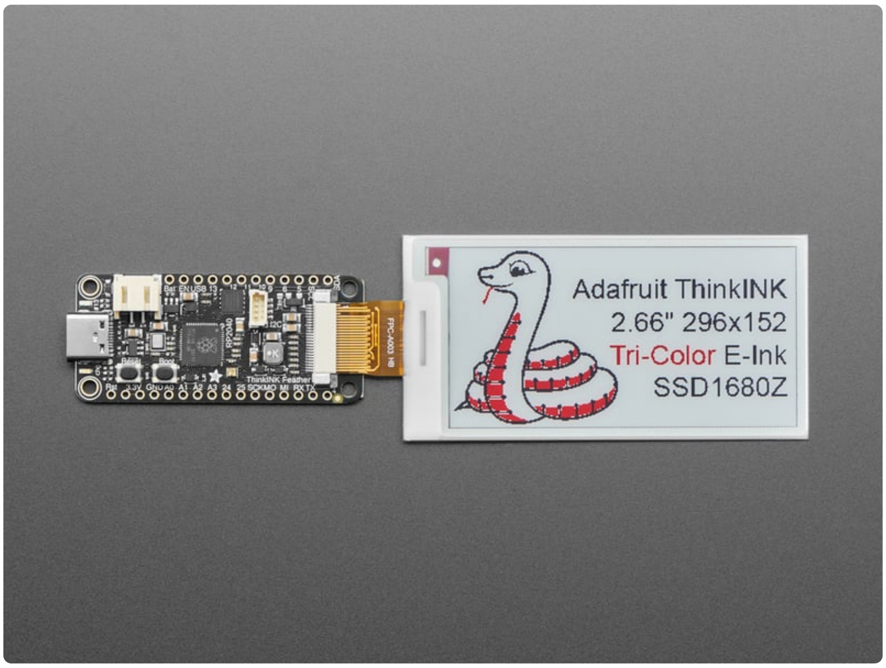
```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
2.13" 250x122 Monochrome
https://www.adafruit.com/product/6383
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.ssd1680 import Adafruit_SSD1680

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
```

```python
# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_SSD1680(152,
    296,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 3
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 5
shape_width = 30
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding
```

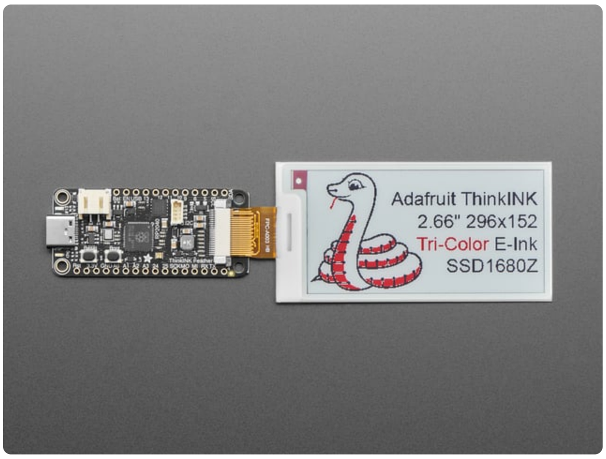
```
# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 20)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 20), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

2.66" 296x152 Tri-Color eInk SSD1680Z



This page will get you up and running with the 2.66" 296x152 Tri-Color eInk display. This display uses the SSD1680Z driver.



[2.66" 296x152 Tri-Color Red / Black / White EInk - Bare Display](https://www.adafruit.com/product/6393)
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6393

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
[Arduino Setup](https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup)

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

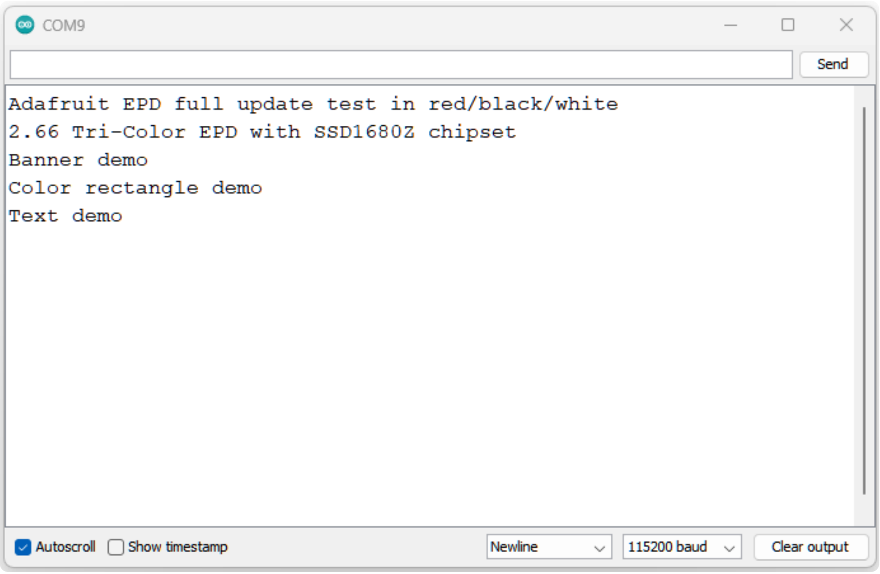
Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
```

```
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.66" Tricolor EPD with SSD1680Z chipset
ThinkInk_266_Tricolor_MFGNR display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS,
                                    EPD_BUSY, EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/black/white");
  Serial.println("2.66 Tri-Color EPD with SSD1680Z chipset");
  display.begin(THINKINK_TRICOLOR);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Tri");
  display.setTextColor(EPD_RED);
  display.print("Color");
  display.display();

  delay(15000);

  Serial.println("Color rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 3, 0, display.width() / 3,
                   display.height(), EPD_BLACK);
  display.fillRect((display.width() * 2) / 3, 0, display.width() / 3,
                   display.height(), EPD_RED);
  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }
```

```
  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  display.display();

  delay(15000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



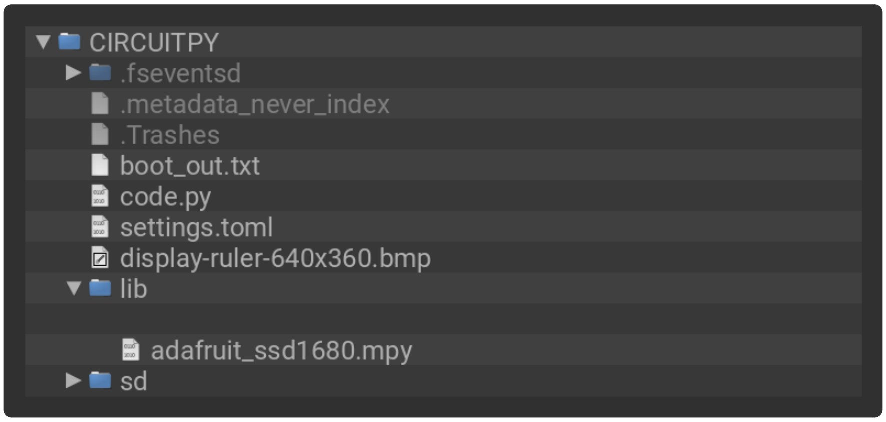
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_SSD1680** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_ssd1680.mpy**



Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: MIT

"""Simple test script for 2.66" 296x152 tri-color display."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_ssd1680

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_ssd1680.SSD1680(
    display_bus,
    width=296,
    height=152,
```

```
    busy_pin=epd_busy,
    highlight_color=0xFF0000,
    rotation=270,
    colstart=0,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
2.66" 296x152 Tri-Color SSD1680Z Chipset
https://www.adafruit.com/product/6393
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.ssd1680 import Adafruit_SSD1680Z

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
RED = (0xFF, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = RED

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_SSD1680Z(152,
    296,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 3
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

WHITE = (0xFF, 0xFF, 0xFF)
RED = (0xFF, 0x00, 0x00)
BLACK = (0x00, 0x00, 0x00)

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
```

```
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 5
shape_width = 30
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=RED, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=RED, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=RED,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=RED)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 20)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 20), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

3.52" 340x180 Quad-Color eInk JD79667

This page will get you up and running with the 3.52" 340x180 Quad-Color eInk display. This display uses the JD79667 driver.

display has a 'flipped' connector, with bottom contacts instead of top. You
d to attach the included extender and Flex A-B cable, which will flip the
acts over and make it work as expected.

[Adafruit 3.52" 340x180 Quad-Color eInk / ePaper - Bare Display](https://www.adafruit.com/product/6414)

What's better than our tri-color E-Paper displays? Quad-color, of course! This is a 3.52" 340x180...

https://www.adafruit.com/product/6414

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

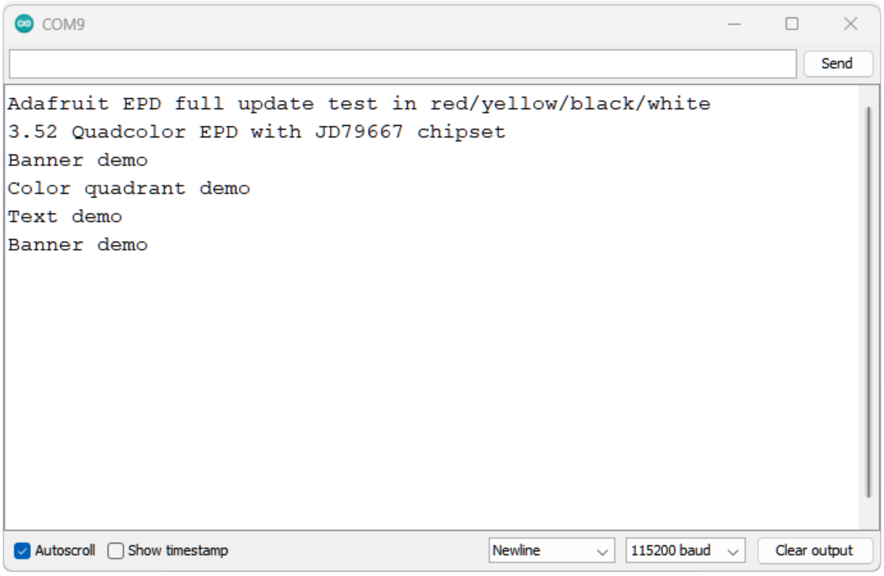
Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 2.13" Quadcolor EPD with JD79661 chipset
//ThinkInk_213_Quadcolor_AJHE5 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

// 3.52" Quadcolor EPD with JD79667 chipset
ThinkInk_352_Quadcolor_AJHE5 display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);
```

```
void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/yellow/black/white");
  Serial.println("3.52 Quadcolor EPD with JD79667 chipset");
  display.begin(THINKINK_QUADCOLOR);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  String text = "QuadColor";
  uint16_t colors[] = {EPD_BLACK, EPD_RED, EPD_YELLOW};

  for (int i = 0; i < text.length(); i++) {
    // Change color for every character (0: BLACK, 1: RED, 2: YELLOW, 3: BLACK, 
etc.)
    display.setTextColor(colors[i % 3]);
    display.print(text.charAt(i));
  }
  display.display();

  delay(15000);

  Serial.println("Color quadrant demo");
  display.clearBuffer();
  // Top-left quadrant - EPD_BLACK
  display.fillRect(0, 0, display.width() / 2, display.height() / 2, EPD_BLACK);
  // Top-right quadrant - EPD_RED
  display.fillRect(display.width() / 2, 0, display.width() / 2, display.height() / 
2, EPD_RED);
  // Bottom-left quadrant - EPD_YELLOW
  display.fillRect(0, display.height() / 2, display.width() / 2, display.height() / 
2, EPD_YELLOW);
  // Bottom-right quadrant - assume you have a 4th color like EPD_WHITE or another 
color
  display.fillRect(display.width() / 2, display.height() / 2, display.width() / 2, 
display.height() / 2, EPD_WHITE);

  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }
  for (int16_t i = 0; i < display.height(); i += 4) {
```

```
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  for (int16_t i = 0; i < display.width(); i += 4) {
     display.drawLine(display.width()/2, display.height()-1, i, 0,
                      EPD_YELLOW);
  }

  display.display();

  delay(15000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



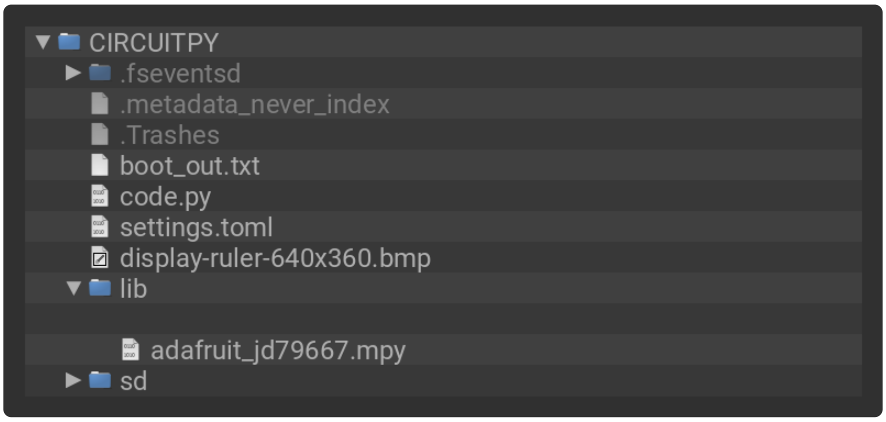
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_JD79667** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_jd79667.mpy**



Example Code

```
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: MIT

"""Simple test for 3.52" 340x180 quad-color eInk display."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_jd79667

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_jd79667.JD79667(
    display_bus,
    width=384,
    height=184,
```

```
    busy_pin=epd_busy,
    rotation=270,
    colstart=0,
    highlight_color=0xFFFF00,
    highlight_color2=0xFF0000,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-640x360.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

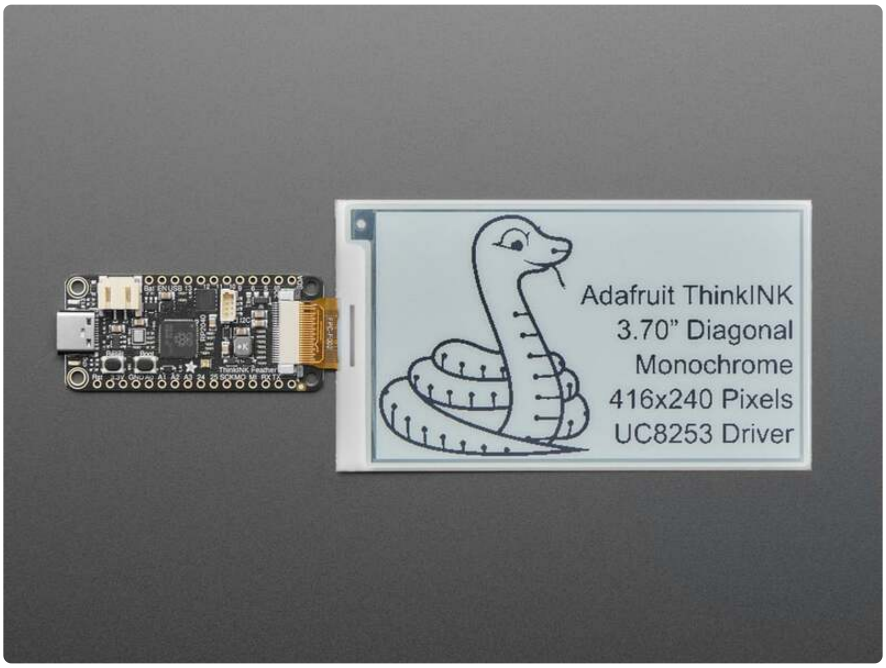
`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
3.52" Quad Color 384x180 display
https://www.adafruit.com/product/6414
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.jd79667 import Adafruit_JD79667

# RGB colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 255, 0)

# Create the SPI device and pins
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None  # No SRAM
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_JD79667(
    180,
    384,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 3

# Get display dimensions
width = display.width
height = display.height
image = Image.new("RGB", (width, height))
# Get drawing object
draw = ImageDraw.Draw(image)

# Fill background with white
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw a border
draw.rectangle((0, 0, width - 1, height - 1), outline=BLACK, fill=WHITE)

# Draw some shapes to demonstrate all four colors
padding = 10
shape_width = 30
top = padding + 10
```

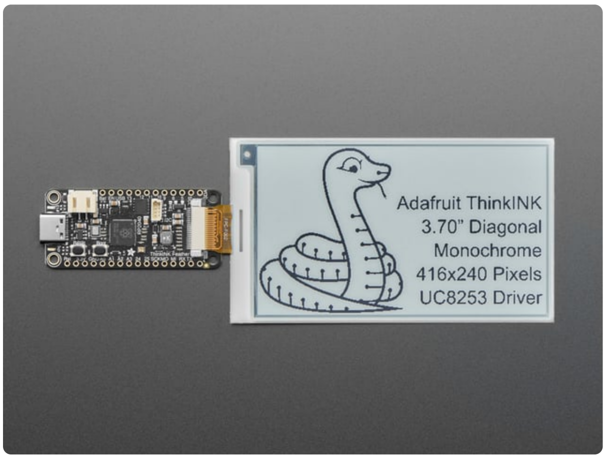
```
bottom = height - padding - 10

# Starting x position
x = padding + 10

# Draw an ellipse filled with red
draw.ellipse((x, top, x + shape_width, bottom), outline=RED, fill=None)
x += shape_width + padding

# Draw a rectangle filled with yellow
draw.rectangle((x, top, x + shape_width, bottom), outline=YELLOW, fill=BLACK)
x += shape_width + padding

# Draw a triangle filled with red
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=YELLOW,
)
x += shape_width + padding

# Draw an X with red and yellow lines
draw.line((x, bottom, x + shape_width, top), fill=RED)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding

# Add some text
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 28)

# Draw text in different colors
draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 28), "World!", font=font, fill=RED)
draw.text((x, top + (28*2)), "Quad", font=font, fill=YELLOW)
draw.text((x, top + (28*3)), "Color!", font=font, fill=BLACK)

display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

3.7" 416x240 Monochrome eInk UC8253



This page will get you up and running with the 3.7" 416x240 monochrome eInk display. This display uses the UC8253 driver.



3.7" 416x240 Monochrome Black/White eInk / ePaper - Bare Display

E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...

https://www.adafruit.com/product/6395

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

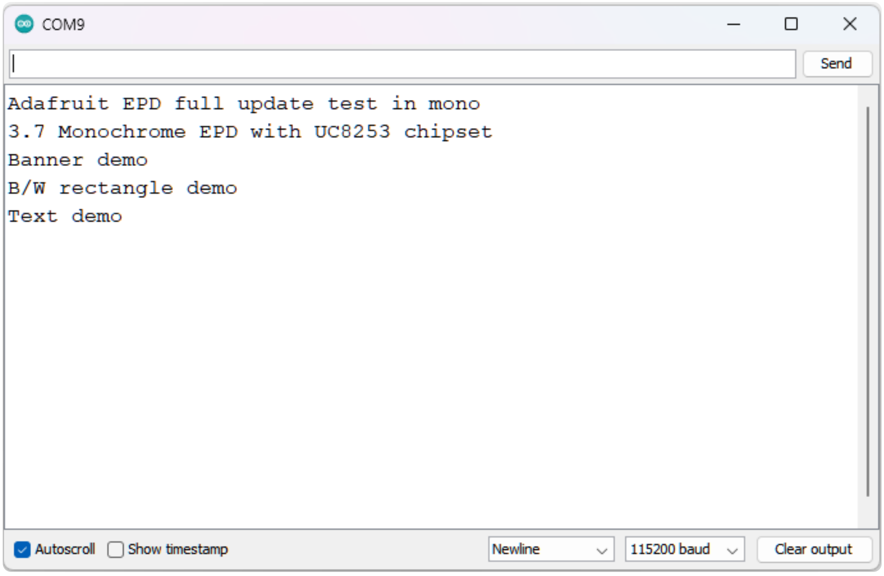
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 3.7" Monochrome Display with 420x240 pixels and UC8253 chipset
ThinkInk_370_Mono_BAAMFGN display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS,
                                  EPD_BUSY, EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("3.7 Monochrome EPD with UC8253 chipset");
  display.begin(THINKINK_MONO);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_UC8253** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_uc8253.mpy**

Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 3.7" 240x416 monochrome display
(ThinkInk_370_Mono_BAAMFGN)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_uc8253

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_uc8253.UC8253(
    display_bus,
    width=416,
    height=240,
    busy_pin=epd_busy,
    rotation=270,
    vcom_cdi=0x97,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

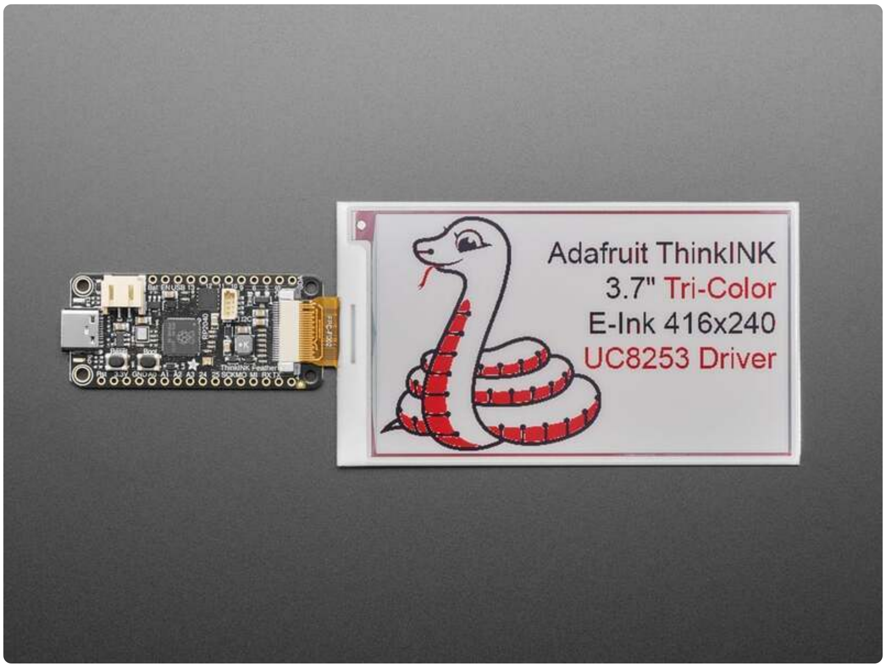
```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
3.7" Monochrome 416x240 display
https://www.adafruit.com/product/6395
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.uc8253 import Adafruit_UC8253_Mono

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
```

```python
# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_UC8253_Mono(240, 416,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy
)

display.set_black_buffer(1, True)
display.set_color_buffer(1, True)
display.rotation = 3

width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 15
shape_width = 45
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding
```

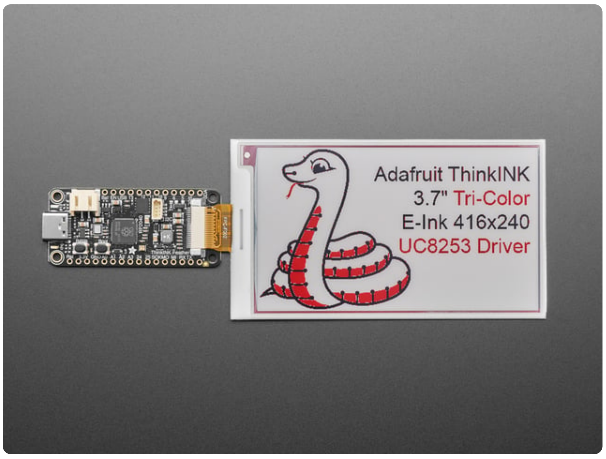
```
# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 30)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 30), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

3.7" 416x240 Tri-Color eInk UC8253



This page will get you up and running with the 3.7" 416x240 Tri-Color eInk display. This display uses the UC8253 driver.



[3.7" 416x240 Tri-Color Red / Black / White eInk - Bare Display](https://www.adafruit.com/product/6394)

E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...

https://www.adafruit.com/product/6394

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course

By Liz Clark

[Arduino Setup](https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup)

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

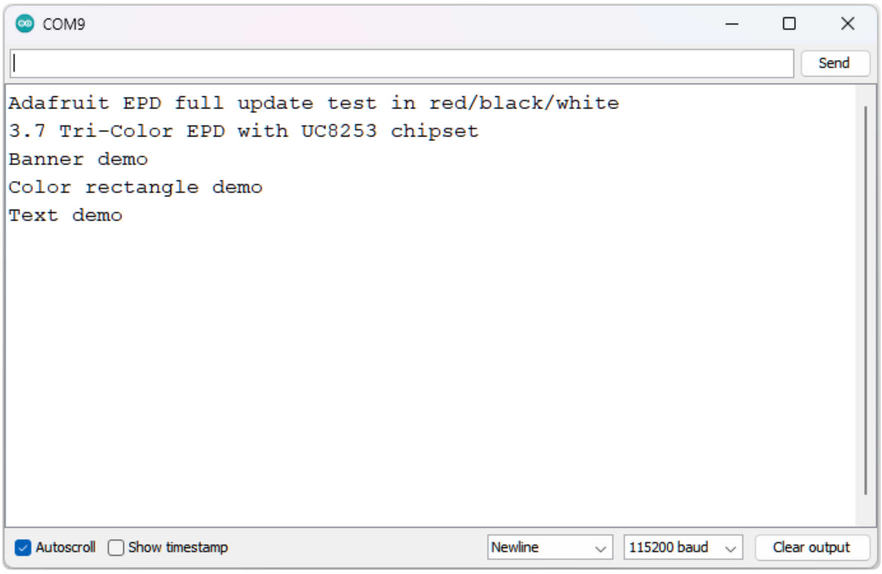
Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
```

```
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 3.7" Tricolor Display with 417x240 pixels and UC8253 chipset
ThinkInk_370_Tricolor_BABMFGNR display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS,
 EPD_BUSY, EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/black/white");
  Serial.println("3.7 Tri-Color EPD with UC8253 chipset");
  display.begin(THINKINK_TRICOLOR);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Tri");
  display.setTextColor(EPD_RED);
  display.print("Color");
  display.display();

  delay(15000);

  Serial.println("Color rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 3, 0, display.width() / 3,
                   display.height(), EPD_BLACK);
  display.fillRect((display.width() * 2) / 3, 0, display.width() / 3,
                   display.height(), EPD_RED);
  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }
```

```
  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  display.display();

  delay(15000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



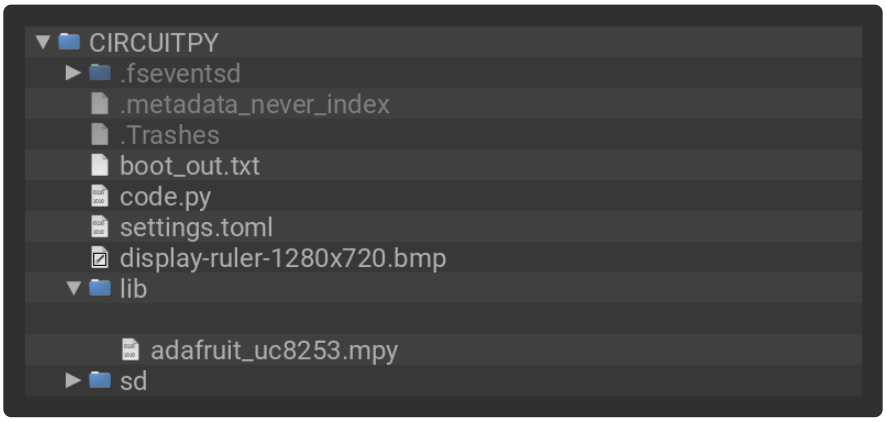
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_UC8253** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_uc8253.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 3.7" 240x416 tricolor display
(ThinkInk_370_Tricolor_BABMFGNR)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_uc8253

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_uc8253.UC8253(
    display_bus, width=416, height=240, busy_pin=epd_busy, rotation=270,
```

```
highlight_color=0xFF0000
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
# Always refresh a little longer. It's not a problem to refresh
# a few seconds more, but it's terrible to refresh too early
# (the display will throw an exception when if the refresh
# is too soon)
print("waited correct time")


# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

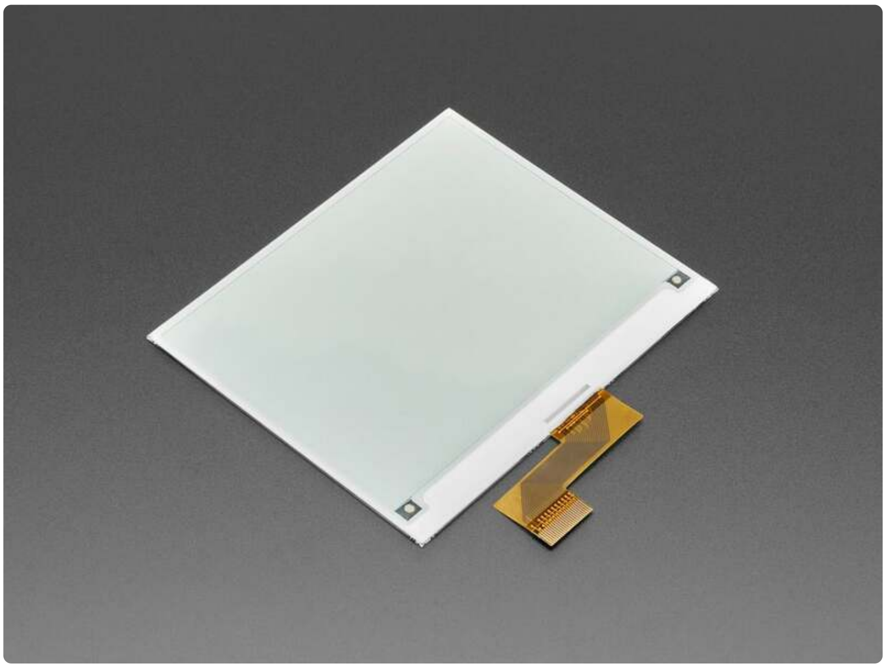
`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
3.7" Tri-Color 416x240 display
https://www.adafruit.com/product/6394
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.uc8253 import Adafruit_UC8253_Tricolor

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
RED = (0xFF, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_UC8253_Tricolor(240, 416,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy
)

display.rotation = 3

width = display.width
height = display.height
image = Image.new("RGB", (width, height))

WHITE = (0xFF, 0xFF, 0xFF)
RED = (0xFF, 0x00, 0x00)
BLACK = (0x00, 0x00, 0x00)

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
```

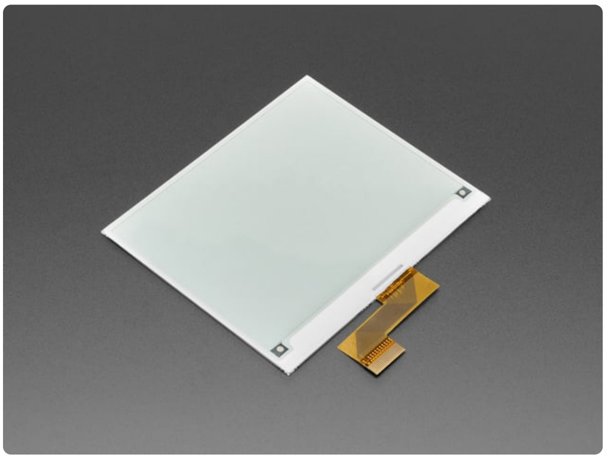
```
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 15
shape_width = 45
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=RED, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=RED, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=RED,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=RED)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 30)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 30), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

4.2" 300x400 Monochrome or 4-Gray eInk SSD1683



This page will get you up and running with the 4.2" 300x400 monochrome eInk display. This display uses the SSD1683 driver.



4.2" 300x400 Monochrome or 4-Gray eInk / ePaper - Bare Display
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6381

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

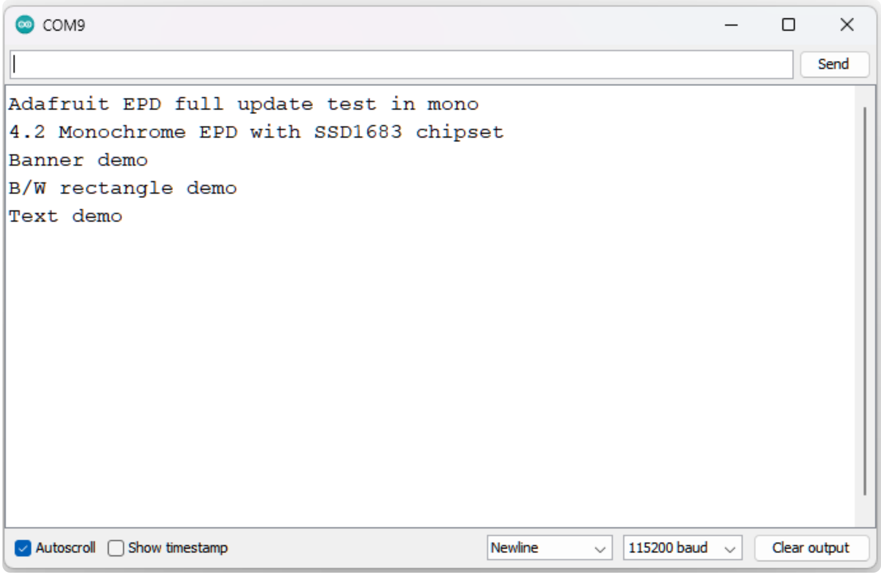
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 4.2" Grayscale/Monochrome displays with 400x300 pixels and SSD1683 chipset
ThinkInk_420_Grayscale4_MFGN display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("4.2 Monochrome EPD with SSD1683 chipset");
  display.begin(THINKINK_MONO);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



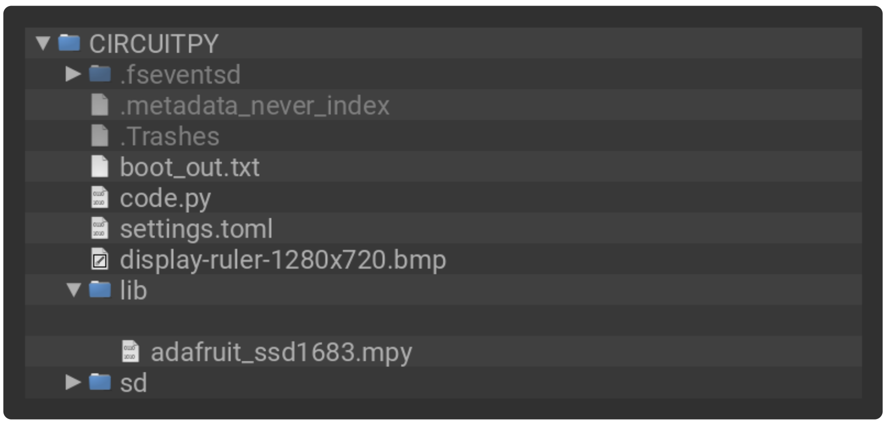
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_SSD1683** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_ssd1683.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 4.2" 400x300 monochrome display
(ThinkInk_420_Grayscale4_MFGN)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_ssd1683

displayio.release_displays()

if "EPD_MOSI" in dir(board):  # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_ssd1683.SSD1683(display_bus, width=400, height=300,
busy_pin=epd_busy)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
# Always refresh a little longer. It's not a problem to refresh
# a few seconds more, but it's terrible to refresh too early
# (the display will throw an exception when if the refresh
# is too soon)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

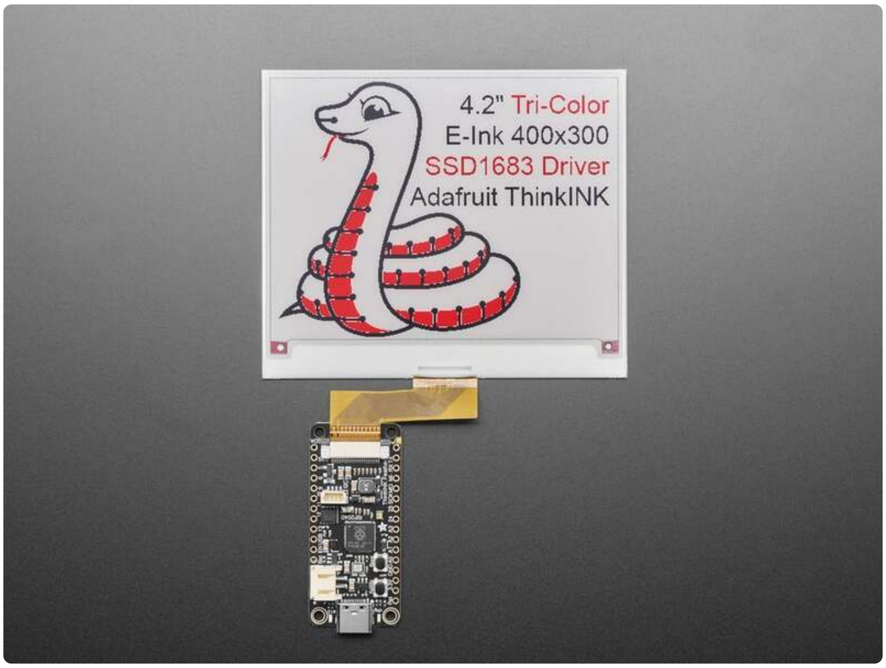
```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
4.2" 300x400 Monochrome
https://www.adafruit.com/product/6381
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.ssd1683 import Adafruit_SSD1683

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0xFF, 0x00, 0x00)
```

```python
# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)


display = Adafruit_SSD1683(400, 300,        # 4.2" 300x400 monochrome display
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 0
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 15
shape_width = 50
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 30)
```

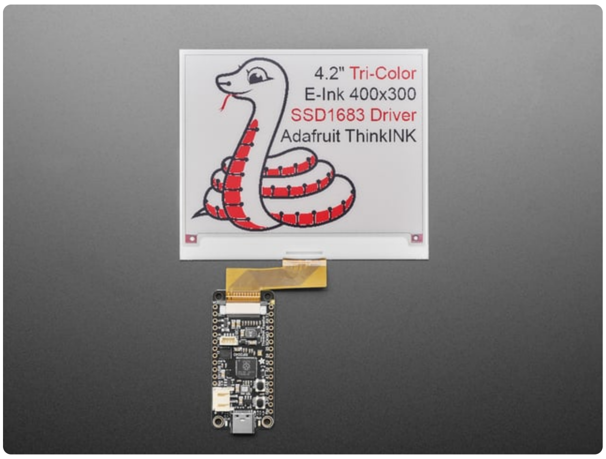
```
draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 30), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

4.2" 300x400 Tri-Color eInk SSD1683



This page will get you up and running with the 4.2" 400x300 Tri-Color eInk display. This display uses the SSD1683 driver.



[4.2" 300x400 Tri-Color eInk / ePaper - Bare Display](https://www.adafruit.com/product/6382)
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6382

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

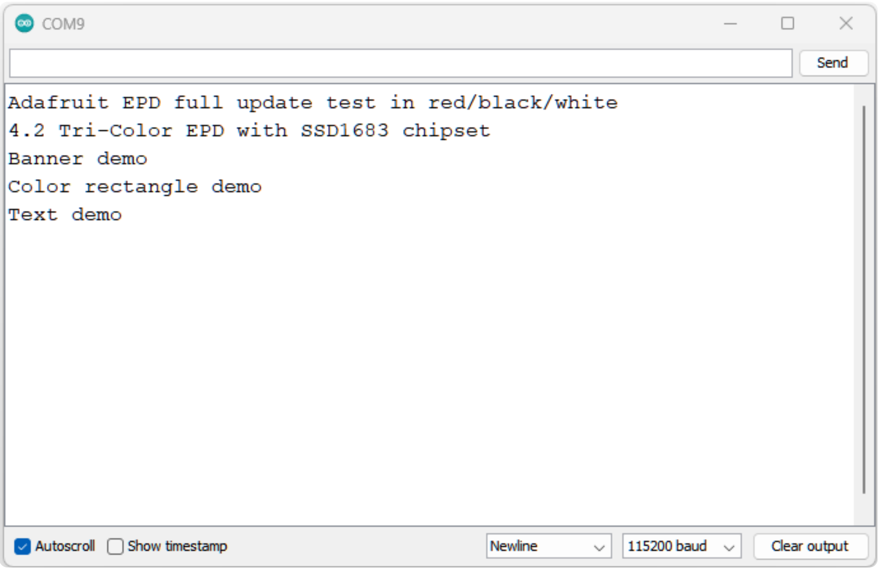
Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                               // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
```

```
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 4.2" Tricolor EPD with SSD1683 chipset
ThinkInk_420_Tricolor_MFGNR display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS,
                                    EPD_BUSY, EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/black/white");
  Serial.println("4.2 Tri-Color EPD with SSD1683 chipset");
  display.begin(THINKINK_TRICOLOR);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Tri");
  display.setTextColor(EPD_RED);
  display.print("Color");
  display.display();

  delay(15000);

  Serial.println("Color rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 3, 0, display.width() / 3,
                   display.height(), EPD_BLACK);
  display.fillRect((display.width() * 2) / 3, 0, display.width() / 3,
                   display.height(), EPD_RED);
  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }
```

```
  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  display.display();

  delay(15000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



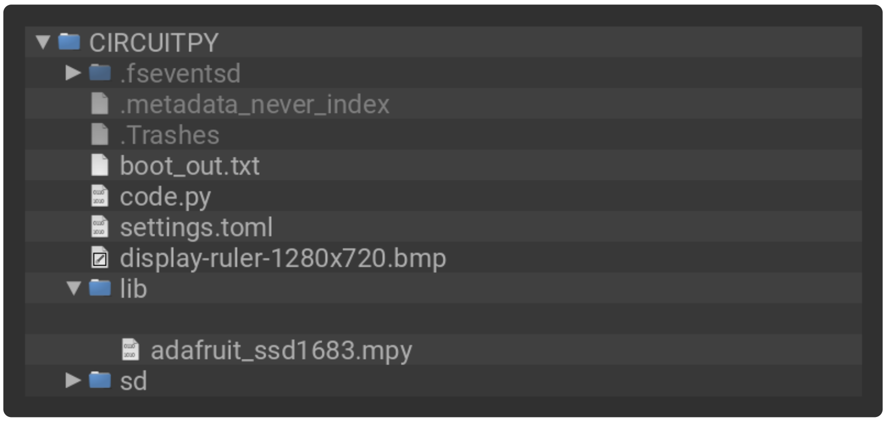
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_SSD1683** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_ssd1683.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 4.2" 400x300 tricolor display
(ThinkInk_420_Tricolor_MFGNR)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_ssd1683

displayio.release_displays()

if "EPD_MOSI" in dir(board):  # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_ssd1683.SSD1683(
    display_bus, width=400, height=300, highlight_color=0xFF0000, busy_pin=epd_busy
```

```
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
# Always refresh a little longer. It's not a problem to refresh
# a few seconds more, but it's terrible to refresh too early
# (the display will throw an exception when if the refresh
# is too soon)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

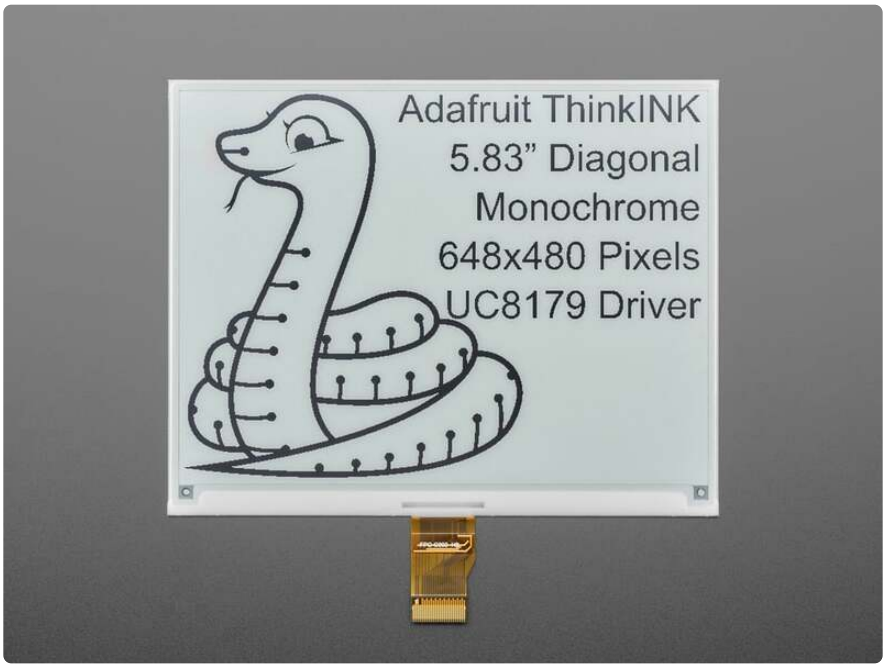
`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
4.2" 300x400 Tri-Color
https://www.adafruit.com/product/6382
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.ssd1683 import Adafruit_SSD1683

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
RED = (0xFF, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

# give them all to our driver
display = Adafruit_SSD1683(400, 300,        # 4.2" 300x400 Tri-Color display
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.rotation = 0
width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
```

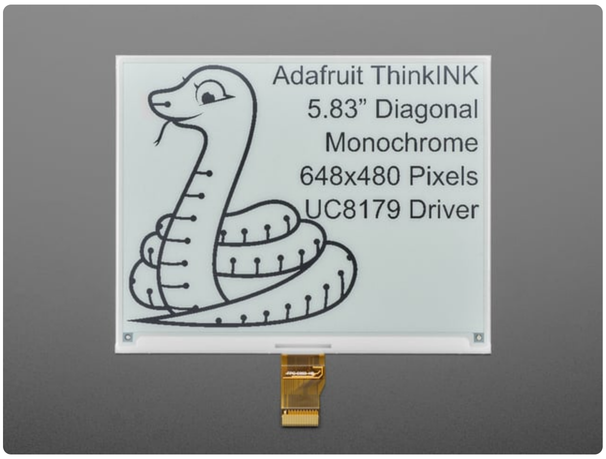
```
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 15
shape_width = 50
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=RED, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=RED, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=RED,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=RED)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 30)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 30), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

5.83" 648x480 Monochrome eInk UC8179



This page will get you up and running with the 5.83" 648x480 monochrome eInk display. This display uses the UC8179 driver.



5.83" 648x480 Monochrome Black / White eInk / ePaper

E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...

https://www.adafruit.com/product/6397

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

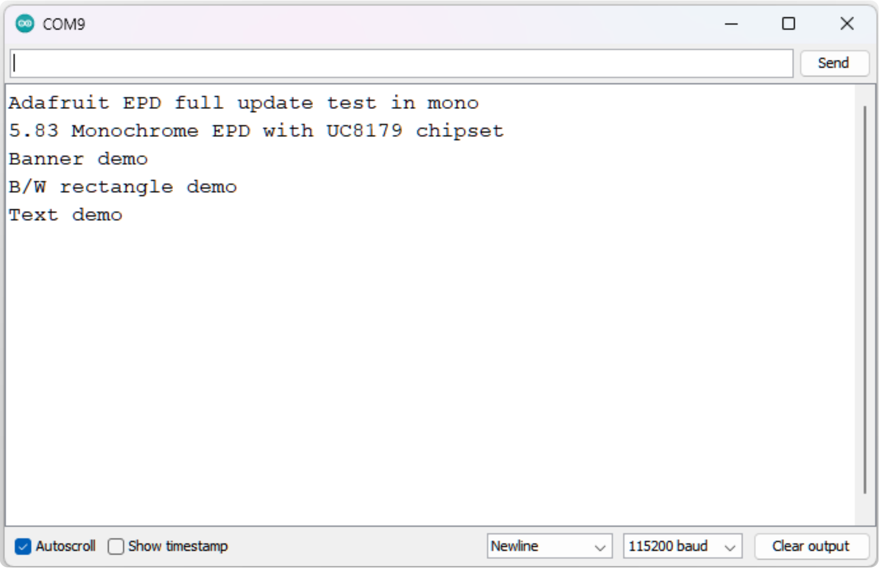
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 5.83" Monochrome displays with 648 x 480 pixels and UC8179 chipset
ThinkInk_583_Mono_AAAMFGN display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("5.83 Monochrome EPD with UC8179 chipset");
  display.begin(THINKINK_MONO);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



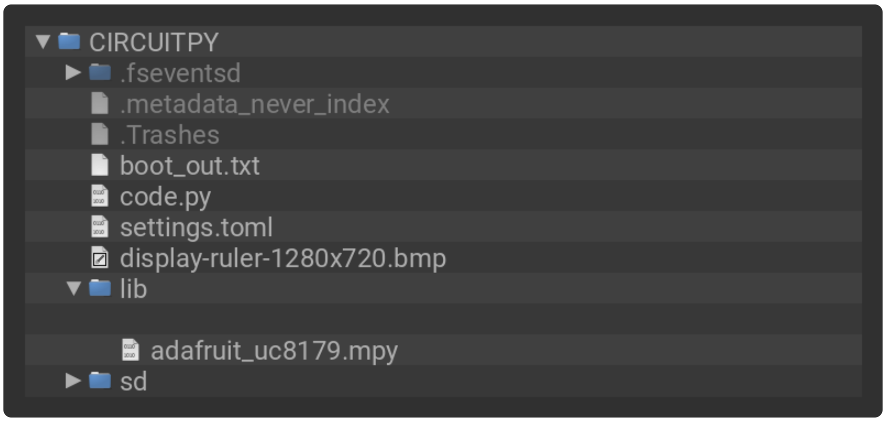
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_UC8179** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_uc8179.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 5.83" 648x480 monochrome display
(ThinkInk_583_Mono_AAAMFGN)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_uc8179

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_uc8179.UC8179(
    display_bus,
    width=648,
    height=480,
    busy_pin=epd_busy,
    rotation=180,
    black_bits_inverted=True,
    colstart=0,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

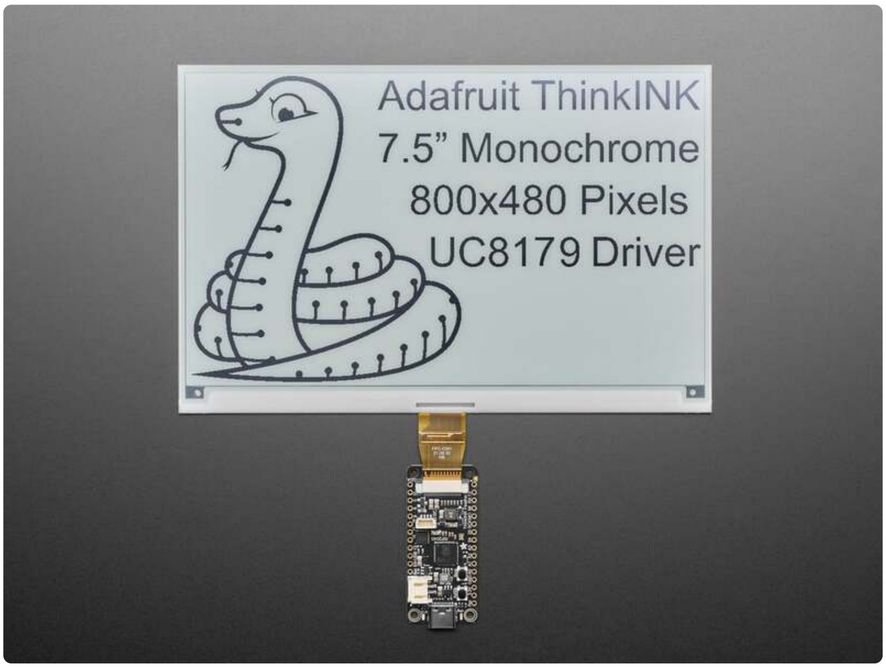
```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
5.83" mono 648x480 display
https://www.adafruit.com/product/6397
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.uc8179 import Adafruit_UC8179

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0xFF, 0x00, 0x00)
```

```python
# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

# give them all to our driver
display = Adafruit_UC8179(648, 480,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.set_black_buffer(1, False)
display.set_color_buffer(1, False)
display.rotation = 0

width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 25
shape_width = 80
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding
```

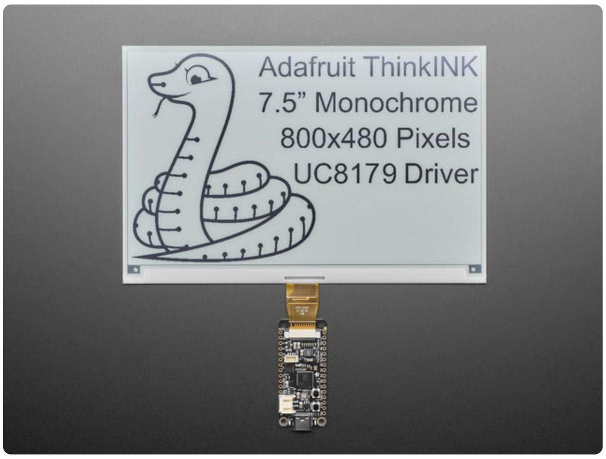
```
# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 40)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 40), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

7.5" 800x480 Monochrome eInk UC8179



This page will get you up and running with the 7.5" 800x480 monochrome eInk display. This display uses the UC8179 driver.



[7.5" 800x480 Monochrome eInk / ePaper - Bare Display](https://www.adafruit.com/product/6396)
E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...
https://www.adafruit.com/product/6396

Arduino

First, follow the steps on the [Arduino Setup page](https://adafru.it/1atw) (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
[Arduino Setup](https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup)

https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

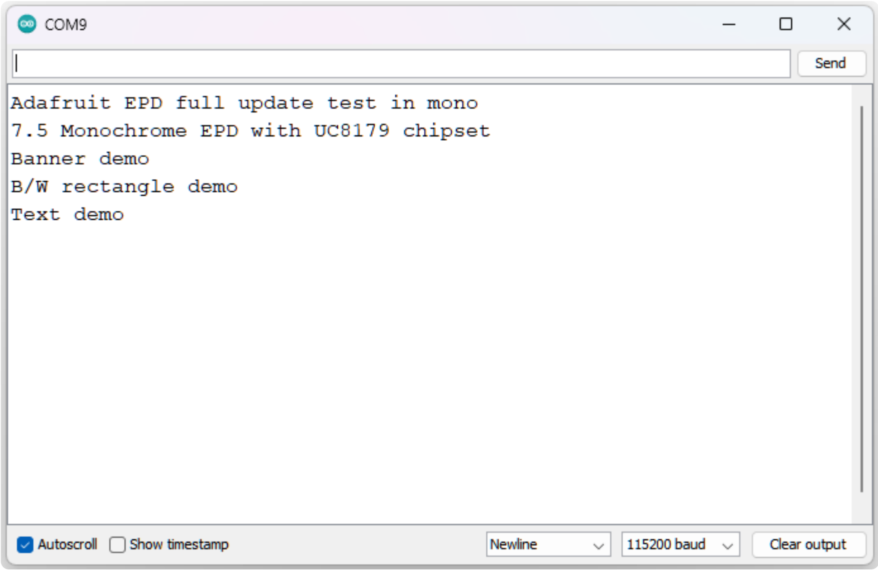
Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                               // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
```

```
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 7.5" Monochrome displays with 800 x 480 pixels and UC8179 chipset
ThinkInk_750_Mono_AAAMFGN display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in mono");
  Serial.println("7.5 Monochrome EPD with UC8179 chipset");
  display.begin(THINKINK_MONO);
}

void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 180) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Monochrome");
  display.display();

  delay(2000);

  Serial.println("B/W rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 2, 0, display.width() / 2,
                   display.height(), EPD_BLACK);
  display.display();

  delay(2000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(2000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_BLACK);
  }
```

```
  display.display();

  delay(2000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_UC8179** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_uc8179.mpy**

Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 7.5" 800x480 monochrome display"""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_uc8179

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_uc8179.UC8179(
    display_bus,
    width=800,
    height=480,
    busy_pin=epd_busy,
    rotation=180,
    black_bits_inverted=True,
    colstart=0,
)

g = displayio.Group()
```

```
pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the [Python Setup page](https://adafru.it/1atx) (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

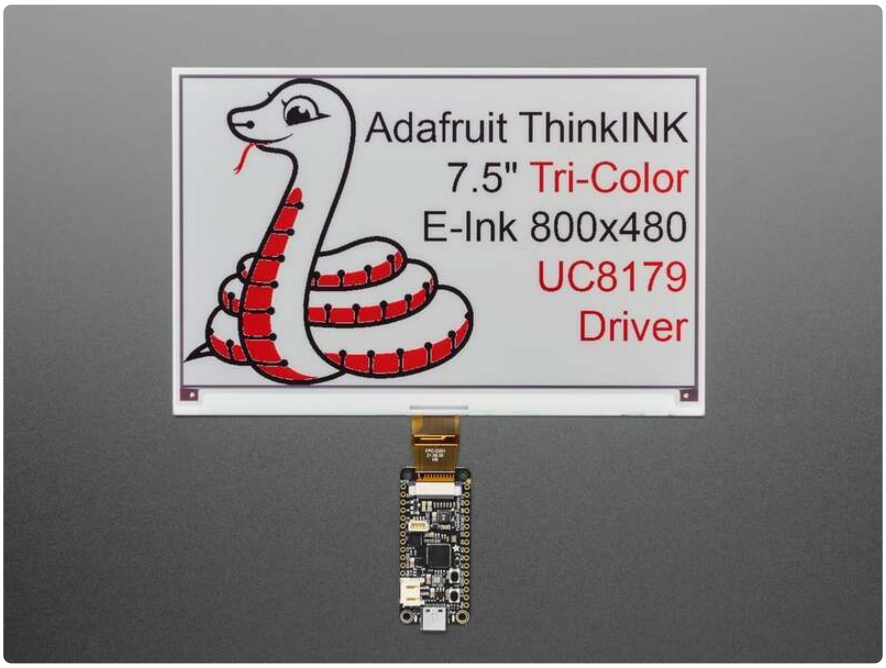
`sudo -E env PATH=$PATH python3 code.py`

```python
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
7.5" mono 800x480 display
https://www.adafruit.com/product/6396
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.uc8179 import Adafruit_UC8179

# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0xFF, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

# give them all to our driver
display = Adafruit_UC8179(800, 480,
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
)

display.set_black_buffer(1, False)
display.set_color_buffer(1, False)
display.rotation = 0

width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 25
shape_width = 100
```

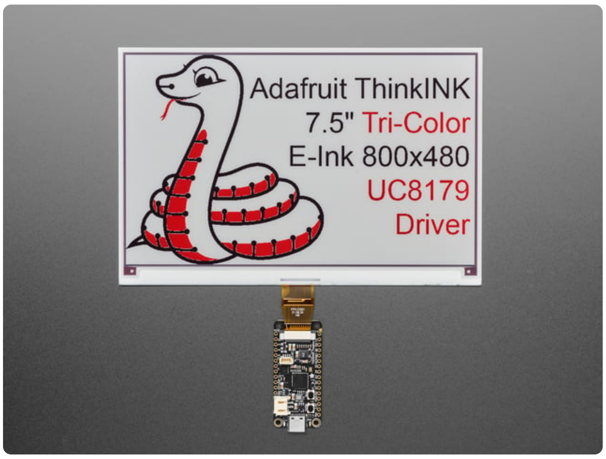
```
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=BLACK, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=BLACK, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=BLACK,
)
x += shape_width + padding
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=BLACK)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 50)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 50), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.

7.5" 800x480 Tri-Color eInk UC8179



This page will get you up and running with the 7.5" 800x480 Tri-Color eInk display. This display uses the UC8179 driver.



7.5" 800x480 Tri-Color eInk / ePaper - Bare Display

E-Ink/E-Paper displays make for great low-power displays that are daylight visible and keep their image even when...

https://www.adafruit.com/product/6415

Arduino

First, follow the steps on the Arduino Setup page (https://adafru.it/1atw) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Arduino Setup

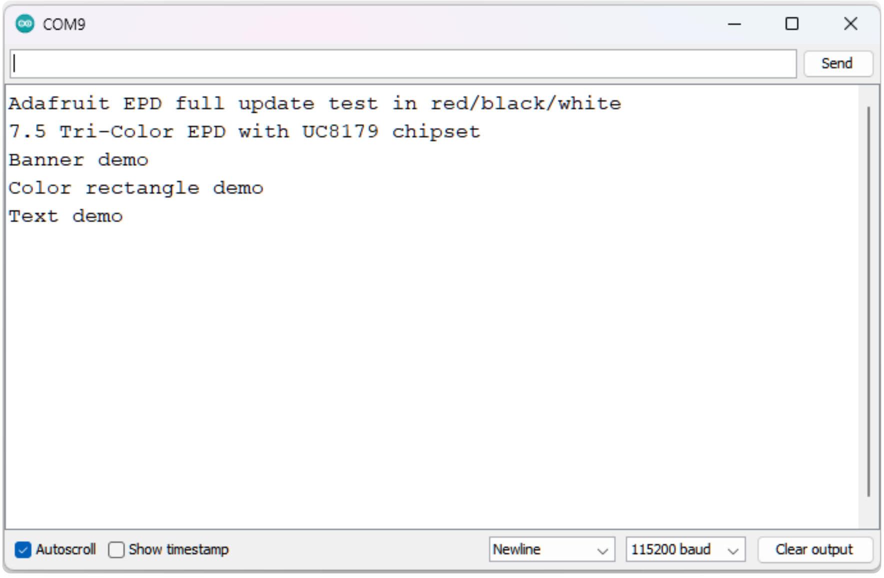
https://learn.adafruit.com/bare-e-ink-displays-crash-course/arduino-setup

Then, you can run this example code on your board:

```
// SPDX-FileCopyrightText: 2025 Ladyada for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/***************************************************
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "Adafruit_ThinkInk.h"

#ifdef ARDUINO_ADAFRUIT_FEATHER_RP2040_THINKINK // detects if compiling for
                                                // Feather RP2040 ThinkInk
#define EPD_DC PIN_EPD_DC       // ThinkInk 24-pin connector DC
#define EPD_CS PIN_EPD_CS       // ThinkInk 24-pin connector CS
#define EPD_BUSY PIN_EPD_BUSY   // ThinkInk 24-pin connector Busy
#define SRAM_CS -1              // use onboard RAM
#define EPD_RESET PIN_EPD_RESET // ThinkInk 24-pin connector Reset
#define EPD_SPI &SPI1           // secondary SPI for ThinkInk
#else
#define EPD_DC 10
#define EPD_CS 9
#define EPD_BUSY 7 // can set to -1 to not use a pin (will wait a fixed delay)
#define SRAM_CS 6
#define EPD_RESET 8  // can set to -1 and share with microcontroller Reset!
#define EPD_SPI &SPI // primary SPI
#endif

// 7.5" Tricolor displays with 800x480 pixels and UC8179 chipset
ThinkInk_750_Tricolor_AABMFGNR display(EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY,
EPD_SPI);

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }
  Serial.println("Adafruit EPD full update test in red/black/white");
  Serial.println("7.5 Tri-Color EPD with UC8179 chipset");
  display.begin(THINKINK_TRICOLOR);
}
```

```
void loop() {
  Serial.println("Banner demo");
  display.clearBuffer();
  display.setTextSize(3);
  display.setCursor((display.width() - 144) / 2, (display.height() - 24) / 2);
  display.setTextColor(EPD_BLACK);
  display.print("Tri");
  display.setTextColor(EPD_RED);
  display.print("Color");
  display.display();

  delay(15000);

  Serial.println("Color rectangle demo");
  display.clearBuffer();
  display.fillRect(display.width() / 3, 0, display.width() / 3,
                   display.height(), EPD_BLACK);
  display.fillRect((display.width() * 2) / 3, 0, display.width() / 3,
                   display.height(), EPD_RED);
  display.display();

  delay(15000);

  Serial.println("Text demo");
  // large block of text
  display.clearBuffer();
  display.setTextSize(1);
  testdrawtext(
      "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur "
      "adipiscing ante sed nibh tincidunt feugiat. Maecenas enim massa, "
      "fringilla sed malesuada et, malesuada sit amet turpis. Sed porttitor "
      "neque ut ante pretium vitae malesuada nunc bibendum. Nullam aliquet "
      "ultrices massa eu hendrerit. Ut sed nisi lorem. In vestibulum purus a "
      "tortor imperdiet posuere. ",
      EPD_BLACK);
  display.display();

  delay(15000);

  display.clearBuffer();
  for (int16_t i = 0; i < display.width(); i += 4) {
    display.drawLine(0, 0, i, display.height() - 1, EPD_BLACK);
  }

  for (int16_t i = 0; i < display.height(); i += 4) {
    display.drawLine(display.width() - 1, 0, 0, i, EPD_RED);
  }
  display.display();

  delay(15000);
}

void testdrawtext(const char *text, uint16_t color) {
  display.setCursor(0, 0);
  display.setTextColor(color);
  display.setTextWrap(true);
  display.print(text);
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the display recognized over SPI. Then, as different demos are sent to the display, you'll see the demo name print to the Serial Monitor.



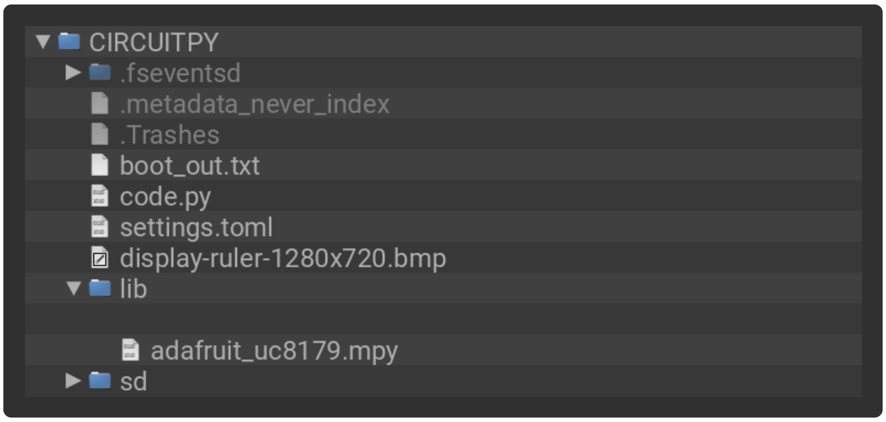
CircuitPython

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_UC8179** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_uc8179.mpy**



Example Code

```
# SPDX-FileCopyrightText: 2025 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Melissa LeBlanc-Williams for Adafruit
Industries
#
# SPDX-License-Identifier: Unlicense

"""Simple test script for 7.5" 800x480 tricolor display
(ThinkInk_750_Tricolor_AABMFGNR)."""

import time

import board
import busio
import displayio
from fourwire import FourWire

import adafruit_uc8179

displayio.release_displays()

if "EPD_MOSI" in dir(board): # Feather RP2040 ThinkInk
    spi = busio.SPI(board.EPD_SCK, MOSI=board.EPD_MOSI, MISO=None)
    epd_cs = board.EPD_CS
    epd_dc = board.EPD_DC
    epd_reset = board.EPD_RESET
    epd_busy = board.EPD_BUSY
else:
    spi = board.SPI()  # Uses SCK and MOSI
    epd_cs = board.D9
    epd_dc = board.D10
    epd_reset = board.D8  # Set to None for FeatherWing
    epd_busy = board.D7  # Set to None for FeatherWing

display_bus = FourWire(spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset,
baudrate=1000000)
time.sleep(1)

display = adafruit_uc8179.UC8179(
    display_bus,
    width=800,
    height=480,
    busy_pin=epd_busy,
    rotation=180,
    black_bits_inverted=True,
    highlight_color=0xFF0000,
    colstart=0,
)

g = displayio.Group()

pic = displayio.OnDiskBitmap("/display-ruler-1280x720.bmp")
t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
g.append(t)

display.root_group = g

display.refresh()

print("refreshed")

time.sleep(display.time_to_refresh + 5)
print("waited correct time")

# Keep the display the same
```

```
while True:
    time.sleep(10)
```

Once everything is saved to the **CIRCUITPY** drive, the code will begin running. Your display will show the display ruler bitmap.

Python

First, follow the steps on the Python Setup page (https://adafru.it/1atx) to install the necessary libraries and dependencies for the example code.

Bare E-Ink Displays Crash Course
By Liz Clark
Python Setup

https://learn.adafruit.com/bare-e-ink-displays-crash-course/python-setup

Pillow Graphics Demo

The great part about using a display on a Raspberry Pi is that you can use Pillow graphics alongside the CircuitPython driver.

The following example uses Pillow for text and drawing shapes. Copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`sudo -E env PATH=$PATH python3 code.py`

```
# SPDX-FileCopyrightText: 2019 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
ePaper Display Shapes and Text demo using the Pillow Library.
7.5" Tri-Color 800x480 display
https://www.adafruit.com/product/6415
"""

import board
import busio
import digitalio
from PIL import Image, ImageDraw, ImageFont

from adafruit_epd.uc8179 import Adafruit_UC8179
```

```
# First define some color constants
WHITE = (0xFF, 0xFF, 0xFF)
BLACK = (0x00, 0x00, 0x00)
RED = (0xFF, 0x00, 0x00)

# Next define some constants to allow easy resizing of shapes and colors
BORDER = 20
FONTSIZE = 24
BACKGROUND_COLOR = BLACK
FOREGROUND_COLOR = WHITE
TEXT_COLOR = BLACK

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.CE0)
dc = digitalio.DigitalInOut(board.D22)
srcs = None
rst = digitalio.DigitalInOut(board.D27)
busy = digitalio.DigitalInOut(board.D17)

display = Adafruit_UC8179(800, 480,         # 7.5" tricolor 800x480 display
    spi,
    cs_pin=ecs,
    dc_pin=dc,
    sramcs_pin=srcs,
    rst_pin=rst,
    busy_pin=busy,
    tri_color = True
)

display.rotation = 0

width = display.width
height = display.height
image = Image.new("RGB", (width, height))

# clear the buffer
display.fill(WHITE)

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)
# empty it
draw.rectangle((0, 0, width, height), fill=WHITE)

# Draw an outline box
draw.rectangle((1, 1, width - 2, height - 2), outline=BLACK, fill=WHITE)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = 25
shape_width = 100
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position for drawing shapes.
x = padding
# Draw an ellipse.
draw.ellipse((x, top, x + shape_width, bottom), outline=RED, fill=WHITE)
x += shape_width + padding
# Draw a rectangle.
draw.rectangle((x, top, x + shape_width, bottom), outline=RED, fill=BLACK)
x += shape_width + padding
# Draw a triangle.
draw.polygon(
    [(x, bottom), (x + shape_width / 2, top), (x + shape_width, bottom)],
    outline=BLACK,
    fill=RED,
)
x += shape_width + padding
```

```
# Draw an X.
draw.line((x, bottom, x + shape_width, top), fill=BLACK)
draw.line((x, top, x + shape_width, bottom), fill=RED)
x += shape_width + padding

# Load default font.
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 50)

draw.text((x, top), "Hello", font=font, fill=BLACK)
draw.text((x, top + 50), "World!", font=font, fill=BLACK)

# Display image.
display.image(image)

display.display()
```

Your display will show a circle, rectangle, triangle and a large X next to "Hello World!" text.