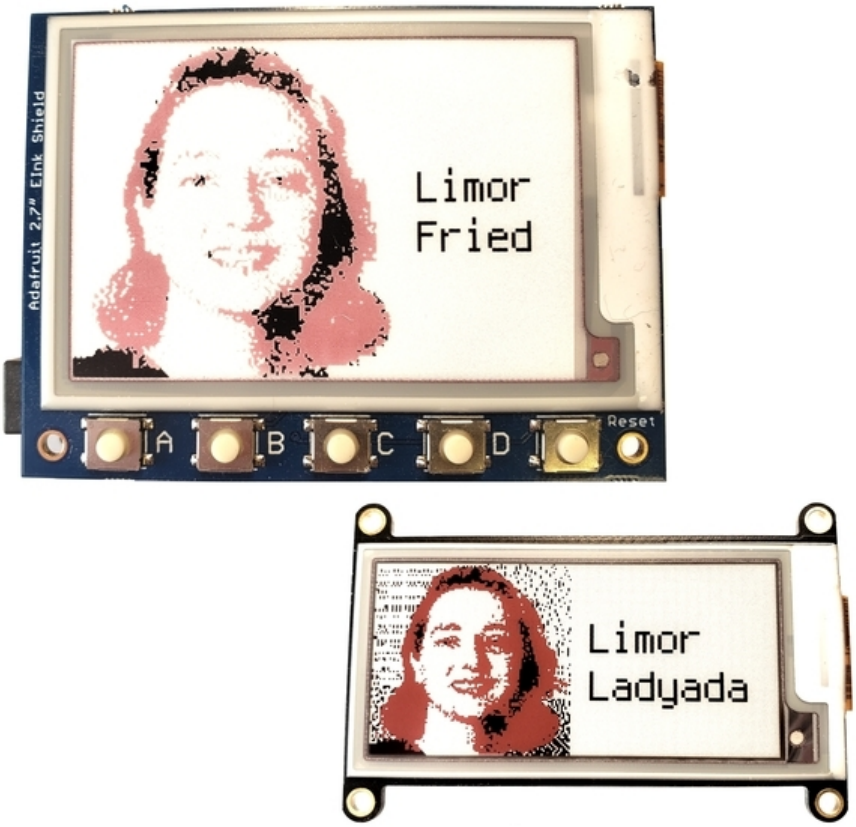
Quickstart using Adafruit eInk/ePaper displays with CircuitPython

Created by Anne Barela

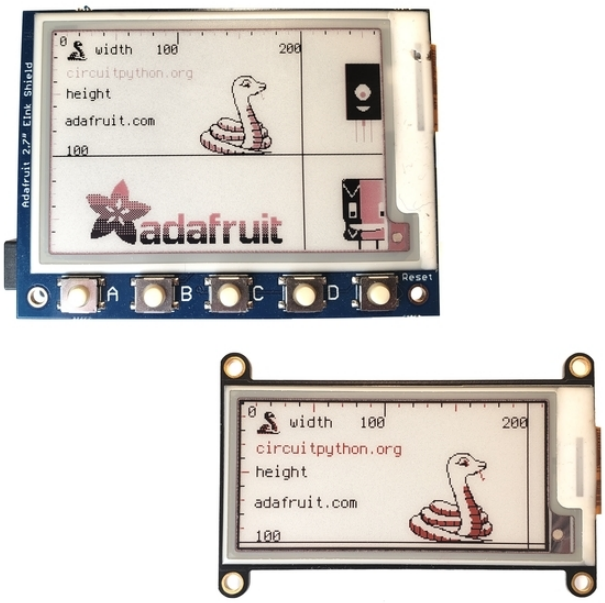


https://learn.adafruit.com/quickstart-using-adafruit-eink-epaper-displays-with-circuitpython

Last updated on 2024-06-03 02:56:17 PM EDT

Table of Contents

Overview



This guide is intended to get you using the Adafruit 2.13" tri-color e-ink FeatherWing and 2.7" tri-color e-ink shield quickly with CircuitPython.

These displays come with headers already soldered on - male headers for the shield version and female headers for the Feather version. This allows them to be plugged in to their respective microcontrollers without soldering. Place on the microcontroller, upload your code and graphics file, and display!

Refresh Times

Most LCD, OLED, and LED displays can be written to quite fast which helps to do moving graphics and colors easily.

For eInk displays, this is not the case. Erasing an image involves sending special signals to the display to "undo" the colored pixels from displaying. Adafruit recommends to not refresh these displays more often than every 180 seconds (3 minutes).

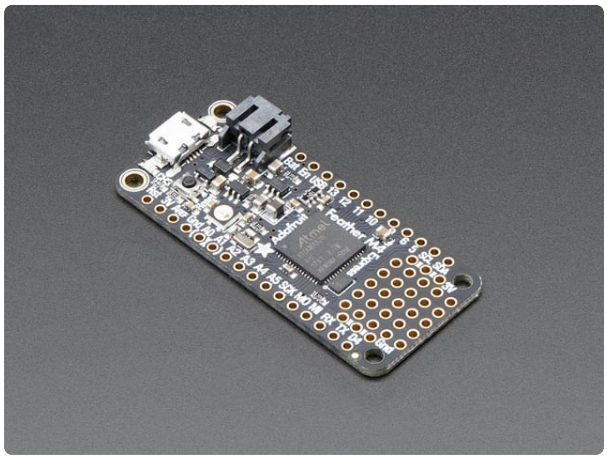
If they are written to more quickly, the display may be damaged. A damaged display may have pixels permanently on or off or other anomalies and voids the warranty. If you want to know more about time between refreshes, see the datasheet on the display you are using here (https://adafru.it/10eL) and here (https://adafru.it/10eM).

eInk displays continue to display an image, even with the power off. If you want a fairly static image, say a picture or a name badge, eInk is wonderful.

If you want a crisp, rarely changing display, eInk is great. For fast changing information, please consider a different class of display like TFT or OLED.

Parts

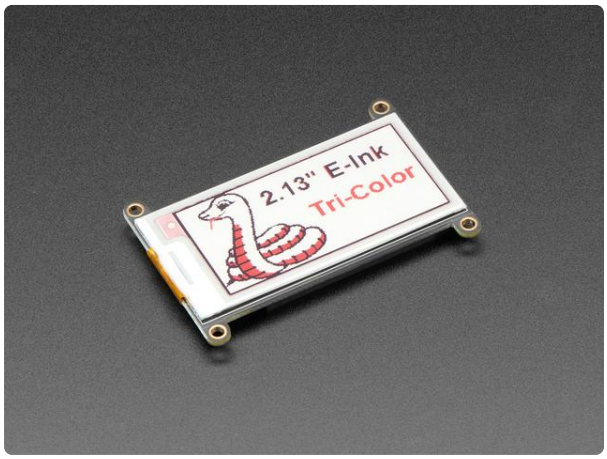
Feather Form Factor



Adafruit Feather M4 Express - Featuring ATSAMD51

It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,...

https://www.adafruit.com/product/3857

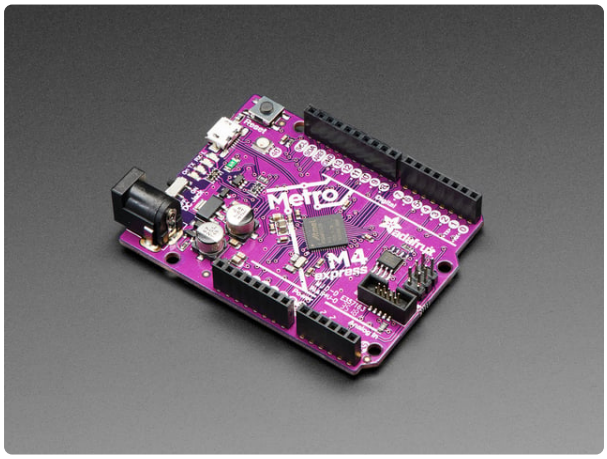


Adafruit 2.13" Tri-Color eInk / ePaper Display FeatherWing

Easy e-paper finally comes to your Feather, with this breakout that's designed to make it a breeze to add a tri-color eInk display. Chances are you've seen one of those...

https://www.adafruit.com/product/4128

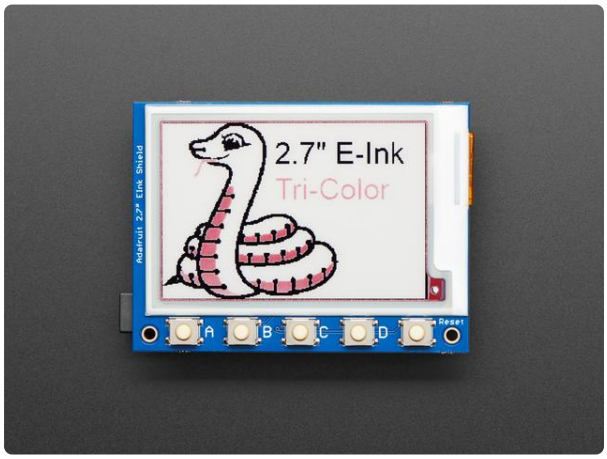
Metro Form Factor



Adafruit Metro M4 feat. Microchip ATSAMD51

Are you ready? Really ready? Cause here comes the fastest, most powerful Metro ever. The Adafruit Metro M4 featuring the Microchip ATSAMD51. This...

https://www.adafruit.com/product/3382



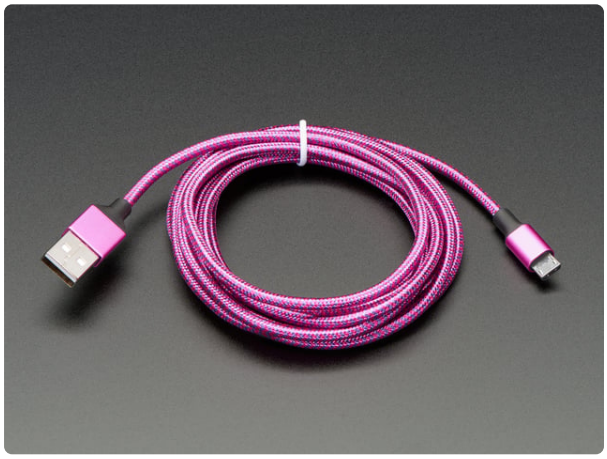
Adafruit 2.7" Tri-Color eInk / ePaper Shield with SRAM

Easy e-paper finally comes to microcontrollers, with this breakout that's designed to make it a breeze to add a tri-color eInk display. Chances are you've seen one of those...

https://www.adafruit.com/product/4229

Programming

Any USB data plus power cable will work if it fits your computer at one end and has a micro B connector on the other end. The one below is a favorite, but pick your own depending on your needs.



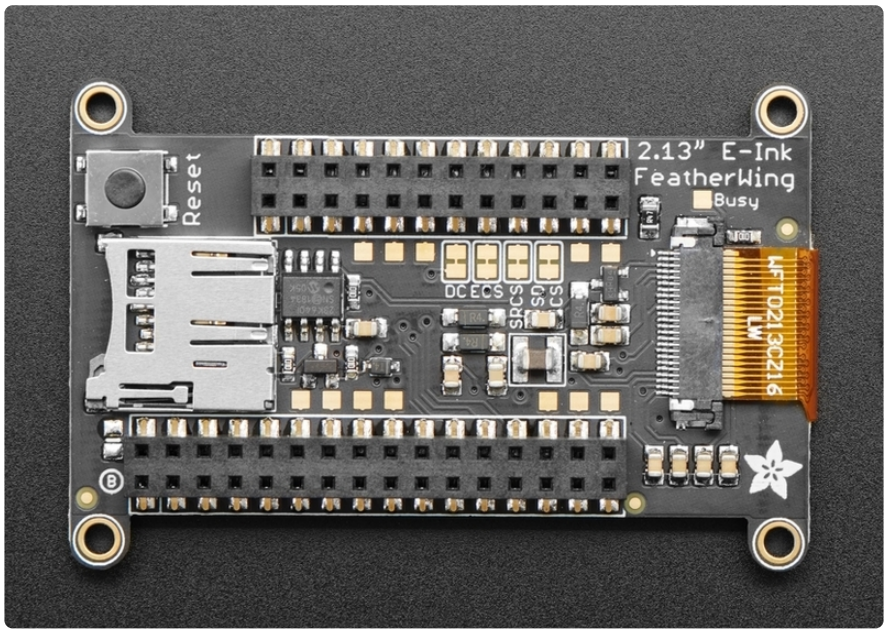
Pink and Purple Braided USB A to Micro B Cable - 2 meter long

This cable is super-fashionable with a woven pink and purple Blinka-like pattern! First let's talk about the cover and over-molding. We got these in custom colors,...

https://www.adafruit.com/product/4148

2.13 Inch Tri-Color eInk FeatherWing

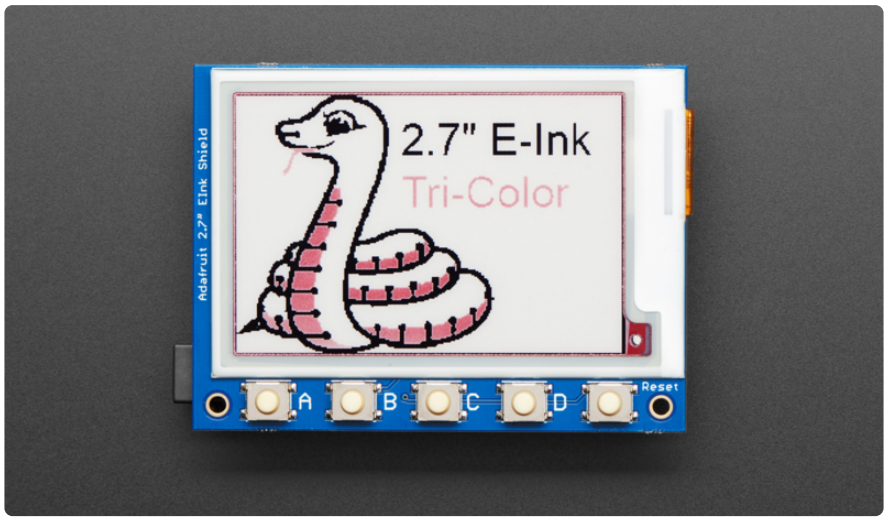
The eInk FeatherWIng is very easy to use. It has an SD card breakout on the back and female headers to plug into a Feather microcontroller. It's solderless if a Feather with male headers is used.



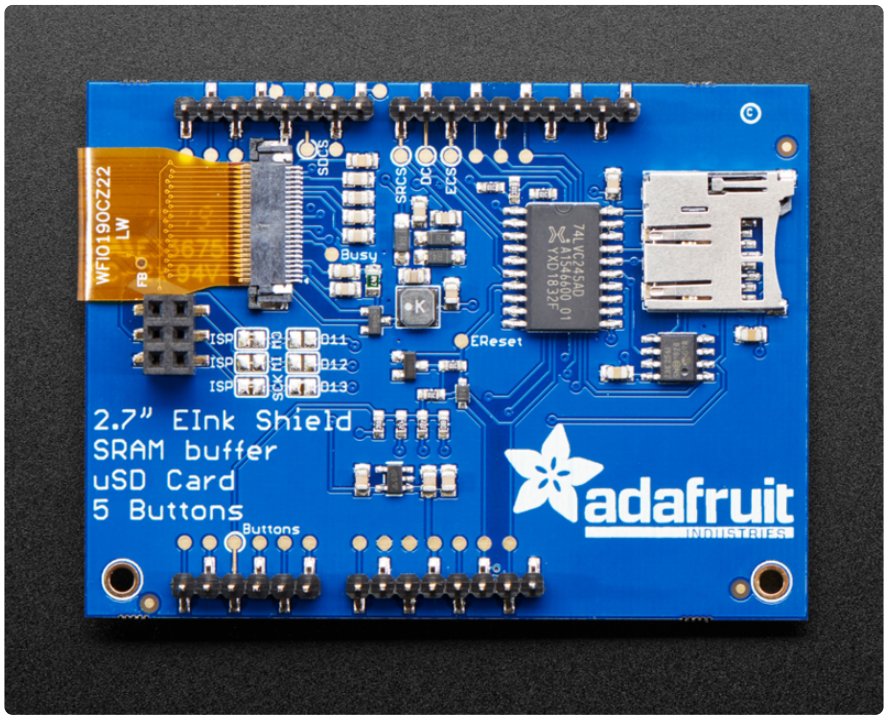
Connection to a Feather Microcontroller

For a speedy chip with lots of memory, the examples use this display with an Adafruit Feather M4 Express microcontroller. Currently this board comes with unsoldered headers.

2.7" Tri-Color eInk Shield



This display, in an Arduino shield form factor, is very easy to use. It has an SD card breakout on the back and male headers to plug into a Metro size microcontroller. It is solderless if a Metro with standard female headers is used.



Connection to a Metro Microcontroller

For a speedy chip with lots of memory, the examples use this display with an Adafruit Metro M4 Express microcontroller. The shield should plug into the Metro M4 with headers. If your Metro does not have female headers, please solder them in.

CircuitPython and Library Setup

Programming

All the quick start examples in this guide for CircuitPython use the new `displayio` driver in CircuitPython 5.0 and higher. The code provides powerful screen composition features and it is optimized. While there are examples of CircuitPython code using the older `adafruit_epd` methods, `displayio` is the future.

Update CircuitPython on your Feather

You should check the version of CircuitPython running on your Feather. Plug the board into your computer with a known good data + power USB cable. The board should show up as a disk drive on your computer named **CIRCUITPY**.

Open up **boot_out.txt**. This will show you the version of CircuitPython the board has loaded.

```
Adafruit CircuitPython 5.0.0-alpha.4 on 2019-09-15; Adafruit Feather M4 Express
with samd51j19
```

`displayio` requires CircuitPython 5 or above. If you have a board that is running a lower version of CircuitPython, you will need to update it. Upgrade to latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Required CircuitPython Libraries

To display a bitmap with `displayio`, there is only one required library, but it is dependent on the display you are using.

First, make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next, you'll need to install the necessary library to use the hardware. Carefully follow the steps to find and install the library from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both Express and non-Express boards.

Due to the number of libraries in the bundle, it is recommended that you manually install the necessary libraries from the bundle. The first library depends on which eInk display you are using:

Feather and the 2.13" Tri Color eInk Display

- **adafruit_IL0373**

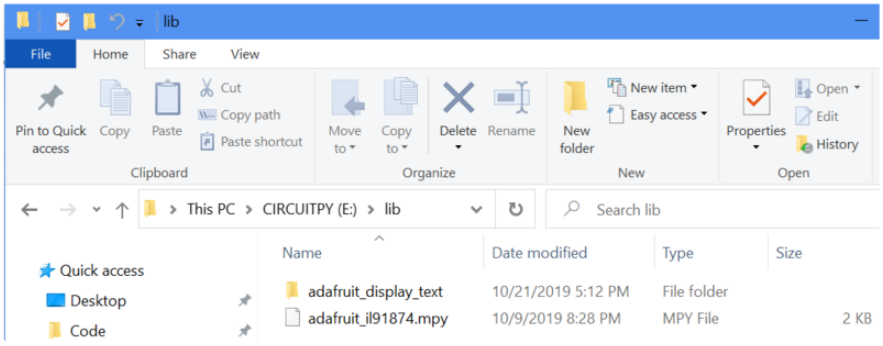
Metro and the 2.7" Tri Color eInk Display

- **adafruit_il91874**

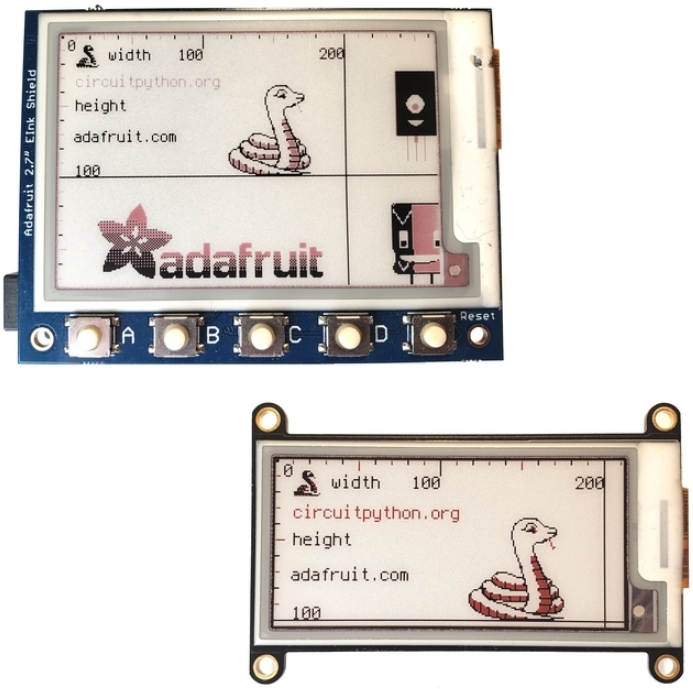
Next, get the following library for displaying text on a `displayio` display. It is used for the second and third examples.

- **adafruit_display_text**

Before continuing, please make sure your board's **lib** folder has the correct driver file and the **adafruit_display_text** libraries copied over.



Example: A Single Bitmap



The first example reads a bitmap image placed on the board's **CIRCUITPY** drive and displays it on the eInk display.

Image

The image used in this example is a CircuitPython text image which shows various elements. You can download it by clicking the green button below.

display-ruler.bmp

https://adafru.it/GOa

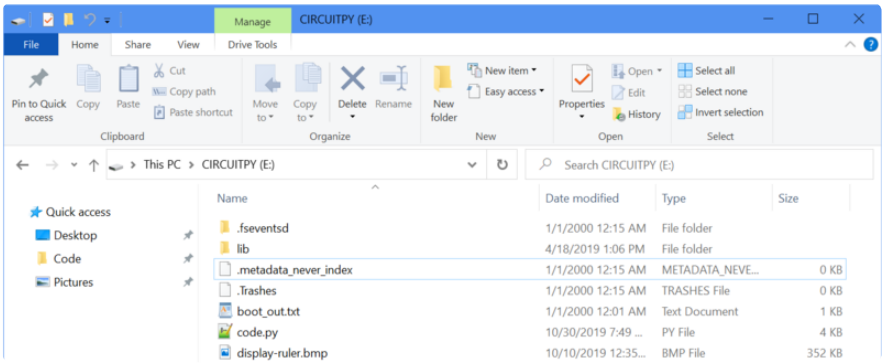
For making your own graphic that can display decently on the three colors available on these displays, see the guide Preparing Graphics for E-Ink Displays (https://adafru.it/GOb).

If the picture is larger than the display width and height, the rest will truncate and not be displayed. It is suggested that you scale the image in an image editor prior to use as the code will not scale a bitmap.

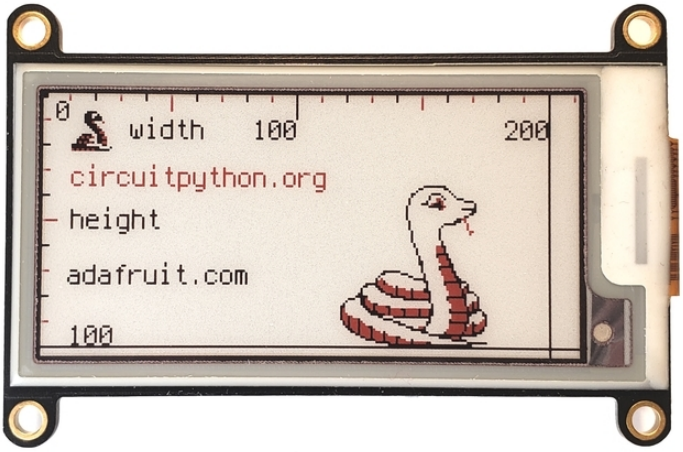
Scale the bitmap you want to use to your display width and height in an image editor and dither or otherwise use only black/white/red. The code cannot perform image scaling on bitmap files.

Code

The code below will take a bitmap placed on the board's flash drive (in the **CIRCUITPY** root directory) and display it on the eInk display. See the comments in the code for how each step contributes to the process.



Please select the code specific to your display and microcontroller as the two code blocks below are not interchangeable for both sets of hardware. The examples are nearly identical though.

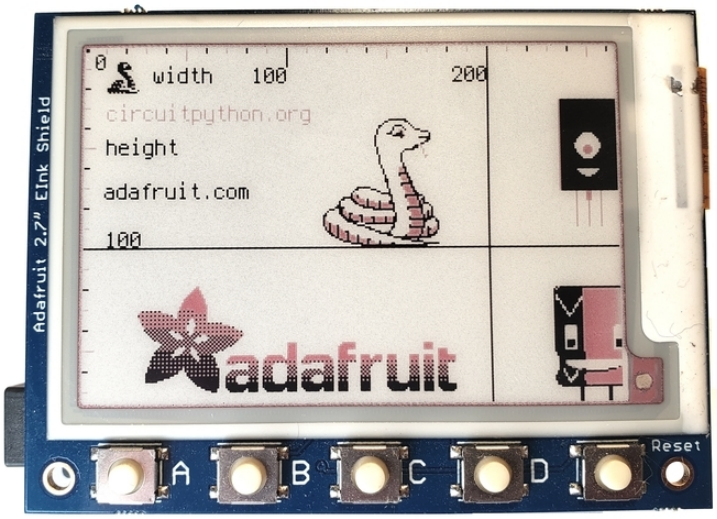


2.13" eInk FeatherWing Example

The pins reflect the combination of the FeatherWing eInk display and a Feather M4. It also has the pixel dimensions for the 2.13" display.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""Simple test script for Adafruit 2.13" 212x104 tri-color display
Supported products:
  * Adafruit 2.13" Tri-Color Display Breakout
  * https://www.adafruit.com/product/4086 (breakout) or
```

```
  * https://www.adafruit.com/product/4128 (FeatherWing)
"""

import time
import board
import displayio
import fourwire
import adafruit_il0373

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use
# This pinout is for a Feather M4 and may be different for other boards
spi = board.SPI()  # Uses SCK and MOSI
epd_cs = board.D9
epd_dc = board.D10
epd_reset = board.D5
epd_busy = board.D6

# Create the displayio connection to the display pins
display_bus = fourwire.FourWire(
    spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset, baudrate=1000000
)
time.sleep(1)  # Wait a bit

# Create the display object - the third color is red (0xff0000)
display = adafruit_il0373.IL0373(
    display_bus,
    width=212,
    height=104,
    rotation=90,
    busy_pin=epd_busy,
    highlight_color=0xFF0000,
)

# Create a display group for our screen objects
g = displayio.Group()

# Display a ruler graphic from the root directory of the CIRCUITPY drive
with open("/display-ruler.bmp", "rb") as f:
    pic = displayio.OnDiskBitmap(f)
    # Create a Tilegrid with the bitmap and put in the displayio group
    # CircuitPython 6 & 7 compatible
    t = displayio.TileGrid(
        pic, pixel_shader=getattr(pic, "pixel_shader", displayio.ColorConverter())
    )
    # CircuitPython 7 compatible only
    # t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
    g.append(t)

    # Place the display group on the screen
    display.root_group = g

    # Refresh the display to have it actually show the image
    # NOTE: Do not refresh eInk displays sooner than 180 seconds
    display.refresh()
    print("refreshed")

    time.sleep(180)
```



2.7" eInk Shield Example

The pins reflect the combination of the shield eInk display and a Metro M4. It also has the pixel dimensions of the 2.7" display.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
  Simple test script for 2.7" 264x176 Tri-Color display shield
  Supported products:
  * Adafruit 2.7" Tri-Color ePaper Display Shield
    https://www.adafruit.com/product/4229

  This program only requires the adafruit_il91874 library in /lib
  for CircuitPython 5.0 and above which has displayio support.
"""

import time
import board
import displayio

# Compatibility with both CircuitPython 8.x.x and 9.x.x.
# Remove after 8.x.x is no longer a supported release.
try:
    from fourwire import FourWire
except ImportError:
    # pylint: disable=ungrouped-imports
    from displayio import FourWire

import adafruit_il91874

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use on the Metro
spi = board.SPI()
epd_cs = board.D10
epd_dc = board.D9
epd_reset = board.D5
```

```
epd_busy = board.D6

# Create the displayio connection to the display pins
display_bus = FourWire(
    spi, command=epd_dc, chip_select=epd_cs, reset=epd_reset, baudrate=1000000
)
time.sleep(1)  # Wait a bit

# Create the display object - the third color is red (0xff0000)
display = adafruit_il91874.IL91874(
    display_bus,
    width=264,
    height=176,
    busy_pin=epd_busy,
    highlight_color=0xFF0000,
    rotation=90,
)

# Create a display group for our screen objects
g = displayio.Group()

# Display a ruler graphic from the root directory of the CIRCUITPY drive
with open("/display-ruler.bmp", "rb") as f:
    pic = displayio.OnDiskBitmap(f)
    # Create a Tilegrid with the bitmap and put in the displayio group
    # CircuitPython 6 & 7 compatible
    t = displayio.TileGrid(
        pic, pixel_shader=getattr(pic, "pixel_shader", displayio.ColorConverter())
    )
    # CircuitPython 7 compatible only
    # t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
    g.append(t)

    # Place the display group on the screen (does not refresh)
    display.root_group = g

    # Show the image on the display
    display.refresh()

    print("refreshed")

    # Do Not refresh the screen more often than every 180 seconds
    #   for eInk displays! Rapid refreshes will damage the panel.
    time.sleep(180)
```

Code Review

For an overview of using `displayio` for displays with CircuitPython, the excellent guide CircuitPython Display Support Using displayio (https://adafru.it/ETG) is your first stop. If you would like a deeper dive into the model used for displays, refer to this guide.

Adafruit suggests using the `displayio.release_displays` function before looking to execute additional code to ensure displays connected to the hardware are released by CircuitPython.

Next is to let the microcontroller know which pins are used on the display. The pins used are very display dependent, it is suggested the guide on the display be referred

to for known a known, working configuration prior to looking to change things up. `displayio.FourWire` sets up the `displayio` connection to the display bus.

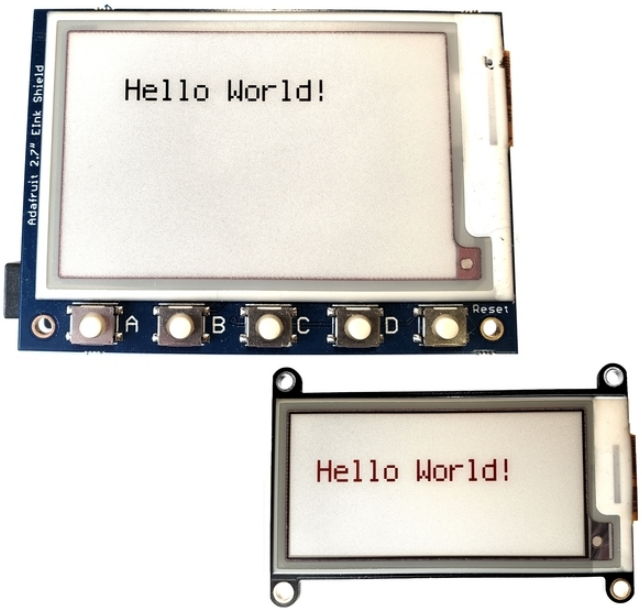
With the bus, the program establishes the connection to the display driver. The size of the display (`width` and `height`), the orientation (`rotation`), busy pin, and the highlight color are given. For this tri-color display, red (`0xff0000`) is specified.

The rest of the code follows the standard `displayio` display setup and use:

- Create a display group
- Open a bitmap to put on the display
- Create a bitmap object
- Create a TileGrid to put objects in with the bitmap and append the tile group to the display group
- Show the display on the screen
- Refresh the screen

Finally the program waits at least 3 minutes. Then the program will complete and go to the REPL. If the bitmap should stay on the screen, add `while True:` and `pass` statements at the end of the program.

Example: Simple Text



This example displays text on the eInk display, the "Hello World" for eInk so to speak. Text can be placed anywhere and scaled larger.

Code

The code below will use the internal `terminalio` font to display a single line of text on the eInk display. You can set the foreground and background color. See the comments for how each step contributes to the process.

Please select the code specific to your display and microcontroller as the two code blocks below are not interchangeable for both sets of hardware. The examples are nearly identical though.

2.13" eInk FeatherWing Example

The pins reflect the combination of the FeatherWing eInk display and a Feather M4. It also has the pixel dimensions for the 2.13" display.

```
# SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Simple text script for Adafruit 2.13" 212x104 tri-color display
Supported products:
  * Adafruit 2.13" Tri-Color Display Breakout
  * Adafruit 2.13" Tri-Color Display FeatherWing
    https://www.adafruit.com/product/4086 (breakout) or
    https://www.adafruit.com/product/4128 (FeatherWing)

  This program requires the adafruit_il0373 library and the
  adafruit_display_text library in the CIRCUITPY /lib folder
  for CircuitPython 5.0 and above which has displayio support.
"""

import time
import board
import displayio
import adafruit_il0373
import terminalio
from adafruit_display_text import label
```

```
BLACK = 0x000000
WHITE = 0xFFFFFF
RED = 0xFF0000

# Change text colors, choose from the following values:
# BLACK, RED, WHITE

FOREGROUND_COLOR = RED
BACKGROUND_COLOR = WHITE

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use
# This pinout is for a Feather M4 and may be different for other boards
spi = board.SPI()  # Uses SCK and MOSI
epd_cs = board.D9
epd_dc = board.D10
epd_reset = board.D5
epd_busy = board.D6

# Create the displayio connection to the display pins
display_bus = displayio.FourWire(spi, command=epd_dc, chip_select=epd_cs,
                                 reset=epd_reset, baudrate=1000000)
time.sleep(1)  # Wait a bit

# Create the display object - the third color is red (0xff0000)
DISPLAY_WIDTH = 212
DISPLAY_HEIGHT = 104

display = adafruit_il0373.IL0373(display_bus, width=DISPLAY_WIDTH,
                                 height=DISPLAY_HEIGHT,
                                 rotation=90, busy_pin=epd_busy,
                                 highlight_color=0xff0000)

# Create a display group for our screen objects
g = displayio.Group()

# Set a background
background_bitmap = displayio.Bitmap(DISPLAY_WIDTH, DISPLAY_HEIGHT, 1)
# Map colors in a palette
palette = displayio.Palette(1)
palette[0] = BACKGROUND_COLOR

# Create a Tilegrid with the background and put in the displayio group
t = displayio.TileGrid(background_bitmap, pixel_shader=palette)
g.append(t)

# Draw simple text using the built-in font into a displayio group
text_group = displayio.Group(scale=2, x=20, y=40)
text = "Hello World!"
text_area = label.Label(terminalio.FONT, text=text, color=FOREGROUND_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Place the display group on the screen
display.root_group = g

# Refresh the display to have everything show on the display
# NOTE: Do not refresh eInk displays more often than 180 seconds!
display.refresh()

time.sleep(120)

while True:
    pass
```

2.7" eInk Shield Example

The pins reflect the combination of the shield eInk display and a Metro M4. It also has the pixel dimensions of the 2.7" display.

```
# SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
  Simple text script for 2.7" 264x176 Tri-Color display shield
  Supported products:
  * Adafruit 2.7" Tri-Color ePaper Display Shield
    https://www.adafruit.com/product/4229
  This program requires the adafruit_il91874 and the
  adafruit_display_text library in /lib on the CIRCUITPY drive
  for CircuitPython 5.0 and above which has displayio support.
"""

import time
import board
import displayio
import adafruit_il91874
import terminalio
from adafruit_display_text import label

BLACK = 0x000000
WHITE = 0xFFFFFF
RED = 0xFF0000

# Change text colors, choose from the following values:
# BLACK, WHITE, RED (note red on this display is not vivid)

FOREGROUND_COLOR = BLACK
BACKGROUND_COLOR = WHITE

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use on the Metro
spi = board.SPI()
epd_cs = board.D10
epd_dc = board.D9
```

```
# Create the displayio connection to the display pins
display_bus = displayio.FourWire(spi, command=epd_dc, chip_select=epd_cs,
                                 baudrate=1000000)
time.sleep(1)  # Wait a bit

# Create the display object - the third color is red (0xff0000)
DISPLAY_WIDTH = 264
DISPLAY_HEIGHT = 176

# Create the display object - the third color is red (0xff0000)
display = adafruit_il91874.IL91874(display_bus, width=DISPLAY_WIDTH,
                                   height=DISPLAY_HEIGHT,
                                   highlight_color=0xff0000, rotation=90)

# Create a display group for our screen objects
g = displayio.Group()

# Set a white background
background_bitmap = displayio.Bitmap(DISPLAY_WIDTH, DISPLAY_HEIGHT, 1)
# Map colors in a palette
palette = displayio.Palette(1)
palette[0] = BACKGROUND_COLOR

# Create a Tilegrid with the background and put in the displayio group
t = displayio.TileGrid(background_bitmap, pixel_shader=palette)
g.append(t)

# Draw simple text using the built-in font into a displayio group
text_group = displayio.Group(scale=2, x=40, y=40)
text = "Hello World!"
text_area = label.Label(terminalio.FONT, text=text, color=FOREGROUND_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Place the display group on the screen
display.root_group = g

# Refresh the display to have everything show on the display
# NOTE: Do not refresh eInk displays more often than 180 seconds!
display.refresh()

time.sleep(180)
while True:
    pass
```

Code Review

For an overview of using `displayio` for displays with CircuitPython, the excellent guide CircuitPython Display Support Using displayio (https://adafru.it/ETG) is your first stop. If you would like a deeper dive into the model used for displays, refer to this guide.

Adafruit suggests using the `displayio.release_displays()` function before looking to execute additional code to ensure displays connected to the hardware are released by CircuitPython.

Next is to let `displayio` know which pins are used on the display. The specific pins used are very display dependent, it is suggested the guide on the display be referred

to for known a known, working configuration prior to looking to change things up. `fourwire.FourWire` sets up the `displayio` connection to the display bus.

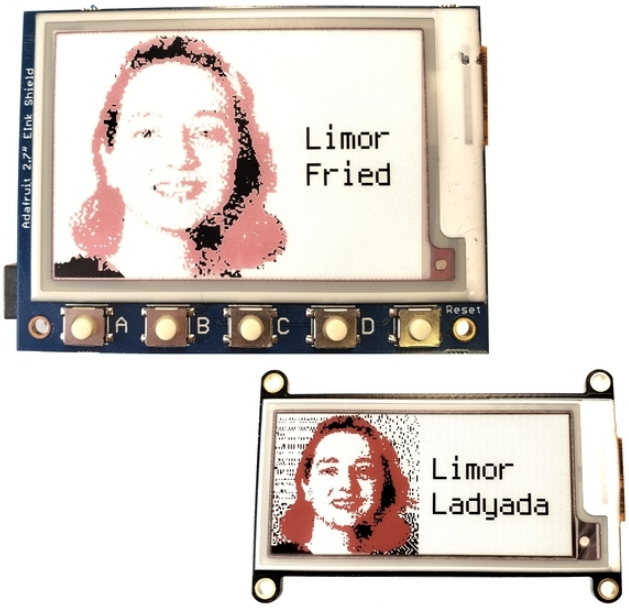
With the bus, the program establishes the connection to the display driver `adafruit_il0373`. The size of the display (`width` and `height`), the orientation (`rotation`), busy pin, and the highlight color are given. For this tri-color display, red (`0xff0000`) is specified.

The rest of the code follows the standard `displayio` display setup and use:

- Create a display group
- Set the desired background color as a bitmap
- Create a TileGrid to put objects in with the bitmap and append the tile group to the display group
- Add a display group item for text and use "Hello World!" as the sample text. Text is scaled by two in size and located at `x=40`, `y=40` to move it from the upper left down and to the right.
- Set your group as the `root_group` on the Display
- Refresh the screen

Finally the program waits 3 minutes (the minimum refresh time. Then the program uses `while True:` and `pass` statements to keep what's on the display until the processor is reset.

Example: A Name Badge



The final example displays text over a bitmap on the eInk display, making a name badge useful for gatherings or conferences.

Images

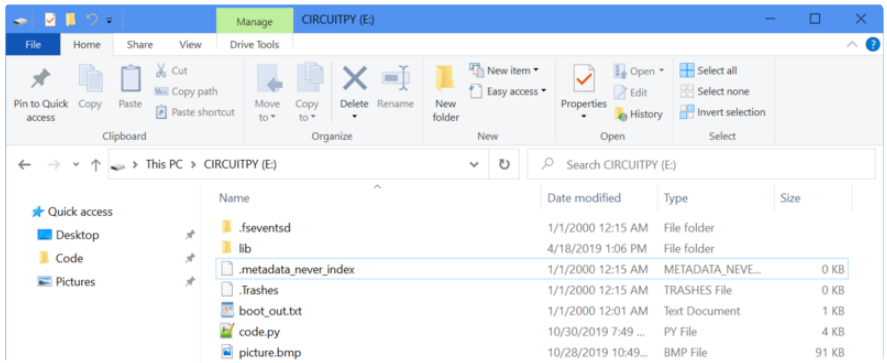
This example uses a square picture file in BMP format to display on the left side of the display. It could be someone's picture, a logo, or other art.

For using the 2.13" display with a height of 104 pixels, make your image 104x104.

For using the 2.7" display with a height of 176 pixels, make your image 176x176.

As the eInk display can only use 3 colors, see the companion guide Preparing Graphics for E-Ink Displays (https://adafru.it/GOb). You may have to play with images to see how they might display best on your chosen display.

Here are sample pictures used with the examples. For the 2.13" display, use the smaller picture and for the 2.7" shield, use the larger picture. Copy the image to the board's **CIRCUITPY** drive as **picture.bmp**.



Code

The code below will display a square picture on the left and use the internal `terminalio` font to place two small lines of text on the right of the eInk display. You can set the foreground and background color. See the comments for how each step contributes to the process.

Please select the code specific to your display and microcontroller as the two code blocks below are not interchangeable for both sets of hardware. The examples are nearly identical though.

The 2.13" display shows more of a true red color and the 2.7" display uses more of a pinkish red. It is all based on how the displays are manufactured and not color issues with the display itself. Please factor this into your planning for color choices.

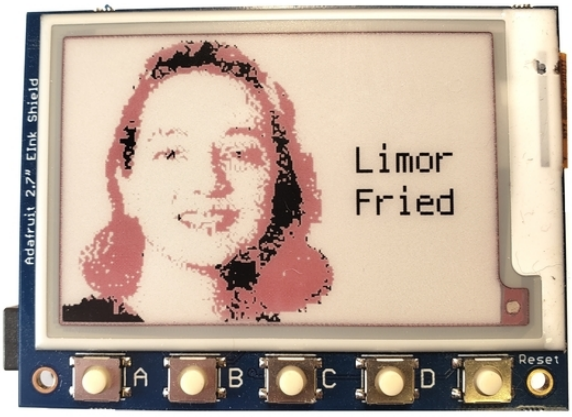
2.13" eInk FeatherWing Example

The pins reflect the combination of the FeatherWing eInk display and a Feather M4. It also has the pixel dimensions for the 2.13" display.

```
# SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Simple badge script for Adafruit 2.13" 212x104 tri-color display
Supported products:
  * Adafruit 2.13" Tri-Color Display Breakout
  * https://www.adafruit.com/product/4086 (breakout) or
  * https://www.adafruit.com/product/4128 (FeatherWing)

  This program requires the adafruit_il0373 library and the
  adafruit_display_text library in the CIRCUITPY /lib folder
```

```
  for CircuitPython 5.0 and above which has displayio support.
"""

import time
import board
import displayio
import adafruit_il0373
import terminalio
from adafruit_display_text import label

BLACK = 0x000000
WHITE = 0xFFFFFF
RED = 0xFF0000

# Change text colors, choose from the following values:
# BLACK, RED, WHITE

TEXT_COLOR = BLACK
BACKGROUND_COLOR = WHITE

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use
# This pinout is for a Feather M4 and may be different for other boards
spi = board.SPI()  # Uses SCK and MOSI
epd_cs = board.D9
epd_dc = board.D10
epd_reset = board.D5
epd_busy = board.D6

# Create the displayio connection to the display pins
display_bus = displayio.FourWire(spi, command=epd_dc, chip_select=epd_cs,
                                 reset=epd_reset, baudrate=1000000)
time.sleep(1)  # Wait a bit

DISPLAY_WIDTH = 212
DISPLAY_HEIGHT = 104
# Create the display object - the third color is red (0xff0000)
display = adafruit_il0373.IL0373(display_bus, width=DISPLAY_WIDTH,
                                 height=DISPLAY_HEIGHT,
                                 rotation=90, busy_pin=epd_busy,
                                 highlight_color=0xff0000)

# Create a display group for our screen objects
g = displayio.Group()

# Set a background
background_bitmap = displayio.Bitmap(DISPLAY_WIDTH, DISPLAY_HEIGHT, 1)
# Map colors in a palette
palette = displayio.Palette(1)
palette[0] = BACKGROUND_COLOR

# Put the background into the display group
bg_sprite = displayio.TileGrid(background_bitmap,
                               pixel_shader=palette,
                               x=0, y=0)
g.append(bg_sprite)

# Display a picture from the root directory of the CIRCUITPY drive
# Picture should be HEIGHTxHEIGHT square ideally for a portrait
# But could be the entire WIDTHxHEIGHT for a non-portrait
filename = "/picture.bmp"

# Create a Tilegrid with the bitmap and put in the displayio group

# CircuitPython 6 & 7 compatible
pic = displayio.OnDiskBitmap(open(filename, "rb"))
t = displayio.TileGrid(
```

```
    pic, pixel_shader=getattr(pic, 'pixel_shader', displayio.ColorConverter())
)
g.append(t)

# # CircuitPython 7+ compatible
# pic = displayio.OnDiskBitmap(filename)
# t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
# g.append(t)

# Draw simple text using the built-in font into a displayio group
# For smaller text, change scale=2 to scale=1
text_group = displayio.Group(scale=2,
                             x=DISPLAY_HEIGHT + 10,
                             y=int(DISPLAY_HEIGHT/2) - 13)
first_name = "Limor"
text_area = label.Label(terminalio.FONT, text=first_name,
                        color=TEXT_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Draw simple text using the built-in font into a displayio group
text_group = displayio.Group(scale=2,
                             x=DISPLAY_HEIGHT + 10,
                             y=int(DISPLAY_HEIGHT/2) + 13)
last_name = "Ladyada"
text_area = label.Label(terminalio.FONT, text=last_name,
                        color=TEXT_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Place the display group on the screen
display.root_group = g

# Refresh the display to have it actually show
# NOTE: Do not refresh eInk displays more often than 180 seconds!
display.refresh()

# Wait the minimum 3 minutes between refreshes. Then loop to freeze.
time.sleep(180)
while True:
    pass
```



2.7" eInk Shield Example

The pins reflect the combination of the shield eInk display and a Metro M4. It has the pixel dimensions of the 2.7" display.

```
# SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Simple Badge script for a 2.7" 264x176 Tri-Color eInk display shield
  Supported products:
  * Adafruit 2.7" Tri-Color ePaper Display Shield
    https://www.adafruit.com/product/4229

  This program requires the adafruit_il91874 and the
  adafruit_display_text library in /lib on the CIRCUITPY drive
  for CircuitPython 5.0 and above which has displayio support.
"""

import time
import board
import displayio
import adafruit_il91874
import terminalio
from adafruit_display_text import label

BLACK = 0x000000
WHITE = 0xFFFFFF
RED = 0xFF0000

# Change text colors, choose from the following values:
# BLACK, RED, WHITE

TEXT_COLOR = BLACK
BACKGROUND_COLOR = WHITE

# Used to ensure the display is free in CircuitPython
displayio.release_displays()

# Define the pins needed for display use on the Metro
spi = board.SPI()
epd_cs = board.D10
epd_dc = board.D9

# Create the displayio connection to the display pins
display_bus = displayio.FourWire(spi, command=epd_dc, chip_select=epd_cs,
                                 baudrate=1000000)
time.sleep(1)  # Wait a bit

DISPLAY_WIDTH = 264
DISPLAY_HEIGHT = 176
# Create the display object - the third color is red (0xff0000)
display = adafruit_il91874.IL91874(display_bus, width=DISPLAY_WIDTH,
                                   height=DISPLAY_HEIGHT,
                                   highlight_color=0xff0000, rotation=90)

# Create a display group for our screen objects
g = displayio.Group()

# Set a background
background_bitmap = displayio.Bitmap(DISPLAY_WIDTH, DISPLAY_HEIGHT, 1)
# Map colors in a palette
palette = displayio.Palette(1)
palette[0] = BACKGROUND_COLOR
```

```
# Put the background into the display group
bg_sprite = displayio.TileGrid(background_bitmap,
                               pixel_shader=palette,
                               x=0, y=0)

g.append(bg_sprite)

# Display a picture from the root directory of the CIRCUITPY drive
# Picture should be HEIGHTxHEIGHT square ideally for a portrait
# But could be the entire WIDTHxHEIGHT for a non-portrait
filename = "/picture.bmp"

# Create a Tilegrid with the bitmap and put in the displayio group

# CircuitPython 6 & 7 compatible
pic = displayio.OnDiskBitmap(open(filename, "rb"))
t = displayio.TileGrid(
    pic, pixel_shader=getattr(pic, 'pixel_shader', displayio.ColorConverter())
)
g.append(t)

# # CircuitPython 7+ compatible
# pic = displayio.OnDiskBitmap(filename)
# t = displayio.TileGrid(pic, pixel_shader=pic.pixel_shader)
# g.append(t)

# Draw simple text using the built-in font into a displayio group
# For smaller text, change scale=2 to scale=1 as scale 2 doesn't
# allow for much text but the text is bigger.
text_group = displayio.Group(scale=2,
                             x=DISPLAY_HEIGHT + 5,
                             y=int(DISPLAY_HEIGHT/2) - 13)
first_name = "Limor"
text_area = label.Label(terminalio.FONT, text=first_name,
                        color=TEXT_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Draw simple text using the built-in font into a displayio group
# For smaller text, change scale=2 to scale=1 as scale 2 doesn't
# allow for much text but the text is bigger.
text_group = displayio.Group(scale=2,
                             x=DISPLAY_HEIGHT + 5,
                             y=int(DISPLAY_HEIGHT/2) + 13)
last_name = "Fried"
text_area = label.Label(terminalio.FONT, text=last_name,
                        color=TEXT_COLOR)
text_group.append(text_area)  # Add this text to the text group
g.append(text_group)

# Place the display group on the screen
display.root_group = g

# Refresh the display to have it actually show
# NOTE: Do not refresh eInk displays more often than 180 seconds!
display.refresh()

# Wait the minimum 3 minutes between refreshes. Then loop to freeze.
time.sleep(180)
while True:
    pass
```

Code Review

For an overview of using `displayio` for displays with CircuitPython, the excellent guide CircuitPython Display Support Using displayio (https://adafru.it/ETG) is your first stop. If you would like a deeper dive into the model used for displays, refer to this guide.

Adafruit suggests using the `displayio.release_displays()` function before looking to execute additional code to ensure displays connected to the hardware are released by CircuitPython.

Next is to let `displayio` know which pins are used on the display. The specific pins used are very display dependent, it is suggested the guide on the display be referred to for known a known, working configuration prior to looking to change things up. `fourwire.FourWire` sets up the `displayio` connection to the display bus.

The display dimensions (display-specific) are defined. Then the bus, the program establishes the connection to the display driver. The size of the display (`width` and `height`), the orientation (`rotation`), busy pin, and the highlight color are given. For this tri-color display, red (`0xff0000`) is specified.

The rest of the code is very similar to the last two examples. First a bitmap graphic named **picture.bmp** is read from the root directory of the **CIRCUITPY** drive. The graphic is set in the display group with `displayio.TileGrid`. Then two lines of text are defined. They are placed a few pixels to the right of the bitmap and equidistant to the middle of the display's height.

Not much text can fit in this manor. There are two methods you can get more text width:

- Changing `scale=2` to `scale=1` will give more characters per line but the text will be smaller.
- If you need ultimate flexibility, it is suggested you create all your design elements into one bitmap picture and use the first example to display just the bitmap.

The program places all the group elements on screen by setting `display.root_group` then refreshes the display with `display.refresh` to actually have the graphics show up. It takes 2-3 seconds for an eInk display to erase the previous image and display the new image.

Finally the program waits 3 minutes. A `while True:` and `pass` loop is at the end of the program so the REPL does not erase the badge graphics. Pressing the reset button will start the program again.

Going Further

This guide is meant as an introduction to using bitmap graphics and text on a three color eInk display with CircuitPython and the `displayio` library.

Additional Adafruit Learning System Guides provide more examples of making different screen elements. Please refer to the following guides for more information:

- Adafruit eInk Display Breakouts (https://adafru.it/GOc)
- Preparing Graphics for E-Ink Displays (https://adafru.it/GOb)
- CircuitPython Display Support Using displayio (https://adafru.it/ETG)

For more advanced name tag badges, you might consider using and Adafruit PyBadge with its TFT LCD display

- PyBadge Conference Badge With Unicode Fonts (https://adafru.it/GOd)