Adafruit CharliePlex LED Matrix Bonnet

Created by Kattni Rembor


Last updated on 2019-03-13 10:30:23 PM UTC

Overview

You won't be able to look away from the mesmerizing patterns created by this **Adafruit CharliePlex LED Matrix Display Bonnet**. This 16x8 LED display can be placed atop any Raspberry Pi computer with a 2x20 connector, for a beautiful, bright grid of 128 CharliePlexed LEDs. It even comes with a built-in CharliePlex driver that is run over I2C.

We carry these Bonnets in a few vivid colors (https://adafru.it/EcS).
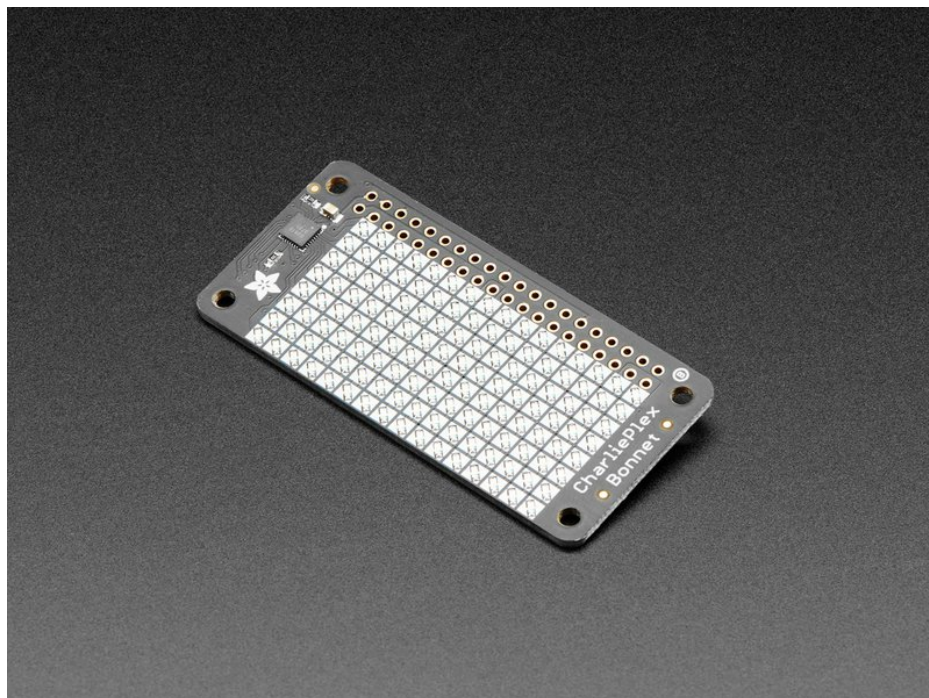
What is particularly nice about this Bonnet is the I2C LED driver chip has the ability to PWM each individual LED in a

16x8 grid so you can have beautiful LED lighting effects, without a lot of pin twiddling. Simply tell the chip which LED on the grid you want lit, and what brightness and it's all taken care of for you. You get 8-bit (256 level) dimming for each individual LED.
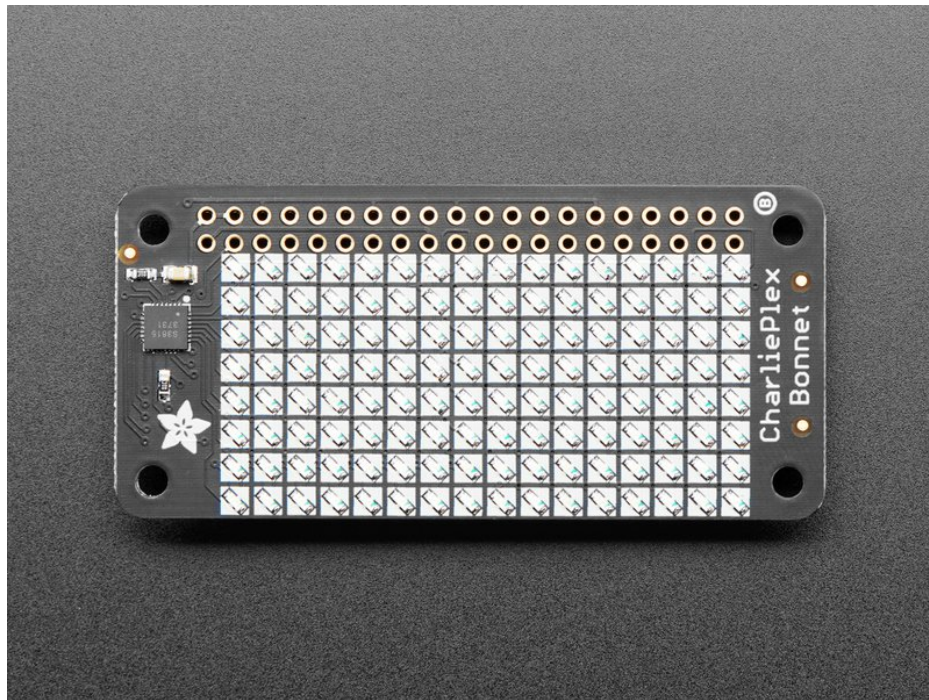

The IS31FL3731 is a nice little chip - and it runs happily over 3.3V power. Inside is enough RAM for 8 separate frames of display memory so you can set up multiple frames of an animation and flip them to be displayed with a single command. Since it uses I2C, it takes up only the SDA/SCL pins on your Pi and can share those pins with other I2C devices and sensors.

The bonnet comes fully assembled, no soldering required, so you can plug it in immediately. To program it, you'll use our CircuitPython library (https://adafru.it/zIE), which works with Linux computers like Raspberry Pi. After a pip install, run some of the examples to display simple graphics or text (https://adafru.it/EcT).
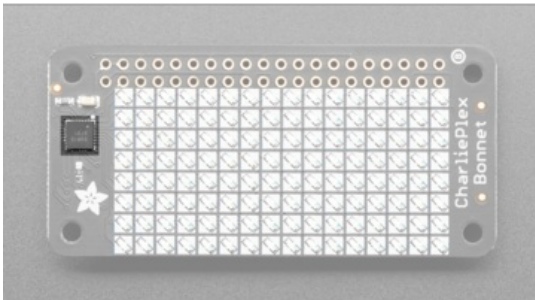
Pinouts
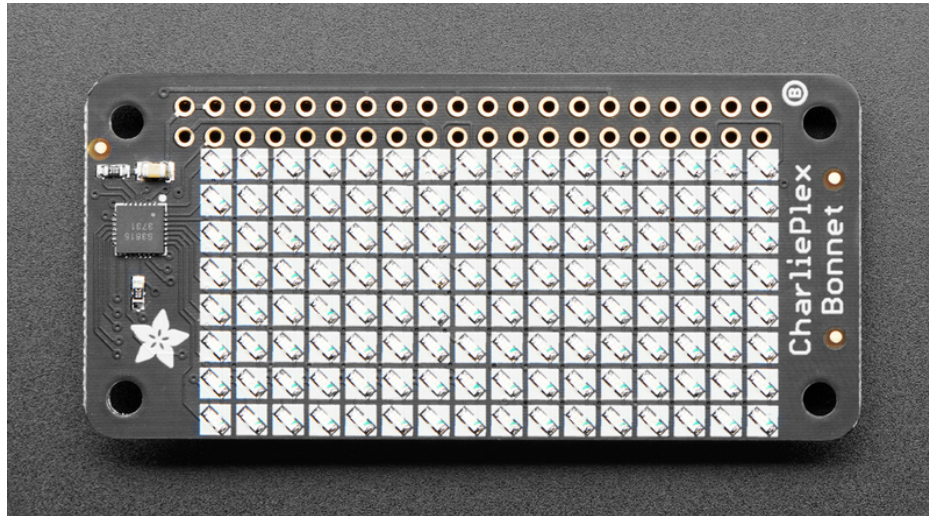
The CharliePlex LED Matrix Bonnet includes the IS31FL3731 driver chip and 128 LEDs in an 8x16 matrix. Let's take a look!

The main pins used by the Bonnet are **5V**, **GND**, **SDA** and **SCL** on the Raspberry Pi. We power the LEDs from the 5V supply, but the SDA/SCL logic level is still only 3.3V so it's safe!
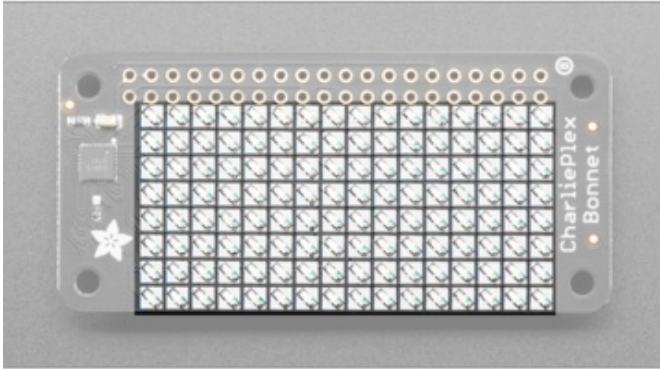
IS31FL3731 Driver Chip

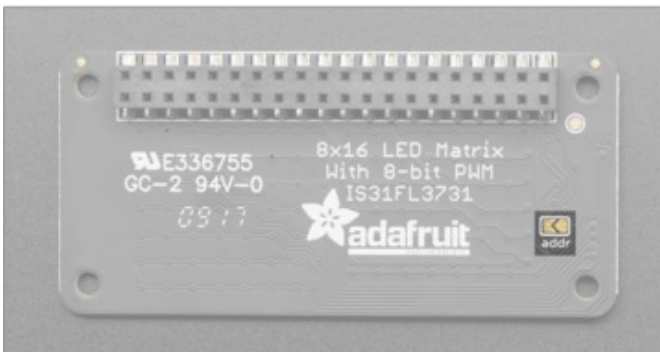The IS31FL3731 drives the LED matrix via I2C.

8 x 16 LED Matrix

The 8 x 16 LED matrix on the CharliePlex Bonnet provides 128 LEDs.

Address Jumper

The default I2C address for the Bonnet is `0x74`. Solder this jumper closed to change it to `0x70` to avoid I2C address conflicts, if needed.

Python & CircuitPython

It's easy to use the IS31FL3731 Charlieplex breakout, the Charlieplex FeatherWing, and the CharliePlex Bonnet with Python or CircuitPython and the Adafruit CircuitPython IS31FL3731 (https://adafru.it/zIE) module. This module allows you to easily write Python code that does all sorts of fun things with the LED matrix.

You can use CharliePlex LED matrices with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a IS31FL3731 breakout to your board exactly as shown on the previous pages for Arduino.
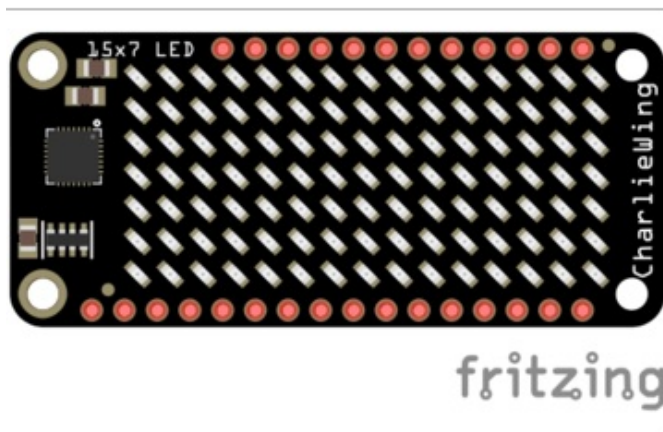
For the FeatherWing, solder on the headers, and attach to the Feather.

Here's an example of wiring a Feather M0 to the breakout with I2C:
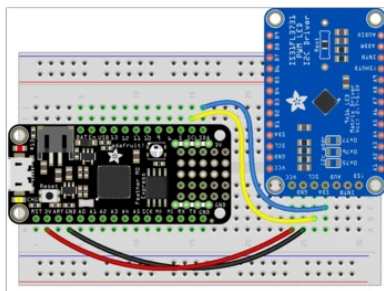
- **Board 3V** to **sensor VCC**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

And here is the CharlieWing on a Feather M4:

- Assemble the CharlieWing by soldering headers onto the board.
- Once assembled, plug it into a Feather!

Python Computer Wiring

Since there's *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).

For the Bonnet, simply attach the Bonnet to your Raspberry Pi header.

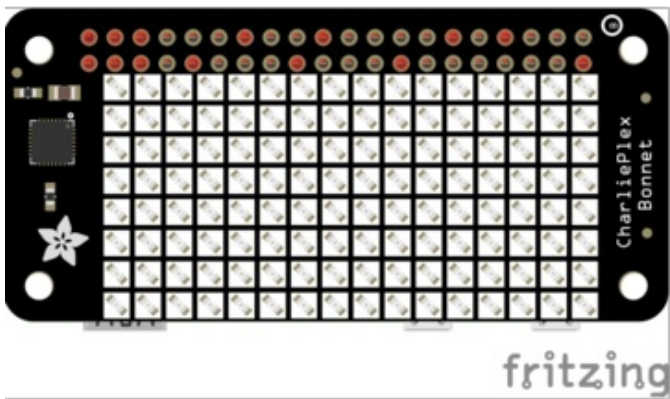Here's the Raspberry Pi wired to the breakout with I2C:
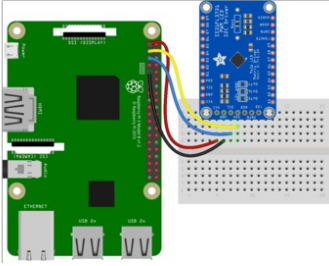
- **Pi 3V3** to **sensor VIN**
- **Pi GND** to **sensor GND**
- **Pi SCL** to **sensor SCL**
- **Pi SDA** to **sensor SDA**

Here is the CharliePlex Bonnet on a Raspberry Pi Zero:

The CharliePlex Bonnet comes fully assembled. Simply plug it into your Raspberry Pi!

CircuitPython Installation of IS31FL3731 Library

You'll need to install the Adafruit CircuitPython IS31FL3731 (https://adafru.it/zlE) library on your CircuitPython board.

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 libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_is31fl3731.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_is31fl3731.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

Python Installation of IS31FL3731 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C 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)!

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

- sudo pip3 install adafruit-circuitpython-is31fl3731

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!

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and manipulate the LED matrix from the board's Python REPL.

NOTE: Due to size and design of each CharliePlex matrix form-factor, import and initialisation is different for each. Make sure you're running the correct code for your matrix!

First, run the following code to import the necessary modules and create the I2C object:

```
import board
import busio
import adafruit_is31fl3731
i2c = busio.I2C(board.SCL, board.SDA)
```

If you're using the CharliePlex breakout, initialise it by running the following code:

```
display = adafruit_is31fl3731.Matrix(i2c)
```

If you're using the CharliePlex FeatherWing, run the following code:

```
display = adafruit_is31fl3731.CharlieWing(i2c)
```

If you're using the CharliePlex Bonnet, run the following code:

```
display = adafruit_is31fl3731.CharlieBonnet(i2c)
```

When the display initializes it will go through and clear each frame (there are 8 frames total) of the display. You might see the display momentarily flash and then turn off to a clear no pixel lit image.

You can control all of the board's pixels using the fill function. Send to this function a value from 0 to 255 where 0 is every LED pixel turned off and 255 is every LED pixel turned on to maximum brightness. For example to set all the pixels to half their brightness run:

```
display.fill(127)
```
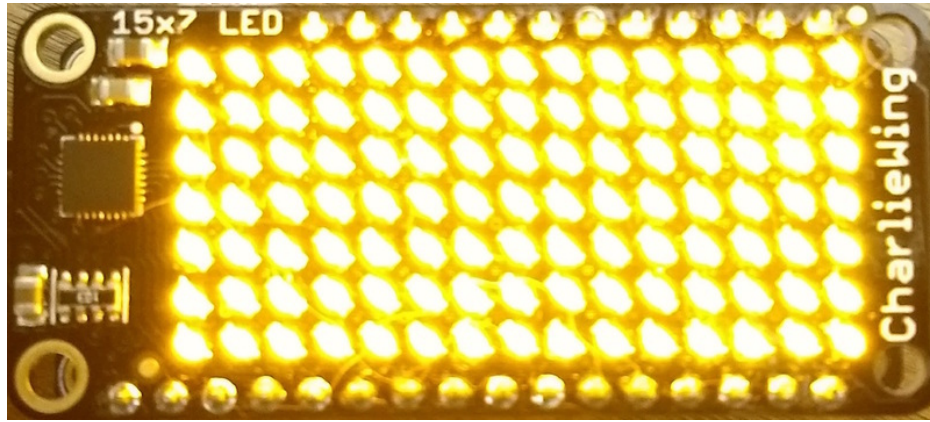
You might notice some buzzing or ringing sounds from the display when all pixels are lit, this is normal as the Charlieplex driver quickly switches LEDs on and off.

If you've used other displays like LED matrices you might notice the Charlieplex module doesn't need to have a show function called to make the changes visible. As soon as you call fill or other display functions the display will update!

You can turn all the pixels off with `fill` set to `0`:

```
display.fill(0)
```
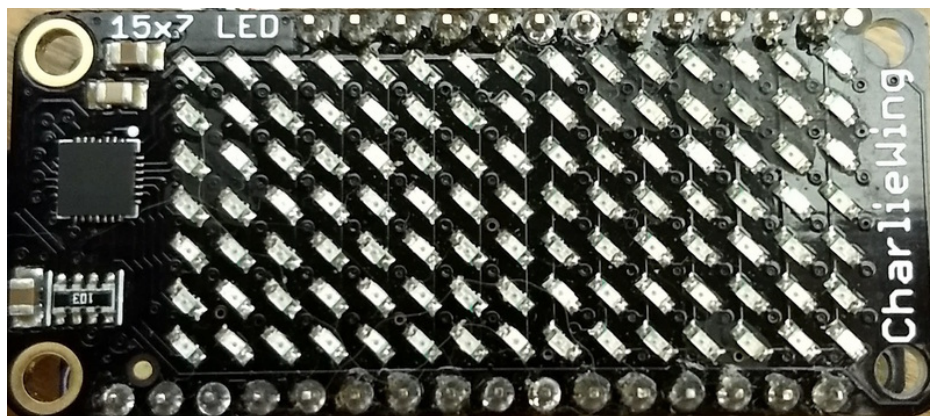

> Be careful setting all pixels to 255 maximum brightness! This might pull more power than your computer's USB port can provide if you are powering your board over USB. Use an external powers supply or battery when lighting lots of LEDs to max brightness.

Now for some fun! You can set any of the LED pixels using the `pixel` function. This function takes the following parameters:

- **X position** - The location of the horizontal / X pixel position.
- **Y position** - The location of the vertical / Y pixel position.
- **Intensity** - This is a value from `0` to `255` which specifies how bright the pixel should be, 0 is off and 255 is maximum brightness. Use an in-between value to show a less bright pixel.
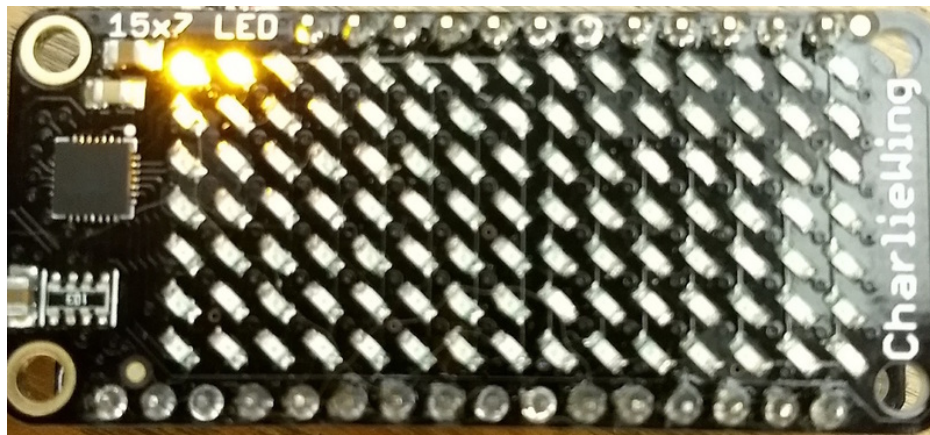
For example to set pixel 0, 0 to full brightness run:

```
display.pixel(0, 0, 255)
```
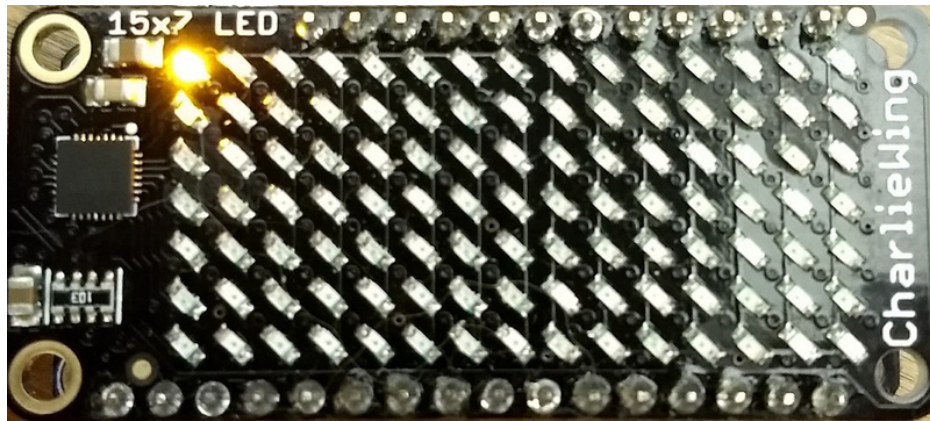

Or to set the pixel next to it horizontally to half brightness run:

```
display.pixel(1, 0, 127)
```


You can turn off individual pixels by setting them to an intensity of zero.

You can even make pixels blink! The board supports a fixed blink rate that you set using the `blink` function. This function takes in the number of milliseconds to use for the blink rate (but internally it can only blink in 270ms increments so you might not get an exact match). For example to blink pixels about once every half second call:

```
display.blink(500)
```

You'll notice nothing actually changes on the board. This is because in addition to intensity each LED pixel has a blink state which can be enabled and disabled. The `fill` command can actually set all pixels and turn them on to blink:

```
display.fill(127, blink=True)
```

You can turn off the blinking by setting `blink=False`.

The `pixel` command supports the blink parameter too! You can turn on and off blinking pixel by pixel as needed. For

example to turn on blinking for pixel `0, 0` :

```
display.pixel(0, 0, 127, blink=True)
```

Currently the Charlieplex module is very simple and only exposes pixel set commands. In the future more advanced graphics commands like line drawing, text display, etc. might be implemented but for now you'll need to manipulate the pixels yourself.

Finally the display supports holding up to 8 frames of pixel data. Each frame contains an entire matrix of LED pixel state (intensity, blinking, etc.) and by default the module starts you on frame 0. You can change to start displaying and drawing on another frame by calling `frame` which takes these parameters:

- **Frame number** - This is the frame number to make the active frame for display or drawing. There are 8 frames total, `0` through `7` .
- `show` - An optional boolean that defaults to `True` and specifies if the frame should be immediately displayed (`True`) or just made active so that pixel and fill commands draw on it but it's not yet shown.

For example to clear frame `1` and draw a few pixels on it, then display it you can run:

```
display.frame(1, show=False)
display.fill(0)
display.pixel(0, 0, 255)
display.pixel(1, 1, 255)
display.pixel(2, 2, 255)
display.frame(1)  # show=True is the default, the frame will be displayed!
```
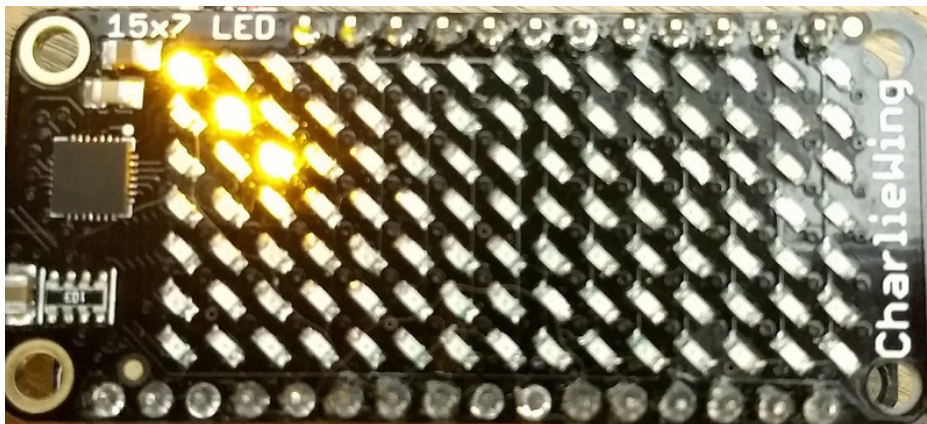

Notice how the first call switches to make frame 1 the active frame but doesn't display it because show is set to false. Then the frame pixel data is changed with fill and pixel commands, and finally the frame is shown by calling frame again but letting the default show = True be used so the frame is displayed.

Using frames you can build simple animations by drawing each frame and swapping between them over time!

That's all there is to the basic Charlieplex driver module usage!

Full Example Code

```
import board
import busio
import adafruit_is31fl3731

i2c = busio.I2C(board.SCL, board.SDA)

# initialize display using Feather CharlieWing LED 15 x 7
display = adafruit_is31fl3731.CharlieWing(i2c)

# uncomment next line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
#display = adafruit_is31fl3731.Matrix(i2c)

# uncomment next line if you are using Adafruit 16x8 Charlieplexed Bonnet
#display = adafruit_is31fl3731.CharlieBonnet(i2c)

# draw a box on the display
# first draw the top and bottom edges
for x in range(display.width):
    display.pixel(x, 0, 50)
    display.pixel(x, display.height - 1, 50)
# now draw the left and right edges
for y in range(display.height):
    display.pixel(0, y, 50)
    display.pixel(display.width - 1, y, 50)
```

Text Scrolling Example

NOTE: When running this example on Raspberry Pi, you must have the font8x5.bin file found here (https://adafru.it/Edh) in the same directory as the program!

```
import board
import busio
import adafruit_framebuf
import adafruit_is31fl3731


i2c = busio.I2C(board.SCL, board.SDA)

# initial display using Feather CharlieWing LED 15 x 7
#display = adafruit_is31fl3731.CharlieWing(i2c)
# uncomment line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
#display = adafruit_is31fl3731.Matrix(i2c)
# uncomment line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
display = adafruit_is31fl3731.CharlieBonnet(i2c)

text_to_show = "Adafruit!!"

# Create a framebuffer for our display
buf = bytearray(32)  # 2 bytes tall x 16 wide = 32 bytes (9 bits is 2 bytes)
fb = adafruit_framebuf.FrameBuffer(buf, display.width, display.height, adafruit_framebuf.MVLSB)


frame = 0 # start with frame 0
while True:
    for i in range(len(text_to_show) * 9):
        fb.fill(0)
        fb.text(text_to_show, -i + display.width, 0, color=1)

        # to improve the display flicker we can use two frame
        # fill the next frame with scrolling text, then
        # show it.
        display.frame(frame, show=False)
        # turn all LEDs off
        display.fill(0)
        for x in range(display.width):
            # using the FrameBuffer text result
            bite = buf[x]
            for y in range(display.height):
                bit = 1 << y & bite
                # if bit > 0 then set the pixel brightness
                if bit:
                    display.pixel(x, y, 50)

        # now that the frame is filled, show it.
        display.frame(frame, show=True)
        frame = 0 if frame else 1
```
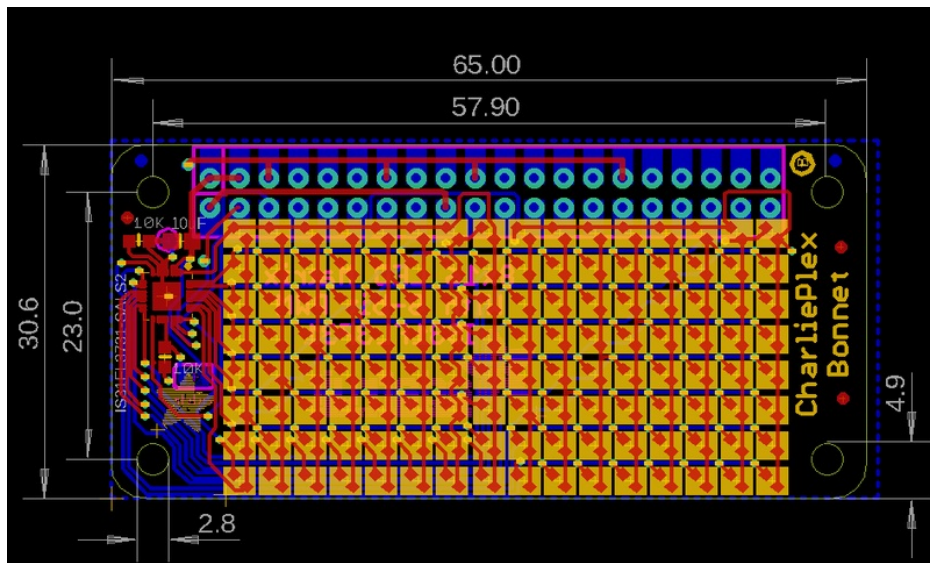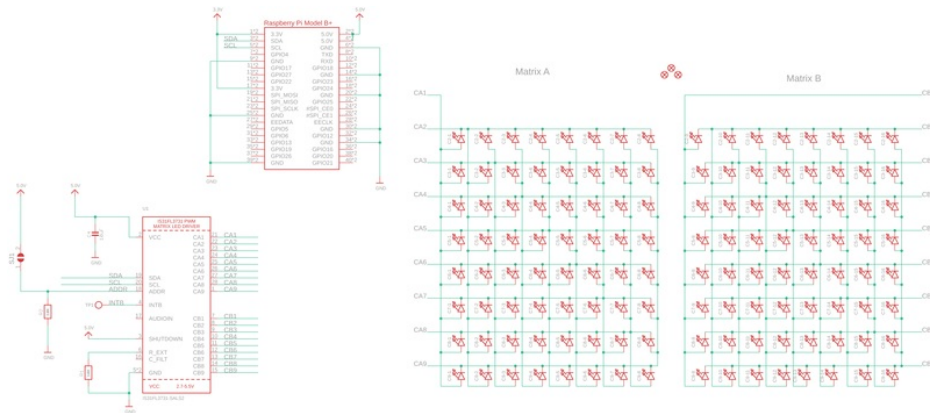
Python Docs

Python Docs (https://adafru.it/C55)

Downloads

Files

Schematic and Fab Print

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Adafruit](#):
  [4119](#) [4121](#)