1 CircuitPython
  1.1 Get CircuitPython .................................................. 3
  1.2 Documentation ..................................................... 4
  1.3 Code Search ......................................................... 4
  1.4 Contributing ........................................................ 4
  1.5 Branding ............................................................. 4
  1.6 Differences from MicroPython .................................... 5
    1.6.1 Behavior ......................................................... 5
    1.6.2 API ............................................................... 6
    1.6.3 Modules ........................................................ 6
  1.7 Project Structure .................................................. 6
    1.7.1 Core ............................................................ 6
    1.7.2 Ports ........................................................... 7
    1.7.3 Boards .......................................................... 7
  1.8 Full Table of Contents ............................................ 7
    1.8.1 Core Modules .................................................. 7
    1.8.2 Supported Ports ............................................... 316
    1.8.3 Troubleshooting ............................................... 326
    1.8.4 Adafruit CircuitPython Libraries ............................ 328
    1.8.5 CircuitPython Library Bundles ............................... 328
    1.8.6 Workflows ..................................................... 328
    1.8.7 Environment Variables ........................................ 338
    1.8.8 Design Guide .................................................. 340
    1.8.9 Architecture ................................................... 354
    1.8.10 Porting ......................................................... 354
    1.8.11 Adding *io support to other ports ......................... 356
    1.8.12 MicroPython libraries ....................................... 358
    1.8.13 Glossary ...................................................... 388
    1.8.14 CircuitPython ................................................ 390
    1.8.15 Contributing .................................................. 394
    1.8.16 Building CircuitPython ..................................... 395
    1.8.17 Building ......................................................... 396
    1.8.18 Testing ........................................................ 397
    1.8.19 Debugging ..................................................... 397
    1.8.20 Code Quality Checks ......................................... 397
    1.8.21 Adafruit Community Code of Conduct ........................ 398
    1.8.22 MicroPython & CircuitPython License ...................... 400
    1.8.23 WebUSB Serial Support ....................................... 400

2 Indices and tables .................................................. 403
<table>
<thead>
<tr>
<th>Python Module Index</th>
<th>405</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>407</td>
</tr>
</tbody>
</table>
Welcome to the API reference documentation for Adafruit CircuitPython. This contains low-level API reference docs which may link out to separate “getting started” guides. Adafruit has many excellent tutorials available through the Adafruit Learning System.
CircuitPython

CircuitPython is a beginner friendly, open source version of Python for tiny, inexpensive computers called microcontrollers. Microcontrollers are the brains of many electronics including a wide variety of development boards used to build hobby projects and prototypes. CircuitPython in electronics is one of the best ways to learn to code because it connects code to reality. Simply install CircuitPython on a supported USB board usually via drag and drop and then edit a code.py file on the CIRCUITPY drive. The code will automatically reload. No software installs are needed besides a text editor (we recommend Mu for beginners.)

Starting with CircuitPython 7.0.0, some boards may only be connectable over Bluetooth Low Energy (BLE). Those boards provide serial and file access over BLE instead of USB using open protocols. (Some boards may use both USB and BLE.) BLE access can be done from a variety of apps including code.circuitpython.org.

CircuitPython features unified Python core APIs and a growing list of 300+ device libraries and drivers that work with it. These libraries also work on single board computers with regular Python via the Adafruit Blinka Library.

CircuitPython is based on MicroPython. See below for differences. Most, but not all, CircuitPython development is sponsored by Adafruit and is available on their educational development boards. Please support both MicroPython and Adafruit.

1.1 Get CircuitPython

Official binaries for all supported boards are available through circuitpython.org/downloads. The site includes stable, unstable and continuous builds. Full release notes are available through GitHub releases as well.
1.2 Documentation

Guides and videos are available through the Adafruit Learning System under the CircuitPython category. An API reference is also available on Read the Docs. A collection of awesome resources can be found at Awesome CircuitPython.

Specifically useful documentation when starting out:

- Welcome to CircuitPython
- CircuitPython Essentials
- Example Code

1.3 Code Search

GitHub doesn't currently support code search on forks. Therefore, CircuitPython doesn't have code search through GitHub because it is a fork of MicroPython. Luckily, SourceGraph has free code search for public repos like CircuitPython. So, visit sourcegraph.com/github.com/adafruit/circuitpython to search the CircuitPython codebase online.

1.4 Contributing

See CONTRIBUTING.md for full guidelines but please be aware that by contributing to this project you are agreeing to the Code of Conduct. Contributors who follow the Code of Conduct are welcome to submit pull requests and they will be promptly reviewed by project admins. Please join the Discord too.

1.5 Branding

While we are happy to see CircuitPython forked and modified, we’d appreciate it if forked releases not use the name “CircuitPython” or the Blinka logo. “CircuitPython” means something special to us and those who learn about it. As a result, we’d like to make sure products referring to it meet a common set of requirements.

If you’d like to use the term “CircuitPython” and Blinka for your product here is what we ask:

- Your product is supported by the primary “adafruit/circuitpython” repo. This way we can update any custom code as we update the CircuitPython internals.
- Your product is listed on circuitpython.org (source here). This is to ensure that a user of your product can always download the latest version of CircuitPython from the standard place.
- Your product supports at least one standard “Workflow” for serial and file access:
  - With a user accessible USB plug which appears as a CIRCUITPY drive when plugged in.
  - With file and serial access over Bluetooth Low Energy using the BLE Workflow.
  - With file access over WiFi using the WiFi Workflow with serial access over USB and/or WebSocket.
- Boards that do not support the USB Workflow should be clearly marked.

If you choose not to meet these requirements, then we ask you call your version of CircuitPython something else (for example, SuperDuperPython) and not use the Blinka logo. You can say it is “CircuitPython-compatible” if most CircuitPython drivers will work with it.
1.6 Differences from MicroPython

CircuitPython:

- Supports native USB on most boards and BLE otherwise, allowing file editing without special tools.
- Floats (aka decimals) are enabled for all builds.
- Error messages are translated into 10+ languages.
- Concurrency within Python is not well supported. Interrupts and threading are disabled. async/await keywords are available on some boards for cooperative multitasking. Some concurrency is achieved with native modules for tasks that require it such as audio file playback.

1.6.1 Behavior

- The order that files are run and the state that is shared between them. CircuitPython’s goal is to clarify the role of each file and make each file independent from each other.
  - `boot.py` runs only once on start up before workflows are initialized. This lays the ground work for configuring USB at startup rather than it being fixed. Since serial is not available, output is written to `boot_out.txt`.
  - `code.py` (or `main.py`) is run after every reload until it finishes or is interrupted. After it is done running, the vm and hardware is reinitialized. This means you cannot read state from `code.py` in the REPL anymore, as the REPL is a fresh vm. CircuitPython’s goal for this change includes reducing confusion about pins and memory being used.
  - After the main code is finished the REPL can be entered by pressing any key.
  - Autoreload state will be maintained across reload.

- Adds a safe mode that does not run user code after a hard crash or brown out. This makes it possible to fix code that causes nasty crashes by making it available through mass storage after the crash. A reset (the button) is needed after it’s fixed to get back into normal mode.

- Safe mode may be handled programmatically by providing a `safemode.py`. `safemode.py` is run if the board has reset due to entering safe mode, unless the safe mode initiated by the user by pressing button(s). USB is not available so nothing can be printed. `safemode.py` can determine why the safe mode occurred using `supervisor.runtime.safe_mode_reason`, and take appropriate action. For instance, if a hard crash occurred, `safemode.py` may do a `microcontroller.reset()` to automatically restart despite the crash. If the battery is low, but is being charged, `safemode.py` may put the board in deep sleep for a while. Or it may simply reset, and have `code.py` check the voltage and do the sleep.

- RGB status LED indicating CircuitPython state. - One green flash - code completed without error. - Two red flashes - code ended due to an exception. - Three yellow flashes - safe mode. May be due to CircuitPython internal error.

- Re-runs `code.py` or other main file after file system writes by a workflow. (Disable with `supervisor.disable_autoreload()`)

- Autoreload is disabled while the REPL is active.

- `code.py` may also be named `code.txt`, `main.py`, or `main.txt`.
- `boot.py` may also be named `boot.txt`.
- `safemode.py` may also be named `safemode.txt`.
1.6.2 API

- Unified hardware APIs. Documented on ReadTheDocs.
- API docs are Python stubs within the C files in shared-bindings.
- No machine API.

1.6.3 Modules

- No module aliasing. (uos and utime are not available as os and time respectively.) Instead os, time, and random are CPython compatible.
- New storage module which manages file system mounts. (Functionality from uos in MicroPython.)
- Modules with a CPython counterpart, such as time, os and random, are strict subsets of their CPython version. Therefore, code from CircuitPython is runnable on CPython but not necessarily the reverse.
- tick count is available as time.monotonic()

1.7 Project Structure

Here is an overview of the top-level source code directories.

1.7.1 Core

The core code of MicroPython is shared amongst ports including CircuitPython:

- docs High level user documentation in Sphinx reStructuredText format.
- drivers External device drivers written in Python.
- examples A few example Python scripts.
- extmod Shared C code used in multiple ports’ modules.
- lib Shared core C code including externally developed libraries such as FATFS.
- logo The CircuitPython logo.
- mpy-cross A cross compiler that converts Python files to byte code prior to being run in MicroPython. Useful for reducing library size.
- py Core Python implementation, including compiler, runtime, and core library.
- shared-bindings Shared definition of Python modules, their docs and backing C APIs. Ports must implement the C API to support the corresponding module.
- shared-module Shared implementation of Python modules that may be based on common-hal.
- tests Test framework and test scripts.
- tools Various tools, including the pyboard.py module.
1.7.2 Ports

Ports include the code unique to a microcontroller line.

<table>
<thead>
<tr>
<th>Supported</th>
<th>Support status</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmel-samd</td>
<td>SAMD21 stable</td>
</tr>
<tr>
<td>cxd56</td>
<td>stable</td>
</tr>
<tr>
<td>espressif</td>
<td>ESP32 beta</td>
</tr>
<tr>
<td>litex</td>
<td>alpha</td>
</tr>
<tr>
<td>mimxrt10xx</td>
<td>alpha</td>
</tr>
<tr>
<td>nrf</td>
<td>stable</td>
</tr>
<tr>
<td>raspberrypi</td>
<td>stable</td>
</tr>
<tr>
<td>efr32</td>
<td>alpha</td>
</tr>
<tr>
<td>stm</td>
<td>F4 stable</td>
</tr>
<tr>
<td>unix</td>
<td>alpha</td>
</tr>
</tbody>
</table>

- **stable** Highly unlikely to have bugs or missing functionality.
- **beta** Being actively improved but may be missing functionality and have bugs.
- **alpha** Will have bugs and missing functionality.

1.7.3 Boards

- Each port has a boards directory containing boards which belong to a specific microcontroller line.
- A list of native modules supported by a particular board can be found here.

1.8 Full Table of Contents

1.8.1 Core Modules

These core modules are intended on being consistent across ports and boards. A module may not exist on a port/board if no underlying hardware support is present or if flash space is limited. For example, a microcontroller without analog features will not have `analogio`. See the Module Support Matrix - Which Modules Are Available on Which Boards page for a list of modules supported on each board.

Module Support Matrix - Which Modules Are Available on Which Boards

The following table lists the available built-in modules for each CircuitPython capable board, as well as each frozen module included on it.

You can filter this list by typing one or more module names or partial names into the search box. Only those boards that provide those modules will be listed. To exclude boards that provide a module, type a “-” in front of the module name. You can also type a regular expression as a filter.
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>01Space 0.42 OLED</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espifd, espnow, fontio, framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2pio, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssd, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>0xCB Helios</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppypo, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>42. Keebs Fwood</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppypo, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>8086 Commander</td>
<td>adafruit_bus_device, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, neopixel_write, nvm, onewireio, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>@sarftashIRtty</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

Frozen Modules: adafruit_hid, adafruit_ssdcard

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit BLM Badge</td>
<td>analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, rainbowio, random, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid</td>
</tr>
<tr>
<td>Adafruit Circuit Playground Bluefruit</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharppixel, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Circuit Playground Express 4-H</td>
<td>adafruit_bus_device, adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, bitbangio, board, builtins, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_circuitplayground, adafruit_hid, adafruit_lis3dh, adafruit_thermistor, neopixel</td>
</tr>
<tr>
<td>Adafruit Circuit-Playground Express</td>
<td>adafruit_bus_device, adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, bitbangio, board, builtins, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_circuitplayground, adafruit_hid, adafruit_lis3dh, adafruit_thermistor, neopixel</td>
</tr>
<tr>
<td>Adafruit Circuit-Playground Express with Crickit libraries</td>
<td>adafruit_bus_device, adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, errno, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_circuitplayground, adafruit_crickit, adafruit_lis3dh, adafruit_motor, adafruit_seesaw, adafruit_thermistor, neopixel</td>
</tr>
<tr>
<td>Adafruit Circuit-Playground Express with displayio</td>
<td>analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_circuitplayground, adafruit_lis3dh, adafruit_thermistor, neopixel</td>
</tr>
</tbody>
</table>

Table 1 – continued from previous page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit CLUE nRF52840 Express</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmappools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit EdgeBadge</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>_stage</code>, adafruit_bus_device, adafruit_pixelbuf, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, binascii, bitbangio, bitmappools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, freqencyio, getpass, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2i0, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Feather Bluefruit Sense</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmappools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Feather ESP32 V2</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmappools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espult, fontio, framebufferio, freqencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2i0, pulseio, pwmio, qrio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

*Frozen Modules:* stage, ugame

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td>ESP32-S2 Reverse</td>
<td></td>
</tr>
<tr>
<td>TFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdif, espnw, espul, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublb, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td>ESP32-S2 TFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdif, espnw, espul, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublb, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td>ESP32-S3 Reverse</td>
<td></td>
</tr>
<tr>
<td>TFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdif, espnw, espul, fontio, framebufferio, frequencyio, getpass, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublb, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td>ESP32-S3 TFT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdif, espnw, espul, fontio, framebufferio, frequencyio, getpass, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublb, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>continues on next page</strong></td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Adafruit Feather ESP32S2</td>
<td><code>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpeledisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Adafruit Feather ESP32S3 4MB Flash 2MB PSRAM</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpeledisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Adafruit Feather ESP32S3 No PSRAM</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpeledisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Adafruit HUZ-ZAH32</td>
<td><code>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpeledisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather M0</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit Feather M0 Basic</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit Feather M0 Express</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit Feather M0 Express with Crickit libraries</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>ADAFRUIT CRICKIT</td>
<td>Frozen Modules: adafruit_cricket, adafruit_motor, adafruit_seesaw, neopixel</td>
</tr>
<tr>
<td>Adafruit Feather M0 RFM69</td>
<td>adafruit_bus_device, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, random, storage, struct, supervisor, sys, time, usb_cdc</td>
</tr>
<tr>
<td>Adafruit Feather M0 RFM9x</td>
<td>adafruit_bus_device, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, storage, struct, supervisor, sys, time, usb_cdc</td>
</tr>
<tr>
<td>Adafruit Feather M4 CAN</td>
<td>_asynio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmapped, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2c_targets, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather M4 Express</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, freqencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharldisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Feather MIMXRT10</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharldisplay, structure, supervisio, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Feather nRF52840 Express</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Feather RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather</td>
<td><code>.asncio</code>, <code>.bleio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>picoio</code>, <code>pmio</code>, <code>qpio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharphelper</code>, <code>store</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>RP2040 CAN</td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td><code>.asncio</code>, <code>.bleio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>picoio</code>, <code>pmio</code>, <code>qpio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharphelper</code>, <code>store</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>RP2040 DVI</td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td><code>.asncio</code>, <code>.bleio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>picoio</code>, <code>pmio</code>, <code>qpio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharphelper</code>, <code>store</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>RP2040 Prop-Maker</td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td><code>.asncio</code>, <code>.bleio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>picoio</code>, <code>pmio</code>, <code>qpio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharphelper</code>, <code>store</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>RP2040 RFM</td>
<td></td>
</tr>
<tr>
<td>Adafruit Feather</td>
<td><code>.asncio</code>, <code>.bleio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>picoio</code>, <code>pmio</code>, <code>qpio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharphelper</code>, <code>store</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Feather</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopypio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>RP2040 Scorpio</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopypio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>ThinkInk</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopypio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>USB Host</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.</code>, <code>pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopypio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>STM32F405 Express</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopypio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit FunHouse</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2cdevice, i2cdevice, imagedisplay, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.Keypad, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Adafruit Gemma M0</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit Gemma M0 PyCon 2018</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit Grand Central M4 Express</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiioio, audiomixer, audiop3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, frequencyio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.Keypad, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, sdioio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Hallowing M4 Express</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiioio, audiomixer, audiop3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, frequencyio, getpass, gifio, i2cdevice, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.Keypad, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Adafruit HUZ-ZAH32 Breakout</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Adafruit ItsyBitsy M0 Express</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleledisplay, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit ItsyBitsy M4 Express</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit ItsyBitsy nRF52840 Express</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ItyBitsy</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>RP2040</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>KB2040</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>LED Glasses</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Driver nRF52840</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
## Modules Available

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit MagTag</td>
<td><code>.asyncio</code>, <code>.pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio</code></td>
</tr>
<tr>
<td></td>
<td><code>UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espifd</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pmwio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectortio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> <code>adafruit_display_text</code>, <code>adafruit_fakerequests</code>, <code>adafruit_lis3dh</code>, <code>adafruit_portalbase</code>, <code>adafruit_requests</code>, <code>neopixel</code></td>
</tr>
</tbody>
</table>

| Adafruit Matrix Portal M4    | `.asyncio`, `.bleio`, `.pixelmap`, `adafruit_bus_device`, `adafruit_pixelbuf`, `alarm`, `analogio`, `array`, `atexit`, `audiobusio`, `audiocore`, `audiomixer`, `audiomp3`, `binascii`, `bitbangio`, `bitmaptools`, `board`, `builtins`, `builtins.pow3`, `busio`, `busio.SPI`, `busio` |
|                              | `UART`, `canio`, `collections`, `countio`, `digitalio`, `displayio`, `errno`, `fontio`, `framebufferio`, `frequencyio`, `getpass`, `gifio`, `i2ctarget`, `io`, `json`, `keypad`, `keypad.KeyMatrix`, `keypad.Keys`, `keypad.ShiftRegisterKeys`, `math`, `microcontroller`, `msgpack`, `neopixel_write`, `nvram`, `onewireio`, `os`, `os.getenv`, `ps2io`, `pulseio`, `pmwio`, `rainbowio`, `random`, `re`, `rgbmatrix`, `rotaryio`, `rtc`, `sdcardio`, `select`, `storage`, `struct`, `supervisor`, `synthio`, `sys`, `terminalio`, `time`, `touchio`, `traceback`, `ulab`, `usb_cdc`, `usb_hid`, `usb_midi`, `vectortio`, `watchdog`, `zlib` |
|                              | **Frozen Modules:** `adafruit_esp32spi`, `adafruit_portalbase`, `adafruit_requests`, `neopixel` |

| Adafruit Matrix Portal S3    | `.asyncio`, `.bleio`, `.pixelmap`, `adafruit_bus_device`, `adafruit_pixelbuf`, `aesio`, `alarm`, `analogbufio`, `analogio`, `array`, `atexit`, `audiobusio`, `audiocore`, `audiomixer`, `binascii`, `bitbangio`, `bitmaptools`, `board`, `builtins`, `builtins.pow3`, `busio`, `busio.SPI`, `busio` |
|                              | `UART`, `canio`, `collections`, `countio`, `digitalio`, `displayio`, `dualbank`, `errno`, `espifd`, `espnow`, `espulp`, `fontio`, `framebufferio`, `frequencyio`, `getpass`, `gifio`, `hashlib`, `i2ctarget`, `io`, `ipaddress`, `json`, `keypad`, `keypad.KeyMatrix`, `keypad.Keys`, `keypad.ShiftRegisterKeys`, `math`, `mdns`, `memorymap`, `microcontroller`, `msgpack`, `neopixel_write`, `nvram`, `onewireio`, `os`, `os.getenv`, `ps2io`, `pulseio`, `pmwio`, `rainbowio`, `random`, `re`, `rgbmatrix`, `rotaryio`, `rtc`, `sdcardio`, `select`, `sharpdisplay`, `socketpool`, `ssl`, `storage`, `struct`, `supervisor`, `synthio`, `sys`, `terminalio`, `time`, `touchio`, `traceback`, `ulab`, `usb_cdc`, `usb_hid`, `usb_midi`, `vectortio`, `watchdog`, `wifi`, `zlib` |
|                              | **Frozen Modules:** `adafruit_esp32spi`, `adafruit_portalbase`, `adafruit_requests`, `neopixel` |

continues on next page
## Modules Available

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Metro ESP32S2</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>escamera</code>, <code>espidf</code>, <code>espnw</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.</code>, shiftRegisterKeys, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulses</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Adafruit Metro ESP32S3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>escamera</code>, <code>espidf</code>, <code>espnw</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.</code>, shiftRegisterKeys, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulses</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Adafruit Metro M0 Express</td>
<td><code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audioio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>paralleldisplay</code>, <code>pulses</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code></td>
</tr>
<tr>
<td>Adafruit Metro M4 Airlift Lite</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audioio</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>flopppio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.</code>, shiftRegisterKeys, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulses</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code></td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Adafruit Metro M4</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf,aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bthangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ernno, floppio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, jswn, keypad, keypad, KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samb, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Metro nRF52840 Express</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmiio, binascii, bthangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ernno, fontio, framebufferio, getpass, gifio, io, jswn, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Metro RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmiio, binascii, bthangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ernno, floppio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, jswn, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit Monster M4SK</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bthangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ernno, floppio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, jswn, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samb, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit NeoKey Trinkey M0</td>
<td>adafruit_pixelbuf, array, board, builtins, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, rainbowio, random, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_hid, neopixel</td>
</tr>
<tr>
<td>Adafruit NeoPixel Trinkey M0</td>
<td>adafruit_pixelbuf, array, board, builtins, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, rainbowio, random, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_hid, neopixel</td>
</tr>
<tr>
<td>Adafruit ProxLight Trinkey M0</td>
<td>adafruit_bus_device, adafruit_pixelbuf, array, board, builtins, busio, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, rainbowio, random, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>adafruit_apds9960, neopixel</td>
</tr>
<tr>
<td>Adafruit PyBadge</td>
<td>__asyncio, __bleio, __bitmapmap, __stage, adafruit_bus_device, adafruit_pixelbuf, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, frequencyio, getpass, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>stage, ugame</td>
</tr>
<tr>
<td>Adafruit PyGamer</td>
<td>__asyncio, __bleio, __bitmapmap, __stage, adafruit_bus_device, adafruit_pixelbuf, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, frequencyio, getpass, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td>stage, ugame</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit PyPortal</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audioio, audiop3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erfmt, floppyst, fontio, framebufferio, frequencyio, getpass, gifio, i2cdevice, io, json, keypad, keypad. AdafruitDisplay, keypad.Keys, keypad.SkipRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallel邑display, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit PyPortal Pynt</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audioio, audiop3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erfmt, floppyst, fontio, framebufferio, frequencyio, getpass, gifio, i2cdevice, io, json, keypad, keypad. AdafruitDisplay, keypad.Keys, keypad.SkipRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallel邑display, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit PyPortal Titano</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audioio, audiop3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erfmt, floppyst, fontio, framebufferio, frequencyio, getpass, gifio, i2cdevice, io, json, keypad, keypad. AdafruitDisplay, keypad.Keys, keypad.SkipRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallel邑display, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit PyRuler</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Adafruit QT Py ESP32 PICO</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>esp32</code>, <code>esp32c3</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2cdevice</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ubab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Adafruit QT Py ESP32-S3 2MB Flash 2MB PSRAM</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>esp32</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2cdevice</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ubab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Adafruit QT Py ESP32-S3 no psram</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>esp32</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2cdevice</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ubab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Adafruit QT Py ESP32C3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>esp32</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2cdevice</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ubab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
## Modules Available

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espifd, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Adafruit</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audiioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Adafruit</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Adafruit</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

*continues on next page*
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adafruit Rotary</td>
<td><code>adafruit_pixelbuf</code>, <code>array</code>, <code>board</code>, <code>builtinss</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>os</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>time</code>, <code>touchio</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>Trinkey M0</td>
<td><strong>Frozen Modules:</strong> <code>adafruit_hid</code>, <code>neopixel</code></td>
</tr>
<tr>
<td>Adafruit Slide</td>
<td><code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>board</code>, <code>builtinss</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>os</code>, <code>rainbowio</code>, <code>random</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>time</code>, <code>touchio</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>Trinkey M0</td>
<td><strong>Frozen Modules:</strong> <code>adafruit_hid</code>, <code>adafruit_simplemath</code>, <code>neopixel</code></td>
</tr>
<tr>
<td>Adafruit Trellis M4</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiocore</code>, <code>audioio</code>, <code>audiomixer</code>, <code>audiomixer</code>, <code>nvm</code>, <code>os</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Express</td>
<td><strong>Frozen Modules:</strong> <code>adafruit_hid</code>, <code>neopixel</code></td>
</tr>
<tr>
<td>Adafruit Trinket M0</td>
<td><code>analogio</code>, <code>array</code>, <code>board</code>, <code>builtinss</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>os</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppyscript</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>AITHinker ESP32-C3S_Kit</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtinss</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espdf</code>, <code>espnow</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2c</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad</code>. <code>KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralelldisplay</code>, <code>ps2</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppyscript</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>AITHinker ESP32-C3S_Kit_2N</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtinss</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espdf</code>, <code>espnow</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2c</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad</code>. <code>KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppyscript</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>AloriumTech</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ermon, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2c sentinel, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ARAM-CON</td>
<td>_asyncio, _bleio, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ermon, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ARAM-CON2</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, ermon, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Arduino</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Arduino MKR1300</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Nano BLE 33</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio, binascii, bitbangio, bitmaps, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, json, keypad, KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Arduino Nano IoT 33</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Arduino Nano ESP32</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaps, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espid, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2c, ipaddress, json, keypad, KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Arduino Nano RP2040 Connect</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio, binascii, bitbangio, bitmaps, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floopyio, fontio, framebufferio, getpass, gifio, i2c, imagecapture, io, json, keypad, KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Arduino Zero</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artisense Reference RD00</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, erro, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>i2c, target, i0, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, scardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

| Atelier-DuMaker nRF52840 Breakout | __asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwio, binascii, bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erro, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c, target, i0, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, scardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib |

| AT-MegaZero ESP32-S2         | __asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erro, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, scardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib |

| BastBLE                     | __asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwio, binascii, bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erro, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, scardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib |

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>BastWiFi</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnw, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdsn, memorymap, microcontroller, msgpack, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgmbmatrix, rotaryio, rtc, sdcardin, select, shardisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>BBQ20KBD</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppgio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmbmatrix, rotaryio, rp2pio, rtc, sdcardin, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>BDMI-CRO VINA-D21</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>BDMI-CRO VINA-D51</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppgio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgmbmatrix, rotaryio, rtc, samd, sdcardin, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
<th>Frozen Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bee-Data-Logger</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.SPI</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espidf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pwoio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgmbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
<td><code>adafruit_bus_device</code>, <code>adafruit_ds3231</code>, <code>adafruit_register</code>, <code>adafruit_sdcard</code>, <code>neopixel</code></td>
</tr>
<tr>
<td>Bee-Motion-S3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.SPI</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espidf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pwoio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgmbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
<td><code>adafruit_bus_device</code>, <code>adafruit_ds3231</code>, <code>adafruit_register</code>, <code>adafruit_sdcard</code>, <code>neopixel</code></td>
</tr>
<tr>
<td>Bee-S3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.SPI</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espidf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pwoio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgmbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
<td><code>adafruit_bus_device</code>, <code>adafruit_ds3231</code>, <code>adafruit_register</code>, <code>adafruit_sdcard</code>, <code>neopixel</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE-SS Multi Sensor</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>BLING! Multi Sensor</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap._stage</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmapools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>doublebank</code>, <code>errno</code>, <code>espcamera</code>, <code>espidf</code>, <code>espnw</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2cimport</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>BlizzardS3</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap._stage</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmapools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>doublebank</code>, <code>errno</code>, <code>espcamera</code>, <code>espidf</code>, <code>espnw</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2cimport</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td><code>adafruit_sdcard</code>, <code>neopixel</code></td>
</tr>
<tr>
<td><strong>Blizzards</strong></td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap._stage</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmapools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>doublebank</code>, <code>errno</code>, <code>espcamera</code>, <code>espidf</code>, <code>espnw</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2cimport</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td><code>neopixel</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOK</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiop3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2ctarget</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Frozen Modules:</td>
<td>neopixel</td>
</tr>
<tr>
<td>BlueMicro833</td>
<td><code>_bleio</code>, <code>adafruit_pixelbuf</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiopwmio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>os</code>, <code>pulsio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>BlueMicro840</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiop3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code></td>
</tr>
<tr>
<td>BPI-Bit-S2</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>esp32</code>, <code>espnow</code>, <code>espuart</code>, <code>frequency</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulsio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharppdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Frozen Modules:</td>
<td>neopixel</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPI-Leaf-S3</td>
<td><code>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espfd, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code>&lt;br&gt;<strong>Frozen Modules:</strong> neopixel</td>
</tr>
<tr>
<td>BPI-Pico-W3</td>
<td><code>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espfd, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code>&lt;br&gt;<strong>Frozen Modules:</strong> neopixel</td>
</tr>
<tr>
<td>Capable Robot Programmable USB Hub</td>
<td><code>alarm, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, nvm, onewireio, os, ps2io, pwmio, rainbowio, random, rtc, samd, storage, struct, supervisor, sys, time, usb_cdc, watchdog</code></td>
</tr>
<tr>
<td>Cedar Grove String-Car M0 Express</td>
<td><code>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid</code></td>
</tr>
<tr>
<td>Challenger NB RP2040 WiFi</td>
<td><code>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c_target, imagemap, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenger</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevio, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rbgmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Challenger</td>
<td>Frozen Modules: adafruit_rfm9x</td>
</tr>
<tr>
<td>Challenger</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevio, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rbgmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Challenger</td>
<td>Frozen Modules: adafruit_sdcard</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenger RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>SubGHz</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins,</td>
</tr>
<tr>
<td></td>
<td>pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture,</td>
</tr>
<tr>
<td></td>
<td>io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_rfm69</td>
</tr>
<tr>
<td>Challenger RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>WiFi</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins,</td>
</tr>
<tr>
<td></td>
<td>pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture,</td>
</tr>
<tr>
<td></td>
<td>io, json, keypad, keypad.KeyMatrix, keypad(Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Challenger RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>WiFi/BLE</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins,</td>
</tr>
<tr>
<td></td>
<td>pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture,</td>
</tr>
<tr>
<td></td>
<td>io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Circuit Playground</td>
<td>adafruit_bus_device, adafruit_pixelbuf, analogio, array, audiobusio, audiocore,</td>
</tr>
<tr>
<td>Express</td>
<td>audiocore, audioio, bitbangio, board, builtins, busio, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>Digi-Key</td>
<td>countio, digitalio, errno, math, microcontroller, neopixel_write, nvm, onewireio, os,</td>
</tr>
<tr>
<td>PyCon 2019</td>
<td>pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys,</td>
</tr>
<tr>
<td></td>
<td>time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_circuitplayground, adafruit_hid, adafruit_lis3dh, adafruit_thermistor, neopixel</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit-Brains</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nmv, onewireio, os, paralleledisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Basic</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nmv, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Deluxe</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopw3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nmv, onewireio, os, os.getenv, paralleledisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>COSMO-Pico</td>
<td>anaalogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nmv, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>CP Sapling M0</td>
<td>adafruit_pixelbuf, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nmv, onewireio, os, paralleledisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>CP Sapling M0 w/ SPI Flash</td>
<td>adafruit_pixelbuf, analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nmv, onewireio, os, paralleledisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP32-M4</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppypio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>CRCibernetica</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufo, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espifd, espnw, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>IdeaBoard</td>
<td>Frozen Modules: adafruit_motor, adafruit_requests, neopixel, simpleio</td>
</tr>
<tr>
<td>CrumpS2</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufo, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espifd, espnw, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Cytron Maker</td>
<td>Frozen Modules: neopixel</td>
</tr>
<tr>
<td>Feather AIoT S3</td>
<td>continues on next page</td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cytron Maker Nano RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules:** neopixel, simpleio

| Cytron Maker Pi RP2040   | _asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib |

**Frozen Modules:** adafruit_motor, neopixel, simpleio

| Datanoise PicoADK       | _asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib |

**Frozen Modules:** neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deneyap Kart</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binacci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qpio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Deneyap Kart 1A</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binacci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qpio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Deneyap Kart 1A v2</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binacci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Deneyap Kart G</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiocore, audiomixer, binacci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnow, fontio, framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deneyap Mini</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pmvio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpparallel, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Deneyap Mini v2</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pmvio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpparallel, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>DFRobot Beetle ESP32-C3</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnow, fontio, framebufferio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pmvio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpparallel, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Diodes Delight Piunora</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, sharpparallel, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, videocore, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dy-LoRa_USI</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, onewireio, os, pwmio, rainbowio, random, rotaryio, rtc, sdcardio, storage, struct, supervisior, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_rfm9x, neopixel</td>
</tr>
<tr>
<td>DynOSSAT-EDU-EPS</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rtc, storage, struct, supervisior, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>DynOSSAT-EDU-OBC</td>
<td>__asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2csource, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisior, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>E-Fidget</td>
<td>__asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2csource, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisior, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ELECFS-REAKS PICO:ED</td>
<td>__asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2csource, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisior, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_is31fl3731, adafruit_motor, adafruit_ticks, __asyncio, elecfreaks_music, neopixel, picoed</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronama minik</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins,</td>
</tr>
<tr>
<td></td>
<td>pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture,</td>
</tr>
<tr>
<td></td>
<td>io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi,</td>
</tr>
<tr>
<td></td>
<td>vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Electronic Cats Bast</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>Pro Mini M0</td>
<td>digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid,</td>
</tr>
<tr>
<td></td>
<td>usb_midi</td>
</tr>
<tr>
<td>Electronic Cats Cat-</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>WAN USBStick</td>
<td>digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid,</td>
</tr>
<tr>
<td></td>
<td>usb_midi</td>
</tr>
<tr>
<td>Electronic Cats Hunter</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins,</td>
</tr>
<tr>
<td>Cat NFC</td>
<td>busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio,</td>
</tr>
<tr>
<td></td>
<td>math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio,</td>
</tr>
<tr>
<td></td>
<td>pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time,</td>
</tr>
<tr>
<td></td>
<td>terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Electronic Cats NFC</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>Copy Cat</td>
<td>digitalio, math, microcontroller, neopixel_write, nvm, onewireio, os, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio,</td>
</tr>
<tr>
<td></td>
<td>usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>Electronut Labs Blip</td>
<td>_asyncio, _bleio, _pixelmap, _stage, adafruit_bus_device, adafruit_pixelbuf,</td>
</tr>
<tr>
<td></td>
<td>aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins, pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad,</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronut Lab Papyr</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Encoder-Pad RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Escornabot Makech</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid</td>
</tr>
<tr>
<td>ESP 12k NodeMCU</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, esp32camera, espdf, espnow, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32-Devkit V1</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio,</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io,</td>
</tr>
<tr>
<td></td>
<td>ipaddress, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys,</td>
</tr>
<tr>
<td></td>
<td>terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

| ESP32-C-DevKitM-1-N4 | _asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,         |
|                     | analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,|
|                     | bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio,|
|                     | UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,|
|                     | espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io,|
|                     | ipaddress, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys,|
|                     | math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,|
|                     | os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio,|
|                     | select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys,|
|                     | terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib        |

| ESP32-S2-DevKitC-1-N4 | _asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,         |
|                      | analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,|
|                      | bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio,|
|                      | UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,|
|                      | espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io,|
|                      | ipaddress, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys,|
|                      | math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,|
|                      | os, os.getenv, parallelledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix,|
|                      | rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,|
|                      | supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc,|
|                      | usb_hid, usb_midi, vectorio, watchdog, wifi, zlib                                 |

| ESP32-S2-DevKitC-1-N4R2 | _asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,         |
|                          | analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,|
|                          | bitbangio, bitmaptools, board, builtins, **builtins.pow3**, busio, busio.SPI, busio,|
|                          | UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,|
|                          | espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,|
|                          | i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad,|
|                          | ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,|
|                          | nvm, onewireio, os, os.getenv, parallelledisplay, ps2io, pulseio, pwmio, qrio,|
|                          | rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,|
|                          | socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio,|
|                          | traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib        |

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32-S2-DevKitC-1-N8R2</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, esp-camera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-Box-2.5</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-Box-Lite</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-DevKitC-1-N32R8</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, esp-camera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32-S3-</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>DevKitC-1N8</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasci,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2cdevice,</td>
</tr>
<tr>
<td></td>
<td>io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor,</td>
</tr>
<tr>
<td></td>
<td>synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi,</td>
</tr>
<tr>
<td></td>
<td>vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>DevKitC-1N8R2</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasci,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage,</td>
</tr>
<tr>
<td></td>
<td>struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab,</td>
</tr>
<tr>
<td></td>
<td>usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>DevKitC-1N8R8</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasci,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage,</td>
</tr>
<tr>
<td></td>
<td>struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab,</td>
</tr>
<tr>
<td></td>
<td>usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>DevKitM-1N8</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasci,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2cdevice,</td>
</tr>
<tr>
<td></td>
<td>io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc,</td>
</tr>
<tr>
<td></td>
<td>usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32-S3-EYE</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasici, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>ESP32-S3-USB-OTG-N8</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasici, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Espressif ESP32-EYE</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasici, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Espressif ESP32-LyraT</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasici, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Espruino</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, alarm,</td>
</tr>
<tr>
<td>Bangle.js 2</td>
<td>analogio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins,</td>
</tr>
<tr>
<td></td>
<td>builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix,</td>
</tr>
<tr>
<td></td>
<td>keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>nvm, os, os.getenv,paralleldisplay, pulseio, pwmio, random, re, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharppdisplay, storage, struct, supervisor, sys, terminalio, time, touchio,</td>
</tr>
<tr>
<td></td>
<td>traceback, ulab, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Espruino</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_pixelbuf, analogio, array, atexit,</td>
</tr>
<tr>
<td>Pico</td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio,</td>
</tr>
<tr>
<td></td>
<td>busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio,</td>
</tr>
<tr>
<td></td>
<td>framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys,</td>
</tr>
<tr>
<td></td>
<td>keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharppdisplay, storage, struct, supervisor, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Espruino</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_pixelbuf, aesio, analogio, array,</td>
</tr>
<tr>
<td>Wifi</td>
<td>atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio,</td>
</tr>
<tr>
<td></td>
<td>framebufferio, getpass, gifio, io, json, keypad, Keypad.KeyMatrix, keypad.Keys,</td>
</tr>
<tr>
<td></td>
<td>Keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharppdisplay, storage, struct, supervisor, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Feather</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_pixelbuf, aesio, analogio, array,</td>
</tr>
<tr>
<td>MIMXRT10</td>
<td>atexit, audiobusio, audiocore, audiomixer, audiomp3, audiowmio, binasci,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio,</td>
</tr>
<tr>
<td></td>
<td>busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio,</td>
</tr>
<tr>
<td></td>
<td>framebufferio, getpass, gifio, io, json, keypad, Keypad.KeyMatrix, keypad.Keys,</td>
</tr>
<tr>
<td></td>
<td>Keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharppdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

Frozen Modules: adafruit esp32spi, adafruit_requests

continues on next page
Table 1 — continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatherS2</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio,audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>FeatherS2 Neo</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>FeatherS2 PreRelease</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio,audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c_target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

Frozen Modules: neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatherS3</td>
<td><code>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c2target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Fig Pi</td>
<td><code>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwpio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c2target, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Fluff M0</td>
<td><code>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</code></td>
</tr>
<tr>
<td>Fomu</td>
<td><code>asyncio, _pixelmap, adafruit_pixelbuf, aesio, array, atexit, binascii, builtins, builtins.pow3, busio.SPI, busio.UART, collections, digitalio, errno, getpass, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, os, os.getenv, rainbowio, random, re, select, storage, struct, supervisor, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, zlib</code></td>
</tr>
<tr>
<td>Franzinho WIFI</td>
<td><code>asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2c2target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
</tbody>
</table>

Continues on next page
### 1.8. Full Table of Contents

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Franzino WIFI w/Wrover</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c Fletcher, i2c target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, s cardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Gravitech Cucumber M</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2c Fletcher, i2c target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, s cardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Gravitech Cucumber MS</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2c Fletcher, i2c target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, s cardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Gravitech Cucumber R</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c Fletcher, i2c target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, s cardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gravitech Cucumber RS</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espncamera</code>, <code>espifd</code>, <code>espnow</code>, <code>espull</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2c</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad</code>, <code>ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulselio</code>, <code>pmwio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Hack Club Sprig</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audioio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppymio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2c</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad</code>, <code>ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulselio</code>, <code>pmwio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rpm</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Hacked Feather M0 Express with 8Mbyte SPI flash</td>
<td><code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audioio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>paralleldisplay</code>, <code>pulselio</code>, <code>pmwio</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>HalloWing M0 Express</td>
<td><code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>audiocore</code>, <code>audioio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>paralleldisplay</code>, <code>pulselio</code>, <code>pmwio</code>, <code>rainbowio</code>, <code>random</code>, <code>rotaryio</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>Hardkernel Odroid Go</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espncamera</code>, <code>espifd</code>, <code>espnow</code>, <code>espull</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2c</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad</code>, <code>ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvram</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulselio</code>, <code>pmwio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

**Frozen Modules:** `adafruit_lis3dh`, `neopixel`
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>HexKyS2</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>HII-BlueFi</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>HMI-DevKit-1.1</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Iki-galVita</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>iLabs</td>
<td>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>Challenger</td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3,</td>
</tr>
<tr>
<td>840</td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, contio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc,</td>
</tr>
<tr>
<td></td>
<td>usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>iMX RT</td>
<td>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>1020 EVK</td>
<td>analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio,</td>
</tr>
<tr>
<td></td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI,</td>
</tr>
<tr>
<td></td>
<td>busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select,</td>
</tr>
<tr>
<td></td>
<td>sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>iMX RT</td>
<td>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>1040 EVK</td>
<td>analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio,</td>
</tr>
<tr>
<td></td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI,</td>
</tr>
<tr>
<td></td>
<td>busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select,</td>
</tr>
<tr>
<td></td>
<td>sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>iMX RT</td>
<td>asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>1050 EVKB</td>
<td>analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio,</td>
</tr>
<tr>
<td></td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI,</td>
</tr>
<tr>
<td></td>
<td>busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select,</td>
</tr>
<tr>
<td></td>
<td>sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time,</td>
</tr>
<tr>
<td></td>
<td>touchio, traceback, ulab, usb_cdc, usb_hid, usb_host, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMX RT 1060 EVK</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pmvio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ublab, usb_cdc, usb_hid, usb_host, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>iMX 1060 EVKB</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pmvio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ublab, usb_cdc, usb_hid, usb_host, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>IMXRT1010 EVK</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pmvio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>IMXRT1015 EVK</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pmvio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules:** adafruit_esp32spi, adafruit_requests

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoTs2</td>
<td>_asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2cstation, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>J&amp;J Studios</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>datum-Distance</td>
<td></td>
</tr>
<tr>
<td>J&amp;J Studios</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>datum-IMU</td>
<td></td>
</tr>
<tr>
<td>J&amp;J Studios</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>datum-Light</td>
<td></td>
</tr>
<tr>
<td>J&amp;J Studios</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>datum-Weather</td>
<td></td>
</tr>
<tr>
<td>Kaluga 1</td>
<td>_asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2cstation, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>keithp.com snekboard</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>LILYGO T-DISPLAY</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmapios, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcario, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Lilygo T-watch 2020 V3</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapios, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rotaryio, rtc, sdcario, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>LILYGO TEMBED ESP32S3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapios, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcario, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>LILYGO TTG T-01C3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnw, fontio, framebufferio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>LILYGO TTG T-DISPLAY v1.1</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnw, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>LILYGO TTG T-OI PLUS</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnw, fontio, framebufferio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>LILYGO TTG T8 ESP32-S2</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espid, espnw, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>LILYGO</td>
<td>_asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>TTGO T8</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td>ESP32-S2 w/Display</td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART,</td>
</tr>
<tr>
<td></td>
<td>canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow,</td>
</tr>
<tr>
<td></td>
<td>espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io,</td>
</tr>
<tr>
<td></td>
<td>ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, oneewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage,</td>
</tr>
<tr>
<td></td>
<td>struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab,</td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Luatos Core-</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>ESP32C3</td>
<td>analogbufio, analogio, array, atexit, audiocore, audiomixer, binascii, bitbangio,</td>
</tr>
<tr>
<td></td>
<td>bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART,</td>
</tr>
<tr>
<td></td>
<td>canio, collections, digitalio, displayio, dualbank, erro, espidf, espnow, fontio,</td>
</tr>
<tr>
<td></td>
<td>framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.</td>
</tr>
<tr>
<td></td>
<td>KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller,</td>
</tr>
<tr>
<td></td>
<td>msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool,</td>
</tr>
<tr>
<td></td>
<td>ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>M5Stack Atom</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>Echo</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, erro, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget,</td>
</tr>
<tr>
<td></td>
<td>io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pvmio, rainbowio, random, re, rotaryio, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys,</td>
</tr>
<tr>
<td></td>
<td>terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>M5Stack Atom</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>Lite</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, erro, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget,</td>
</tr>
<tr>
<td></td>
<td>io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pvmio, rainbowio, random, re, rotaryio, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys,</td>
</tr>
<tr>
<td></td>
<td>terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>M5Stack Atom</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>Matrix</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, erro, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget,</td>
</tr>
<tr>
<td></td>
<td>io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, ps2io, pulseio, pvmio, rainbowio, random, re, rotaryio, rtc, sdcardio,</td>
</tr>
<tr>
<td></td>
<td>select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys,</td>
</tr>
<tr>
<td></td>
<td>terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>M5Stack Atom U</td>
<td><code>asynctio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtin</code>, <code>builtin.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio</code></td>
</tr>
<tr>
<td>M5Stack AtomS3 Lite</td>
<td><code>asynctio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtin</code>, <code>builtin.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio</code></td>
</tr>
<tr>
<td>M5Stack Core Basic</td>
<td><code>asynctio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtin</code>, <code>builtin.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio</code></td>
</tr>
<tr>
<td>M5Stack Core Fire</td>
<td><code>asynctio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtin</code>, <code>builtin.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>M5Stack Core2</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espif,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, vectorio,</td>
</tr>
<tr>
<td></td>
<td>watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_display_text, adafruit_fakerequests, adafruit_requests, neopixel</td>
</tr>
<tr>
<td>M5STACK STAMP-C3</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espif,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm,</td>
</tr>
<tr>
<td></td>
<td>onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor,</td>
</tr>
<tr>
<td></td>
<td>synthio, sys, terminalio, time, touchio, traceback, ubl, vectorio, watchdog, wifi,</td>
</tr>
<tr>
<td></td>
<td>zlib</td>
</tr>
<tr>
<td>M5Stack Stick C</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espif,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, vectorio,</td>
</tr>
<tr>
<td></td>
<td>watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>MagiClick S3 N4R2</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2cableio, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rtcio, shadepic, stringio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Maker badge by Czech maker</td>
<td><code>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2cableio, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDISPLAY, ps2io, pulseio, pwmio, rainbowio, random, re, rtcio, shadepic, stringio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Maker diary M60 Keyboard</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDISPLAY, pulseio, pwmio, rainbowio, random, re, rtcio, shadepic, stringio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Maker diary nRF52840 Connect Kit</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDISPLAY, pulseio, pwmio, rainbowio, random, re, rtcio, shadepic, stringio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
</tbody>
</table>

Frozen Modules: adafruit_bus_device, adafruit_display_text, adafruit_register, adafruit_requests, adafruit_ssd1680, adafruit_uc8151d, neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.2 Developer Kit</td>
<td>`_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiodpmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad.</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>nRF52840</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiodpmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad.</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>nRF52840 MDK</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiodpmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad.</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pitaya Go</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiodpmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad.</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDBT50Q-DB-40</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memmap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>MDBT50Q-RX Dongle</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memmap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Melopero Shake RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Meow Meow</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>
| MEOW-BIT | _asyncio, _pixelmap, _stage, adafruit_bus_device, adafruit_pixelbuf, analogio, array, atexit, audiocore, audiomp3, audiopwmio, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, usb_cdc, usb_hid, usb_midi, vectorio  
**Frozen Modules:** stage, ugame |

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metro</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiom3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>MIMXRT10</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, dualbank, errno, espdf, espnow, fontio, framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>micro:bit v2</td>
<td>_bleio, alarm, analogio, array, audiobusio, audiocore, audiomixer, audiopwmio, board, builtins, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>MicroDev microC3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espdf, espnow, fontio, framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad. KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>MicroDev microS2</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdbdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini SAM M4</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_dotstar</td>
</tr>
<tr>
<td>MixGo CE</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharldisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> Image, adafruit_framebuf, adafruit_irremote, adafruit_minimqtt, adafruit_requests, adafruit_rtttl, blynklib, blyntimer, button, infrared, irremote, led, matcher, matrix, mixgoce, mixpy, mmatrix, msa301, music, mxc6655xa, neopixel, pixels, sensor, touchpad</td>
</tr>
<tr>
<td>MOR-PHEANS MorphESP-240</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharldisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>nanoESP32-S2 w/Wrover</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>nanoESP32-S2 w/Wroom</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>NanoS3</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmapools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espncamera, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Frozen Modules:</td>
<td>neopixel</td>
</tr>
<tr>
<td>ndGarage[n°] Bit6: FeatherSnow v2</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>ndGarage[n°]</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neuron</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bithbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espdif, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>nice!nano</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiodpwmio, binascii, bithbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paraledispplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>NUCLEO F446RE</td>
<td><code>_asyncio, adafruit_bus_device, aesio, analogio, array, atexit, binascii, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, errno, getpass, math, microcontroller, msgpack, onewireio, os, os.getenv, pulseio, pwmio, random, re, rtc, select, storage, struct, supervisor, sys, time, traceback, usb_cdc, usb_hid, usb_midi, vectorio</code></td>
</tr>
<tr>
<td>NUCLEO STM32F746</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bithbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</code></td>
</tr>
<tr>
<td>NUCLEO STM32F767</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bithbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUCLEO STM32H743</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, select, sharpprogram, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>nullbits Bit-C PRO</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpprogram, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Oak Dev Tech BREAD204</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpprogram, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Oak Dev Tech Cast-Away RP2040</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpprogram, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
## Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oak Dev Tech PixelWing</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>ESP32S2</td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bithbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio,</td>
</tr>
<tr>
<td></td>
<td>traceback, ulab, usb_cdc, usb_hid, usb_midio, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Open Hardware Summit</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td>2020 Badge</td>
<td>alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bithbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.</td>
</tr>
<tr>
<td></td>
<td>Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio,</td>
</tr>
<tr>
<td></td>
<td>rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midio, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>OPENMV-H7 R1</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, array, atexit, binascii, bithbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midio, vectorio, zlib</td>
</tr>
<tr>
<td>Pajenicko PicoPad</td>
<td>_asyncio, _bleio, _pixelmap, _stage, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer,</td>
</tr>
<tr>
<td></td>
<td>audiomp3, audiopwmio, binascii, bithbangio, bitmaptools, bitops, board, builtins,</td>
</tr>
<tr>
<td></td>
<td><strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, cyw43,</td>
</tr>
<tr>
<td></td>
<td>digitalio, displayio, errno, floppyo, fontio, framebufferio, getpass, gifio,</td>
</tr>
<tr>
<td></td>
<td>hashlib, i2ctarget, imagecapture, io, ipaddress, json, keypad, keypad.KeyMatrix,</td>
</tr>
<tr>
<td></td>
<td>keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio,</td>
</tr>
<tr>
<td></td>
<td>os, os.getenv, paralleldisplay, picodvi, pulseio, pwmio, qrio, rainbowio, random,</td>
</tr>
<tr>
<td></td>
<td>re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, socketpool,</td>
</tr>
<tr>
<td></td>
<td>ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio,</td>
</tr>
<tr>
<td></td>
<td>traceback, ulab, usb_cdc, usb_hid, usb_midio, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> stage, ugame</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle Argon</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomixer, audiopwmio, analogio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Particle Boron</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomixer, audiopwmio, analogio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Particle Xenon</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomixer, audiopwmio, analogio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>PCA10056 nRF52840-DK</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomixer, audiopwmio, analogio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA10059 nRF52840 Dongle</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binscii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>PewPew 10.2 1.2 nRF52833 DK</td>
<td>_bleio, alarm, analogio, array, audiobusio, audiocore, audiomixer, audiopwmio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, onewireio, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid</td>
</tr>
<tr>
<td>Frozen Modules: pew</td>
<td></td>
</tr>
<tr>
<td>PewPew LCD</td>
<td>array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, fontio, microcontroller, os, pwmio, random, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc</td>
</tr>
<tr>
<td>Frozen Modules: pew</td>
<td></td>
</tr>
<tr>
<td>PewPew M4</td>
<td>_stage, array, audiocore, audioio, audiomixer, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, fontio, keypad, keypad.Keys, math, microcontroller, nvm, os, random, storage, struct, supervisor, sys, terminalio, time, usb_cdc, zlib</td>
</tr>
<tr>
<td>Frozen Modules: pew, stage, ugame</td>
<td></td>
</tr>
<tr>
<td>PicoPlanet</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>PillBug</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binscii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad. Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimoroni Badger 2040</td>
<td>_asyncio, __bleio, __bitmapmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiopmp, audiopwmi, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c_get, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usbhid, usbhid, usbhid, usbhid, vectortio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Badger 2040 W</td>
<td>_asyncio, __bleio, __bitmapmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiopmp, audiopwmi, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, cyw43, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, hashlib, i2c_get, imagecapture, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usbhid, usbhid, usbhid, vectortio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Pimoroni Inky Frame 5.7</td>
<td>_asyncio, __bleio, __bitmapmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiopmp, audiopwmi, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, cyw43, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, hashlib, i2c_get, imagecapture, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usbhid, usbhid, usbhid, vectortio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_register, pcfs063a</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> neopixel</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimoroni Interstate 75</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpprimary, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Keybow 2040</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpenprimary, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Motor 2040</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpenprimary, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni PGA2040</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpenprimary, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimoroni Pico LiPo (16MB)</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmiio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, picodvi, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubal, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Pimoroni Pico LiPo (4MB)</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmiio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubal, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
</tbody>
</table>

Frozen Modules: adafruit_sdcard

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimoroni PicoSystem</td>
<td>_asyncio, _bleio, _pixelmap, _stage, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c conglomerate, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Plasma 2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c conglomerate, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Plasma 2040W</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, cyw43, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, hashlib, i2c conglomerate, imagecapture, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

Frozen Modules: neopixel

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pimoroni Servo 2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagcaptcha, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Tiny 2040 (2MB)</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagcaptcha, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Pimoroni Tiny 2040 (8MB)</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagcaptcha, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ProS3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad. ShiftRegisterKeys, math, ndns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules:** neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>PYB LR Nano V2</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_hid, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Py-boardV1_1</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rgmatrix, rtc, sdcardio, sdioio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Py-Cubedv04</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, floppyio, frequencyio, getpass, i2cdevice, io, json, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_hid, usb_midi, watchdog, zlib</td>
</tr>
<tr>
<td>PyCubedv04 MRAM</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, floppyio, frequencyio, getpass, i2cdevice, io, json, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_hid, usb_midi, watchdog, zlib</td>
</tr>
<tr>
<td>Py-Cubedv05</td>
<td>__asyncio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, floppyio, frequencyio, getpass, i2cdevice, io, json, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_hid, usb_midi, watchdog, zlib</td>
</tr>
</tbody>
</table>

Frozen Modules: adafruit_register, neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyCubed05</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, floppyio, frequencyio, getpass, i2c targetType, io, json, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, synthio, sys, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, watchdog, zlib</td>
</tr>
<tr>
<td>MRAM</td>
<td><strong>Frozen Modules:</strong> adafruit_register, neopixel</td>
</tr>
<tr>
<td>PyKey 18</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmMatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardeploy, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Numpad</td>
<td><strong>Frozen Modules:</strong> adafruit_register, neopixel</td>
</tr>
<tr>
<td>PyKey 44</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmMatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardeploy, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Ergo</td>
<td><strong>Frozen Modules:</strong> adafruit_register, neopixel</td>
</tr>
<tr>
<td>PyKey 60</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cTarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, parallelDisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmMatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardeploy, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyKey TKL 87</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cpi, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpsilonio, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Raspberry Pi 4B</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, sharpsilonio, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, videocore, zlib</td>
</tr>
<tr>
<td>Raspberry Pi Compute Module 4</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, sharpsilonio, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, videocore, zlib</td>
</tr>
<tr>
<td>Raspberry Pi Compute Module 4 IO Board</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, sharpsilonio, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, videocore, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi Pico</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2cdevice, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, picodvi, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Raspberry Pi Pico W</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, cyw43, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, hashlib, i2cdevice, imagecapture, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleledisplay, picodvi, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Raspberry Pi Zero</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, shardisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, videocore, zlib</td>
</tr>
<tr>
<td>Raspberry Pi Zero 2W</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, rainbowio, random, re, rtc, sdcardio, sdioio, select, shardisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, videocore, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi Zero W</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>array</code>, <code>atexit</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio_UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rtc</code>, <code>sdcardio</code>, <code>sdiio</code>, <code>select</code>, <code>sharidisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>videocore</code>, <code>zlib</code></td>
</tr>
<tr>
<td>Robo HAT MM1 M4</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analoggbufio</code>, <code>analoggio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>binascii</code>, <code>bitbangio</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio_UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>errno</code>, <code>floppyio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2c_target</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> <code>neopixel</code></td>
</tr>
<tr>
<td>RP2.65-F</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analoggbufio</code>, <code>analoggio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiompi3</code>, <code>audiopwmiio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio_UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2c_target</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharidisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> <code>adafruit_hid</code>, <code>adafruit_register</code>, <code>neopixel</code>, <code>stamp_carrier_board</code>, <code>stamp_round_carrier_board</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2Mini</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdif, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td>Frozen Modules: neopixel</td>
</tr>
<tr>
<td>S2Pico</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espdif, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td>Frozen Modules: neopixel</td>
</tr>
<tr>
<td>SAM E54 Xplained Pro</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, canio, collections, countio, digitalio, displayio, errno, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, sdioio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td>Frozen Modules: neopixel</td>
</tr>
<tr>
<td>SAM32v26</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio. UART, collections, countio, digitalio, displayio, errno, floppio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td></td>
<td>Frozen Modules: neopixel</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saola 1 w/Wroom</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, eromo, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keypad, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Saola 1 w/Wrover</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, eromo, espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keypad, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Seeed Studio XIAO ESP32C3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, eromo, espidf, espnow, fontio, framebufferio, getpass, hashlib, i2ctarget, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keypad, keypad.ShiftRegisterKeys, math, mdns, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Seeed XIAO nRF52840 Sense</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, eromo, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keypad, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_mid, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Board</td>
<td>Modules Available</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Seeeduino Wio</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>Terminal</td>
<td>analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI,</td>
</tr>
<tr>
<td></td>
<td>busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio,</td>
</tr>
<tr>
<td></td>
<td>framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.</td>
</tr>
<tr>
<td></td>
<td>KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack,</td>
</tr>
<tr>
<td></td>
<td>neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio,</td>
</tr>
<tr>
<td></td>
<td>pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select,</td>
</tr>
<tr>
<td></td>
<td>sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback,</td>
</tr>
<tr>
<td></td>
<td>ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Seeeduino</td>
<td>analogio, array, board, builtins, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>XIAO</td>
<td>digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid,</td>
</tr>
<tr>
<td></td>
<td>usb_midi</td>
</tr>
<tr>
<td>Seeeduino</td>
<td>analogio, array, board, builtins, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>XIAO KB</td>
<td>digitalio, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules</strong>: adafruit_hid, neopixel</td>
</tr>
<tr>
<td>Seeeduino</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td>XIAO RP2040</td>
<td>analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.</td>
</tr>
<tr>
<td></td>
<td>pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio,</td>
</tr>
<tr>
<td></td>
<td>errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture,</td>
</tr>
<tr>
<td></td>
<td>io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>senseBox</td>
<td>analogio, array, board, builtins, busio.SPI, busio.UART, collections,</td>
</tr>
<tr>
<td>MCU</td>
<td>digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid,</td>
</tr>
<tr>
<td></td>
<td>usb_midi</td>
</tr>
<tr>
<td>Serpente</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio,</td>
</tr>
<tr>
<td></td>
<td>math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio,</td>
</tr>
<tr>
<td></td>
<td>pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio,</td>
</tr>
<tr>
<td></td>
<td>time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>
### Modules Available

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
<th>Frozen Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiLabs xG24 Dev Kit</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>binascii</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>pmnio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
<td></td>
</tr>
<tr>
<td>SiLabs xG24 Explorer Kit</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>binascii</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>pmnio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>traceback</code>, <code>ulab</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
<td></td>
</tr>
<tr>
<td>Silicogniion LLC M4-Shim</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>binascii</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cformat</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad</code>. <code>KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pseuio</code>, <code>pmnio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
<td></td>
</tr>
<tr>
<td>Silicogniion LLC RP2040-Shim</td>
<td><code>asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomixed</code>, <code>audiomixed</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>bitops</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cformat</code>, <code>imagecapture</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pseuio</code>, <code>pseuio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rp2pio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
<td></td>
</tr>
<tr>
<td>Simmel</td>
<td><code>_bleio</code>, <code>aesio</code>, <code>analogio</code>, <code>array</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixed</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>math</code>, <code>microcontroller</code>, <code>os</code>, <code>pmnio</code>, <code>random</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>time</code>, <code>usb_hid</code>, <code>watchdog</code></td>
<td></td>
</tr>
</tbody>
</table>

*continues on next page*
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparkFun LU-MIDrive</td>
<td>adafruit_pixelbuf, analogio, array, board, builtins, busio, busio.SPI, busio. UART, collections, digitalio, displayio, erorro, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td></td>
<td><strong>Frozen Modules:</strong> adafruit_dotstar</td>
</tr>
<tr>
<td>SparkFun MicroMod nRF52840 Processor</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaiptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erorro, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>SparkFun MicroMod RP2040 Processor</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaiptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erorro, flopppio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>SparkFun MicroMod SAMD51 Processor</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaiptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erorro, flopppio, fontio, framebufferio, freencycio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, shardisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparkFun Pro Micro RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascrii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpprivdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubal, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>SparkFun Pro nRF52840 Mini</td>
<td>_bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascrii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, sharpprivdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubal, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>SparkFun Qwiic Micro</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, onewireio, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>SparkFun RedBoard Turbo</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, math, microcontroller, neopixel_write, nvm, onewireio, os, paralleldisplay, pulseio, pwmio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>SparkFun SAMD21 Dev Breakout</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
<tr>
<td>SparkFun SAMD21 Mini Breakout</td>
<td>analogio, array, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_midi</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparkFun STM32 MicroMod</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audiomixer, audiompi3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, erro, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, sdioio, select, sharpsdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>SparkFun Teensy MicroMod</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, erro, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharpsdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>SparkFun Thing Plus - RP2040</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, audiopwmio, binasci, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erro, floppiyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagemcapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qpio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpsdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>SparkFun Thing Plus - SAMD51</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiompi3, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erro, floppiyio, fontio, framebufferio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpsdiplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparkFun Thing Plus - STM32</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiocore, audiomixer, audiopwmio, builtins, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, sdioio, select, sharpsdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Sparkfun Thing Plus MGM240P</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiopwmio, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, math, microcontroller, msgpack, nvm, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpsdisplay, storage, struct, supervisor, sys, terminalio, time, traceback, ulab, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>splitkb.com Liatris</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiopwmio, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, floppio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpsdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>PRESENSE</td>
<td>_asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, binascii, bitbangio, board, builtins, busio, busio.SPI, busio.UART, camera, collections, digitalio, errno, getpass, gnss, io, json, math, microcontroller, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, sdioio, select, storage, struct, supervisor, sys, time, traceback, ulab, usb_cdc, zlib</td>
</tr>
<tr>
<td>Sprite_v2b</td>
<td>_asyncio, adafruit_bus_device, adafruit_pixelbuf, alarm, analogio, array, atexit, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, errno, freencyio, i2ctarget, io, json, math, microcontroller, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, random, re, rtc, samd, sdcardio, select, storage, struct, supervisor, sys, time, traceback, usb_cdc, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCI ISP1807</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busioUART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Micro Board</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busioUART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>ST STM32F746</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busioUART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>StackRduino M0 PRO</td>
<td>adafruit_pixelbuf, analogio, array, audiobusio, audiocore, audioio, board, builtins, busio, busio.SPI, busioUART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, paralleldisplay, pulseio, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ublab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>stm32f411ceblackpill</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busioUART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

continues on next page
## Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>stm32f11ce</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiocore, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>blackpill-with-flash</td>
<td></td>
</tr>
<tr>
<td>STM32F411</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, binasci, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>STM32F412</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiocore, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>STM32F4_D</td>
<td>`_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiocore, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, sdcardio, sdioio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
<tr>
<td>Swan R5</td>
<td>`_asyncio, aesio, alarm, analogio, array, atexit, audibusio, audiocore, audiomp3, audiopwmio, binasci, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rtc, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubl, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>takayoshiotake Octave RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiorecore, audiomixer, audiomp3,</td>
</tr>
<tr>
<td></td>
<td>audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3,</td>
</tr>
<tr>
<td></td>
<td>busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno,</td>
</tr>
<tr>
<td></td>
<td>floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io,</td>
</tr>
<tr>
<td></td>
<td>json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math,</td>
</tr>
<tr>
<td></td>
<td>microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv,</td>
</tr>
<tr>
<td></td>
<td>paralleldisplay, pulseio, pwmio, qrio, rainboxio, random, re, rgbmatrix, rotaryio,</td>
</tr>
<tr>
<td></td>
<td>rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio,</td>
</tr>
<tr>
<td></td>
<td>sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi,</td>
</tr>
<tr>
<td></td>
<td>vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Targett Module Clip w/Wroom</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiorecore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, haslib, i2ctarget, io,</td>
</tr>
<tr>
<td></td>
<td>iopaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys,</td>
</tr>
<tr>
<td></td>
<td>math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os,</td>
</tr>
<tr>
<td></td>
<td>os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainboxio, random, re, rgbmatrix,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc,</td>
</tr>
<tr>
<td></td>
<td>usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Targett Module Clip w/Wrover</td>
<td>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiorecore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espidf,</td>
</tr>
<tr>
<td></td>
<td>espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, haslib, i2ctarget, io, iopaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, qrio, rainboxio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td>Teensy 4.0</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio,</td>
</tr>
<tr>
<td></td>
<td>analogio, array, atexit, audiobusio, audiorecore, audiomixer, audiomp3, audiopwmio,</td>
</tr>
<tr>
<td></td>
<td>binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI,</td>
</tr>
<tr>
<td></td>
<td>busio.UART, collections, digitalio, displayio, errno, fontio, framebufferio,</td>
</tr>
<tr>
<td></td>
<td>getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pwmio, rainboxio, random, re, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teensy 4.1</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erumno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, onewireio, os, os.getenv, pwmio, rainbowio, random, re, rotaryio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb, usb_cdc, usb_hid, usb_host, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Teknikio Bluebird</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erumno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>TG-Boards’ Datalore IP M4</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, flopppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2ctarget, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharldisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>TG-Watch</td>
<td>__asyncio, __bleio, __pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, erumno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharldisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules**: adafruit_ble, adafruit_ble_apple_notification_center, adafruit_display_shapes, adafruit_display_text, adafruit_drv2605, adafruit_ds3231, adafruit_focaltouch, adafruit_lcd709203f, adafruit_lsm6ds, adafruit_progressbar, adafruit_register, adafruit_st7789

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Open Book Feather</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audioio</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>floppyio</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>i2cdevice</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulsio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>samd</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>THUNDERPACK_v11</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiocore</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>pulsio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>THUNDERPACK_v12</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiocore</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>pulsio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
<tr>
<td>TinkeringTech Scout-Makes Azul</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>audiomp3</code>, <code>audiopwmio</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.SPI</code>, <code>busio.UART</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>framebufferio</code>, <code>getpass</code>, <code>gifio</code>, <code>io</code>, <code>json</code>, <code>keypad</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>pulsio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyPICO</td>
<td>`_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td>Nano</td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, gpio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio,</td>
</tr>
<tr>
<td></td>
<td>watchdog, wifi, zlib</td>
</tr>
<tr>
<td>TinyS2</td>
<td>`_asyncio, _pixelmap, _stage, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm,</td>
</tr>
<tr>
<td></td>
<td>analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii,</td>
</tr>
<tr>
<td></td>
<td>bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.</td>
</tr>
<tr>
<td></td>
<td>UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espcamera,</td>
</tr>
<tr>
<td></td>
<td>espidf, espnow, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib,</td>
</tr>
<tr>
<td></td>
<td>ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write,</td>
</tr>
<tr>
<td></td>
<td>nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, gpio, rainbowio, random, re,</td>
</tr>
<tr>
<td></td>
<td>rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct,</td>
</tr>
<tr>
<td></td>
<td>supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio,</td>
</tr>
<tr>
<td></td>
<td>watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules:** neopixel

continues on next page
### Board Modules Available

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyS3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>_stage</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espcamera</code>, <code>espdf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td><code>neopixel</code></td>
</tr>
<tr>
<td>Tiny-WATCH S3</td>
<td><code>_asyncio</code>, <code>_bleio</code>, <code>_pixelmap</code>, <code>_stage</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espcamera</code>, <code>espdf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>qrio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
<tr>
<td><strong>Frozen Modules:</strong></td>
<td><code>adafruit_s7789</code></td>
</tr>
<tr>
<td>Trinket M0 Haxpress</td>
<td><code>adafruit_pixelbuf</code>, <code>analogio</code>, <code>array</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiolio</code>, <code>board</code>, <code>builtins</code>, <code>busio</code>, <code>busio.UART</code>, <code>collections</code>, <code>digitalio</code>, <code>displayio</code>, <code>errno</code>, <code>fontio</code>, <code>math</code>, <code>microcontroller</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>paralleldisplay</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code></td>
</tr>
<tr>
<td>TTGO T8 ESP32-S2-WROOM</td>
<td><code>_asyncio</code>, <code>_pixelmap</code>, <code>adafruit_bus_device</code>, <code>adafruit_pixelbuf</code>, <code>aesio</code>, <code>alarm</code>, <code>analogbufio</code>, <code>analogio</code>, <code>array</code>, <code>atexit</code>, <code>audiobusio</code>, <code>audiocore</code>, <code>audiomixer</code>, <code>binascii</code>, <code>bitbangio</code>, <code>bitmaptools</code>, <code>board</code>, <code>builtins</code>, <code>builtins.pow3</code>, <code>busio</code>, <code>busio.UART</code>, <code>canio</code>, <code>collections</code>, <code>countio</code>, <code>digitalio</code>, <code>displayio</code>, <code>dualbank</code>, <code>errno</code>, <code>espcamera</code>, <code>espdf</code>, <code>espnow</code>, <code>espulp</code>, <code>fontio</code>, <code>framebufferio</code>, <code>frequencyio</code>, <code>getpass</code>, <code>gifio</code>, <code>hashlib</code>, <code>i2ctarget</code>, <code>io</code>, <code>ipaddress</code>, <code>json</code>, <code>keypad</code>, <code>keypad.KeyMatrix</code>, <code>keypad.Keys</code>, <code>keypad.ShiftRegisterKeys</code>, <code>math</code>, <code>mdns</code>, <code>memorymap</code>, <code>microcontroller</code>, <code>msgpack</code>, <code>neopixel_write</code>, <code>nvm</code>, <code>onewireio</code>, <code>os</code>, <code>os.getenv</code>, <code>paralleldisplay</code>, <code>ps2io</code>, <code>pulseio</code>, <code>pwmio</code>, <code>rainbowio</code>, <code>random</code>, <code>re</code>, <code>rgbmatrix</code>, <code>rotaryio</code>, <code>rtc</code>, <code>sdcardio</code>, <code>select</code>, <code>sharpdisplay</code>, <code>socketpool</code>, <code>ssl</code>, <code>storage</code>, <code>struct</code>, <code>supervisor</code>, <code>synthio</code>, <code>sys</code>, <code>terminalio</code>, <code>time</code>, <code>touchio</code>, <code>traceback</code>, <code>ulab</code>, <code>usb_cdc</code>, <code>usb_hid</code>, <code>usb_midi</code>, <code>vectorio</code>, <code>watchdog</code>, <code>wifi</code>, <code>zlib</code></td>
</tr>
</tbody>
</table>

continues on next page
## Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART-Logger I</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, frequencyio, getpass, gifio, i2c, target, io, json, keypad, kayakpd. KeyMatrix, keypad.Key, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, samd, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, touchio, traceback, ubuf, usb_cdc, usb_hid, usb_hid, usb_media, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>uChip</td>
<td>analogio, array, busio, busio.SPI, busio.UART, collections, digitalio, math, microcontroller, neopixel_write, nvm, os, pwmio, rainbowio, random, rotaryio, rtc, storage, struct, supervisor, sys, time, touchio, usb_cdc, usb_hid, usb_media</td>
</tr>
<tr>
<td>uGame10</td>
<td>_stage, analogio, array, audiocore, audioio, board, builtins, busio, busio.SPI, busio.UART, collections, digitalio, displayio, errno, fontio, keypad, keypad.KeyMatrix, keypad.Key, keypad.ShiftRegisterKeys, math, microcontroller, nvm, onewireio, os, pwmio, rainbowio, random, rotaryio, storage, struct, supervisor, sys, terminalio, time, usb_cdc</td>
</tr>
<tr>
<td>VCC-GND Studio YD RP2040</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, audiomp3, audiowinio, binascii, bitbangio, bitmaptools, bitops, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2c, target, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Key, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubuf, usb_cdc, usb_hid, usb_hid, usb_media, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>VCC-GND YD ESP32-S3 (N16R8)</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogobufio, analogio, array, atexit, audiobusio, audiocore, audioio, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, <strong>builtins.pow3</strong>, busio, busio.SPI, busio.UART, canio, collections, countio, digitalio, displayio, dualbank, errno, espconnected, espfind, espnano, espulp, fontio, framebufferio, frequencyio, getpass, gifio, hashlib, i2c, target, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mDNS, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ubuf, usb_cdc, usb_hid, usb_hid, usb_media, vectorio, watchdog, wifi, zlib</td>
</tr>
</tbody>
</table>

**Frozen Modules:** neopixel

continues on next page
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC-GND YD ESP32-S3</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, dualbank, errno, espcamera, espidf, espnow, espulp, fontio, framebufferio, frequency, getpass, gifio, hashlib, i2circuit, iot, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, mdns, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.environ, ps2io, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, socketpool, ssl, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, wifi, zlib</td>
</tr>
<tr>
<td></td>
<td>Frozen Modules: neopixel</td>
</tr>
<tr>
<td>W5100S-EVB-Pico</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2circuit, imagemap, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.environ, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>W5500-EVB-Pico</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2circuit, imagemap, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.environ, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>WarmBit BluePixel nRF52840</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, memorymap, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.environ, paralleldisplay, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rotaryio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
<p>| Board                     | Modules Available                                                                                                                                 |</p>
<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waveshare RP2040-LCD-1.28</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, flopppio, fontio, framebufferio, getpass, gifio, i2cctarget, imagemapencode, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvms, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmbmatrix, rotaryio, rp2pio, rtc, sddcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ubab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Waveshare RP2040-TOUCH-LCD-1.28</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, flopppio, fontio, framebufferio, getpass, gifio, i2cctarget, imagemapencode, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvms, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmbmatrix, rotaryio, rp2pio, rtc, sddcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ubab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>Waveshare RP2040-Zero</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, flopppio, fontio, framebufferio, getpass, gifio, i2cctarget, imagemapencode, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvms, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmbmatrix, rotaryio, rp2pio, rtc, sddcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ubab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
<tr>
<td>WeAct Studio Pico</td>
<td>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins, pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, flopppio, fontio, framebufferio, getpass, gifio, i2cctarget, imagemapencode, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvms, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qrio, rainbowio, random, re, rgmbmatrix, rotaryio, rp2pio, rtc, sddcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchscreen, traceback, ubab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Board</th>
<th>Modules Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeAct Studio Pico 16MB</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, cgroup, debug, device, digitalio, displayio, errno, fio, fontio, framebufferio, getpass, gifio, i2cdevice, i2cinterface, imagecapture, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, pulseio, pwmio, qio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharptext, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib</code></td>
</tr>
<tr>
<td>Wemos Lolin C3 Mini &quot;</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnow, fontio, framebufferio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharptext, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Wemos Lolin C3 Pico &quot;</td>
<td><code>_asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogbufio, analogio, array, atexit, audiocore, audiomixer, binascii, bitbangio, bitmaptools, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, canio, collections, digitalio, displayio, dualbank, errno, espidf, espnow, fontio, framebufferio, getpass, hashlib, i2cdevice, io, ipaddress, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, ps2io, pulseio, pwmio, rainbowio, random, re, rgbmatrix, rtc, sdcardio, select, sharptext, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, vectorio, watchdog, wifi, zlib</code></td>
</tr>
<tr>
<td>Winterbloom Big Honking Button</td>
<td><code>adafruit_pixelbuf, analogio, array, audioio, board, builtins, busio.SPI, busio.UART, collections, digitalio, errno, math, microcontroller, neopixel_write, nvm, onewireio, os, pulseio, pwmio, rainbowio, random, rotaryio, storage, struct, supervisor, sys, time, usb_cdc</code></td>
</tr>
<tr>
<td>Winterbloom Sol</td>
<td><code>_asyncio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogio, array, atexit, binascii, bitbangio, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, errno, floppyio, freqencyio, getpass, io, json, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, pulseio, pwmio, rainbowio, random, re, rotaryio, rtc, samd, sdcardio, select, storage, struct, supervisor, sys, time, traceback, ulab, usb_cdc, usb_midi, watchdog, zlib</code></td>
</tr>
</tbody>
</table>

*Frozen Modules:* `neopixel`
The BLE adapter used to manage device discovery and connections. This object is the sole instance of _bleio.Adapter.

**exception _bleio.BluetoothError**

Bases: Exception

Catchall exception for Bluetooth related errors.
Initialize self. See help(type(self)) for accurate signature.

exception _bleio.RoleError
   Bases: BluetoothError
   Raised when a resource is used as the mismatched role. For example, if a local CCCD is attempted to be set but they can only be set when remote.
   Initialize self. See help(type(self)) for accurate signature.

exception _bleio.SecurityError
   Bases: BluetoothError
   Raised when a security related error occurs.
   Initialize self. See help(type(self)) for accurate signature.

_bleio.set_adapter(adapter: Adapter | None) → None
   Set the adapter to use for BLE, such as when using an HCI adapter. Raises NotImplementedException when the adapter is a singleton and cannot be set.

   The BLE Adapter object manages the discovery and connection to other nearby Bluetooth Low Energy devices. This part of the Bluetooth Low Energy Specification is known as Generic Access Profile (GAP).
   Discovery of other devices happens during a scanning process that listens for small packets of information, known as advertisements, that are broadcast unencrypted. The advertising packets have two different uses. The first is to broadcast a small piece of data to anyone who cares and nothing more. These are known as beacons. The second class of advertisement is to promote additional functionality available after the devices establish a connection. For example, a BLE heart rate monitor would advertise that it provides the standard BLE Heart Rate Service.
   The Adapter can do both parts of this process: it can scan for other device advertisements and it can advertise its own data. Furthermore, Adapters can accept incoming connections and also initiate connections.
   On boards that do not have native BLE, you can use an HCI co-processor. Pass the uart and pins used to communicate with the co-processor, such as an Adafruit AirLift. The co-processor must have been reset and put into BLE mode beforehand by the appropriate pin manipulation. The uart, rts, and cts objects are used to communicate with the HCI co-processor in HCI mode. The Adapter object is enabled during this call.
   After instantiating an Adapter, call _bleio.set_adapter() to set _bleio.adapter

On boards with native BLE, you cannot create an instance of _bleio.Adapter; this constructor will raise NotImplementedException. Use _bleio.adapter to access the sole instance already available.

enabled: bool
   State of the BLE adapter.

address: Address
   MAC address of the BLE adapter.

name: str
   name of the BLE adapter used once connected. The name is “CIRCUITPY” + the last four hex digits of adapter.address, to make it easy to distinguish multiple CircuitPython boards.

advertising: bool
   True when the adapter is currently advertising. (read-only)

connected: bool
   True when the adapter is connected to another device regardless of who initiated the connection. (read-only)
connections:  Tuple[Connection]

Tuple of active connections including those initiated through _bleio.Adapter.connect(). (read-only)

**start_advertising**(data: circuitpython_typing.ReadableBuffer, *, scan_response:

circuitpython_typing.ReadableBuffer | None = None, connectable: bool = True,
anonymous: bool = False, timeout: int = 0, interval: float = 0.1, tx_power: int = 0,
directed_to: Address | None = None) → None

Starts advertising until **stop_advertising** is called or if connectable, another device connects to us.

**Warning:** If data is longer than 31 bytes, then this will automatically advertise as an extended advertisement that older BLE 4.x clients won’t be able to scan for.

**Note:** If you set anonymous=True, then a timeout must be specified. If no timeout is specified, then the maximum allowed timeout will be selected automatically.

**Parameters**

- **data** *(ReadableBuffer)* – advertising data packet bytes
- **scan_response** *(ReadableBuffer)* – scan response data packet bytes. None if no scan response is needed.
- **connectable** *(bool)* – If True then other devices are allowed to connect to this peripheral.
- **anonymous** *(bool)* – If True then this device’s MAC address is randomized before advertising.
- **timeout** *(int)* – If set, we will only advertise for this many seconds. Zero means no timeout.
- **interval** *(float)* – advertising interval, in seconds
- **int** *(tx_power)* – transmitter power while advertising in dBm
- **Address** *(directed_to)* – peer to advertise directly to

**stop_advertising**() → None

Stop sending advertising packets.

**start_scan**(prefixes: circuitpython_typing.ReadableBuffer = b", *, buffer_size: int = 512, extended: bool = False, timeout: float | None = None, interval: float = 0.1, window: float = 0.1, minimum_rssi: int = -80, active: bool = True) → Iterable[ScanEntry]

Starts a BLE scan and returns an iterator of results. Advertisements and scan responses are filtered and returned separately.

**Parameters**

- **prefixes** *(ReadableBuffer)* – Sequence of byte string prefixes to filter advertising packets with. A packet without an advertising structure that matches one of the prefixes is ignored. Format is one byte for length (n) and n bytes of prefix and can be repeated.
- **buffer_size** *(int)* – the maximum number of advertising bytes to buffer.
- **extended** *(bool)* – When True, support extended advertising packets. Increasing buffer_size is recommended when this is set.
• **timeout** (*float*) – the scan timeout in seconds. If None or zero, will scan until `stop_scan` is called.

• **interval** (*float*) – the interval (in seconds) between the start of two consecutive scan windows. Must be in the range 0.0025 - 40.959375 seconds.

• **window** (*float*) – the duration (in seconds) to scan a single BLE channel. window must be <= interval.

• **minimum_rssi** (*int*) – the minimum rssi of entries to return.

• **active** (*bool*) – retrieve scan responses for scannable advertisements.

**Returns**  
an iterable of `_bleio.ScanEntry` objects

**Return type**  
iterable

`stop_scan()` → *None*  
Stop the current scan.

`connect(address: Address, *, timeout: float) → Connection`  
Attempts a connection to the device with the given address.

**Parameters**

• **address** (*Address*) – The address of the peripheral to connect to

• **timeout** (*float/int*) – Try to connect for timeout seconds.

`erase_bonding()` → *None*  
Erase all bonding information stored in flash memory.

**class** `_bleio.Address(address: circuitpython_typing.ReadableBuffer, address_type: int)`  
Encapsulates the address of a BLE device.

Create a new Address object encapsulating the address value. The value itself can be one of:

**Parameters**

• **address** (*ReadableBuffer*) – The address value to encapsulate. A buffer object (bytearray, bytes) of 6 bytes.

• **address_type** (*int*) – one of the integer values: `PUBLIC`, `RANDOM_STATIC`, `RANDOM_PRIVATE_RESOLVABLE`, or `RANDOM_PRIVATE_NON_RESOLVABLE`.

**address_bytes**: *bytes*  
The bytes that make up the device address (read-only).

Note that the bytes object returned is in little-endian order: The least significant byte is `address_bytes[0]`. So the address will appear to be reversed if you print the raw `bytes` object. If you print or use `str()` on the Attribute object itself, the address will be printed in the expected order. For example:

```python
>>> import _bleio
>>> _bleio.adapter.address
<Address c8:1d:f5:ed:a8:35>
>>> _bleio.adapter.address.address_bytes
b'\xa8\xed\xf5\xd1\xc8'
```
**type**: `int`

The address type (read-only).

One of the integer values: `PUBLIC`, `RANDOM_STATIC`, `RANDOM_PRIVATE_RESOLVABLE`, or `RANDOM_PRIVATE_NON_RESOLVABLE`.

**PUBLIC**: `int`

A publicly known address, with a company ID (high 24 bits) and company-assigned part (low 24 bits).

**RANDOM_STATIC**: `int`

A randomly generated address that does not change often. It may never change or may change after a power cycle.

**RANDOM_PRIVATE_RESOLVABLE**: `int`

An address that is usable when the peer knows the other device’s secret Identity Resolving Key (IRK).

**RANDOM_PRIVATE_NON_RESOLVABLE**: `int`

A randomly generated address that changes on every connection.

**__eq__**(other: `object`) → `bool`

Two Address objects are equal if their addresses and address types are equal.

**__hash__**() → `int`

Returns a hash for the Address data.

**class** `_bleio.Attribute`

Definitions associated with all BLE attributes: characteristics, descriptors, etc.

*Attribute* is, notionally, a superclass of *Characteristic* and *Descriptor*, but is not defined as a Python superclass of those classes.

You cannot create an instance of *Attribute*.

**NO_ACCESS**: `int`

security mode: access not allowed

**OPEN**: `int`

security_mode: no security (link is not encrypted)

**ENCRYPT_NO_MITM**: `int`

security_mode: unauthenticated encryption, without man-in-the-middle protection

**ENCRYPT_WITH_MITM**: `int`

security_mode: authenticated encryption, with man-in-the-middle protection

**LESC_ENCRYPT_WITH_MITM**: `int`

security_mode: LESC encryption, with man-in-the-middle protection

**SIGNED_NO_MITM**: `int`

security_mode: unauthenticated data signing, without man-in-the-middle protection

**SIGNED_WITH_MITM**: `int`

security_mode: authenticated data signing, without man-in-the-middle protection

**class** `_bleio.Characteristic`

Stores information about a BLE service characteristic and allows reading and writing of the characteristic’s value.

There is no regular constructor for a Characteristic. A new local Characteristic can be created and attached to a Service by calling *add_to_service()*.

Remote Characteristic objects are created by *Connection.discover_remote_services()* as part of remote Services.
properties: int
An int bitmask representing which properties are set, specified as bitwise or’ing of these possible values.
BROADCAST, INDICATE, NOTIFY, READ, WRITE, WRITE_NO_RESPONSE.

uuid: UUID | None
The UUID of this characteristic. (read-only)
Will be None if the 128-bit UUID for this characteristic is not known.

value: bytearray
The value of this characteristic.

max_length: int
The max length of this characteristic.

descriptors: Descriptor
A tuple of Descriptor objects related to this characteristic. (read-only)

service: Service
The Service this Characteristic is a part of.

BROADCAST: int
property: allowed in advertising packets

INDICATE: int
property: server will indicate to the client when the value is set and wait for a response

NOTIFY: int
property: server will notify the client when the value is set

READ: int
property: clients may read this characteristic

WRITE: int
property: clients may write this characteristic; a response will be sent back

WRITE_NO_RESPONSE: int
property: clients may write this characteristic; no response will be sent back

add_to_service(service: Service, uuid: UUID, *, properties: int = 0, read_perm: int = Attribute.OPEN,
write_perm: int = Attribute.OPEN, max_length: int = 20, fixed_length: bool = False,
initial_value: circuitpython_typing.ReadableBuffer | None = None, user_description: str | None = None) → Characteristic
Create a new Characteristic object, and add it to this Service.

Parameters

• service (Service) – The service that will provide this characteristic

• uuid (UUID) – The uuid of the characteristic

• properties (int) – The properties of the characteristic, specified as a bitmask of these values bitwise-or’d together: BROADCAST, INDICATE, NOTIFY, READ, WRITE, WRITE_NO_RESPONSE.

• read_perm (int) – Specifies whether the characteristic can be read by a client, and if so, which security mode is required. Must be one of the integer values Attribute.NO_ACCESS, Attribute.OPEN, Attribute.ENCRYPT_NO_MITM, Attribute.ENCRYPT_WITH_MITM, Attribute.LESC_ENCRYPT_WITH_MITM, Attribute.SIGNED_NO_MITM, or Attribute.SIGNED_WITH_MITM.
• **write_perm**(int) – Specifies whether the characteristic can be written by a client, and if so, which security mode is required. Values allowed are the same as **read_perm**.

• **max_length**(int) – Maximum length in bytes of the characteristic value. The maximum allowed is 512, or possibly 510 if **fixed_length** is False. The default, 20, is the maximum number of data bytes that fit in a single BLE 4.x ATT packet.

• **fixed_length**(bool) – True if the characteristic value is of fixed length.

• **initial_value**(ReadableObject) – The initial value for this characteristic. If not given, will be filled with zeros.

• **user_description**(str) – User friendly description of the characteristic

Returns
the new Characteristic.

**set_cccd**(*, notify: bool = False, indicate: bool = False) → None
Set the remote characteristic’s CCCD to enable or disable notification and indication.

Parameters

• **notify**(bool) – True if Characteristic should receive notifications of remote writes

• **indicate**(bool) – True if Characteristic should receive indications of remote writes

**class** _bleio.CharacteristicBuffer_(characteristic: Characteristic, *, timeout: int = 1, buffer_size: int = 64)
Accumulates a Characteristic’s incoming values in a FIFO buffer.

Monitor the given Characteristic. Each time a new value is written to the Characteristic add the newly-written bytes to a FIFO buffer.

Parameters

• **characteristic**(Characteristic) – The Characteristic to monitor. It may be a local Characteristic provided by a Peripheral Service, or a remote Characteristic in a remote Service that a Central has connected to.

• **timeout**(int) – The timeout in seconds to wait for the first character and between subsequent characters.

• **buffer_size**(int) – Size of ring buffer that stores incoming data coming from client. Must be $\geq$ 1.

**in_waiting**: int
The number of bytes in the input buffer, available to be read

**read**(nbytes: int | None = None) → bytes | None
Read characters. If nbytes is specified then read at most that many bytes. Otherwise, read everything that arrives until the connection times out. Providing the number of bytes expected is highly recommended because it will be faster.

Returns
Data read

Return type
bytes or None

**readinto**(buf: circuitpython_typing.WriteableBuffer) → int | None
Read bytes into the buf. Read at most len(buf) bytes.

Returns
number of bytes read and stored into buf
Return type

`int` or `None` (on a non-blocking error)

`readline()` → `bytes`

Read a line, ending in a newline character.

Returns
the line read

Return type

`int` or `None`

`reset_input_buffer()` → `None`

Discard any unread characters in the input buffer.

deinit() → `None`

Disable permanently.

class _bleio.Connection

A BLE connection to another device. Used to discover and interact with services on the other device.

Usage:

```python
import _bleio

my_entry = None
for entry in _bleio.adapter.scan(2.5):
    if entry.name is not None and entry.name == 'InterestingPeripheral':
        my_entry = entry
        break

if not my_entry:
    raise Exception("InterestingPeripheral' not found")

connection = _bleio.adapter.connect(my_entry.address, timeout=10)
```

Connections cannot be made directly. Instead, to initiate a connection use `Adapter.connect`. Connections may also be made when another device initiates a connection. To use a Connection created by a peer, read the `Adapter.connections` property.

- **connected**: `bool`
  True if connected to the remote peer.

- **paired**: `bool`
  True if paired to the remote peer.

- **connection_interval**: `float`
  Time between transmissions in milliseconds. Will be multiple of 1.25ms. Lower numbers increase speed and decrease latency but increase power consumption.

  When setting `connection_interval`, the peer may reject the new interval and `connection_interval` will then remain the same.

  Apple has additional guidelines that dictate should be a multiple of 15ms except if HID is available. When HID is available Apple devices may accept 11.25ms intervals.

- **max_packet_length**: `int`
  The maximum number of data bytes that can be sent in a single transmission, not including overhead bytes.
This is the maximum number of bytes that can be sent in a notification, which must be sent in a single packet. But for a regular characteristic read or write, may be sent in multiple packets, so this limit does not apply.

**disconnect()** → None

Disconnects from the remote peripheral. Does nothing if already disconnected.

**pair(*, bond: bool = True) → None**

Pair to the peer to improve security.

**discover_remote_services(service_uuids_whitelist: Iterable[UUID] | None = None) → Tuple[Service, Ellipsis]**

Do BLE discovery for all services or for the given service UUIDS, to find their handles and characteristics, and return the discovered services. *Connection.connected* must be True.

**Parameters**

- **service_uuids_whitelist** *(iterable)* – an iterable of *UUID* objects for the services provided by the peripheral that you want to use.

  The peripheral may provide more services, but services not listed are ignored and will not be returned.

  If service_uuids_whitelist is None, then all services will undergo discovery, which can be slow.

  If the service UUID is 128-bit, or its characteristic UUID’s are 128-bit, you you must have already created a *UUID* object for that UUID in order for the service or characteristic to be discovered. Creating the UUID causes the UUID to be registered for use. (This restriction may be lifted in the future.)

**Returns**

- A tuple of *_bleio.Service* objects provided by the remote peripheral.

---

**class _bleio.Descriptor**

Stores information about a BLE descriptor.

Descriptors are attached to BLE characteristics and provide contextual information about the characteristic.

There is no regular constructor for a Descriptor. A new local Descriptor can be created and attached to a Characteristic by calling **add_to_characteristic()**. Remote Descriptor objects are created by *Connection.discover_remote_services()* as part of remote Characteristics in the remote Services that are discovered.

**uuid**: *UUID*

The descriptor uuid. (read-only)

**characteristic**: *Characteristic*

The Characteristic this Descriptor is a part of.

**value**: *bytearray*

The value of this descriptor.

**classmethod add_to_characteristic(characteristic: Characteristic, uuid: UUID, *, read_perm: int = Attribute.OPEN, write_perm: int = Attribute.OPEN, max_length: int = 20, fixed_length: bool = False, initial_value: circuitpython_typing.ReadableBuffer = b") → Descriptor**

Create a new Descriptor object, and add it to this Service.

**Parameters**
• **characteristic** (**Characteristic**) – The characteristic that will hold this descriptor

• **uuid** (**UUID**) – The uuid of the descriptor

• **read_perm** (**int**) – Specifies whether the descriptor can be read by a client, and if so, which security mode is required. Must be one of the integer values `Attribute.NO_ACCESS`, `Attribute.OPEN`, `Attribute.ENCRYPT_NO_MITM`, `Attribute.ENCRYPT_WITH_MITM`, `Attribute.LESC_ENCRYPT_WITH_MITM`, `Attribute.SIGNED_NO_MITM`, or `Attribute.SIGNED_WITH_MITM`.

• **write_perm** (**int**) – Specifies whether the descriptor can be written by a client, and if so, which security mode is required. Values allowed are the same as `read_perm`.

• **max_length** (**int**) – Maximum length in bytes of the descriptor value. The maximum allowed is 512, or possibly 510 if `fixed_length` is False. The default, 20, is the maximum number of data bytes that fit in a single BLE 4.x ATT packet.

• **fixed_length** (**bool**) – True if the descriptor value is of fixed length.

• **initial_value** (**ReadableBuffer**) – The initial value for this descriptor.

Returns
the new Descriptor.

```python
class _bleio.PacketBuffer(characteristic: Characteristic, *, buffer_size: int, max_packet_size: int | None = None)
```

Accumulates a Characteristic’s incoming packets in a FIFO buffer and facilitates packet aware outgoing writes. A packet’s size is either the characteristic length or the maximum transmission unit (MTU) minus overhead, whichever is smaller. The MTU can change so check `incoming_packet_length` and `outgoing_packet_length` before creating a buffer to store data.

When we’re the server, we ignore all connections besides the first to subscribe to notifications.

Monitor the given Characteristic. Each time a new value is written to the Characteristic add the newly-written bytes to a FIFO buffer.

Monitor the given Characteristic. Each time a new value is written to the Characteristic add the newly-written packet of bytes to a FIFO buffer.

Parameters

• **characteristic** (**Characteristic**) – The Characteristic to monitor. It may be a local Characteristic provided by a Peripheral Service, or a remote Characteristic in a remote Service that a Central has connected to.

• **buffer_size** (**int**) – Size of ring buffer (in packets of the Characteristic’s maximum length) that stores incoming packets coming from the peer.

• **max_packet_size** (**int**) – Maximum size of packets. Overrides value from the characteristic. (Remote characteristics may not have the correct length.)

```python
incoming_packet_length: int
Maximum length in bytes of a packet we are reading.
```

```python
outgoing_packet_length: int
Maximum length in bytes of a packet we are writing.
```

```python
readinto(buf: circuitpython_typing.WriteableBuffer) → int
Reads a single BLE packet into the buf. Raises an exception if the next packet is longer than the given buffer. Use `incoming_packet_length` to read the maximum length of a single packet.
```

Returns
number of bytes read and stored into buf
Return type
  int
write\(data: \text{circuitpython\_typing.ReadableBuffer}, *, header: \text{bytes | None = None}) \rightarrow \text{int}\)
  Writes all bytes from data into the same outgoing packet. The bytes from header are included before data
  when the pending packet is currently empty.

  This does not block until the data is sent. It only blocks until the data is pending.

  Returns
  number of bytes written. May include header bytes when packet is empty.

Return type
  int
deinit() \rightarrow \text{None}
  Disable permanently.

class \_bleio.ScanEntry
Encapsulates information about a device that was received during scanning. It can be advertisement or scan
response data. This object may only be created by a \_bleio.ScanResults: it has no user-visible constructor.

Cannot be instantiated directly. Use \_bleio.Adapter.start_scan.

address: Address
  The address of the device (read-only), of type \_bleio.Address.

advertisement\_bytes: \text{bytes}
  All the advertisement data present in the packet, returned as a \text{bytes} object. (read-only)

rssi: \text{int}
  The signal strength of the device at the time of the scan, in integer dBm. (read-only)

connectable: \text{bool}
  True if the device can be connected to. (read-only)

scan\_response: \text{bool}
  True if the entry was a scan response. (read-only)

matches\(prefixes: \text{ScanEntry}, *, match\_all: \text{bool = True}) \rightarrow \text{bool}\)
  Returns True if the ScanEntry matches all prefixes when match\_all is True. This is stricter than the scan
  filtering which accepts any advertisements that match any of the prefixes where match\_all is False.

class \_bleio.ScanResults
Iterates over advertising data received while scanning. This object is always created by a \_bleio.Adapter: it
has no user-visible constructor.

Cannot be instantiated directly. Use \_bleio.Adapter.start_scan.

__iter__() \rightarrow \text{Iterator[\_bleio.ScanEntry]}
  Returns itself since it is the iterator.

__next__() \rightarrow \text{ScanEntry}
  Returns the next \_bleio.ScanEntry. Blocks if none have been received and scanning is still active.
  Raises StopIteration if scanning is finished and no other results are available.

class \_bleio.Service\(\text{uuid: UUID}, *, secondary: \text{bool = False})
Stores information about a BLE service and its characteristics.

Create a new Service identified by the specified UUID. It can be accessed by all connections. This is known as
a Service server. Client Service objects are created via Connection.discover\_remote\_services.
To mark the Service as secondary, pass `True` as `secondary`.

**Parameters**

- `uuid (UUID)` – The uuid of the service
- `secondary (bool)` – If the service is a secondary one

**Returns**

the new Service

**characteristics:** Tuple[Characteristic, Ellipsis]

A tuple of Characteristic designating the characteristics that are offered by this service. (read-only)

**remote:** bool

True if this is a service provided by a remote device. (read-only)

**secondary:** bool

True if this is a secondary service. (read-only)

**uuid:** UUID | None

The UUID of this service. (read-only)

Will be None if the 128-bit UUID for this service is not known.

class _bleio.UUID(value: int | circuitpython_typing.ReadableBuffer | str)

A 16-bit or 128-bit UUID. Can be used for services, characteristics, descriptors and more.

Create a new UUID or UUID object encapsulating the uuid value. The value can be one of:

- an `int` value in range 0 to 0xFFFF (Bluetooth SIG 16-bit UUID)
- a buffer object (bytearray, bytes) of 16 bytes in little-endian order (128-bit UUID)
- a string of hex digits of the form ‘xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx’

Creating a 128-bit UUID registers the UUID with the onboard BLE software, and provides a temporary 16-bit UUID that can be used in place of the full 128-bit UUID.

**Parameters**

value (`int`, `ReadableBuffer` or `str`) – The uuid value to encapsulate

**uuid16:** int

The 16-bit part of the UUID. (read-only)

**uuid128:** bytes

The 128-bit value of the UUID Raises AttributeError if this is a 16-bit UUID. (read-only)

**size:** int

128 if this UUID represents a 128-bit vendor-specific UUID. 16 if this UUID represents a 16-bit Bluetooth SIG assigned UUID. (read-only) 32-bit UUIDs are not currently supported.

**pack_into**(buffer: circuitpython_typing.WriteableBuffer, offset: int = 0) → None

Packs the UUID into the given buffer at the given offset.
__eq__(other: object) → bool
Two UUID objects are equal if their values match and they are both 128-bit or both 16-bit.

_eve – Low-level BridgeTek EVE bindings

The _eve module provides a class _EVE which contains methods for constructing EVE command buffers and appending basic graphics commands.

class _eve._EVE

Create an _EVE object

    register(o: object) → None

flush() → None
    Send any queued drawing commands directly to the hardware.

    Parameters
    width (int) – The width of the grid in tiles, or 1 for sprites.

cc(b: circuitpython_typing.ReadableBuffer) → None
    Append bytes to the command FIFO.

    Parameters
    b (ReadableBuffer) – The bytes to add

AlphaFunc(func: int, ref: int) → None
    Set the alpha test function

    Parameters
    • func (int) – specifies the test function, one of NEVER, LESS, LEQUAL, GREATER, GEQUAL, EQUAL, NOTEQUAL, or ALWAYS. Range 0-7. The initial value is ALWAYS(7)
    • ref (int) – specifies the reference value for the alpha test. Range 0-255. The initial value is 0

    These values are part of the graphics context and are saved and restored by SaveContext() and RestoreContext().

Begin(prim: int) → None
    Begin drawing a graphics primitive

    Parameters
    prim (int) – graphics primitive.

    Valid primitives are BITMAPS, POINTS, LINES, LINE_STRIP, EDGE_STRIP_R, EDGE_STRIP_L, EDGE_STRIP_A, EDGE_STRIP_B and RECTS.

BitmapExtFormat(format: int) → None
    Set the bitmap format

    Parameters
    format (int) – bitmap pixel format.

BitmapHandle(handle: int) → None
    Set the bitmap handle

    Parameters
    handle (int) – bitmap handle. Range 0-31. The initial value is 0
This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapLayoutH** (*linestride: int, height: int*) → None

Set the source bitmap memory format and layout for the current handle. high bits for large bitmaps

Parameters

- **linestride** (*int*) – high part of bitmap line stride, in bytes. Range 0-7
- **height** (*int*) – high part of bitmap height, in lines. Range 0-3

**BitmapLayout** (*format: int, linestride: int, height: int*) → None

Set the source bitmap memory format and layout for the current handle

Parameters

- **format** (*int*) – bitmap pixel format, or GLFORMAT to use BITMAP_EXT_FORMAT instead. Range 0-31
- **linestride** (*int*) – bitmap line stride, in bytes. Range 0-1023
- **height** (*int*) – bitmap height, in lines. Range 0-511

**BitmapSizeH** (*width: int, height: int*) → None

Set the screen drawing of bitmaps for the current handle. high bits for large bitmaps

Parameters

- **width** (*int*) – high part of drawn bitmap width, in pixels. Range 0-3
- **height** (*int*) – high part of drawn bitmap height, in pixels. Range 0-3

**BitmapSize** (*filter: int, wrapx: int, wrapy: int, width: int, height: int*) → None

Set the screen drawing of bitmaps for the current handle

Parameters

- **filter** (*int*) – bitmap filtering mode, one of NEAREST or BILINEAR. Range 0-1
- **wrapx** (*int*) – bitmap x wrap mode, one of REPEAT or BORDER. Range 0-1
- **wrapy** (*int*) – bitmap y wrap mode, one of REPEAT or BORDER. Range 0-1
- **width** (*int*) – drawn bitmap width, in pixels. Range 0-511
- **height** (*int*) – drawn bitmap height, in pixels. Range 0-511

**BitmapSource** (*addr: int*) → None

Set the source address for bitmap graphics

Parameters

- **addr** (*int*) – Bitmap start address, pixel-aligned. May be in SRAM or flash. Range 0-16777215

**BitmapSwizzle** (*r: int, g: int, b: int, a: int*) → None

Set the source for the r,g,b and a channels of a bitmap

Parameters

- **r** (*int*) – red component source channel. Range 0-7
- **g** (*int*) – green component source channel. Range 0-7
- **b** (*int*) – blue component source channel. Range 0-7
- **a** (*int*) – alpha component source channel. Range 0-7
**BitmapTransformA** \((p: \text{int}, v: \text{int}) \rightarrow \text{None}\)

Set the \(a\) component of the bitmap transform matrix

**Parameters**

- **\(p\) (\text{int})** – precision control: 0 is 8.8, 1 is 1.15. Range 0-1. The initial value is 0
- **\(v\) (\text{int})** – The \(a\) component of the bitmap transform matrix, in signed 8.8 or 1.15 bit fixed-point form. Range 0-131071. The initial value is 256

The initial value is \(p = 0, v = 256\). This represents the value 1.0.

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapTransformB** \((p: \text{int}, v: \text{int}) \rightarrow \text{None}\)

Set the \(b\) component of the bitmap transform matrix

**Parameters**

- **\(p\) (\text{int})** – precision control: 0 is 8.8, 1 is 1.15. Range 0-1. The initial value is 0
- **\(v\) (\text{int})** – The \(b\) component of the bitmap transform matrix, in signed 8.8 or 1.15 bit fixed-point form. Range 0-131071. The initial value is 0

The initial value is \(p = 0, v = 0\). This represents the value 0.0.

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapTransformC** \((v: \text{int}) \rightarrow \text{None}\)

Set the \(c\) component of the bitmap transform matrix

**Parameters**

- **\(v\) (\text{int})** – The \(c\) component of the bitmap transform matrix, in signed 15.8 bit fixed-point form. Range 0-16777215. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapTransformD** \((p: \text{int}, v: \text{int}) \rightarrow \text{None}\)

Set the \(d\) component of the bitmap transform matrix

**Parameters**

- **\(p\) (\text{int})** – precision control: 0 is 8.8, 1 is 1.15. Range 0-1. The initial value is 0
- **\(v\) (\text{int})** – The \(d\) component of the bitmap transform matrix, in signed 8.8 or 1.15 bit fixed-point form. Range 0-131071. The initial value is 0

The initial value is \(p = 0, v = 0\). This represents the value 0.0.

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapTransformE** \((p: \text{int}, v: \text{int}) \rightarrow \text{None}\)

Set the \(e\) component of the bitmap transform matrix

**Parameters**

- **\(p\) (\text{int})** – precision control: 0 is 8.8, 1 is 1.15. Range 0-1. The initial value is 0
- **\(v\) (\text{int})** – The \(e\) component of the bitmap transform matrix, in signed 8.8 or 1.15 bit fixed-point form. Range 0-131071. The initial value is 256
The initial value is \( p = 0, v = 256 \). This represents the value 1.0.

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**BitmapTransformF(v: int) \( \to \) None**

Set the \( f \) component of the bitmap transform matrix

**Parameters**

\( v \ (\text{int}) \) – The \( f \) component of the bitmap transform matrix, in signed 15.8 bit fixed-point form. Range 0-16777215. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**BlendFunc(src: int, dst: int) \( \to \) None**

Set pixel arithmetic

**Parameters**

- **src (int)** – specifies how the source blending factor is computed. One of `ZERO`, `ONE`, `SRC_ALPHA`, `DST_ALPHA`, `ONE_MINUS_SRC_ALPHA` or `ONE_MINUS_DST_ALPHA`. Range 0-7. The initial value is `SRC_ALPHA(2)`

- **dst (int)** – specifies how the destination blending factor is computed, one of the same constants as `src`. Range 0-7. The initial value is `ONE_MINUS_SRC_ALPHA(4)`

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**Call(dest: int) \( \to \) None**

Execute a sequence of commands at another location in the display list

**Parameters**

- **dest (int)** – display list address. Range 0-65535

**Cell(cell: int) \( \to \) None**

Set the bitmap cell number for the vertex2f command

**Parameters**

- **cell (int)** – bitmap cell number. Range 0-127. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**ClearColorA(alpha: int) \( \to \) None**

Set clear value for the alpha channel

**Parameters**

- **alpha (int)** – alpha value used when the color buffer is cleared. Range 0-255. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**ClearColorRGB(red: int, green: int, blue: int) \( \to \) None**

Set clear values for red, green and blue channels

**Parameters**

- **red (int)** – red value used when the color buffer is cleared. Range 0-255. The initial value is 0
• **green** (*int*) – green value used when the color buffer is cleared. Range 0-255. The initial value is 0

• **blue** (*int*) – blue value used when the color buffer is cleared. Range 0-255. The initial value is 0

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**Clear**(*c: int, s: int, t: int*) → None

Clear buffers to preset values

**Parameters**

• **c** (*int*) – clear color buffer. Range 0-1

• **s** (*int*) – clear stencil buffer. Range 0-1

• **t** (*int*) – clear tag buffer. Range 0-1

**ClearStencil**(*s: int*) → None

Set clear value for the stencil buffer

**Parameters**

• **s** (*int*) – value used when the stencil buffer is cleared. Range 0-255. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**ClearTag**(*s: int*) → None

Set clear value for the tag buffer

**Parameters**

• **s** (*int*) – value used when the tag buffer is cleared. Range 0-255. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**ColorA**(*alpha: int*) → None

Set the current color alpha

**Parameters**

• **alpha** (*int*) – alpha for the current color. Range 0-255. The initial value is 255

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**ColorMask**(*r: int, g: int, b: int, a: int*) → None

Enable and disable writing of frame buffer color components

**Parameters**

• **r** (*int*) – allow updates to the frame buffer red component. Range 0-1. The initial value is 1

• **g** (*int*) – allow updates to the frame buffer green component. Range 0-1. The initial value is 1

• **b** (*int*) – allow updates to the frame buffer blue component. Range 0-1. The initial value is 1

• **a** (*int*) – allow updates to the frame buffer alpha component. Range 0-1. The initial value is 1
These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**ColorRGB(red: int, green: int, blue: int) → None**

Set the drawing color

- **Parameters**
  - `red (int)` – red value for the current color. Range 0-255. The initial value is 255
  - `green (int)` – green for the current color. Range 0-255. The initial value is 255
  - `blue (int)` – blue for the current color. Range 0-255. The initial value is 255

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

**Display() → None**

End the display list

**End() → None**

End drawing a graphics primitive

`Vertex2ii()` and `Vertex2if()` calls are ignored until the next `Begin()`.

**Jump(dest: int) → None**

Execute commands at another location in the display list

- **Parameters**
  - `dest (int)` – display list address. Range 0-65535

**Macro(m: int) → None**

Execute a single command from a macro register

- **Parameters**
  - `m (int)` – macro register to read. Range 0-1

**Nop() → None**

No operation

**PaletteSource(addr: int) → None**

Set the base address of the palette

- **Parameters**
  - `addr (int)` – Address in graphics SRAM, 2-byte aligned. Range 0-4194303. The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**RestoreContext() → None**

Restore the current graphics context from the context stack

**Return() → None**

Return from a previous call command

**SaveContext() → None**

Push the current graphics context on the context stack
ScissorSize(width: int, height: int) → None

Set the size of the scissor clip rectangle

Parameters

• width (int) – The width of the scissor clip rectangle, in pixels. Range 0-4095. The initial value is hsize

• height (int) – The height of the scissor clip rectangle, in pixels. Range 0-4095. The initial value is 2048

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

ScissorXY(x: int, y: int) → None

Set the top left corner of the scissor clip rectangle

Parameters

• x (int) – The x coordinate of the scissor clip rectangle, in pixels. Range 0-2047. The initial value is 0

• y (int) – The y coordinate of the scissor clip rectangle, in pixels. Range 0-2047. The initial value is 0

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

StencilFunc(func: int, ref: int, mask: int) → None

Set function and reference value for stencil testing

Parameters

• func (int) – specifies the test function, one of NEVER, LESS, LEQUAL, GREATER, GEQUAL, EQUAL, NOTEQUAL, or ALWAYS. Range 0-7. The initial value is ALWAYS(7)

• ref (int) – specifies the reference value for the stencil test. Range 0-255. The initial value is 0

• mask (int) – specifies a mask that is ANDed with the reference value and the stored stencil value. Range 0-255. The initial value is 255

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

StencilMask(mask: int) → None

Control the writing of individual bits in the stencil planes

Parameters

• mask (int) – the mask used to enable writing stencil bits. Range 0-255. The initial value is 255

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

StencilOp(sfail: int, spass: int) → None

Set stencil test actions

Parameters

• sfail (int) – specifies the action to take when the stencil test fails, one of KEEP, ZERO, REPLACE, INCR, INCWRAP, DECR, DECR_WRAP, and INVERT. Range 0-7. The initial value is KEEP(1)
• `spass (int)` – specifies the action to take when the stencil test passes, one of the same constants as `sfail`. Range 0-7. The initial value is KEEP(1).

These values are part of the graphics context and are saved and restored by `SaveContext()` and `RestoreContext()`.

`TagMask(mask: int) → None`

Control the writing of the tag buffer

**Parameters**

`mask (int)` – allow updates to the tag buffer. Range 0-1. The initial value is 1.

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

`Tag(s: int) → None`

Set the current tag value

**Parameters**

`s (int)` – tag value. Range 0-255. The initial value is 255.

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

`Vertex2ii(x: int, y: int, handle: int, cell: int) → None`

**Parameters**

• `x (int)` – x-coordinate in pixels. Range 0-511
• `y (int)` – y-coordinate in pixels. Range 0-511
• `handle (int)` – bitmap handle. Range 0-31
• `cell (int)` – cell number. Range 0-127

This method is an alternative to `Vertex2f()`.

`Vertex2f(b: float) → None`

Draw a point.

**Parameters**

• `x (float)` – pixel x-coordinate
• `y (float)` – pixel y-coordinate

`LineWidth(width: float) → None`

Set the width of rasterized lines

**Parameters**

`width (float)` – line width in pixels. Range 0-511. The initial value is 1.

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

`PointSize(size: float) → None`

Set the diameter of rasterized points

**Parameters**

`size (float)` – point diameter in pixels. Range 0-1023. The initial value is 1.

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.
**VertexTranslateX** \((x: \text{float}) \rightarrow \text{None}\)

Set the vertex transformation’s x translation component

**Parameters**

\(x (\text{float})\) – signed x-coordinate in pixels. Range \(\pm 4095\). The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**VertexTranslateY** \((y: \text{float}) \rightarrow \text{None}\)

Set the vertex transformation’s y translation component

**Parameters**

\(y (\text{float})\) – signed y-coordinate in pixels. Range \(\pm 4095\). The initial value is 0

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**VertexFormat** \((\text{frac}: \text{int}) \rightarrow \text{None}\)

Set the precision of vertex2f coordinates

**Parameters**

\(\text{frac} (\text{int})\) – Number of fractional bits in X,Y coordinates, 0-4. Range 0-7. The initial value is 4

This value is part of the graphics context and is saved and restored by `SaveContext()` and `RestoreContext()`.

**cmd0** \((n: \text{int}) \rightarrow \text{None}\)

Append the command word \(n\) to the FIFO

**Parameters**

\(n (\text{int})\) – The command code

This method is used by the `eve` module to efficiently add commands to the FIFO.

**cmd** \((n: \text{int}, \text{fmt}: \text{str}, \text{args}: \text{Tuple}[\text{str}, \text{Ellipsis}] ) \rightarrow \text{None}\)

Append a command packet to the FIFO.

**Parameters**

- \(n (\text{int})\) – The command code
- \(\text{fmt} (\text{str})\) – The command format `struct` layout
- \(\text{args} (\text{tuple}(\text{str}, \ldots))\) – The command’s arguments

Supported format codes: h, H, i, I.

This method is used by the `eve` module to efficiently add commands to the FIFO.

**_pew – LED matrix driver**


This is an internal module to be used by the `pew.py` library from https://github.com/pewpew-game/pew-pewpew-standalone-10.x to handle the LED matrix display and buttons on the pewpew10 board.

Usage:
This singleton class is instantiated by the `pew` library, and used internally by it. All user-visible interactions are done through that library.

Initializes matrix scanning routines.

The `buffer` is a 64 byte long `bytearray` that stores what should be displayed on the matrix. `rows` and `cols` are both lists of eight `DigitalInputOutput` objects that are connected to the matrix rows and columns. `buttons` is a `DigitalInputOutput` object that is connected to the common side of all buttons (the other sides of the buttons are connected to rows of the matrix).

_pixelmap – A fast pixel mapping library

The `_pixelmap` module provides the `PixelMap` class to accelerate RGB(W) strip/matrix manipulation, such as DotStar and Neopixel.

_class _pixelmap.PixelMap(pixbuf: adafruit_pixelbuf.PixelBuf, indices: Tuple[int | Tuple[int]])

Construct a `PixelMap` object that uses the given indices of the underlying `pixbuf`.

auto_write: bool

True if updates should be automatically written

bpp: int

The number of bytes per pixel in the buffer (read-only)

byteorder: str

byteorder string for the buffer (read-only)

fill(color: adafruit_pixelbuf.PixelType) → None

Fill all the pixels in the map with the given color

indices(index: int) → Tuple[int]

Return the `Pixbuf` indices for a `PixelMap` index

__getitem__(index: slice) → PixelReturnSequence

__getitem__(index: int) → adafruit_pixelbuf.PixelReturnType

__setitem__(index: slice, value: adafruit_pixelbuf.PixelSequence) → None

__setitem__(index: int, value: adafruit_pixelbuf.PixelType) → None

Sets the pixel value at the given index. Value can either be a tuple or integer. Tuples are The individual (Red, Green, Blue[, White]) values between 0 and 255. If given an integer, the red, green and blue values are packed into the lower three bytes (0xRRGGBB). For RGBW byteorders, if given only RGB values either as an int or as a tuple, the white value is used instead when the red, green, and blue values are the same.

__len__() → int

Length of the map
Transmit the color data to the pixels so that they are shown. This is done automatically when auto_write is True.

stage – C-level helpers for animation of sprites on a stage

The _stage module contains native code to speed-up the stage Library.

```python
def _stage.render(x0: int, y0: int, x1: int, y1: int, layers: List[Layer], buffer: WriteableBuffer, display: displayio.Display, scale: int, background: int) -> None:
    Render and send to the display a fragment of the screen.
```

Parameters:
- `x0 (int)` – Left edge of the fragment.
- `y0 (int)` – Top edge of the fragment.
- `x1 (int)` – Right edge of the fragment.
- `y1 (int)` – Bottom edge of the fragment.
- `layers (list[Layer])` – A list of the Layer objects.
- `buffer (WriteableBuffer)` – A buffer to use for rendering.
- `display (Display)` – The display to use.
- `scale (int)` – How many times should the image be scaled up.
- `background (int)` – What color to display when nothing is there.

There are also no sanity checks, outside of the basic overflow checking. The caller is responsible for making the passed parameters valid.

This function is intended for internal use in the stage library and all the necessary checks are performed there.

```python
class _stage.Layer(width: int, height: int, graphic: ReadableBuffer, palette: ReadableBuffer, grid: ReadableBuffer)
```

Keep internal information about a single layer of graphics

Keep internal information about a layer of graphics (either a Grid or a Sprite) in a format suitable for fast rendering with the render() function.

```python
Parameters:
- `width (int)` – The width of the grid in tiles, or 1 for sprites.
- `height (int)` – The height of the grid in tiles, or 1 for sprites.
- `graphic (ReadableBuffer)` – The graphic data of the tiles.
- `palette (ReadableBuffer)` – The color palette to be used.
- `grid (ReadableBuffer)` – The contents of the grid map.
```

This class is intended for internal use in the stage library and it shouldn't be used on its own.

```python
def move(x: int, y: int) -> None
    Set the offset of the layer to the specified values.
```
frame(frame: int, rotation: int) → None

Set the animation frame of the sprite, and optionally rotation its graphic.


Keep information about a single grid of text

Keep internal information about a grid of text in a format suitable for fast rendering with the render() function.

Parameters
• width(int) – The width of the grid in tiles, or 1 for sprites.
• height(int) – The height of the grid in tiles, or 1 for sprites.
• font(ReadableBuffer) – The font data of the characters.
• palette(ReadableBuffer) – The color palette to be used.
• chars(ReadableBuffer) – The contents of the character grid.

This class is intended for internal use in the stage library and it shouldn’t be used on its own.

move(x: int, y: int) → None

Set the offset of the text to the specified values.

adafruit_bus_device – Hardware accelerated external bus access

The I2CDevice and SPIDevice helper classes make managing transaction state on a bus easy. For example, they manage locking the bus to prevent other concurrent access. For SPI devices, it manages the chip select and protocol changes such as mode. For I2C, it manages the device address.

adafruit_bus_device.i2c_device – I2C Device Manager

class adafruit_bus_device.i2c_device.I2CDevice(i2c: busio.I2C, device_address: int, probe: bool = True)

Represents a single I2C device and manages locking the bus and the device address.

Parameters
• i2c(I2C) – The I2C bus the device is on
• device_address(int) – The 7 bit device address
• probe(bool) – Probe for the device upon object creation, default is true

Example:

```python
import busio
from board import *
from adafruit_bus_device.i2c_device import I2CDevice
with busio.I2C(SCL, SDA) as i2c:
    device = I2CDevice(i2c, 0x70)
    bytes_read = bytearray(4)
    with device:
        device.readinto(bytes_read)
# A second transaction
```

(continues on next page)
with device:
    device.write(bytes_read)

__enter__() \rightarrow I2CDevice

Context manager entry to lock bus.

__exit__() \rightarrow None

Automatically unlocks the bus on exit.

readinto(buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int = sys.maxsize) \rightarrow None

Read into buffer from the device.

If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed. The number of bytes read will be the length of buffer[start:end].

Parameters

- buffer (WriteableBuffer) – read bytes into this buffer
- start (int) – beginning of buffer slice
- end (int) – end of buffer slice; if not specified, use len(buffer)

write(buffer: circuitpython_typing.ReadableBuffer, *, start: int = 0, end: int = sys.maxsize) \rightarrow None

Write the bytes from buffer to the device, then transmit a stop bit.

If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed, but without copying the data. The number of bytes written will be the length of buffer[start:end].

Parameters

- buffer (ReadableBuffer) – write out bytes from this buffer
- start (int) – beginning of buffer slice
- end (int) – end of buffer slice; if not specified, use len(buffer)

write_then_readinto(out_buffer: circuitpython_typing.ReadableBuffer, in_buffer: circuitpython_typing.WriteableBuffer, *, out_start: int = 0, out_end: int = sys.maxsize, in_start: int = 0, in_end: int = sys.maxsize) \rightarrow None

Write the bytes from out_buffer to the device, then immediately reads into in_buffer from the device.

If out_start or out_end is provided, then the buffer will be sliced as if out_buffer[out_start:out_end] were passed, but without copying the data. The number of bytes written will be the length of out_buffer[out_start:out_end].

If in_start or in_end is provided, then the input buffer will be sliced as if in_buffer[in_start:in_end] were passed. The number of bytes read will be the length of out_buffer[out_start:out_end].

Parameters

- out_buffer (ReadableBuffer) – write out bytes from this buffer
- in_buffer (WriteableBuffer) – read bytes into this buffer
- out_start (int) – beginning of out_buffer slice
- out_end (int) – end of out_buffer slice; if not specified, use len(out_buffer)
- in_start (int) – beginning of in_buffer slice
- in_end (int) – end of in_buffer slice; if not specified, use len(in_buffer)
**adafruit_bus_device.spi_device** – SPI Device Manager

```python
class adafruit_bus_device.spi_device.SPIDevice(spi: busio.SPI, chip_select: digitalio.DigitalInOut | None = None, *, baudrate: int = 100000, polarity: int = 0, phase: int = 0, extra_clocks: int = 0)
```

Represents a single SPI device and manages locking the bus and the device address.

**Parameters**

- `spi` *(SPI)* — The SPI bus the device is on
- `chip_select` *(DigitalInOut)* — The chip select pin object that implements the DigitalInOut API. `None` if a chip select pin is not being used.
- `cs_active_value` *(bool)* — Set to true if your device requires CS to be active high. Defaults to false.
- `extra_clocks` *(int)* — The minimum number of clock cycles to cycle the bus after CS is high. (Used for SD cards.)

**Example:**

```python
import busio
import digitalio
from board import *
from adafruit_bus_device.spi_device import SPIDevice

with busio.SPI(SCK, MOSI, MISO) as spi_bus:
    cs = digitalio.DigitalInOut(D10)
    device = SPIDevice(spi_bus, cs)

    bytes_read = bytearray(4)
    # The object assigned to spi in the with statements below
    # is the original spi_bus object. We are using the busio.SPI
    # operations busio.SPI.readinto() and busio.SPI.write().

    with device as spi:
        spi.readinto(bytes_read)
        # A second transaction

    with device as spi:
        spi.write(bytes_read)
```

**__enter__**() → `busio.SPI`

Starts a SPI transaction by configuring the SPI and asserting chip select.

**__exit__**() → `None`

Ends a SPI transaction by deasserting chip select. See `Lifetime and ContextManagers` for more info.

**adafruit_pixelbuf** – A fast RGB(W) pixel buffer library for like NeoPixel and DotStar

The `adafruit_pixelbuf` module provides the `PixelBuf` class to accelerate RGB(W) strip/matrix manipulation, such as DotStar and Neopixel.

Byteorders are configured with strings, such as “RGB” or “RGBD”.

**adafruit_pixelbuf.PixelReturnType**

**adafruit_pixelbuf.PixelReturnSequence**
A fast RGB[W] pixel buffer for LED and similar devices.

Create a PixelBuf object of the specified size, byteorder, and bits per pixel.

When brightness is less than 1.0, a second buffer will be used to store the color values before they are adjusted for brightness.

When P (PWM duration) is present as the 4th character of the byteorder string, the 4th value in the tuple/list for a pixel is the individual pixel brightness (0.0-1.0) and will enable a Dotstar compatible 1st byte for each pixel.

Parameters

- `size (int)` – Number of pixels
- `byteorder (str)` – Byte order string (such as “RGB”, “RGBW” or “PBGR”)
- `brightness (float)` – Brightness (0 to 1.0, default 1.0)
- `auto_write (bool)` – Whether to automatically write pixels (Default False)
- `header (ReadableBuffer)` – Sequence of bytes to always send before pixel values.
- `trailer (ReadableBuffer)` – Sequence of bytes to always send after pixel values.

**bpp:** int

The number of bytes per pixel in the buffer (read-only)

**brightness:** float

Float value between 0 and 1. Output brightness.

When brightness is less than 1.0, a second buffer will be used to store the color values before they are adjusted for brightness.

**auto_write:** bool

Whether to automatically write the pixels after each update.

**byteorder:** str

byteorder string for the buffer (read-only)

**show()** → None

Transmits the color data to the pixels so that they are shown. This is done automatically when `auto_write` is True.

**fill(color: PixelType) → None**

Fills the given pixelbuf with the given color.

**__getitem__(index: slice) → PixelReturnType**

Returns the pixel value at the given index as a tuple of (Red, Green, Blue[, White]) values between 0 and 255. When in PWM (DotStar) mode, the 4th tuple value is a float of the pixel intensity from 0-1.0.

**__setitem__(index: slice, value: PixelSequence) → None**
**__setitem__** (index: int, value: PixelType) → None

Sets the pixel value at the given index. Value can either be a tuple or integer. Tuples are The individual (Red, Green, Blue[, White]) values between 0 and 255. If given an integer, the red, green and blue values are packed into the lower three bytes (0xRRGGBB). For RGBW byteorders, if given only RGB values either as an int or as a tuple, the white value is used instead when the red, green, and blue values are the same.

### aesio – AES encryption routines

The AES module contains classes used to implement encryption and decryption. It aims to be low overhead in terms of memory.

For more information on AES, refer to the Wikipedia entry.

```python
aesio.MODE_ECB: int
aesio.MODE_CBC: int
aesio.MODE_CTR: int
```

```python
class aesio.AES(key: circuitpython_typing.ReadableBuffer, mode: int = 0, IV: circuitpython_typing.ReadableBuffer | None = None, segment_size: int = 8)
```

Encrypt and decrypt AES streams

Create a new AES state with the given key.

**Parameters**

- **key (ReadableBuffer)** – A 16-, 24-, or 32-byte key
- **mode (int)** – AES mode to use. One of: MODE_ECB, MODE_CBC, or MODE_CTR
- **IV (ReadableBuffer)** – Initialization vector to use for CBC or CTR mode

Additional arguments are supported for legacy reasons.

Encrypting a string:

```python
import aesio
from binascii import hexlify

key = b'Sixteen byte key'
inp = b'CircuitPython!!!'  # Note: 16-bytes long
outp = bytearray(len(inp))
cipher = aesio.AES(key, aesio.MODE_ECB)
cipher.encrypt_into(inp, outp)
hexlify(outp)
```

```python
rekey(key: circuitpython_typing.ReadableBuffer, IV: circuitpython_typing.ReadableBuffer | None = None) → None
```

Update the AES state with the given key.

**Parameters**

- **key (ReadableBuffer)** – A 16-, 24-, or 32-byte key
- **IV (ReadableBuffer)** – Initialization vector to use for CBC or CTR mode
encrypt_into(src: circuitpython_typing.ReadableBuffer, dest: circuitpython_typing.WriteableBuffer) → None

Encrypt the buffer from src into dest.

For ECB mode, the buffers must be 16 bytes long. For CBC mode, the buffers must be a multiple of 16 bytes, and must be equal length. For CTR mode, there are no restrictions.

decrypt_into(src: circuitpython_typing.ReadableBuffer, dest: circuitpython_typing.WriteableBuffer) → None

Decrypt the buffer from src into dest. For ECB mode, the buffers must be 16 bytes long. For CBC mode, the buffers must be a multiple of 16 bytes, and must be equal length. For CTR mode, there are no restrictions.

alarm – Alarms and sleep

Provides alarms that trigger based on time intervals or on external events, such as pin changes. The program can simply wait for these alarms, or go to sleep and be awoken when they trigger.

There are two supported levels of sleep: light sleep and deep sleep.

Light sleep keeps sufficient state so the program can resume after sleeping. It does not shut down WiFi, BLE, or other communications, or ongoing activities such as audio playback. It reduces power consumption to the extent possible that leaves these continuing activities running. In some cases there may be no decrease in power consumption.

Deep sleep shuts down power to nearly all of the microcontroller including the CPU and RAM. This can save a more significant amount of power, but CircuitPython must restart code.py from the beginning when awakened.

For both light sleep and deep sleep, if CircuitPython is connected to a host computer, maintaining the connection takes priority and power consumption may not be reduced.

For more information about working with alarms and light/deep sleep in CircuitPython, see this Learn guide.

alarm.pin – Trigger an alarm when a pin changes state.

class alarm.pin.PinAlarm(pin: microcontroller.Pin, value: bool, edge: bool = False, pull: bool = False)

Create an alarm triggered by a microcontroller.Pin level. The alarm is not active until it is passed to an alarm-enabling function, such as alarm.light_sleep_until_alarms() or alarm.exit_and_deep_sleep_until_alarms().

Parameters

- **pin** (`microcontroller.Pin`) – The pin to monitor. On some ports, the choice of pin may be limited due to hardware restrictions, particularly for deep-sleep alarms.

- **value** (`bool`) – When active, trigger when the pin value is high (True) or low (False). On some ports, multiple PinAlarm objects may need to have coordinated values for deep-sleep alarms.

- **edge** (`bool`) – If True, trigger only when there is a transition to the specified value of value. If True, if the alarm becomes active when the pin value already matches value, the alarm is not triggered: the pin must transition from not value to value to trigger the alarm. On some ports, edge-triggering may not be available, particularly for deep-sleep alarms.

- **pull** (`bool`) – Enable a pull-up or pull-down which pulls the pin to the level opposite that of value. For instance, if value is set to True, setting pull to True will enable a pull-down, to hold the pin low normally until an outside signal pulls it high.
pin: `microcontroller.Pin`
   The trigger pin.

value: `bool`
   The value on which to trigger.

**alarm.time** – Trigger an alarm when the specified time is reached.

```python
class alarm.time.TimeAlarm(*, monotonic_time: float | None = None, epoch_time: int | None = None)
```

Create an alarm that will be triggered when `time.monotonic()` would equal `monotonic_time`, or when `time.time()` would equal `epoch_time`. Only one of the two arguments can be given. The alarm is not active until it is passed to an `alarm`-enabling sleep function, such as `alarm.light_sleep_until_alarms()` or `alarm.exit_and_deep_sleep_until_alarms()`.

If the given time is already in the past, then an exception is raised. If the sleep happens after the given time, then it will wake immediately due to this time alarm.

**monotonic_time**: `float`
   When this time is reached, the alarm will trigger, based on the `time.monotonic()` clock. The time may be given as `epoch_time` in the constructor, but it is returned by this property only as a `time.monotonic()` time.

**alarm.touch** – Trigger an alarm when touch is detected.

```python
class alarm.touch.TouchAlarm(*pin: microcontroller.Pin)
```

Create an alarm that will be triggered when the given pin is touched. The alarm is not active until it is passed to an `alarm`-enabling function, such as `alarm.light_sleep_until_alarms()` or `alarm.exit_and_deep_sleep_until_alarms()`.

**Parameters**
   - `pin` (`microcontroller.Pin`) – The pin to monitor. On some ports, the choice of pin may be limited due to hardware restrictions, particularly for deep-sleep alarms.

**Limitations**: Not available on SAMD, nRF, or RP2040.

**pin**: `microcontroller.Pin`
   The trigger pin.

**alarm.sleep_memory**: `SleepMemory`
   Memory that persists during deep sleep. This object is the sole instance of `alarm.SleepMemory`.

**alarm.wake_alarm**: `circuitpython_typing.Alarm` | `None`
   The most recently triggered alarm. If CircuitPython was sleeping, the alarm that woke it from sleep. If no alarm occurred since the last hard reset or soft restart, value is `None`.

**alarm.light_sleep_until_alarms**(*alarms: circuitpython_typing.Alarm*) → `circuitpython_typing.Alarm`
   Go into a light sleep until awakened one of the alarms. The alarm causing the wake-up is returned, and is also available as `alarm.wake_alarm`.

   If no alarms are specified, return immediately.

   **If CircuitPython is connected to a host computer, the connection will be maintained, and the microcontroller may not actually go into a light sleep.** This allows the user to interrupt an existing program with ctrl-C, and to edit the files in CIRCUITPY, which would not be possible in true light sleep. Thus, to use light sleep and save significant power, it may be necessary to disconnect from the host.
alarm.exit_and_deep_sleep_until_alarms(*alarms: circuitpython_typing.Alarm, preserve_dios: Sequence[digitalio.DigitalInOut] = () → None

Exit the program and go into a deep sleep, until awakened by one of the alarms. This function does not return. When awakened, the microcontroller will restart and will run boot.py and code.py from the beginning. After restart, an alarm equivalent to the one that caused the wake-up will be available as alarm.wake_alarm. Its type and/or attributes may not correspond exactly to the original alarm. For time-base alarms, currently, an alarm.time.TimeAlarm() is created.

If no alarms are specified, the microcontroller will deep sleep until reset.

Parameters

- **alarms** ([circuitpython_typing.Alarm]) – the alarms that can wake the microcontroller.
- **preserve_dios** ([Sequence[digitalio.DigitalInOut]]) – A sequence of DigitalInOut objects whose state should be preserved during deep sleep. If a DigitalInOut in the sequence is set to be an output, its current DigitalInOut.value (True or False) will be preserved during the deep sleep. If a DigitalInOut in the sequence is set to be an input, its current DigitalInOut.pull value (DOWN, UP, or None) will be preserved during deep sleep.

Preserving DigitalInOut states during deep sleep can be used to ensure that external or on-board devices are powered or unpowered during sleep, among other purposes.

On some microcontrollers, some pins cannot remain in their original state for hardware reasons.

**Limitations:** preserve_dios is currently only available on Espressif.

**Note:** On Espressif chips, preserving pin settings during deep sleep may consume extra current. On ESP32, this was measured to be 250 uA or more. Consider not preserving pins unless you need to. Measure power consumption carefully both with no pins preserved and with the pins you might want to preserve to achieve the lowest consumption.

If CircuitPython is connected to a host computer via USB or BLE the first time a deep sleep is requested, the connection will be maintained and the system will not go into deep sleep. This allows the user to interrupt an existing program with ctrl-C, and to edit the files in CIRCUITPY, which would not be possible in true deep sleep.

If CircuitPython goes into a true deep sleep, and USB or BLE is reconnected, the next deep sleep will still be a true deep sleep. You must do a hard reset or power-cycle to exit a true deep sleep loop.

Here is a skeletal example:

```python
import alarm
import time
import board

print("Waking up")

# Create an alarm for 60 seconds from now, and also a pin alarm.
time_alarm = alarm.time.TimeAlarm(monotonic_time=time.monotonic() + 60)
pin_alarm = alarm.pin.PinAlarm(board.D7, False)

# Deep sleep until one of the alarm goes off. Then restart the program.
alarm.exit_and_deep_sleep_until_alarms(time_alarm, pin_alarm)
```
class `alarm.SleepMemory`

Store raw bytes in RAM that persists during deep sleep. The class acts as a `bytearray`. If power is lost, the memory contents are lost.

Note that this class can't be imported and used directly. The sole instance of `SleepMemory` is available at `alarm.sleep_memory`.

**Limitations:** Not supported on RP2040.

Usage:

```python
import alarm
alarm.sleep_memory[0] = True
alarm.sleep_memory[1] = 12
```

Not used. Access the sole instance through `alarm.sleep_memory`.

**bool** () → bool

sleep_memory is True if its length is greater than zero. This is an easy way to check for its existence.

**len** () → int

Return the length. This is used by `len`.

**getitem** (index: `slice`) → `bytearray`

**getitem** (index: `int`) → `int`

Returns the value at the given index.

**setitem** (index: `slice`, value: `circuitpython_typing.ReadableBuffer`) → None

**setitem** (index: `int`, value: `int`) → None

Set the value at the given index.

**analogbufio – Analog Buffered IO Hardware Support**

The `analogbufio` module contains classes to provide access to analog-to-digital conversion and digital-to-analog (DAC) for multiple value transfer.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See *Lifetime and Context Managers* for more info.

**TODO:** For the essentials of `analogbufio`, see the CircuitPython Essentials Learn guide

**TODO:** For more information on using `analogbufio`, see this additional Learn guide

class `analogbufio.BufferedIn(pin: microcontroller.Pin, *, sample_rate: `int```

Capture multiple analog voltage levels to the supplied buffer

Usage:

```python
import board
import analogbufio
import array

length = 1000
mybuffer = array.array("H", [0x0000] * length)
rate = 500000
adcbuf = analogbufio.BufferedIn(board.GP26, sample_rate=rate)
adcbuf.readinto(mybuffer)
```

(continues on next page)
Create a `BufferedIn` on the given pin and given sample rate.

**Parameters**

- **pin (Pin)** – the pin to read from

- **sample_rate (~int)** – rate: sampling frequency, in samples per second

**deinit() → None**

Shut down the `BufferedIn` and release the pin for other use.

**__enter__() → BufferedIn**

No-op used by Context Managers.

**__exit__() → None**

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**readinto(buffer: circuitpython_typing.WriteableBuffer) → int**

Fills the provided buffer with ADC voltage values.

ADC values will be read into the given buffer at the supplied sample_rate. Depending on the buffer type-code, ‘B’, ‘H’, samples are 8-bit byte-arrays or 16-bit half-words and are always unsigned. The ADC most significant bits of the ADC are kept. (See [https://docs.circuitpython.org/en/latest/docs/library/array.html](https://docs.circuitpython.org/en/latest/docs/library/array.html))

**Parameters**

- **buffer (WriteableBuffer)** – buffer: A buffer for samples

---

**analogio – Analog hardware support**

The `analogio` module contains classes to provide access to analog IO typically implemented with digital-to-analog (DAC) and analog-to-digital (ADC) converters.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See *Lifetime and ContextManagers* for more info.

For example:

```python
import analogio
from board import *

pin = analogio.AnalogIn(A0)
print(pin.value)
pin.deinit()
```

This example will initialize the the device, read value and then `deinit()` the hardware. The last step is optional because CircuitPython will do it automatically after the program finishes.

For the essentials of `analogio`, see the CircuitPython Essentials Learn guide
For more information on using `analogio`, see this additional Learn guide

```python
import analogio
from board import *
adc = analogio.AnalogIn(A1)
val = adc.value
```

Use the AnalogIn on the given pin. The reference voltage varies by platform so use `reference_voltage` to read the configured setting.

**Parameters**

- **pin** (`Pin`) – the pin to read from

**value**: `int`

The value on the analog pin between 0 and 65535 inclusive (16-bit). (read-only)

Even if the underlying analog to digital converter (ADC) is lower resolution, the value is 16-bit.

**reference_voltage**: `float`

The maximum voltage measurable (also known as the reference voltage) as a `float` in Volts. Note the ADC value may not scale to the actual voltage linearly at ends of the analog range.

- **deinit** () → None
  
  Turn off the AnalogIn and release the pin for other use.

- **__enter__** () → AnalogIn
  
  No-op used by Context Managers.

- **__exit__** () → None
  
  Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

```python
import analogio
from board import *

dac = analogio.AnalogOut(A2)
# output on pin A2
dac.value = 32768  
# makes A2 1.65V
```

Use the AnalogOut on the given pin.

**Parameters**

- **pin** (`Pin`) – the pin to output to
value: int
The value on the analog pin between 0 and 65535 inclusive (16-bit). (write-only)
Even if the underlying digital to analog converter (DAC) is lower resolution, the value is 16-bit.

definit() → None
Turn off the AnalogOut and release the pin for other use.

__enter__() → AnalogOut
No-op used by Context Managers.

__exit__() → None
Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

atexit – Atexit Module

This module defines functions to register and unregister cleanup functions. Functions thus registered are automatically
executed upon normal vm termination.
These functions are run in the reverse order in which they were registered; if you register A, B, and C, they will be run
in the order C, B, A.

This module implements a subset of the corresponding CPython module, as described below. For more information,
refer to the original CPython documentation: atexit.

atexit.register(func: Callable[Ellipsis, Any], *args: Any | None, **kwargs: Any | None) → Callable[Ellipsis, Any]
Register func as a function to be executed at termination.
Any optional arguments that are to be passed to func must be passed as arguments to register(). It is possible
to register the same function and arguments more than once.
At normal program termination (for instance, if sys.exit() is called or the vm execution completes), all func-
tions registered are called in last in, first out order.
If an exception is raised during execution of the exit handler, a traceback is printed (unless SystemExit is raised)
and the execution stops.
This function returns func, which makes it possible to use it as a decorator.

atexit.unregister(func: Callable[Ellipsis, Any]) → None
Remove func from the list of functions to be run at termination.
unregister() silently does nothing if func was not previously registered. If func has been registered more than
once, every occurrence of that function in the atexit call stack will be removed.

audiobusio – Support for audio input and output over digital buses

The audiobusio module contains classes to provide access to audio IO over digital buses. These protocols are used to
communicate audio to other chips in the same circuit. It doesn’t include audio interconnect protocols such as S/PDIF.
All classes change hardware state and should be deinitialized when they are no longer needed. To do so, either call
definit() or use a context manager.

Output an I2S audio signal
Create a I2SOut object associated with the given pins.

Parameters

- **bit_clock** (Pin) – The bit clock (or serial clock) pin
- **word_select** (Pin) – The word select (or left/right clock) pin
- **data** (Pin) – The data pin
- **left_justified** (bool) – True when data bits are aligned with the word select clock. False when they are shifted by one to match classic I2S protocol.

Simple 8ksps 440 Hz sine wave on Metro M0 Express using UDA1334 Breakout:

```python
import audiobusio
import audiocore
import board
import array
import time
import math

# Generate one period of sine wave.
length = 8000 // 440
sine_wave = array.array("H", [0] * length)
for i in range(length):
    sine_wave[i] = int(math.sin(math.pi * 2 * i / length) * (2 ** 15) + 2 ** 15)

sine_wave = audiocore.RawSample(sine_wave, sample_rate=8000)
i2s = audiobusio.I2SOut(board.D1, board.D0, board.D9)
i2s.play(sine_wave, loop=True)
time.sleep(1)
i2s.stop()
```

Playing a wave file from flash:

```python
import board
import audiocore
import audiobusio
import digitalio

f = open("cplay-5.1-16bit-16khz.wav", "rb")
wav = audiocore.WaveFile(f)

a = audiobusio.I2SOut(board.D1, board.D0, board.D9)

print("playing")
a.play(wav)
while a.playing:
    pass
print("stopped")
```
playing:  bool
    True when the audio sample is being output. (read-only)

paused:  bool
    True when playback is paused. (read-only)

deinit() → None
    Deinitialises the I2SOut and releases any hardware resources for reuse.

__enter__() → I2SOut
    No-op used by Context Managers.

__exit__() → None
    Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

play(sample: circuitpython_typing.AudioSample, *, loop: bool = False) → None
    Plays the sample once when loop=False and continuously when loop=True. Does not block. Use playing to block.
    Sample must be an audiocore.WaveFile, audiocore.RawSample, audiomixer.Mixer or audiomp3.MP3Decoder.
    The sample itself should consist of 8 bit or 16 bit samples.

stop() → None
    Stops playback.

pause() → None
    Stops playback temporarily while remembering the position. Use resume to resume playback.

resume() → None
    Resumes sample playback after pause().

class audiobusio.PDMIn(clock_pin: microcontroller.Pin, data_pin: microcontroller.Pin, *, sample_rate: int = 16000, bit_depth: int = 8, mono: bool = True, oversample: int = 64, startup_delay: float = 0.11)

    Record an input PDM audio stream

    Create a PDMIn object associated with the given pins. This allows you to record audio signals from the given pins. Individual ports may put further restrictions on the recording parameters. The overall sample rate is determined by sample_rate x oversample, and the total must be 1MHz or higher, so sample_rate must be a minimum of 16000.

    Parameters

    • clock_pin (Pin) – The pin to output the clock to
    • data_pin (Pin) – The pin to read the data from
    • sample_rate (int) – Target sample_rate of the resulting samples. Check sample_rate for actual value. Minimum sample_rate is about 16000 Hz.
    • bit_depth (int) – Final number of bits per sample. Must be divisible by 8
    • mono (bool) – True when capturing a single channel of audio, captures two channels otherwise
    • oversample (int) – Number of single bit samples to decimate into a final sample. Must be divisible by 8
• **startup_delay** (*float*) -- seconds to wait after starting microphone clock to allow microphone to turn on. Most require only 0.01s; some require 0.1s. Longer is safer. Must be in range 0.0-1.0 seconds.

**Limitations:** On SAMD and RP2040, supports only 8 or 16 bit mono input, with 64x oversampling. On nRF52840, supports only 16 bit mono input at 16 kHz; oversampling is fixed at 64x. Not provided on nRF52833 for space reasons. Not available on Espressif.

For example, to record 8-bit unsigned samples to a buffer:

```python
import audiobusio
import board

# Prep a buffer to record into
b = bytearray(200)
with audiobusio.PDMIn(board.MICROPHONE_CLOCK, board.MICROPHONE_DATA, sample_rate=16000) as mic:
    mic.record(b, len(b))
```

To record 16-bit unsigned samples to a buffer:

```python
import audiobusio
import board

# Prep a buffer to record into.
b = array.array("H", [0] * 200)
with audiobusio.PDMIn(board.MICROPHONE_CLOCK, board.MICROPHONE_DATA, sample_rate=16000, bit_depth=16) as mic:
    mic.record(b, len(b))
```

**sample_rate**: *int*

The actual sample_rate of the recording. This may not match the constructed sample rate due to internal clock limitations.

**deinit()** → None

Deinitialises the PDMIn and releases any hardware resources for reuse.

**__enter__()** → *PDMIn*

No-op used by Context Managers.

**__exit__()** → None

Automatically deinitializes the hardware when exiting a context.

**record**(destination: circuitpython_typing.WriteableBuffer, destination_length: *int*) → None

Records destination_length bytes of samples to destination. This is blocking.

An IOError may be raised when the destination is too slow to record the audio at the given rate. For internal flash, writing all 1s to the file before recording is recommended to speed up writes.

**Returns**

The number of samples recorded. If this is less than destination_length, some samples were missed due to processing time.
audiocore – Support for audio samples

class audiocore.RawSample(buffer: circuitpython_typing.ReadableBuffer, *, channel_count: int = 1, sample_rate: int = 8000):
    A raw audio sample buffer in memory

    Create a RawSample based on the given buffer of values. If channel_count is more than 1 then each channel’s samples should alternate. In other words, for a two channel buffer, the first sample will be for channel 1, the second sample will be for channel two, the third for channel 1 and so on.

    Parameters

    • buffer (ReadableBuffer) – A buffer with samples
    • channel_count (int) – The number of channels in the buffer
    • sample_rate (int) – The desired playback sample rate

    Simple 8ksps 440 Hz sin wave:

    ```python
    import audiocore
    import audioio
    import board
    import array
    import time
    import math

    # Generate one period of sine wav.
    length = 8000 // 440
    sine_wave = array.array("h", [0] * length)
    for i in range(length):
        sine_wave[i] = int(math.sin(math.pi * 2 * i / length) * (2 ** 15))

    dac = audioio.AudioOut(board.SPEAKER)
    sine_wave = audiocore.RawSample(sine_wave)
    dac.play(sine_wave, loop=True)
    time.sleep(1)
    dac.stop()
    ```

    sample_rate: int | None

    32 bit value that dictates how quickly samples are played in Hertz (cycles per second). When the sample is looped, this can change the pitch output without changing the underlying sample. This will not change the sample rate of any active playback. Call play again to change it.

decinit() → None

    Deinitialises the RawSample and releases any hardware resources for reuse.

    __enter__() → RawSample

    No-op used by Context Managers.

    __exit__(self, _type, _value, _traceback) → None

    Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

class audiocore.WaveFile(file: str | BinaryIO, buffer: circuitpython_typing.WriteableBuffer):
    Load a wave file for audio playback
A .wav file prepped for audio playback. Only mono and stereo files are supported. Samples must be 8 bit unsigned or 16 bit signed. If a buffer is provided, it will be used instead of allocating an internal buffer, which can prevent memory fragmentation.

Load a .wav file for playback with `audioio.AudioOut` or `audiobusio.I2SOut`.

**Parameters**

- `file (Union[str, BinaryIO])` – The name of a wave file (preferred) or an already opened wave file

- `buffer (WriteableBuffer)` – Optional pre-allocated buffer, that will be split in half and used for double-buffering of the data. The buffer must be 8 to 1024 bytes long. If not provided, two 256 byte buffers are initially allocated internally.

Playing a wave file from flash:

```python
import board
import audiocore
import audioio
import digitalio

# Required for CircuitPlayground Express
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.switch_to_output(value=True)

wav = audiocore.WaveFile("cplay-5.1-16bit-16khz.wav")
a = audioio.AudioOut(board.A0)

print("playing")
a.play(wav)
while a.playing:
    pass
print("stopped")
```

- `sample_rate: int` – 32 bit value that dictates how quickly samples are loaded into the DAC in Hertz (cycles per second). When the sample is looped, this can change the pitch output without changing the underlying sample.

- `bits_per_sample: int` – Bits per sample. (read only)

- `channel_count: int` – Number of audio channels. (read only)

- `deinit() → None` – Deinitialises the WaveFile and releases all memory resources for reuse.

- `__enter__() → WaveFile` – No-op used by Context Managers.

- `__exit__() → None` – Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.
audioio – Support for audio output

The `audioio` module contains classes to provide access to audio IO.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See `Lifetime and Context Managers` for more info.

For more information on working with this module, refer to the CircuitPython Essentials Learn Guide.

Since CircuitPython 5, `RawSample` and `WaveFile` are moved to `audiocore`, and `Mixer` is moved to `audiomixer`.

For compatibility with CircuitPython 4.x, some builds allow the items in `audiocore` to be imported from `audioio`. This will be removed for all boards in a future build of CircuitPython.

```python
class audioio.AudioOut(left_channel: microcontroller.Pin, *, right_channel: microcontroller.Pin | None = None, quiescent_value: int = 32768)

Output an analog audio signal

Create a AudioOut object associated with the given pin(s). This allows you to play audio signals out on the given pin(s).

Parameters

- `left_channel` (Pin) – The pin to output the left channel to
- `right_channel` (Pin) – The pin to output the right channel to
- `quiescent_value` (int) – The output value when no signal is present. Samples should start and end with this value to prevent audible popping.
```

Simple 8ksps 440 Hz sin wave:

```python
import audiocore
import audioio
import board
import array
import time
import math

# Generate one period of sine wav.
length = 8000 // 440
sine_wave = array.array("H", [0] * length)
for i in range(length):
    sine_wave[i] = int(math.sin(math.pi * 2 * i / length) * (2 ** 15) + 2 ** 15)

dac = audioio.AudioOut(board.SPEAKER)
sine_wave = audiocore.RawSample(sine_wave, sample_rate=8000)
dac.play(sine_wave, loop=True)
time.sleep(1)
dac.stop()
```

Playing a wave file from flash:

```python
import board
import audioio
import digitalio

# Required for CircuitPlayground Express
```
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.switch_to_output(value=True)

data = open("cplay-5.1-16bit-16khz.wav", "rb")
wav = audiocore.WaveFile(data)
a = audioio.AudioOut(board.A0)

print("playing")
a.play(wav)
while a.playing:
    pass
print("stopped")

playing: bool
    True when an audio sample is being output even if paused. (read-only)

paused: bool
    True when playback is paused. (read-only)

deinit() → None
    Deinitialises the AudioOut and releases any hardware resources for reuse.

__enter__() → AudioOut
    No-op used by Context Managers.

__exit__() → None
    Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

play(sample: circuitpython_typing.AudioSample, *, loop: bool = False) → None
    Plays the sample once when loop=False and continuously when loop=True. Does not block. Use playing to block.

    Sample must be an audiocore.WaveFile, audiocore.RawSample, audiomixer.Mixer or audiomp3.MP3Decoder.

    The sample itself should consist of 16 bit samples. Microcontrollers with a lower output resolution will use the highest order bits to output. For example, the SAMD21 has a 10 bit DAC that ignores the lowest 6 bits when playing 16 bit samples.

stop() → None
    Stops playback and resets to the start of the sample.

pause() → None
    Stops playback temporarily while remembering the position. Use resume to resume playback.

resume() → None
    Resumes sample playback after pause().
**audiomixer – Support for audio mixing**

**class audiomixer.Mixer**

```python
class audiomixer.Mixer(voice_count: int = 2, buffer_size: int = 1024, channel_count: int = 2,
                       bits_per_sample: int = 16, samples_signed: bool = True, sample_rate: int = 8000)
```

Mixes one or more audio samples together into one sample.

Create a Mixer object that can mix multiple channels with the same sample rate. Samples are accessed and controlled with the mixer’s `audiomixer.MixerVoice` objects.

**Parameters**

- `voice_count (int)` – The maximum number of voices to mix
- `buffer_size (int)` – The total size in bytes of the buffers to mix into
- `channel_count (int)` – The number of channels the source samples contain. 1 = mono; 2 = stereo.
- `bits_per_sample (int)` – The bits per sample of the samples being played
- `samples_signed (bool)` – Samples are signed (True) or unsigned (False)
- `sample_rate (int)` – The sample rate to be used for all samples

Playing a wave file from flash:

```python
import board
import audioio
import audiocore
import audiomixer
import digitalio

da = audioio.AudioOut(board.A0)
music = audiocore.WaveFile(open("cplay-5.1-16bit-16khz.wav", "rb"))
drum = audiocore.WaveFile(open("drum.wav", "rb"))
mixer = audiomixer.Mixer(voice_count=2, sample_rate=16000, channel_count=1,
                       bits_per_sample=16, samples_signed=True)

print("playing")
# Have AudioOut play our Mixer source
a.play(mixer)
# Play the first sample voice
mixer.voice[0].play(music)
while mixer.playing:
    # Play the second sample voice
    mixer.voice[1].play(drum)
    time.sleep(1)
print("stopped")
```

**playing:** `bool`

True when any voice is being output. (read-only)

**sample_rate:** `int`

32 bit value that dictates how quickly samples are played in Hertz (cycles per second).

**voice:** `Tuple[MixerVoice, Ellipsis]`

A tuple of the mixer’s `audiomixer.MixerVoice` object(s).
```python
>>> mixer.voice
(<MixerVoice>,)
```

**deinit() → None**
Deinitialises the Mixer and releases any hardware resources for reuse.

**__enter__() → Mixer**
No-op used by Context Managers.

**__exit__() → None**
Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**play(sample: circuitpython_typing.AudioSample, *, voice: int = 0, loop: bool = False) → None**
Plays the sample once when loop=False and continuously when loop=True. Does not block. Use *playing* to block.

Sample must be an `audiocore.WaveFile`, `audiocore.RawSample`, `audiomixer.Mixer` or `audiomp3.MP3Decoder`.

The sample must match the Mixer's encoding settings given in the constructor.

**stop_voice(voice: int = 0) → None**
Stops playback of the sample on the given voice.

**class audiomixer.MixerVoice**
Voice objects used with Mixer

Used to access and control samples with `audiomixer.Mixer`.

MixerVoice instance object(s) created by `audiomixer.Mixer`.

**level: float**
The volume level of a voice, as a floating point number between 0 and 1.

**playing: bool**
True when this voice is being output. (read-only)

**play(sample: circuitpython_typing.AudioSample, *, loop: bool = False) → None**
Plays the sample once when loop=False, and continuously when loop=True. Does not block. Use *playing* to block.

Sample must be an `audiocore.WaveFile`, `audiocore.RawSample`, `audiomixer.Mixer` or `audiomp3.MP3Decoder`.

The sample must match the `audiomixer.Mixer`'s encoding settings given in the constructor.

**stop() → None**
Stops playback of the sample on this voice.
audiomp3 – Support for MP3-compressed audio files

For more information about working with MP3 files in CircuitPython, see this CircuitPython Essentials Learn guide page.

class audiomp3.MP3Decoder(file: str | BinaryIO, buffer: circuitpython_typing.WriteableBuffer)

Load a mp3 file for audio playback

Note: MP3Decoder uses a lot of contiguous memory, so care should be given to optimizing memory usage. More information and recommendations can be found here: https://learn.adafruit.com/Memory-saving-tips-for-CircuitPython/reducing-memory-fragmentation

Load a .mp3 file for playback with audioio.AudioOut or audiobusio.I2SOut.

Parameters

- **file** (Union[str, BinaryIO]) – The name of a mp3 file (preferred) or an already opened mp3 file
- **buffer** (WriteableBuffer) – Optional pre-allocated buffer, that will be split in half and used for double-buffering of the data. If not provided, two buffers are allocated internally. The specific buffer size required depends on the mp3 file.

Playback of mp3 audio is CPU intensive, and the exact limit depends on many factors such as the particular microcontroller, SD card or flash performance, and other code in use such as displayio. If playback is garbled, skips, or plays as static, first try using a “simpler” mp3:

- Use constant bit rate (CBR) not VBR or ABR (variable or average bit rate) when encoding your mp3 file
- Use a lower sample rate (e.g., 11.025kHz instead of 48kHz)
- Use a lower bit rate (e.g., 32kbit/s instead of 256kbit/s)

Reduce activity taking place at the same time as mp3 playback. For instance, only update small portions of a displayio screen if audio is playing. Disable auto-refresh and explicitly call refresh.

Playing a mp3 file from flash:

```python
import board
import audiomp3
import audioio
import digitalio

# Required for CircuitPlayground Express
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.switch_to_output(value=True)

mp3 = audiomp3.MP3Decoder("cplay-16bit-16khz-64kbps.mp3")
a = audioio.AudioOut(board.A0)

print("playing")
a.play(mp3)
while a.playing:
    pass
print("stopped")
```
file: BinaryIO
File to play back.
sample_rate: int
  32 bit value that dictates how quickly samples are loaded into the DAC in Hertz (cycles per second). When
  the sample is looped, this can change the pitch output without changing the underlying sample.
bits_per_sample: int
  Bits per sample. (read only)
channel_count: int
  Number of audio channels. (read only)
rms_level: float
  The RMS audio level of a recently played moment of audio. (read only)
samples_decoded: int
  The number of audio samples decoded from the current file. (read only)
deinit() → None
  Deinitialises the MP3 and releases all memory resources for reuse.
__enter__() → MP3Decoder
  No-op used by Context Managers.
__exit__() → None
  Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for
  more info.
open(filepath: str) → None
  Takes in the name of a mp3 file, opens it, and replaces the old playback file.

audiopwmio – Audio output via digital PWM

The audiopwmio module contains classes to provide access to audio IO.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues
after use. To do so, either call deinit() or use a context manager. See Lifetime and ContextManagers for more info.

Since CircuitPython 5, Mixer, RawSample and WaveFile are moved to audiocore.

class audiopwmio.PWMAudioOut(left_channel: microcontroller.Pin, *, right_channel: microcontroller.Pin | None
  = None, quiescent_value: int = 32768)

  Output an analog audio signal by varying the PWM duty cycle.

  Create a PWMAudioOut object associated with the given pin(s). This allows you to play audio signals out on
  the given pin(s). In contrast to mod:audioio, the pin(s) specified are digital pins, and are driven with a device-
  dependent PWM signal.

  Parameters
    • left_channel (Pin) – The pin to output the left channel to
    • right_channel (Pin) – The pin to output the right channel to
    • quiescent_value (int) – The output value when no signal is present. Samples should
      start and end with this value to prevent audible popping.
Limitations: On mimxrt10xx, low sample rates may have an audible “carrier” frequency. The manufacturer datasheet states that the “MQS” peripheral is intended for 44 kHz or 48kHz input signals.

Simple 8ksps 440 Hz sin wave:

```python
import audiocore
import audiopwmio
import board
import array
import time
import math

# Generate one period of sine wav.
length = 8000 // 440
sine_wave = array.array("H", [0] * length)
for i in range(length):
    sine_wave[i] = int(math.sin(math.pi * 2 * i / length) * (2 ** 15) + 2 ** 15)

dac = audiopwmio.PWMAudioOut(board.SPEAKER)
sine_wave = audiocore.RawSample(sine_wave, sample_rate=8000)
dac.play(sine_wave, loop=True)
time.sleep(1)
dac.stop()
```

Playing a wave file from flash:

```python
import board
import audiocore
import audiopwmio
import digitalio

# Required for CircuitPlayground Express
speaker_enable = digitalio.DigitalInOut(board.SPEAKER_ENABLE)
speaker_enable.switch_to_output(value=True)

data = open("cplay-5.1-16bit-16khz.wav", "rb")
wav = audiocore.WaveFile(data)
a = audiopwmio.PWMAudioOut(board.SPEAKER)

print("playing")
a.play(wav)
while a.playing:
    pass
print("stopped")
```

playing: bool

True when an audio sample is being output even if paused. (read-only)

paused: bool

True when playback is paused. (read-only)

deinit() → None

Deinitialises the PWMAudioOut and releases any hardware resources for reuse.

__enter__() → PWMAudioOut

No-op used by Context Managers.
__exit__() → None

Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

\[\text{play}(\text{sample: circuitpython_typing.AudioSample}, *, \text{loop: bool} = \text{False}) \rightarrow \text{None}\]

Plays the sample once when loop=False and continuously when loop=True. Does not block. Use \text{playing} to block.

Sample must be an \text{audiocore.WaveFile}, \text{audiocore.RawSample}, \text{audiomixer.Mixer} or \text{audiomp3.MP3Decoder}.

The sample itself should consist of 16 bit samples. Microcontrollers with a lower output resolution will use the highest order bits to output.

\[\text{stop()} \rightarrow \text{None}\]

Stops playback and resets to the start of the sample.

\[\text{pause()} \rightarrow \text{None}\]

Stops playback temporarily while remembering the position. Use \text{resume} to resume playback.

\[\text{resume()} \rightarrow \text{None}\]

Resumes sample playback after \text{pause()}.

\[\text{bitbangio} – \text{Digital protocols implemented by the CPU}\]

The \text{bitbangio} module contains classes to provide digital bus protocol support regardless of whether the underlying hardware exists to use the protocol.

First try to use \text{busio} module instead which may utilize peripheral hardware to implement the protocols. Native implementations will be faster than bitbanged versions and have more capabilities.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call \text{deinit()} or use a context manager. See \text{Lifetime and ContextManagers} for more info.

For example:

```python
import bitbangio
from board import *

i2c = bitbangio.I2C(SCL, SDA)
p = i2c.scan()
i2c.deinit()
```

This example will initialize the the device, run \text{scan()} and then \text{deinit()} the hardware. The last step is optional because CircuitPython automatically resets hardware after a program finishes.


Two wire serial protocol

I2C is a two-wire protocol for communicating between devices. At the physical level it consists of 2 wires: SCL and SDA, the clock and data lines respectively.

\text{See also:}

Using this class directly requires careful lock management. Instead, use \text{I2CDevice} to manage locks.
See also:

Using this class to directly read registers requires manual bit unpacking. Instead, use an existing driver or make one with Register data descriptors.

Parameters

- **scl** *(Pin)* – The clock pin
- **sda** *(Pin)* – The data pin
- **frequency** *(int)* – The clock frequency of the bus
- **timeout** *(int)* – The maximum clock stretching timeout in microseconds

**deinit()** → None

Releases control of the underlying hardware so other classes can use it.

**__enter__()** → **I2C**

No-op used in Context Managers.

**__exit__()** → None

Automatically deinitializes the hardware on context exit. See *Lifetime and ContextManagers* for more info.

**scan()** → List[int]

Scan all I2C addresses between 0x08 and 0x77 inclusive and return a list of those that respond. A device responds if it pulls the SDA line low after its address (including a read bit) is sent on the bus.

**try_lock()** → bool

Attempts to grab the I2C lock. Returns True on success.

**unlock()** → None

Releases the I2C lock.

**readfrom_into(address: int, buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int = sys.maxsize)** → None

Read into *buffer* from the device selected by *address*. The number of bytes read will be the length of *buffer*. At least one byte must be read.

If *start* or *end* is provided, then the buffer will be sliced as if *buffer[start:end]*. This will not cause an allocation like *buf[start:end]* will so it saves memory.

Parameters

- **address** *(int)* – 7-bit device address
- **buffer** *(WriteableBuffer)* – buffer to write into
- **start** *(int)* – Index to start writing at
- **end** *(int)* – Index to write up to but not include

**writeto(address: int, buffer: circuitpython_typing.ReadableBuffer, *, start: int = 0, end: int = sys.maxsize)** → None

Write the bytes from *buffer* to the device selected by *address* and then transmits a stop bit. Use *writeto_then_readfrom* when needing a write, no stop and repeated start before a read.

If *start* or *end* is provided, then the buffer will be sliced as if *buffer[start:end]* were passed, but without copying the data. The number of bytes written will be the length of *buffer[start:end]*.

Writing a buffer or slice of length zero is permitted, as it can be used to poll for the existence of a device.
Parameters

- **address** (*int*) – 7-bit device address
- **buffer** (*ReadableBuffer*) – buffer containing the bytes to write
- **start** (*int*) – beginning of buffer slice
- **end** (*int*) – end of buffer slice; if not specified, use `len(buffer)`

```python
writeto_then_readfrom(address: int, out_buffer: circuitpython_typing.ReadableBuffer, in_buffer: circuitpython_typing.ReadableBuffer, *, out_start: int = 0, out_end: int = sys.maxsize, in_start: int = 0, in_end: int = sys.maxsize) → None
```

Write the bytes from `out_buffer` to the device selected by `address`, generate no stop bit, generate a repeated start and read into `in_buffer`. `out_buffer` and `in_buffer` can be the same buffer because they are used sequentially.

If `out_start` or `out_end` is provided, then the buffer will be sliced as if `out_buffer[out_start:out_end]` were passed, but without copying the data. The number of bytes written will be the length of `out_buffer[start:end]`.

If `in_start` or `in_end` is provided, then the input buffer will be sliced as if `in_buffer[in_start:in_end]` were passed, The number of bytes read will be the length of `out_buffer[in_start:in_end]`.

Parameters

- **address** (*int*) – 7-bit device address
- **out_buffer** (*ReadableBuffer*) – buffer containing the bytes to write
- **in_buffer** (*WriteableBuffer*) – buffer to write into
- **out_start** (*int*) – beginning of `out_buffer` slice
- **out_end** (*int*) – end of `out_buffer` slice; if not specified, use `len(out_buffer)`
- **in_start** (*int*) – beginning of `in_buffer` slice
- **in_end** (*int*) – end of `in_buffer` slice; if not specified, use `len(in_buffer)`

```python
class bitbangio.SPI(clock: microcontroller.Pin, MOSI: microcontroller.Pin | None = None, MISO: microcontroller.Pin | None = None)
```

A 3-4 wire serial protocol

SPI is a serial protocol that has exclusive pins for data in and out of the main device. It is typically faster than **I2C** because a separate pin is used to select a device rather than a transmitted address. This class only manages three of the four SPI lines: `clock`, `MOSI`, `MISO`. Its up to the client to manage the appropriate select line, often abbreviated `CS` or `SS`. (This is common because multiple secondaries can share the `clock`, `MOSI` and `MISO` lines and therefore the hardware.)

Construct an SPI object on the given pins.

See also:

Using this class directly requires careful lock management. Instead, use `SPIDevice` to manage locks.

See also:

Using this class to directly read registers requires manual bit unpacking. Instead, use an existing driver or make one with `Register` data descriptors.

Parameters

- **clock** (*Pin*) – the pin to use for the clock.
• **MOSI** (Pin) – the Main Out Selected In pin.
• **MISO** (Pin) – the Main In Selected Out pin.

`deinit()` → None
Turn off the SPI bus.

`__enter__` () → `SPI`
No-op used by Context Managers.

`__exit__` () → None
Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

`configure(*, baudrate: int = 100000, polarity: int = 0, phase: int = 0, bits: int = 8) → None`
Configures the SPI bus. Only valid when locked.

**Parameters**

• `baudrate (int)` – the clock rate in Hertz
• `polarity (int)` – the base state of the clock line (0 or 1)
• `phase (int)` – the edge of the clock that data is captured. First (0) or second (1). Rising or falling depends on clock polarity.
• `bits (int)` – the number of bits per word

`try_lock()` → `bool`
Attempts to grab the SPI lock. Returns True on success.

**Returns**

True when lock has been grabbed

**Return type**

`bool`

`unlock()` → None
Releases the SPI lock.

`write(buf: circuitpython_typing.ReadableBuffer, *, start: int = 0, end: int = sys.maxsize) → None`
Write the data contained in `buf`. Requires the SPI being locked. If the buffer is empty, nothing happens.

If `start` or `end` is provided, then the buffer will be sliced as if `buffer[start:end]` were passed, but without copying the data. The number of bytes written will be the length of `buffer[start:end]`.

**Parameters**

• `buffer (ReadableBuffer)` – buffer containing the bytes to write
• `start (int)` – beginning of buffer slice
• `end (int)` – end of buffer slice; if not specified, use `len(buffer)`

`readinto(buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int = sys.maxsize, write_value: int = 0) → None`
Read into `buffer` while writing `write_value` for each byte read. The SPI object must be locked. If the number of bytes to read is 0, nothing happens.

If `start` or `end` is provided, then the buffer will be sliced as if `buffer[start:end]` were passed. The number of bytes read will be the length of `buffer[start:end]`.

**Parameters**
• **buffer** (*WriteableBuffer*) – read bytes into this buffer
• **start** (*int*) – beginning of buffer slice
• **end** (*int*) – end of buffer slice; if not specified, use `len(buffer)`
• **write_value** (*int*) – value to write while reading

```python
write_readinto(out_buffer: circuitpython_typing.ReadableBuffer, in_buffer: circuitpython_typing.WriteableBuffer, *, out_start: int = 0, out_end: int = sys.maxsize, in_start: int = 0, in_end: int = sys.maxsize) → None
```

Write out the data in `out_buffer` while simultaneously reading data into `in_buffer`. The SPI object must be locked.

If `out_start` or `out_end` is provided, then the buffer will be sliced as if `out_buffer[out_start:out_end]` were passed, but without copying the data. The number of bytes written will be the length of `out_buffer[out_start:out_end]`.

If `in_start` or `in_end` is provided, then the input buffer will be sliced as if `in_buffer[in_start:in_end]` were passed. The number of bytes read will be the length of `out_buffer[in_start:in_end]`.

The lengths of the slices defined by `out_buffer[out_start:out_end]` and `in_buffer[in_start:in_end]` must be equal. If buffer slice lengths are both 0, nothing happens.

**Parameters**

• **out_buffer** (*ReadableBuffer*) – write out bytes from this buffer
• **in_buffer** (*WriteableBuffer*) – read bytes into this buffer
• **out_start** (*int*) – beginning of `out_buffer` slice
• **out_end** (*int*) – end of `out_buffer` slice; if not specified, use `len(out_buffer)`
• **in_start** (*int*) – beginning of `in_buffer` slice
• **in_end** (*int*) – end of `in_buffer` slice; if not specified, use `len(in_buffer)`

---

**bitmaptools** – Collection of bitmap manipulation tools

**Note:** If you’re looking for information about displaying bitmaps on screens in CircuitPython, see this Learn guide for information about using the `displayio` module.

```python
bitmaptools.rotozoom(dest_bitmap: displayio.Bitmap, source_bitmap: displayio.Bitmap, *, ox: int, oy: int, dest_clip0: Tuple[int, int], dest_clip1: Tuple[int, int], px: int, py: int, source_clip0: Tuple[int, int], source_clip1: Tuple[int, int], angle: float, scale: float, skip_index: int) → None
```

Inserts the source bitmap region into the destination bitmap with rotation (angle), scale and clipping (both on source and destination bitmaps).

**Parameters**

• **dest_bitmap** (*bitmap*) – Destination bitmap that will be copied into
• **source_bitmap** (*bitmap*) – Source bitmap that contains the graphical region to be copied
• **ox** (*int*) – Horizontal pixel location in destination bitmap where source bitmap point (px,py) is placed. Defaults to None which causes it to use the horizontal midway point of the destination bitmap.
• **oy** *(int)* – Vertical pixel location in destination bitmap where source bitmap point (px,py) is placed. Defaults to None which causes it to use the vertical midway point of the destination bitmap.

• **dest_clip0** *(Tuple[int,int])* – First corner of rectangular destination clipping region that constrains region of writing into destination bitmap

• **dest_clip1** *(Tuple[int,int])* – Second corner of rectangular destination clipping region that constrains region of writing into destination bitmap

• **px** *(int)* – Horizontal pixel location in source bitmap that is placed into the destination bitmap at (ox,oy). Defaults to None which causes it to use the horizontal midway point in the source bitmap.

• **py** *(int)* – Vertical pixel location in source bitmap that is placed into the destination bitmap at (ox,oy). Defaults to None which causes it to use the vertical midway point in the source bitmap.

• **source_clip0** *(Tuple[int,int])* – First corner of rectangular source clipping region that constrains region of reading from the source bitmap

• **source_clip1** *(Tuple[int,int])* – Second corner of rectangular source clipping region that constrains region of reading from the source bitmap

• **angle** *(float)* – Angle of rotation, in radians (positive is clockwise direction). Defaults to None which gets treated as 0.0 radians or no rotation.

• **scale** *(float)* – Scaling factor. Defaults to None which gets treated as 1.0 or same as original source size.

• **skip_index** *(int)* – Bitmap palette index in the source that will not be copied, set to None to copy all pixels


Alpha blend the two source bitmaps into the destination.

It is permitted for the destination bitmap to be one of the two source bitmaps.

**Parameters**

• **dest_bitmap** *(bitmap)* – Destination bitmap that will be written into

• **source_bitmap_1** *(bitmap)* – The first source bitmap

• **source_bitmap_2** *(bitmap)* – The second source bitmap

• **factor1** *(float)* – The proportion of bitmap 1 to mix in

• **factor2** *(float)* – The proportion of bitmap 2 to mix in. If specified as None, 1-factor1 is used. Usually the proportions should sum to 1.

• **colorspace** *(displayio.Colorspace)* – The colorspace of the bitmaps. They must all have the same colorspace. Only the following colorspaces are permitted: L8, RGB565, RGB565_SWAPPED, BGR565 and BGR565_SWAPPED.

For the L8 colorspace, the bitmaps must have a bits-per-value of 8. For the RGB colorspaces, they must have a bits-per-value of 16.

**bitmaptools.fill_region** *(dest_bitmap: displayio.Bitmap, x1: int, y1: int, x2: int, y2: int, value: int) → None

Draws the color value into the destination bitmap within the rectangular region bounded by (x1,y1) and (x2,y2), exclusive.
Parameters

- **dest_bitmap** *(bitmap)* – Destination bitmap that will be written into
- **x1** *(int)* – x-pixel position of the first corner of the rectangular fill region
- **y1** *(int)* – y-pixel position of the first corner of the rectangular fill region
- **x2** *(int)* – x-pixel position of the second corner of the rectangular fill region (exclusive)
- **y2** *(int)* – y-pixel position of the second corner of the rectangular fill region (exclusive)
- **value** *(int)* – Bitmap palette index that will be written into the rectangular fill region in the destination bitmap

bitmaptools.boundary_fill(dest_bitmap: displayio.Bitmap, x: int, y: int, fill_color_value: int, replaced_color_value: int) → None

Draws the color value into the destination bitmap enclosed area of pixels of the background_value color. Like “Paint Bucket” fill tool.

Parameters

- **dest_bitmap** *(bitmap)* – Destination bitmap that will be written into
- **x** *(int)* – x-pixel position of the first pixel to check and fill if needed
- **y** *(int)* – y-pixel position of the first pixel to check and fill if needed
- **fill_color_value** *(int)* – Bitmap palette index that will be written into the enclosed area in the destination bitmap
- **replaced_color_value** *(int)* – Bitmap palette index that will filled with the value color in the enclosed area in the destination bitmap

bitmaptools.draw_line(dest_bitmap: displayio.Bitmap, x1: int, y1: int, x2: int, y2: int, value: int) → None

Draws a line into a bitmap specified two endpoints (x1,y1) and (x2,y2).

Parameters

- **dest_bitmap** *(bitmap)* – Destination bitmap that will be written into
- **x1** *(int)* – x-pixel position of the line’s first endpoint
- **y1** *(int)* – y-pixel position of the line’s first endpoint
- **x2** *(int)* – x-pixel position of the line’s second endpoint
- **y2** *(int)* – y-pixel position of the line’s second endpoint
- **value** *(int)* – Bitmap palette index that will be written into the line in the destination bitmap


Draw a polygon connecting points on provided bitmap with provided value

Parameters

- **dest_bitmap** *(bitmap)* – Destination bitmap that will be written into
- **xs** *(ReadableBuffer)* – x-pixel position of the polygon’s vertices
- **ys** *(ReadableBuffer)* – y-pixel position of the polygon’s vertices
- **value** *(int)* – Bitmap palette index that will be written into the line in the destination bitmap
- **close** *(bool)* – (Optional) Whether to connect first and last point. (True)
```python
import board
import displayio
import bitmaptools

display = board.DISPLAY
main_group = displayio.Group()
display.root_group = main_group

palette = displayio.Palette(3)
palette[0] = 0xffffff
palette[1] = 0x0000ff
palette[2] = 0xff0000

bmp = displayio.Bitmap(128,128, 3)
bmp.fill(0)

xs = bytes([4, 101, 101, 19])
ys = bytes([4, 19, 121, 101])
bitmaptools.draw_polygon(bmp, xs, ys, 1)

xs = bytes([14, 60, 110])
ys = bytes([14, 24, 90])
bitmaptools.draw_polygon(bmp, xs, ys, 2)

tilegrid = displayio.TileGrid(bitmap=bmp, pixel_shader=palette)
main_group.append(tilegrid)

while True:
    pass
```

bitmaptools.arrayblit(bitmap: displayio.Bitmap, data: circuitpython_typing.ReadableBuffer, x1: int = 0, y1: int = 0, x2: int | None = None, y2: int | None = None, skip_index: int | None = None) → None

Inserts pixels from data into the rectangle of width×height pixels with the upper left corner at (x,y).

The values from data are taken modulo the number of color values available in the destination bitmap.

If x1 or y1 are not specified, they are taken as 0. If x2 or y2 are not specified, or are given as -1, they are taken as the width and height of the image.

The coordinates affected by the blit are x1 <= x < x2 and y1 <= y < y2.

data must contain at least as many elements as required. If it contains excess elements, they are ignored.

The blit takes place by rows, so the first elements of data go to the first row, the next elements to the next row, and so on.

**Parameters**

- **bitmap** (displayio.Bitmap) – A writable bitmap
- **data** (ReadableBuffer) – Buffer containing the source pixel values
- **x1** (int) – The left corner of the area to blit into (inclusive)
- **y1** (int) – The top corner of the area to blit into (inclusive)
- **x2** (int) – The right of the area to blit into (exclusive)
• **y2 (int)** – The bottom corner of the area to blit into (exclusive)

• **skip_index (int)** – Bitmap palette index in the source that will not be copied, set to None to copy all pixels

```python
```

Reads from a binary file into a bitmap.

The file must be positioned so that it consists of `bitmap.height` rows of pixel data, where each row is the smallest multiple of `element_size` bytes that can hold `bitmap.width` pixels.

The bytes in an element can be optionally swapped, and the pixels in an element can be reversed. Also, the row loading direction can be reversed, which may be required for loading certain bitmap files.

This function doesn’t parse image headers, but is useful to speed up loading of uncompressed image formats such as PCF glyph data.

**Parameters**

• **bitmap (displayio.Bitmap)** – A writable bitmap

• **file (BinaryIO)** – A file opened in binary mode

• **bits_per_pixel (int)** – Number of bits per pixel. Values 1, 2, 4, 8, 16, 24, and 32 are supported;

• **element_size (int)** – Number of bytes per element. Values of 1, 2, and 4 are supported, except that 24 `bits_per_pixel` requires 1 byte per element.

• **reverse_pixels_in_element (bool)** – If set, the first pixel in a word is taken from the Most Significant Bits; otherwise, it is taken from the Least Significant Bits.

• **swap_bytes_in_element (bool)** – If the `element_size` is not 1, then reverse the byte order of each element read.

• **reverse_rows (bool)** – Reverse the direction of the row loading (required for some bitmap images).

```python
class bitmaptools.DitherAlgorithm
```

Identifies the algorithm for dither to use

**Atkinson:** `DitherAlgorithm`

The classic Atkinson dither, often associated with the Hypercard esthetic

**FloydStenberg:** `DitherAlgorithm`

The Floyd-Stenberg dither

```python
```

Convert the input image into a 2-level output image using the given dither algorithm.

**Parameters**

• **dest_bitmap (bitmap)** – Destination bitmap. It must have a value_count of 2 or 65536. The stored values are 0 and the maximum pixel value.

• **source_bitmap (bitmap)** – Source bitmap that contains the graphical region to be dithered. It must have a value_count of 65536.

• **colorspace** – The colorspace of the image. The supported colorspaces are RGB565, BGR565, RGB565_SWAPPED, and BGR565_SWAPPED
• **algorithm** – The dither algorithm to use, one of the `DitherAlgorithm` values.

`bitmaptools.draw_circle(dest_bitmap: displayio.Bitmap, x: int, y: int, radius: int, value: int) → None`

Draws a circle into a bitmap specified using a center (x0, y0) and radius r.

**Parameters**

- **dest_bitmap** (`bitmap`) – Destination bitmap that will be written into
- **x** (`int`) – x-pixel position of the circle’s center
- **y** (`int`) – y-pixel position of the circle’s center
- **radius** (`int`) – circle’s radius
- **value** (`int`) – Bitmap palette index that will be written into the circle in the destination bitmap

```python
import board
import displayio
import bitmaptools

display = board.DISPLAY
main_group = displayio.Group()
display.root_group = main_group

palette = displayio.Palette(2)
palette[0] = 0xffffff
palette[1] = 0x440044

bmp = displayio.Bitmap(128, 128, 2)
bmp.fill(0)

bitmaptools.circle(64, 64, 32, 1)

tilegrid = displayio.TileGrid(bitmap=bmp, pixel_shader=palette)
main_group.append(tilegrid)

while True:
    pass
```

**bitops – Routines for low-level manipulation of binary data**

`bitops.bit_transpose(input: circuitpython_typing.ReadableBuffer, output: circuitpython_typing.WriteableBuffer, width: int = 8) → circuitpython_typing.WriteableBuffer`

“Transpose” a buffer by assembling each output byte with bits taken from each of width different input bytes.

This can be useful to convert a sequence of pixel values into a single stream of bytes suitable for sending via a parallel conversion method.

The number of bytes in the input buffer must be a multiple of the width, and the width can be any value from 2 to 8. If the width is fewer than 8, then the remaining (less significant) bits of the output are set to zero.

Let stride = len(input) // width. Then the first byte is made out of the most significant bits of [input[0], input[stride], input[2*stride], ...]. The second byte is made out of the second bits, and so on until
the 8th output byte which is made of the first bits of \texttt{input[1]}\, \texttt{input[1+stride]}\, \texttt{input[2*stride]}\, ..

The required output buffer size is \texttt{len(input) * 8 // width}.

Returns the output buffer.

\textbf{board – Board specific pin names}

Common container for board base pin names. These will vary from board to board so don’t expect portability when using this module.

Another common use of this module is to use serial communication buses with the default pins and settings. For more information about serial communication in CircuitPython, see the \texttt{busio}.

For more information regarding the typical usage of \texttt{board}, refer to the CircuitPython Essentials Learn guide

| Warning: The board module varies by board. The APIs documented here may or may not be available on a specific board. |

\begin{verbatim}
board.board_id: str
    Board ID string. The unique identifier for the board model in circuitpython, as well as on circuitpython.org. Example: “hallowing_m0_express”.

board.I2C() \to busio.I2C
    Returns the \texttt{busio.I2C} object for the board’s designated I2C bus(es). The object created is a singleton, and uses the default parameter values for \texttt{busio.I2C}.

board.SPI() \to busio.SPI
    Returns the \texttt{busio.SPI} object for the board’s designated SPI bus(es). The object created is a singleton, and uses the default parameter values for \texttt{busio.SPI}.

board.UART() \to busio.UART
    Returns the \texttt{busio.UART} object for the board’s designated UART bus(es). The object created is a singleton, and uses the default parameter values for \texttt{busio.UART}.

\end{verbatim}

\textbf{busio – Hardware accelerated external bus access}

The \texttt{busio} module contains classes to support a variety of serial protocols.

When the microcontroller does not support the behavior in a hardware accelerated fashion it may internally use a bitbang routine. However, if hardware support is available on a subset of pins but not those provided, then a \texttt{RuntimeError} will be raised. Use the \texttt{bitbangio} module to explicitly bitbang a serial protocol on any general purpose pins.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call \texttt{deinit()} or use a context manager. See \textit{Lifetime and Context Managers} for more info.

For example:

\begin{verbatim}
import busio
from board import *

i2c = busio.I2C(SCL, SDA)
i2c.try_lock()
\end{verbatim}

(continues on next page)
print(i2c.scan())
i2c.unlock()
i2c.deinit()

This example will initialize the device, lock the I2C bus, run `scan()`, unlock the bus, and then `deinit()` the hardware. The last step is optional because CircuitPython automatically resets hardware after a program finishes.

Note that drivers will typically handle communication if provided the bus instance (such as `busio.I2C(board.SCL, board.SDA)`), and that many of the methods listed here are lower level functionalities that are needed for working with custom drivers.

Tutorial for I2C and SPI: https://learn.adafruit.com/circuitpython-basics-i2c-and-spi

Tutorial for UART: https://learn.adafruit.com/circuitpython-essentials/circuitpython-uart-serial


Two wire serial protocol

I2C is a two-wire protocol for communicating between devices. At the physical level it consists of 2 wires: SCL and SDA, the clock and data lines respectively.

See also:
Using this class directly requires careful lock management. Instead, use `I2CDevice` to manage locks.

See also:
Using this class to directly read registers requires manual bit unpacking. Instead, use an existing driver or make one with `Register` data descriptors.

Parameters

- `scl` (Pin) – The clock pin
- `sda` (Pin) – The data pin
- `frequency` (int) – The clock frequency in Hertz
- `timeout` (int) – The maximum clock stretching time out - (used only for `bitbangio.I2C`; ignored for `busio.I2C`)

deinit() → None

Releases control of the underlying hardware so other classes can use it.

__enter__() → I2C

No-op used in Context Managers.

__exit__() → None

Automatically deinitializes the hardware on context exit. See `Lifetime and Context Managers` for more info.

scan() → List[int]

Scan all I2C addresses between 0x08 and 0x77 inclusive and return a list of those that respond.

Returns

List of device ids on the I2C bus

Return type

list
try_lock() → bool
Attempts to grab the I2C lock. Returns True on success.

Returns
True when lock has been grabbed

Return type
bool
unlock() → None
Releases the I2C lock.

readfrom_into(address: int, buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int = sys.maxsize) → None
Read into buffer from the device selected by address. At least one byte must be read.
If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed, but without copying the data. The number of bytes read will be the length of buffer[start:end].

Parameters
• address (int) – 7-bit device address
• buffer (WriteableBuffer) – buffer to write into
• start (int) – beginning of buffer slice
• end (int) – end of buffer slice; if not specified, use len(buffer)

writeto(address: int, buffer: circuitpython_typing.ReadableBuffer, *, start: int = 0, end: int = sys.maxsize) → None
Write the bytes from buffer to the device selected by address and then transmit a stop bit.
If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed, but without copying the data. The number of bytes written will be the length of buffer[start:end].
Writing a buffer or slice of length zero is permitted, as it can be used to poll for the existence of a device.

Parameters
• address (int) – 7-bit device address
• buffer (ReadableBuffer) – buffer containing the bytes to write
• start (int) – beginning of buffer slice
• end (int) – end of buffer slice; if not specified, use len(buffer)

writeto_then_readfrom(address: int, out_buffer: circuitpython_typing.ReadableBuffer, in_buffer: circuitpython_typing.WriteableBuffer, *, out_start: int = 0, out_end: int = sys.maxsize, in_start: int = 0, in_end: int = sys.maxsize) → None
Write the bytes from out_buffer to the device selected by address, generate no stop bit, generate a repeated start and read into in_buffer. out_buffer and in_buffer can be the same buffer because they are used sequentially.
If out_start or out_end is provided, then the buffer will be sliced as if out_buffer[out_start:out_end] were passed, but without copying the data. The number of bytes written will be the length of out_buffer[start:end].
If in_start or in_end is provided, then the input buffer will be sliced as if in_buffer[in_start:in_end] were passed, The number of bytes read will be the length of out_buffer[in_start:in_end].

Parameters
• **address** *(int)* – 7-bit device address
• **out_buffer** *(ReadableBuffer)* – buffer containing the bytes to write
• **in_buffer** *(WriteableBuffer)* – buffer to write into
• **out_start** *(int)* – beginning of **out_buffer** slice
• **out_end** *(int)* – end of **out_buffer** slice; if not specified, use `len(out_buffer)`
• **in_start** *(int)* – beginning of **in_buffer** slice
• **in_end** *(int)* – end of **in_buffer** slice; if not specified, use `len(in_buffer)`

class **busio.SPI** *(clock: microcontroller.Pin, MOSI: microcontroller.Pin | None = None, MISO: microcontroller.Pin | None = None, half_duplex: bool = False)*

A 3-4 wire serial protocol

SPI is a serial protocol that has exclusive pins for data in and out of the main device. It is typically faster than **I2C** because a separate pin is used to select a device rather than a transmitted address. This class only manages three of the four SPI lines: **clock**, **MOSI**, **MISO**. Its up to the client to manage the appropriate select line, often abbreviated CS or SS. (This is common because multiple secondaries can share the **clock**, **MOSI** and **MISO** lines and therefore the hardware.)

Construct an SPI object on the given pins.

---

**Note:** The SPI peripherals allocated in order of desirability, if possible, such as highest speed and not shared use first. For instance, on the nRF52840, there is a single 32MHz SPI peripheral, and multiple 8MHz peripherals, some of which may also be used for **I2C**. The 32MHz SPI peripheral is returned first, then the exclusive 8MHz SPI peripheral, and finally the shared 8MHz peripherals.

---

**See also:**

Using this class directly requires careful lock management. Instead, use **SPIDevice** to manage locks.

**See also:**

Using this class to directly read registers requires manual bit unpacking. Instead, use an existing driver or make one with **Register** data descriptors.

**Parameters**

• **clock** *(Pin)* – the pin to use for the clock.
• **MOSI** *(Pin)* – the Main Out Selected In pin.
• **MISO** *(Pin)* – the Main In Selected Out pin.
• **half_duplex** *(bool)* – True when MOSI is used for bidirectional data. False when SPI is full-duplex or simplex.

**Limitations:** **half_duplex** is available only on STM; other chips do not have the hardware support.

**frequency**: **int**

The actual SPI bus frequency. This may not match the frequency requested due to internal limitations.

**deinit()** → **None**

Turn off the SPI bus.

**__enter__**() → **SPI**

No-op used by Context Managers. Provided by context manager helper.
__exit__() → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**configure**(*, baudrate: int = 100000, polarity: int = 0, phase: int = 0, bits: int = 8) → None

Configures the SPI bus. The SPI object must be locked.

**Parameters**

- **baudrate**(int) – the desired clock rate in Hertz. The actual clock rate may be higher or lower due to the granularity of available clock settings. Check the *frequency* attribute for the actual clock rate.
- **polarity**(int) – the base state of the clock line (0 or 1)
- **phase**(int) – the edge of the clock that data is captured. First (0) or second (1). Rising or falling depends on clock polarity.
- **bits**(int) – the number of bits per word

**Note:** On the SAMD21, it is possible to set the baudrate to 24 MHz, but that speed is not guaranteed to work. 12 MHz is the next available lower speed, and is within spec for the SAMD21.

**Note:** On the nRF52840, these baudrates are available: 125kHz, 250kHz, 1MHz, 2MHz, 4MHz, and 8MHz. If you pick a a baudrate other than one of these, the nearest lower baudrate will be chosen, with a minimum of 125kHz. Two SPI objects may be created, except on the Circuit Playground Bluefruit, which allows only one (to allow for an additional I2C object).

try_lock() → bool

Attempts to grab the SPI lock. Returns True on success.

**Returns**

True when lock has been grabbed

**Return type**

bool

unlock() → None

Releases the SPI lock.

**write**(buffer: circuitpython_typing.ReadableBuffer, *, start: int = 0, end: int = sys.maxsize) → None

Write the data contained in buffer. The SPI object must be locked. If the buffer is empty, nothing happens. If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed, but without copying the data. The number of bytes written will be the length of buffer[start:end].

**Parameters**

- **buffer**(ReadableBuffer) – write out bytes from this buffer
- **start**(int) – beginning of buffer slice
- **end**(int) – end of buffer slice; if not specified, use len(buffer)

**readinto**(buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int = sys.maxsize, write_value: int = 0) → None

Read into buffer while writing write_value for each byte read. The SPI object must be locked. If the number of bytes to read is 0, nothing happens.
If start or end is provided, then the buffer will be sliced as if buffer[start:end] were passed. The number of bytes read will be the length of buffer[start:end].

Parameters

- buffer (WriteableBuffer) – read bytes into this buffer
- start (int) – beginning of buffer slice
- end (int) – end of buffer slice; if not specified, it will be the equivalent value of len(buffer) and for any value provided it will take the value of min(end, len(buffer))
- write_value (int) – value to write while reading

write_readinto(out_buffer: circuitpython_typing.ReadableBuffer, in_buffer: circuitpython_typing.WriteableBuffer, *, out_start: int = 0, out_end: int = sys.maxsize, in_start: int = 0, in_end: int = sys.maxsize) → None

Write out the data in out_buffer while simultaneously reading data into in_buffer. The SPI object must be locked.

If out_start or out_end is provided, then the buffer will be sliced as if out_buffer[out_start:out_end] were passed, but without copying the data. The number of bytes written will be the length of out_buffer[out_start:out_end].

If in_start or in_end is provided, then the input buffer will be sliced as if in_buffer[in_start:in_end] were passed, The number of bytes read will be the length of out_buffer[in_start:in_end].

The lengths of the slices defined by out_buffer[out_start:out_end] and in_buffer[in_start:in_end] must be equal. If buffer slice lengths are both 0, nothing happens.

Parameters

- out_buffer (ReadableBuffer) – write out bytes from this buffer
- in_buffer (WriteableBuffer) – read bytes into this buffer
- out_start (int) – beginning of out_buffer slice
- out_end (int) – end of out_buffer slice; if not specified, use len(out_buffer)
- in_start (int) – beginning of in_buffer slice
- in_end (int) – end of in_buffer slice; if not specified, use len(in_buffer)

class busio.UART(tx: microcontroller.Pin | None = None, rx: microcontroller.Pin | None = None, *, rts: microcontroller.Pin | None = None, cts: microcontroller.Pin | None = None, rs485_dir: microcontroller.Pin | None = None, rs485_invert: bool = False, baudrate: int = 9600, bits: int = 8, parity: Parity | None = None, stop: int = 1, timeout: float = 1, receiver_buffer_size: int = 64)

A bidirectional serial protocol

A common bidirectional serial protocol that uses an an agreed upon speed rather than a shared clock line.

Parameters

- tx (Pin) – the pin to transmit with, or None if this UART is receive-only.
- rx (Pin) – the pin to receive on, or None if this UART is transmit-only.
- rts (Pin) – the pin for rts, or None if rts not in use.
- cts (Pin) – the pin for cts, or None if cts not in use.
• **rs485_dir** (*Pin*) – the output pin for rs485 direction setting, or None if rs485 not in use.

• **rs485_invert** (*bool*) – rs485_dir pin active high when set. Active low otherwise.

• **baudrate** (*int*) – the transmit and receive speed.

• **bits** (*int*) – the number of bits per byte, 5 to 9.

• **parity** (*Parity*) – the parity used for error checking.

• **stop** (*int*) – the number of stop bits, 1 or 2.

• **timeout** (*float*) – the timeout in seconds to wait for the first character and between subsequent characters when reading. Raises **ValueError** if timeout > 100 seconds.

• **receiver_buffer_size** (*int*) – the character length of the read buffer (0 to disable). (When a character is 9 bits the buffer will be 2 * receiver_buffer_size bytes.)

 tx and rx cannot both be None.

*New in CircuitPython 4.0:* timeout has incompatibly changed units from milliseconds to seconds. The new upper limit on timeout is meant to catch mistaken use of milliseconds.

**Limitations:** RS485 is not supported on SAMD, nRF, Broadcom, Spresense, or STM. On i.MX and Raspberry Pi RP2040, RS485 support is implemented in software: The timing for the rs485_dir pin signal is done on a best-effort basis, and may not meet RS485 specifications intermittently.

**baudrate:** *int*

The current baudrate.

**in_waiting:** *int*

The number of bytes in the input buffer, available to be read

**timeout:** *float*

The current timeout, in seconds (float).

**deinit** () → None

Deinitialises the UART and releases any hardware resources for reuse.

**__enter__** () → **UART**

No-op used by Context Managers.

**__exit__** () → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**read**(nbytes: *int | None = None*) → bytes | None

Read bytes. If nbytes is specified then read at most that many bytes. Otherwise, read everything that arrives until the connection times out. Providing the number of bytes expected is highly recommended because it will be faster. If no bytes are read, return None.

**Note:** When no bytes are read due to a timeout, this function returns None. This matches the behavior of **io.RawIOBase.read** in Python 3, but differs from pyserial which returns b'' in that situation.

**Returns**

Data read

**Return type**

bytes or None
readinto(buf: circuitpython_typing.WriteableBuffer) → int | None
Read bytes into the buf. Read at most len(buf) bytes.

Returns
number of bytes read and stored into buf

Return type
int or None (on a non-blocking error)

New in CircuitPython 4.0: No length parameter is permitted.

readline() → bytes
Read a line, ending in a newline character, or return None if a timeout occurs sooner, or return everything readable if no newline is found and timeout=0

Returns
the line read

Return type
bytes or None

write(buf: circuitpython_typing.ReadableBuffer) → int | None
Write the buffer of bytes to the bus.

New in CircuitPython 4.0: buf must be bytes, not a string.

return
the number of bytes written

rtype
int or None

reset_input_buffer() → None
Discard any unread characters in the input buffer.

class busio.Parity
Enum-like class to define the parity used to verify correct data transfer.

ODD: int
Total number of ones should be odd.

EVEN: int
Total number of ones should be even.

camera – Support for camera input

The camera module contains classes to control the camera and take pictures.

class camera.Camera
The class to control camera.

Usage:

import board
import sdioio
import storage
import camera

(continues on next page)
sd = sdioio.SDCard(
    clock=board.SDIO_CLOCK,
    command=board.SDIO_COMMAND,
    data=board.SDIO_DATA,
    frequency=25000000)
vfs = storage.VfsFat(sd)
storage.mount(vfs, '/sd')
cam = camera.Camera()

buffer = bytearray(512 * 1024)
file = open("/sd/image.jpg","wb")
size = cam.take_picture(buffer, width=1920, height=1080, format=camera.ImageFormat.JPG)
file.write(buffer, size)
file.close()

Initialize camera.

definit() \rightarrow None
    De-initialize camera.

take_picture(buf: circuitpython_typing.WriteableBuffer, format: ImageFormat) \rightarrow int
    Take picture and save to buf in the given format. The size of the picture taken is width by height in pixels.
    
    **Returns**
    the number of bytes written into buf

    **Return type**
    int

class camera.ImageFormat
    Image format
    Enum-like class to define the image format.

    **JPG**: ImageFormat
        JPG format.

    **RGB565**: ImageFormat
        RGB565 format.

canio – CAN bus access

The canio module contains low level classes to support the CAN bus protocol.

CAN and Listener classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call definit() or use a context manager. See [Lifetime and Context Managers](#) for more info.

For example:

```python
import canio
from board import *
```
This example will write the data ‘adafruit’ onto the CAN bus to any device listening for message id 0x0408.

A CAN bus involves a transceiver, which is often a separate chip with a “standby” pin. If your board has a CAN_STANDBY pin, ensure to set it to an output with the value False to enable the transceiver.

Other implementations of the CAN device may exist (for instance, attached via an SPI bus). If so their constructor arguments may differ, but otherwise we encourage implementors to follow the API that the core uses.

For more information on working with this module, refer to this Learn Guide on using it.

class canio.BusState

The state of the CAN bus

ERROR_ACTIVE: object
The bus is in the normal (active) state

ERROR_WARNING: object
The bus is in the normal (active) state, but a moderate number of errors have occurred recently.

Note: Not all implementations may use ERROR_WARNING. Do not rely on seeing ERROR_WARNING before ERROR_PASSIVE.

ERROR_PASSIVE: object
The bus is in the passive state due to the number of errors that have occurred recently.

This device will acknowledge packets it receives, but cannot transmit messages. If additional errors occur, this device may progress to BUS_OFF. If it successfully acknowledges other packets on the bus, it can return to ERROR_WARNING or ERROR_ACTIVE and transmit packets.

BUS_OFF: object
The bus has turned off due to the number of errors that have occurred recently. It must be restarted before it will send or receive packets. This device will neither send nor acknowledge packets on the bus.


CAN bus protocol
A common shared-bus protocol. The rx and tx pins are generally connected to a transceiver which controls the H and L pins on a shared bus.

Parameters

• rx (Pin) – the pin to receive with
• tx (Pin) – the pin to transmit with
• baudrate (int) – The bit rate of the bus in Hz. All devices on the bus must agree on this value.
• loopback (bool) – When True the rx pin’s value is ignored, and the device receives the packets it sends.
• **silent** (*bool*) – When True the `tx` pin is always driven to the high logic level. This mode can be used to “sniff” a CAN bus without interfering.

• **auto_restart** (*bool*) – If True, will restart communications after entering bus-off state

```python
class CAN:
    # ... (rest of the class definition)
```

```python
auto_restart: bool
    # ... (implementation details)
```

If True, will restart communications after entering bus-off state

```python
baudrate: int
    # ... (implementation details)
```

The baud rate (read-only)

```python
transmit_error_count: int
    # ... (implementation details)
```

The number of transmit errors (read-only). Increased for a detected transmission error, decreased for successful transmission. Limited to the range from 0 to 255 inclusive. Also called TEC.

```python
receive_error_count: int
    # ... (implementation details)
```

The number of receive errors (read-only). Increased for a detected reception error, decreased for successful reception. Limited to the range from 0 to 255 inclusive. Also called REC.

```python
state: BusState
    # ... (implementation details)
```

The current state of the bus. (read-only)

```python
loopback: bool
    # ... (implementation details)
```

True if the device was created in loopback mode, False otherwise (read-only)

```python
silent: bool
    # ... (implementation details)
```

True if the device was created in silent mode, False otherwise (read-only)

```python
restart() -> None
    # ... (implementation details)
```

If the device is in the bus off state, restart it.

```python
listen(matches: Sequence[Match] | None = None, *, timeout: float = 10) -> Listener
    # ... (implementation details)
```

Start receiving messages that match any one of the filters.

Creating a listener is an expensive operation and can interfere with reception of messages by other listeners.

There is an implementation-defined maximum number of listeners and limit to the complexity of the filters.

If the hardware cannot support all the requested matches, a ValueError is raised. Note that generally there are some number of hardware filters shared among all fifos.

A message can be received by at most one Listener. If more than one listener matches a message, it is undefined which one actually receives it.

An empty filter list causes all messages to be accepted.

Timeout dictates how long receive() and next() will block.

Platform specific notes:

SAM E5x supports two Listeners. Filter blocks are shared between the two listeners. There are 4 standard filter blocks and 4 extended filter blocks. Each block can either match 2 single addresses or a mask of addresses. The number of filter blocks can be increased, up to a hardware maximum, by rebuilding CircuitPython, but this decreases the CircuitPython free memory even if canio is not used.

STM32F405 supports two Listeners. Filter blocks are shared between the two listeners. There are 14 filter blocks. Each block can match 2 standard addresses with mask or 1 extended address with mask.

ESP32S2 supports one Listener. There is a single filter block, which can either match a standard address with mask or an extended address with mask.
send(message: RemoteTransmissionRequest | Message) → None

Send a message on the bus with the given data and id. If the message could not be sent due to a full fifo or a bus error condition, RuntimeError is raised.

definit() → None

Deinitialize this object, freeing its hardware resources

__enter__() → CAN

Returns self, to allow the object to be used in a The with statement statement for resource control

__exit__(unused1: Type[BaseException] | None, unused2: BaseException | None, unused3: types.TracebackType | None) → None

Calls definit()

class canio.Listener

Listens for CAN message

canio.Listener is not constructed directly, but instead by calling canio.CAN.listen.

In addition to using the receive method to retrieve a message or the in_waiting method to check for an available message, a listener can be used as an iterable, yielding messages until no message arrives within self. timeout seconds.

timeout: float

receive() → RemoteTransmissionRequest | Message | None

Reads a message, after waiting up to self.timeout seconds

If no message is received in time, None is returned. Otherwise, a Message or RemoteTransmissionRequest is returned.

in_waiting() → int

Returns the number of messages (including remote transmission requests) waiting

__iter__() → Listener

Returns self

This method exists so that Listener can be used as an iterable

__next__() → RemoteTransmissionRequest | Message

Reads a message, after waiting up to self.timeout seconds

If no message is received in time, raises StopIteration. Otherwise, a Message or is returned.

This method enables the Listener to be used as an iterable, for instance in a for-loop.

definit() → None

Deinitialize this object, freeing its hardware resources

__enter__() → CAN

Returns self, to allow the object to be used in a The with statement statement for resource control

__exit__(unused1: Type[BaseException] | None, unused2: BaseException | None, unused3: types.TracebackType | None) → None

Calls definit()

class canio.Match(id: int, *, mask: int | None = None, extended: bool = False)

Describe CAN bus messages to match

Construct a Match with the given properties.
If mask is not None, then the filter is for any id which matches all the nonzero bits in mask. Otherwise, it matches exactly the given id. If extended is true then only extended ids are matched, otherwise only standard ids are matched.

```python
def filter_message(id, mask=None, extended=False):
    if mask is not None:
        return id & mask == id
    return id == extended
```

- `id: int` The id to match
- `mask: int` The optional mask of ids to match
- `extended: bool` True to match extended ids, False to match standard ids

```python
class canio.Message(id: int, data: bytes, *, extended: bool = False)
```

Construct a Message to send on a CAN bus.

**Parameters**

- `id (int)` – The numeric ID of the message
- `data (bytes)` – The content of the message
- `extended (bool)` – True if the message has an extended identifier, False if it has a standard identifier

In CAN, messages can have a length from 0 to 8 bytes.

```python
def construct_message(id, data, *, extended=False):
    return canio.Message(id, data, extended=extended)
```

```python
class canio.RemoteTransmissionRequest(id: int, length: int, *, extended: bool = False)
```

Construct a RemoteTransmissionRequest to send on a CAN bus.

**Parameters**

- `id (int)` – The numeric ID of the requested message
- `length (int)` – The length of the requested message
- `extended (bool)` – True if the message has an extended identifier, False if it has a standard identifier

In CAN, messages can have a length from 0 to 8 bytes.

```python
def construct_remote_request(id, length, *, extended=False):
    return canio.RemoteTransmissionRequest(id, length, extended=extended)
```
countio – Support for edge counting

The `countio` module contains logic to read and count edge transitions. For more information on the applications of counting edges, see this Learn Guide on sequential circuits.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See `Lifetime and ContextManagers` for more info.

**class countio.Edge**

Enumerates which signal transitions can be counted.

- **RISE**: Edge
  - Count the rising edges.
- **FALL**: Edge
  - Count the falling edges.
- **RISE_AND_FALL**: Edge
  - Count the rising and falling edges.

**Limitations**: RISE_AND_FALL is not available to RP2040 due to hardware limitations.

**class countio.Counter**

Create a Counter object associated with the given pin that counts rising- and/or falling-edge transitions. At least one of `rise` and `fall` must be True. The default is to count only falling edges, and is for historical backward compatibility.

**Parameters**

- **pin** (Pin) – pin to monitor
- **edge** (Edge) – which edge transitions to count
- **pull** (Optional[digitalio.Pull]) – enable a pull-up or pull-down if not None

For example:

```python
import board
import countio

# Count rising edges only.
pin_counter = countio.Counter(board.D1, edge=countio.Edge.RISE)
# Reset the count after 100 counts.
while True:
    if pin_counter.count >= 100:
        pin_counter.reset()
    print(pin_counter.count)
```

**Limitations**: On RP2040, `Counter` uses the PWM peripheral, and is limited to using PWM channel B pins due to hardware restrictions. See the pin assignments for your board to see which pins can be used.

**count**: int

The current count in terms of pulses.
deinit() → None
Deinitializes the Counter and releases any hardware resources for reuse.

__enter__() → Counter
No-op used by Context Managers.

__exit__() → None
Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

reset() → None
Resets the count back to 0.

cyw43 – A class that represents a GPIO pin attached to the wifi chip.

class cyw43.CywPin
Cannot be constructed at runtime, but may be the type of a pin object in board. A CywPin can be used as a DigitalInOut, but not with other peripherals such as PWMOut.

cyw43.PM_STANDARD: int
The standard power management mode

cyw43.PM_AGGRESSIVE: int
Aggressive power management mode for optimal power usage at the cost of performance

cyw43.PM_PERFORMANCE: int
Performance power management mode where more power is used to increase performance

cyw43.PM_DISABLED: int
Disable power management and always use highest power mode. CircuitPython sets this value at reset time, because it provides the best connectivity reliability.

cyw43.set_power_management(value: int) → None
Set the power management register

For transmitter power, see wifi.Radio.txpower. This controls software power saving features inside the cyw43 chip. It does not control transmitter power.

The value is interpreted as a 24-bit hexadecimal number of the form 0x00adbrrm.

The low 4 bits, m, are the power management mode:
- 0: disabled
- 1: aggressive power saving which reduces wifi throughput
- 2: Power saving with high throughput

The next 8 bits, r, specify “the maximum time to wait before going back to sleep” for power management mode 2. The units of r are 10ms.

The next 4 bits, b, are the “wake period is measured in beacon periods”.

The next 4 bits, d, specify the “wake interval measured in DTIMs. If this is set to 0, the wake interval is measured in beacon periods”.

The top 4 bits, a, specifies the “wake interval sent to the access point”

Several PM_ constants gathered from various sources are included in this module. According to Raspberry Pi documentation, the value 0xa11140 (called cyw43.PM_DISABLED here) increases responsiveness at the cost of higher power usage.

1.8. Full Table of Contents
cyw43.get_power_management() → int
Retrieve the power management register

digitalio – Basic digital pin support

The digitalio module contains classes to provide access to basic digital IO.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call deinit() or use a context manager. See Lifetime and ContextManagers for more info.

For example:

```python
import digitalio
import board

pin = digitalio.DigitalInOut(board.LED)
print(pin.value)
```

This example will initialize the device, read value and then deinit() the hardware.

Here is blinky:

```python
import time
import digitalio
import board

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT
while True:
    led.value = True
    time.sleep(0.1)
    led.value = False
    time.sleep(0.1)
```

For the essentials of digitalio, see the CircuitPython Essentials Learn guide

For more information on using digitalio, see this additional Learn guide

class digitalio.DriveMode

Defines the drive mode of a digital pin

Enum-like class to define the drive mode used when outputting digital values.

PUSH_PULL: DriveMode
Output both high and low digital values

OPEN_DRAIN: DriveMode
Output low digital values but go into high z for digital high. This is useful for i2c and other protocols that share a digital line.

class digitalio.DigitalInOut(pin: microcontroller.Pin)

Digital input and output

A DigitalInOut is used to digitally control I/O pins. For analog control of a pin, see the analogio.AnalogIn and analogio.AnalogOut classes.

Create a new DigitalInOut object associated with the pin. Defaults to input with no pull. Use switch_to_input() and switch_to_output() to change the direction.
Parameters

param (Pin) – The pin to control

direction: Direction

The direction of the pin.

Setting this will use the defaults from the corresponding switch_to_input() or switch_to_output() method. If you want to set pull, value or drive mode prior to switching, then use those methods instead.

value: bool

The digital logic level of the pin.

drive_mode: DriveMode

The pin drive mode. One of:

- `digitalio.DriveMode.PUSH_PULL`
- `digitalio.DriveMode.OPEN_DRAIN`

pull: Pull | None

The pin pull direction. One of:

- `digitalio.Pull.UP`
- `digitalio.Pull.DOWN`
- None

Raises

- `AttributeError` – if `direction` is `OUTPUT`.

decor() → None

Turn off the DigitalInOut and release the pin for other use.

__enter__() → DigitalInOut

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

switch_to_output(value: bool = False, drive_mode: DriveMode = DriveMode.PUSH_PULL) → None

Set the drive mode and value and then switch to writing out digital values.

Parameters

- value (bool) – default value to set upon switching
- drive_mode (DriveMode) – drive mode for the output

switch_to_input(pull: Pull | None = None) → None

Set the pull and then switch to read in digital values.

Parameters

- pull (Pull) – pull configuration for the input

Example usage:
```python
import digitalio
import board

switch = digitalio.DigitalInOut(board.SLIDE_SWITCH)
switch.switch_to_input(pull=digitalio.Pull.UP)
# Or, after switch_to_input
switch.pull = digitalio.Pull.UP
print(switch.value)
```

class digitalio.Direction

Defines the direction of a digital pin

Enum-like class to define which direction the digital values are going.

**INPUT:** Direction

Read digital data in

**OUTPUT:** Direction

Write digital data out

class digitalio.Pull

Defines the pull of a digital input pin

Enum-like class to define the pull value, if any, used while reading digital values in.

**UP:** Pull

When the input line isn’t being driven the pull up can pull the state of the line high so it reads as true.

**DOWN:** Pull

When the input line isn’t being driven the pull down can pull the state of the line low so it reads as false.

displayio – Native helpers for driving displays

The `displayio` module contains classes to manage display output including synchronizing with refresh rates and partial updating.

For more a more thorough explanation and guide for using `displayio`, please refer to this Learn guide.

displayio.CIRCUITPYTHON_TERMINAL: Group

The `displayio.Group` that is the displayed serial terminal (REPL).

displayio.release_displays() → None

Releases any actively used displays so their buses and pins can be used again. This will also release the builtin display on boards that have one. You will need to reinitialize it yourself afterwards. This may take seconds to complete if an active EPaperDisplay is refreshing.

Use this once in your code.py if you initialize a display. Place it right before the initialization so the display is active as long as possible.

class displayio.Colorspace

The colorspace for a `ColorConverter` to operate in

**RGB888:** Colorspace

The standard 24-bit colorspace. Bits 0-7 are blue, 8-15 are green, and 16-24 are red. (0xRRGGBB)
RGB565: *Colorsace*

The standard 16-bit colorspace. Bits 0-4 are blue, bits 5-10 are green, and 11-15 are red (0bR-RRRRGGGGBBBBB)

RGB565_SWAPPED: *Colorsace*

The swapped 16-bit colorspace. First, the high and low 8 bits of the number are swapped, then they are interpreted as for RGB565

RGB555: *Colorsace*

The standard 15-bit colorspace. Bits 0-4 are blue, bits 5-9 are green, and 11-14 are red. The top bit is ignored. (0bxRRRRGOGGBBBB)

RGB555_SWAPPED: *Colorsace*

The swapped 15-bit colorspace. First, the high and low 8 bits of the number are swapped, then they are interpreted as for RGB555

```python
class displayio.Bitmap(width: int, height: int, value_count: int)
```

Stores values of a certain size in a 2D array

Bitmaps can be treated as read-only buffers. If the number of bits in a pixel is 8, 16, or 32; and the number of bytes per row is a multiple of 4, then the resulting memoryview will correspond directly with the bitmap’s contents. Otherwise, the bitmap data is packed into the memoryview with unspecified padding.

A Bitmap can be treated as a buffer, allowing its content to be viewed and modified using e.g., with `ulab.numpy.frombuffer`, but the `displayio.Bitmap.dirty` method must be used to inform `displayio` when a bitmap was modified through the buffer interface.

`bitmaptools.arrayblit` can also be useful to move data efficiently into a Bitmap.

Create a Bitmap object with the given fixed size. Each pixel stores a value that is used to index into a corresponding palette. This enables differently colored sprites to share the underlying Bitmap. `value_count` is used to minimize the memory used to store the Bitmap.

**Parameters**

- `width` *(int)* – The number of values wide
- `height` *(int)* – The number of values high
- `value_count` *(int)* – The number of possible pixel values.

**width**: `int`

Width of the bitmap. (read only)

**height**: `int`

Height of the bitmap. (read only)

```python
__getitem__(index: Tuple[int, int] | int) → int
```

Returns the value at the given index. The index can either be an x,y tuple or an int equal to y * width + x.

This allows you to:

```python
print(bitmap[0,1])
```

```python
__setitem__(index: Tuple[int, int] | int, value: int) → None
```

Sets the value at the given index. The index can either be an x,y tuple or an int equal to y * width + x.

This allows you to:
(bitmap[0,1] = 3)

```python
blit(x: int, y: int, source_bitmap: Bitmap, *, x1: int, y1: int, x2: int, y2: int, skip_index: int) -> None
```

Inserts the source_bitmap region defined by rectangular boundaries (x1, y1) and (x2, y2) into the bitmap at the specified (x, y) location.

**Parameters**

- **x (int)** – Horizontal pixel location in bitmap where source_bitmap upper-left corner will be placed
- **y (int)** – Vertical pixel location in bitmap where source_bitmap upper-left corner will be placed
- **source_bitmap (bitmap)** – Source bitmap that contains the graphical region to be copied
- **x1 (int)** – Minimum x-value for rectangular bounding box to be copied from the source bitmap
- **y1 (int)** – Minimum y-value for rectangular bounding box to be copied from the source bitmap
- **x2 (int)** – Maximum x-value (exclusive) for rectangular bounding box to be copied from the source bitmap
- **y2 (int)** – Maximum y-value (exclusive) for rectangular bounding box to be copied from the source bitmap
- **skip_index (int)** – Bitmap palette index in the source that will not be copied, set to None to copy all pixels

```python
fill(value: int) -> None
```

Fills the bitmap with the supplied palette index value.

```python
dirty(x1: int = 0, y1: int = 0, x2: int = -1, y2: int = -1) -> None
```

Inform displayio of bitmap updates done via the buffer protocol.

**Parameters**

- **x1 (int)** – Minimum x-value for rectangular bounding box to be considered as modified
- **y1 (int)** – Minimum y-value for rectangular bounding box to be considered as modified
- **x2 (int)** – Maximum x-value (exclusive) for rectangular bounding box to be considered as modified
- **y2 (int)** – Maximum y-value (exclusive) for rectangular bounding box to be considered as modified

If x1 or y1 are not specified, they are taken as 0. If x2 or y2 are not specified, or are given as -1, they are taken as the width and height of the image. Thus, calling dirty() with the default arguments treats the whole bitmap as modified.

When a bitmap is modified through the buffer protocol, the display will not be properly updated unless the bitmap is notified of the “dirty rectangle” that encloses all modified pixels.

```python
deinit() -> None
```

Release resources allocated by Bitmap.
class displayio.ColorConverter(*, input_colorspace: Colorspace = Colorspace.RGB888, dither: bool = False)

Converts one color format to another.

Create a ColorConverter object to convert color formats.

Parameters

• colorspace (Colorspace) – The source colorspace, one of the Colorspace constants

• dither (bool) – Adds random noise to dither the output image

dither: bool

When True the ColorConverter dithers the output by adding random noise when truncating to display bitdepth

convert(color: int) → int

Converts the given color to RGB565 according to the Colorspace

make_transparent(color: int) → None

Set the transparent color or index for the ColorConverter. This will raise an Exception if there is already a selected transparent index.

Parameters

• color (int) – The color to be transparent

make_opaque(color: int) → None

Make the ColorConverter be opaque and have no transparent pixels.

Parameters

• color (int) – [IGNORED] Use any value

displayio._DisplayBus

FourWire, parallel_display.ParallelBus or I2CDisplay

class displayio.Display(display_bus: _DisplayBus, init_sequence: circuitpython_typing.ReadableBuffer, *,
width: int, height: int, colstart: int = 0, rowstart: int = 0, rotation: int = 0,
color_depth: int = 16, grayscale: bool = False, pixels_in_byte_share_row: bool = True,
bytes_per_cell: int = 1, reverse_pixels_in_byte: bool = False,
set_column_command: int = 42, set_row_command: int = 43, write_ram_command: int = 44,
backlight_pin: microcontroller.Pin | None = None,
brightness_command: int | None = None, brightness: float = 1.0, single_byte_bounds: bool = False,
data_as_commands: bool = False, auto_refresh: bool = True,
native_frames_per_second: int = 60, backlight_on_high: bool = True,
SH1107_addressing: bool = False)

Manage updating a display over a display bus

This initializes a display and connects it into CircuitPython. Unlike other objects in CircuitPython, Display objects live until displayio.release_displays() is called. This is done so that CircuitPython can use the display itself.

Most people should not use this class directly. Use a specific display driver instead that will contain the initialization sequence at minimum.

Create a Display object on the given display bus (FourWire, ParallelBus or I2CDisplay).

The init_sequence is bitpacked to minimize the ram impact. Every command begins with a command byte followed by a byte to determine the parameter count and delay. When the top bit of the second byte is 1 (0x80), a delay will occur after the command parameters are sent. The remaining 7 bits are the parameter count excluding any delay byte. The bytes following are the parameters. When the delay bit is set, a single byte after the parameters
specifies the delay duration in milliseconds. The value 0xff will lead to an extra long 500 ms delay instead of
255 ms. The next byte will begin a new command definition. Here is an example:

```python
init_sequence = (b"\x1e\x0f\x00\x0e\x14\x03\x11\x07\x31\xC1\x48\x08\x0F\x0C\x31\x36\n\x0F" # Set Gamma
    b"\x11\x80\x78" # Exit Sleep then delay 0x78 (120ms)
    b"\x29\x81\xaa\x78" # Display on then delay 0x78 (120ms)
)

display = displayio.Display(display_bus, init_sequence, width=320, height=240)
```

The first command is 0xe1 with 15 (0xf) parameters following. The second is 0x11 with 0 parameters and a
120ms (0x78) delay. The third command is 0x29 with one parameter 0xaa and a 120ms delay (0x78). Multiple
byte literals (b""') are merged together on load. The parents are needed to allow byte literals on subsequent lines.
The initialization sequence should always leave the display memory access inline with the scan of the display to
minimize tearing artifacts.

Parameters

- **display_bus** – The bus that the display is connected to
- **init_sequence** ([ReadableBuffer]) – Byte-packed initialization sequence.
- **width** (int) – Width in pixels
- **height** (int) – Height in pixels
- **colstart** (int) – The index if the first visible column
- **rowstart** (int) – The index if the first visible row
- **rotation** (int) – The rotation of the display in degrees clockwise. Must be in 90 degree
  increments (0, 90, 180, 270)
- **color_depth** (int) – The number of bits of color per pixel transmitted. (Some displays
  support 18 bit but 16 is easier to transmit. The last bit is extrapolated.)
- **grayscale** (bool) – True if the display only shows a single color.
- **pixels_in_byte_share_row** (bool) – True when pixels are less than a byte and a byte
  includes pixels from the same row of the display. When False, pixels share a column.
- **bytes_per_cell** (int) – Number of bytes per addressable memory location when
  color_depth < 8. When greater than one, bytes share a row or column according to pix-
  els_in_byte_share_row.
- **reverse_pixels_in_byte** (bool) – Reverses the pixel order within each byte when
  color_depth < 8. Does not apply across multiple bytes even if there is more than one byte
  per cell (bytes_per_cell.)
- **reverse_bytes_in_word** (bool) – Reverses the order of bytes within a word when
  color_depth == 16
- **set_column_command** (int) – Command used to set the start and end columns to update
- **set_row_command** (int) – Command used so set the start and end rows to update
- **write_ram_command** (int) – Command used to write pixels values into the update region.
  Ignored if data_as_commands is set.
- **backlight_pin** ([microcontroller.Pin]) – Pin connected to the display’s backlight
- **brightness_command** (int) – Command to set display brightness. Usually available in
  OLED controllers.
• **brightness** (*float*) – Initial display brightness.
• **single_byte_bounds** (*bool*) – Display column and row commands use single bytes
• **data_as_commands** (*bool*) – Treat all init and boundary data as SPI commands. Certain displays require this.
• **auto_refresh** (*bool*) – Automatically refresh the screen
• **native_frames_per_second** (*int*) – Number of display refreshes per second that occur with the given init_sequence.
• **backlight_on_high** (*bool*) – If True, pulling the backlight pin high turns the backlight on.
• **SH1107_addressing** (*bool*) – Special quirk for SH1107, use upper/lower column set and page set
• **set_vertical_scroll** (*int*) – This parameter is accepted but ignored for backwards compatibility. It will be removed in a future release.
• **backlight_pwm_frequency** (*int*) – The frequency to use to drive the PWM for backlight brightness control. Default is 50000.

```python
class Display:
    def __init__(self, init_sequence):
        self.init_sequence = init_sequence

    @property
    def auto_refresh(self) -> bool:
        return self._auto_refresh

    @auto_refresh.setter
    def auto_refresh(self, value: bool):
        self._auto_refresh = value

    @property
    def brightness(self) -> float:
        return self._brightness

    @brightness.setter
    def brightness(self, value: float):
        self._brightness = value

    @property
    def width(self) -> int:
        return self._width

    @width.setter
    def width(self, value: int):
        self._width = value

    @property
    def height(self) -> int:
        return self._height

    @height.setter
    def height(self, value: int):
        self._height = value

    @property
    def rotation(self) -> int:
        return self._rotation

    @rotation.setter
    def rotation(self, value: int):
        self._rotation = value

    @property
    def bus(self) -> DisplayBus:
        return self._bus

    @bus.setter
    def bus(self, value: DisplayBus):
        self._bus = value

    @property
    def root_group(self) -> Group:
        return self._root_group

    @root_group.setter
    def root_group(self, value: Group):
        self._root_group = value

    def show(self, group: Group) -> None:
        if group is None:
            self.root_group = displayio.CIRCUITPYTHON_TERMINAL
        else:
            self.root_group = group
```

**Note:** `show()` is deprecated and will be removed in CircuitPython 9.0.0. Use `.root_group = group` instead.

Switches to displaying the given group of layers. When group is None, the default CircuitPython terminal will be shown.

**Parameters**

- **group** (*Group*) – The group to show.
refresh(*, target_frames_per_second: int | None = None, minimum_frames_per_second: int = 0) → bool

When auto_refresh is off, and target_frames_per_second is not None this waits for the target frame rate and then refreshes the display, returning True. If the call has taken too long since the last refresh call for the given target frame rate, then the refresh returns False immediately without updating the screen to hopefully help getting caught up.

If the time since the last successful refresh is below the minimum frame rate, then an exception will be raised. The default minimum_frames_per_second of 0 disables this behavior.

When auto_refresh is off, and target_frames_per_second is None this will update the display immediately.

When auto_refresh is on, updates the display immediately. (The display will also update without calls to this.)

Parameters

- target_frames_per_second (Optional[int]) – The target frame rate that refresh() should try to achieve. Set to None for immediate refresh.
- minimum_frames_per_second (int) – The minimum number of times the screen should be updated per second.

fill_row(y: int, buffer: circuitpython_typing.WriteableBuffer) → circuitpython_typing.WriteableBuffer

Extract the pixels from a single row

Parameters

- y (int) – The top edge of the area
- buffer (WriteableBuffer) – The buffer in which to place the pixel data


Manage updating an epaper display over a display bus

This initializes an epaper display and connects it into CircuitPython. Unlike other objects in CircuitPython, EPaperDisplay objects live until displayio.release_displays() is called. This is done so that CircuitPython can use the display itself.

Most people should not use this class directly. Use a specific display driver instead that will contain the startup and shutdown sequences at minimum.

Create a EPaperDisplay object on the given display bus (displayio.FourWire or paralleldisplay.ParallelBus).
The `start_sequence` and `stop_sequence` are bitpacked to minimize the ram impact. Every command begins with a command byte followed by a byte to determine the parameter count and delay. When the top bit of the second byte is 1 (0x80), a delay will occur after the command parameters are sent. The remaining 7 bits are the parameter count excluding any delay byte. The bytes following are the parameters. When the delay bit is set, a single byte after the parameters specifies the delay duration in milliseconds. The value 0xff will lead to an extra long 500 ms delay instead of 255 ms. The next byte will begin a new command definition.

Parameters

- `display_bus` – The bus that the display is connected to
- `start_sequence (ReadableBuffer)` – Byte-packed command sequence.
- `stop_sequence (ReadableBuffer)` – Byte-packed command sequence.
- `width (int)` – Width in pixels
- `height (int)` – Height in pixels
- `ram_width (int)` – RAM width in pixels
- `ram_height (int)` – RAM height in pixels
- `colstart (int)` – The index if the first visible column
- `rowstart (int)` – The index if the first visible row
- `rotation (int)` – The rotation of the display in degrees clockwise. Must be in 90 degree increments (0, 90, 180, 270)
- `set_column_window_command (int)` – Command used to set the start and end columns to update
- `set_row_window_command (int)` – Command used so set the start and end rows to update
- `set_current_column_command (int)` – Command used to set the current column location
- `set_current_row_command (int)` – Command used to set the current row location
- `write_black_ram_command (int)` – Command used to write pixels values into the update region
- `black_bits_inverted (bool)` – True if 0 bits are used to show black pixels. Otherwise, 1 means to show black.
- `write_color_ram_command (int)` – Command used to write pixels values into the update region
- `color_bits_inverted (bool)` – True if 0 bits are used to show the color. Otherwise, 1 means to show color.
- `highlight_color (int)` – RGB888 of source color to highlight with third ePaper color.
- `refresh_display_command (int)` – Command used to start a display refresh. Single int or byte-packed command sequence
- `refresh_time (float)` – Time it takes to refresh the display before the stop_sequence should be sent. Ignored when busy_pin is provided.
- `busy_pin (microcontroller.Pin)` – Pin used to signify the display is busy
- `busy_state (bool)` – State of the busy pin when the display is busy
- `seconds_per_frame (float)` – Minimum number of seconds between screen refreshes
- `always_toggle_chip_select (bool)` – When True, chip select is toggled every byte
- grayscale *(bool)* – When true, the color ram is the low bit of 2-bit grayscale
- advanced_color_epaper *(bool)* – When true, the display is a 7-color advanced color epaper (ACeP)
- two_byte_sequence_length *(bool)* – When true, use two bytes to define sequence length
- start_up_time *(float)* – Time to wait after reset before sending commands
- address_little_endian *(bool)* – Send the least significant byte (not bit) of multi-byte addresses first. Ignored when ram is addressed with one byte

*time_to_refresh: float*

Time, in fractional seconds, until the ePaper display can be refreshed.

*busy: bool*

True when the display is refreshing. This uses the *busy_pin* when available or the *refresh_time* otherwise.

*width: int*

Gets the width of the display in pixels

*height: int*

Gets the height of the display in pixels

*rotation: int*

The rotation of the display as an int in degrees.

*bus: _DisplayBus*

The bus being used by the display

*root_group: Group*

The root group on the epaper display. If the root group is set to *displayio.CIRCUITPYTHON_TERMINAL*, the default CircuitPython terminal will be shown. If the root group is set to *None*, no output will be shown.

**Note:** *show()* is deprecated and will be removed in CircuitPython 9.0.0. Use *.root_group = group* instead.

Switches to displaying the given group of layers. When group is None, the default CircuitPython terminal will be shown.

**Parameters**

*group: Group* – The group to show.

**update_refresh_mode**(start_sequence: circuitpython_typing.ReadableBuffer, seconds_per_frame: float = 180) → None

Updates the *start_sequence* and *seconds_per_frame* parameters to enable varying the refresh mode of the display.

**refresh() → None**

Refreshes the display immediately or raises an exception if too soon. Use *time.sleep(display.time_to_refresh)* to sleep until a refresh can occur.

Manage updating a display over SPI four wire protocol in the background while Python code runs. It doesn't handle display initialization.

Create a FourWire object associated with the given pins.

The SPI bus and pins are then in use by the display until `displayio.release_displays()` is called even after
a reload. (It does this so CircuitPython can use the display after your code is done.) So, the first time you initialize
a display bus in code.py you should call `displayio.release_displays()` first, otherwise it will error after
the first code.py run.

If the command pin is not specified, a 9-bit SPI mode will be simulated by adding a data/command bit to every
bit being transmitted, and splitting the resulting data back into 8-bit bytes for transmission. The extra bits that
this creates at the end are ignored by the receiving device.

**Parameters**

- `spi_bus (busio.SPI)` – The SPI bus that make up the clock and data lines
- `command (microcontroller.Pin)` – Data or command pin. When None, 9-bit SPI is sim-
ulated.
- `chip_select (microcontroller.Pin)` – Chip select pin
- `reset (microcontroller.Pin)` – Reset pin. When None only software reset can be used
- `baudrate (int)` – Maximum baudrate in Hz for the display on the bus
- `polarity (int)` – the base state of the clock line (0 or 1)
- `phase (int)` – the edge of the clock that data is captured. First (0) or second (1). Rising or
falling depends on clock polarity.

`reset()` → None

Performs a hardware reset via the reset pin. Raises an exception if called when no reset pin is available.

`send(command: int, data: circuitpython_typing.ReadableBuffer, *, toggle_every_byte: bool = False) → None`

Sends the given command value followed by the full set of data. Display state, such as vertical scroll, set via `send`
may or may not be reset once the code is done.

```python
class displayio.Group(*, scale: int = 1, x: int = 0, y: int = 0)
```

Manage a group of sprites and groups and how they are inter-related.

Create a Group of a given size and scale. Scale is in one dimension. For example, scale=2 leads to a layer’s pixel
being 2x2 pixels when in the group.

**Parameters**

- `scale (int)` – Scale of layer pixels in one dimension.
- `x (int)` – Initial x position within the parent.
- `y (int)` – Initial y position within the parent.

**hidden: bool**

True when the Group and all of it’s layers are not visible. When False, the Group’s layers are visible if they
haven’t been hidden.

**scale: int**

Scales each pixel within the Group in both directions. For example, when scale=2 each pixel will be rep-
represented by 2x2 pixels.
x: `int`
   X position of the Group in the parent.

y: `int`
   Y position of the Group in the parent.

`append(layer: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → None`
   Append a layer to the group. It will be drawn above other layers.

`insert(index: int, layer: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → None`
   Insert a layer into the group.

`index(layer: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → int`
   Returns the index of the first copy of layer. Raises ValueError if not found.

`pop(i: int = -1) → vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid`
   Remove the ith item and return it.

`remove(layer: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → None`
   Remove the first copy of layer. Raises ValueError if it is not present.

`__bool__() → bool`

`__contains__(item: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → bool`

`__iter__() → Iterator[vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid]`

`__len__() → int`
   Returns the number of layers in a Group

`__getitem__(index: int) → vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid`
   Returns the value at the given index.
   This allows you to:

   ```python
   print(group[0])
   ```

`__setitem__(index: int, value: vectorio.Circle | vectorio.Rectangle | vectorio.Polygon | Group | TileGrid) → None`
   Sets the value at the given index.
   This allows you to:

   ```python
   group[0] = sprite
   ```

`__delitem__(index: int) → None`
   Deletes the value at the given index.
   This allows you to:

   ```python
   del group[0]
   ```

`sort(key: function, reverse: bool) → None`
   Sort the members of the group.
**class** `displayio.I2CDisplay(i2c_bus: busio.I2C, *, device_address: int, reset: microcontroller.Pin | None = None)`

Manage updating a display over I2C in the background while Python code runs. It doesn’t handle display initialization.

Create a I2CDisplay object associated with the given I2C bus and reset pin.

The I2C bus and pins are then in use by the display until `displayio.release_displays()` is called even after a reload. (It does this so CircuitPython can use the display after your code is done.) So, the first time you initialize a display bus in code.py you should call `displayio.release_displays()` first, otherwise it will error after the first code.py run.

**Parameters**

- `i2c_bus` *(busio.I2C)* – The I2C bus that make up the clock and data lines
- `device_address` *(int)* – The I2C address of the device
- `reset` *(microcontroller.Pin)* – Reset pin. When None only software reset can be used

**reset()** → None

Performs a hardware reset via the reset pin. Raises an exception if called when no reset pin is available.

**send(command: int, data: circuitpython_typing.ReadableBuffer)** → None

Sends the given command value followed by the full set of data. Display state, such as vertical scroll, set via send may or may not be reset once the code is done.

**class** `displayio.OnDiskBitmap(file: str | BinaryIO)`

Loads values straight from disk. This minimizes memory use but can lead to much slower pixel load times. These load times may result in frame tearing where only part of the image is visible.

It’s easiest to use on a board with a built in display such as the Hallowing M0 Express.

```python
import board
import displayio
import time
import pulseio

board.DISPLAY.brightness = 0
splash = displayio.Group()
board.DISPLAY.show(splash)

odb = displayio.OnDiskBitmap('/sample.bmp')
face = displayio.TileGrid(odb, pixel_shader=odb.pixel_shader)
splash.append(face)
# Wait for the image to load.
board.DISPLAY.refresh(target_frames_per_second=60)

# Fade up the backlight
for i in range(100):
    board.DISPLAY.brightness = 0.01 * i
    time.sleep(0.05)

# Wait forever
while True:
    pass
```

Create an OnDiskBitmap object with the given file.
Parameters

- **file** (*file*) – The name of the bitmap file. For backwards compatibility, a file opened in binary mode may also be passed.

Older versions of CircuitPython required a file opened in binary mode. CircuitPython 7.0 modified OnDiskBitmap so that it takes a filename instead, and opens the file internally. A future version of CircuitPython will remove the ability to pass in an opened file.

- **width**: *int*
  Width of the bitmap. (read only)

- **height**: *int*
  Height of the bitmap. (read only)

- **pixel_shader**: `ColorConverter` | `Palette`
  The image's pixel shader. The type depends on the underlying bitmap's structure. The pixel shader can be modified (e.g., to set the transparent pixel or, for palette shaded images, to update the palette.)

class `displayio.Palette(color_count: int, *, dither: bool = False)`

Map a pixel palette_index to a full color. Colors are transformed to the display's format internally to save memory.

Create a Palette object to store a set number of colors.

Parameters

- • **color_count** (*int*) – The number of colors in the Palette
- • **dither** (*bool*) – When true, dither the RGB color before converting to the display’s color space

- **dither**: *bool*
  When True the Palette dithers the output color by adding random noise when truncating to display bitdepth

- **__bool__** () → *bool*

- **__len__** () → *int*
  Returns the number of colors in a Palette

- **__getitem__** (index: *int*) → *int* | *None*
  Return the pixel color at the given index as an integer.

- **__setitem__** (index: *int*, value: *int* | `circuitpython_typing.ReadableBuffer` | `Tuple[int, int, int]`) → *None*
  Sets the pixel color at the given index. The index should be an integer in the range 0 to color_count-1.

  The value argument represents a color, and can be from 0x000000 to 0xFFFFFFFF (to represent an RGB value). Value can be an int, bytes (3 bytes (RGB) or 4 bytes (RGB + pad byte)), bytearray, or a tuple or list of 3 integers.

  This allows you to:

  ```python
  palette[0] = 0xFFFFFF # set using an integer
  palette[1] = b'\xff\xff\x00' # set using 3 bytes
  palette[2] = b'\xff\xff\x00\x00' # set using 4 bytes
  palette[3] = bytearray(b'\x00\x00\x0FF') # set using a bytearray of 3 or 4 bytes
  palette[4] = (10, 20, 30) # set using a tuple of 3 integers
  ```

- **make_transparent** (palette_index: *int*) → *None*

- **make_opaque** (palette_index: *int*) → *None*
is_transparent(palette_index: int) → bool

Returns True if the palette index is transparent. Returns False if opaque.

class displayio.Shape(width: int, height: int, *, mirror_x: bool = False, mirror_y: bool = False)

Represents a shape made by defining boundaries that may be mirrored.

Create a Shape object with the given fixed size. Each pixel is one bit and is stored by the column boundaries of
the shape on each row. Each row’s boundary defaults to the full row.

Parameters

• width (int) – The number of pixels wide
• height (int) – The number of pixels high
• mirror_x (bool) – When true the left boundary is mirrored to the right.
• mirror_y (bool) – When true the top boundary is mirrored to the bottom.

set_boundary(y: int, start_x: int, end_x: int) → None

Loads pre-packed data into the given row.

class displayio.TileGrid(bitmap: Bitmap | OnDiskBitmap | Shape, *, pixel_shader: ColorConverter | Palette, width: int = 1, height: int = 1, tile_width: int | None = None, tile_height: int | None = None, default_tile: int = 0, x: int = 0, y: int = 0)

A grid of tiles sourced out of one bitmap

Position a grid of tiles sourced from a bitmap and pixel_shader combination. Multiple grids can share bitmaps
and pixel shaders.

A single tile grid is also known as a Sprite.

Create a TileGrid object. The bitmap is source for 2d pixels. The pixel_shader is used to convert the value and
its location to a display native pixel color. This may be a simple color palette lookup, a gradient, a pattern or a
color transformer.

To save RAM usage, tile values are only allowed in the range from 0 to 255 inclusive (single byte values).

tile_width and tile_height match the height of the bitmap by default.

Parameters

• bitmap (Bitmap, OnDiskBitmap, Shape) – The bitmap storing one or more tiles.
• pixel_shader (ColorConverter, Palette) – The pixel shader that produces colors from values
• width (int) – Width of the grid in tiles.
• height (int) – Height of the grid in tiles.
• tile_width (int) – Width of a single tile in pixels. Defaults to the full Bitmap and must
  evenly divide into the Bitmap’s dimensions.
• tile_height (int) – Height of a single tile in pixels. Defaults to the full Bitmap and must
  evenly divide into the Bitmap’s dimensions.
• default_tile (int) – Default tile index to show.
• x (int) – Initial x position of the left edge within the parent.
• y (int) – Initial y position of the top edge within the parent.
• hidden: bool

True when the TileGrid is hidden. This may be False even when a part of a hidden Group.
x: int
  X position of the left edge in the parent.

y: int
  Y position of the top edge in the parent.

width: int
  Width of the tilegrid in tiles.

height: int
  Height of the tilegrid in tiles.

tile_width: int
  Width of a single tile in pixels.

tile_height: int
  Height of a single tile in pixels.

flip_x: bool
  If true, the left edge rendered will be the right edge of the right-most tile.

flip_y: bool
  If true, the top edge rendered will be the bottom edge of the bottom-most tile.

transpose_xy: bool
  If true, the TileGrid’s axis will be swapped. When combined with mirroring, any 90 degree rotation can
  be achieved along with the corresponding mirrored version.

pixel_shader: ColorConverter | Palette
  The pixel shader of the tilegrid.

bitmap: Bitmap | OnDiskBitmap | Shape
  The bitmap of the tilegrid.

contains(touch_tuple: tuple) → bool
  Returns True if the first two values in touch_tuple represent an x,y coordinate inside the tilegrid rectangle
  bounds.

__getitem__(index: Tuple[int, int] | int) → int
  Returns the tile index at the given index. The index can either be an x,y tuple or an int equal to y * width
  + x.

This allows you to:

print(grid[0])

__setitem__(index: Tuple[int, int] | int, value: int) → None
  Sets the tile index at the given index. The index can either be an x,y tuple or an int equal to y * width +
  x.

This allows you to:

grid[0] = 10

or:

grid[0,0] = 10
dualbank – Dualbank Module

The dualbank module adds ability to update and switch between the two identical app partitions, which can contain different firmware versions.

Having two partitions enables rollback functionality.

The two partitions are defined as the boot partition and the next-update partition. Calling `dualbank.flash()` writes the next-update partition.

After the next-update partition is written a validation check is performed and on a successful validation this partition is set as the boot partition. On next reset, firmware will be loaded from this partition.

Use cases:

- Can be used for OTA Over-The-Air updates.
- Can be used for dual-boot of different firmware versions or platforms.

Note:

Boards with flash =2MB:
This module is unavailable as the flash is only large enough for one app partition.

Boards with flash >2MB:
This module is enabled/disabled at runtime based on whether the CIRCUITPY drive is extended or not. See `storage.erase_filesystem()` for more information.

```python
import dualbank
dualbank.flash(buffer, offset)
dualbank.switch()
```

dualbank.flash(buffer: circuitpython_typing.ReadableBuffer, offset: int = 0) → None

Writes one of the two app partitions at the given offset.

This can be called multiple times when flashing the firmware in smaller chunks.

Parameters

- `buffer` (ReadableBuffer) – The entire firmware or a partial chunk.
- `offset` (int) – Start writing at this offset in the app partition.

dualbank.switch() → None

Switches to the next-update partition.

On next reset, firmware will be loaded from the partition just switched over to.
espcamera – Wrapper for the espcamera library

This library enables access to any camera sensor supported by the library, including OV5640 and OV2640.

See also:
Non-Espressif microcontrollers use the imagecapture module together with wrapper libraries such as adafruit_ov5640.

```python
class espcamera.GrabMode
    Controls when a new frame is grabbed.
    WHEN_EMPTY: GrabMode
        Fills buffers when they are empty. Less resources but first fb_count frames might be old
    LATEST: GrabMode
        Except when 1 frame buffer is used, queue will always contain the last fb_count frames
```

```python
class espcamera.PixelFormat
    Format of data in the captured frames
    RGB565: PixelFormat
        A 16-bit format with 5 bits of Red and Blue and 6 bits of Green
    GRAYSCALE: PixelFormat
        An 8-bit format with 8-bits of luminance
    JPEG: PixelFormat
        A compressed format
```

```python
class espcamera.FrameSize
    The pixel size of the captured frames
    R96X96: FrameSize
        96x96
    QQVGA: FrameSize
        160x120
    QCIF: FrameSize
        176x144
    HQVGA: FrameSize
        240x176
    R240X240: FrameSize
        240x240
    QVGA: FrameSize
        320x240
    CIF: FrameSize
        400x296
    HVGA: FrameSize
        480x320
    VGA: FrameSize
        640x480
```
SVGA: FrameSize
800x600

XGA: FrameSize
1024x768

HD: FrameSize
1280x720

SXGA: FrameSize
1280x1024

UXGA: FrameSize
1600x1200

FHD: FrameSize
1920x1080

P_HD: FrameSize
720x1280

P_3MP: FrameSize
864x1536

QXGA: FrameSize
2048x1536

QHD: FrameSize
2560x1440

WQXGA: FrameSize
2560x1600

P_FHD: FrameSize
1080x1920

QSXGA: FrameSize
2560x1920

class esp-camera.GainCeiling

The maximum amount of gain applied to raw sensor data.

Higher values are useful in darker conditions, but increase image noise.

GAIN_2X: GainCeiling

GAIN_4X: GainCeiling

GAIN_8X: GainCeiling

GAIN_16X: GainCeiling

GAIN_32X: GainCeiling

GAIN_64X: GainCeiling

GAIN_128X: GainCeiling
class espcamera.Camera(*, data_pins: List[microcontroller.Pin], pixel_clock_pin: microcontroller.Pin,
                        vsync_pin: microcontroller.Pin, href_pin: microcontroller.Pin, i2c: busio.I2C,
                        external_clock_pin: microcontroller.Pin | None = None, external_clock_frequency: int
                        = 20000000, powerdown_pin: microcontroller.Pin | None = None, reset_pin:
                        microcontroller.Pin | None = None, pixel_format:PixelFormat =
                        PixelFormat.RGB565, frame_size: FrameSize = FrameSize.QQVGA, jpeg_quality: int
                        = 15, framebuffer_count: int = 1, grab_mode: GrabMode =
                        GrabMode.WHEN_EMPTY)

Configure and initialize a camera with the given properties

This driver requires that the `CIRCUITPY_RESERVED_PSRAM` in `settings.toml` be large enough to hold the camera framebuffer(s). Generally, boards with built-in cameras will have a default setting that is large enough. If the constructor raises a MemoryError or an IDFError, this probably indicates the setting is too small and should be increased.

**Important:** Not all supported sensors have all of the properties listed below. For instance, the OV5640 supports `denoise`, but the OV2640 does not. The underlying esp32-camera library does not provide a reliable API to check which settings are supported. CircuitPython makes a best effort to determine when an unsupported property is set and will raise an exception in that case.

**Parameters**

- **data_pins** – The 8 data data_pins used for image data transfer from the camera module,
  least significant bit first

- **pixel_clock_pin** – The pixel clock output from the camera module

- **vsync_pin** – The vertical sync pulse output from the camera module

- **href_pin** – The horizontal reference output from the camera module

- **i2c** – The I2C bus connected to the camera module

- **external_clock_pin** – The pin on which to generate the external clock

- **external_clock_frequency** – The frequency generated on the external clock pin

- **powerdown_pin** – The powerdown input to the camera module

- **reset_pin** – The reset input to the camera module

- **pixel_format** – The pixel format of the captured image

- **frame_size** – The size of captured image

- **jpeg_quality** – For `PixelFormat.JPG`, the quality. Higher numbers increase quality. If the quality is too high, the JPEG data will be larger than the available buffer size and the image will be unusable or truncated. The exact range of appropriate values depends on the sensor and must be determined empirically.

- **framebuffer_count** – The number of framebuffers (1 for single-buffered and 2 for double-buffered)

- **grab_mode** – When to grab a new frame

**frame_available:** bool

True if a frame is available, False otherwise
**pixel_format**: *PixelFormat*

The pixel format of captured frames

**frame_size**: *FrameSize*

The size of captured frames

**contrast**: int

The sensor contrast. Positive values increase contrast, negative values lower it. The total range is device-specific but is often from -2 to +2 inclusive.

**brightness**: int

The sensor brightness. Positive values increase brightness, negative values lower it. The total range is device-specific but is often from -2 to +2 inclusive.

**saturation**: int

The sensor saturation. Positive values increase saturation (more vibrant colors), negative values lower it (more muted colors). The total range is device-specific but the value is often from -2 to +2 inclusive.

**sharpness**: int

The sensor sharpness. Positive values increase sharpness (more defined edges), negative values lower it (softer edges). The total range is device-specific but the value is often from -2 to +2 inclusive.

**denoise**: int

The sensor ‘denoise’ setting. Any camera sensor has inherent ‘noise’, especially in low brightness environments. Software algorithms can decrease noise at the expense of fine detail. A larger value increases the amount of software noise removal. The total range is device-specific but the value is often from 0 to 10.

**gain_ceiling**: *GainCeiling*

The sensor ‘gain ceiling’ setting. “Gain” is an analog multiplier applied to the raw sensor data. The ‘ceiling’ is the maximum gain value that the sensor will use. A higher gain means that the sensor has a greater response to light, but also makes sensor noise more visible.

**quality**: int

The ‘quality’ setting when capturing JPEG images. This is similar to the quality setting when exporting a jpeg image from photo editing software. Typical values range from 5 to 40, with higher numbers leading to larger image sizes and better overall image quality. However, when the quality is set to a high number, the total size of the JPEG data can exceed the size of an internal buffer, causing image capture to fail.

**colorbar**: bool

When True, a test pattern image is captured and the real sensor data is not used.

**whitebal**: bool

When True, the camera attempts to automatically control white balance. When False, the wb_mode setting is used instead.

**gain_ctrl**: bool

When True, the camera attempts to automatically control the sensor gain, up to the value in the gain_ceiling property. When False, the agc_gain setting is used instead.

**exposure_ctrl**: bool

When True the camera attempts to automatically control the exposure. When False, the aec_value setting is used instead.

**hmirror**: bool

When True the camera image is mirrored left-to-right
vflip: bool
   When True the camera image is flipped top-to-bottom

aec2: bool
   When True the sensor’s “night mode” is enabled, extending the range of automatic gain control.

awb_gain: bool
   Access the awb_gain property of the camera sensor

agc_gain: int
   Access the gain level of the sensor. Higher values produce brighter images. Typical settings range from 0 to 30.

aec_value: int
   Access the exposure value of the camera. Higher values produce brighter images. Typical settings range from 0 to 1200.

special_effect: int
   Enable a “special effect”. Zero is no special effect. On OV5640, special effects range from 0 to 6 inclusive and select various color modes.

wb_mode: int
   The white balance mode. 0 is automatic white balance. Typical values range from 0 to 4 inclusive.

ae_level: int
   The exposure offset for automatic exposure. Typical values range from -2 to +2.

dcw: bool
   When True an advanced white balance mode is selected.

bpc: bool
   When True, “black point compensation” is enabled. This can make black parts of the image darker.

wpc: bool
   When True, “white point compensation” is enabled. This can make white parts of the image whiter.

raw_gma: bool
   When True, raw gamma mode is enabled.

dlenc: bool
   Enable “lens correction”. This can help compensate for light fall-off at the edge of the sensor area.

max_frame_size: FrameSize
   The maximum frame size that can be captured

address: int
   The I2C (SCCB) address of the sensor

sensor_name: str
   The name of the sensor

supports_jpeg: bool
   True if the sensor can capture images in JPEG format

height: int
   The height of the image being captured
width: int
   The width of the image being captured

grab_mode: GrabMode
   The grab mode of the camera

framebuffer_count: int
   True if double buffering is used

definit() → None
   Deinitialises the camera and releases all memory resources for reuse.

__enter__() → Camera
   No-op used by Context Managers.

__exit__() → None
   Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

take(timeout: float | None = 0.25) → displayio.Bitmap | circuitpython_typing.ReadableBuffer | None
   Record a frame. Wait up to ‘timeout’ seconds for a frame to be captured.

   In the case of timeout, None is returned. If pixel_format is PixelFormat.JPG, the returned value is a read-only memoryview. Otherwise, the returned value is a read-only displayio.Bitmap.

reconfigure(frame_size: FrameSize | None = None, pixel_format: PixelFormat | None = None, grab_mode: GrabMode | None = None, framebuffer_count: int | None = None) → None
   Change multiple related camera settings simultaneously

   Because these settings interact in complex ways, and take longer than the other properties to set, they are set together in a single function call.

   If an argument is unspecified or None, then the setting is unchanged.

espidf – Return the total size of the ESP-IDF, which includes the CircuitPython heap.

espidf.heap_caps_get_total_size() → int

espidf.heap_caps_get_free_size() → int
   Return total free memory in the ESP-IDF heap.

espidf.heap_caps_get_largest_free_block() → int
   Return the size of largest free memory block in the ESP-IDF heap.

espidf.erase_nvs() → None
   Erase all data in the non-volatile storage (nvs), including data stored by with microcontroller.nvm

   This is necessary when upgrading from CircuitPython 6.3.0 or earlier to CircuitPython 7.0.0, because the layout of data in nvs has changed. The old data will be lost when you perform this operation.

exception espidf.IDFError
   Bases: OSError

   Raised when an ESP-IDF function returns an error code. esp_err_t

   Initialize self. See help(type(self)) for accurate signature.
exception espidf.MemoryError
    Bases: MemoryError
    Raised when an ESP-IDF memory allocation fails.
    Initialize self. See help(type(self)) for accurate signature.

espidf.get_total_psram() → int
    Returns the number of bytes of psram detected, or 0 if psram is not present or not configured

espidf.get_reserved_psram() → int
    Returns number of bytes of psram reserved for use by esp-idf, either a board-specific default value or the value defined in settings.toml.

**espnow – ESP-NOW Module**

The *espnow* module provides an interface to the ESP-NOW protocol provided by Espressif on its SoCs.

**Sender**

```python
import espnow

e = espnow.ESPNow()
peer = espnow.Peer(mac=b'aaaaa')
e.peers.append(peer)

e.send("Starting...")
for i in range(10):
    e.send(str(i)*20)
e.send(b'end')
```

**Receiver**

```python
import espnow

e = espnow.ESPNow()
packets = []

while True:
    if e:
        packet = e.read()
packets.append(packet)
        if packet.msg == b'end':
            break

print("packets: ", f"length={len(packets)}")
for packet in packets:
    print(packet)
```

**class espnow.ESPNow(buffer_size: int = 526, phy_rate: int = 0)**

Provides access to the ESP-NOW protocol.

Allocate and initialize ESPNow instance as a singleton.

**Parameters**

- **buffer_size** (int) – The size of the internal ring buffer. Default: 526 bytes.
• **phy_rate (int)** – The ESP-NOW physical layer rate. Default: 1 Mbps. wifi_phy_rate_t

**send_success**: int

The number of tx packets received by the peer(s) ESP_NOW_SEND_SUCCESS. (read-only)

**send_failure**: int

The number of failed tx packets ESP_NOW_SEND_FAIL. (read-only)

**read_success**: int

The number of rx packets captured in the buffer. (read-only)

**read_failure**: int

The number of dropped rx packets due to buffer overflow. (read-only)

**buffer_size**: int

The size of the internal ring buffer. (read-only)

**phy_rate**: int

The ESP-NOW physical layer rate. wifi_phy_rate_t

**peers**: Peers

The peer info records for all registered ESPNow peers. (read-only)

deinit () → None

Deinitializes ESP-NOW and releases it for another program.

__enter__ () → ESPNow

No-op used by Context Managers.

__exit__ () → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**send**(message: circuitpython_typing.ReadableBuffer, peer: Peer | None = None) → None

Send a message to the peer’s mac address.

This blocks until a timeout of 2 seconds if the ESP-NOW internal buffers are full.

**Parameters**

• message (ReadableBuffer) – The message to send (length <= 250 bytes).

• peer (Peer) – Send message to this peer. If None, send to all registered peers.

**read()** → ESPNowPacket | None

Read a packet from the receive buffer.

This is non-blocking, the packet is received asynchronously from the peer(s).

**Returns**

An ESPNowPacket if available in the buffer, otherwise None.

**set_pmk**(pmk: circuitpython_typing.ReadableBuffer) → None

Set the ESP-NOW Primary Master Key (pmk) for encrypted communications.

**Parameters**

• pmk (ReadableBuffer) – The ESP-NOW Primary Master Key (length = 16 bytes).

__bool__ () → bool

True if len() is greater than zero. This is an easy way to check if the buffer is empty.
__len__() → int
Return the number of bytes available to read. Used to implement len().

class espnow.ESPNowPacket
A packet retrieved from ESP-NOW communication protocol. A namedtuple.

mac: circuitpython_typing.ReadableBuffer
The sender’s mac address (length = 6 bytes).

msg: circuitpython_typing.ReadableBuffer
The message sent by the peer (length <= 250 bytes).

rssi: int
The received signal strength indication (in dBm from -127 to 0).

time: int
The time in milliseconds since the device last booted when the packet was received.

class espnow.Peer(mac: bytes, *, lmk: bytes | None, channel: int = 0, interface: int = 0, encrypted: bool = False)
A data class to store parameters specific to a peer.

Construct a new peer object.

Parameters

• mac (bytes) – The mac address of the peer.
• lmk (bytes) – The Local Master Key (lmk) of the peer.
• channel (int) – The peer’s channel. Default: 0 ie. use the current channel.
• interface (int) – The WiFi interface to use. Default: 0 ie. STA.
• encrypted (bool) – Whether or not to use encryption.

mac: circuitpython_typing.ReadableBuffer
The WiFi mac to use.

lmk: circuitpython_typing.ReadableBuffer
The WiFi lmk to use.

channel: int
The WiFi channel to use.

interface: int
The WiFi interface to use.

encrypted: bool
Whether or not to use encryption.

class espnow.Peers
Maintains a list of Peer internally and only exposes a subset of list methods.

You cannot create an instance of Peers.

append(peer: Peer) → None
Append peer.

Parameters

peer (Peer) – The peer object to append.
**remove** *(peer: Peer) → None*

Remove peer.

**Parameters**

*peer (Peer)* – The peer object to remove.

---

**espulp – ESP Ultra Low Power Processor Module**

The *espulp* module adds ability to load and run programs on the ESP32-Sx’s ultra-low-power RISC-V processor.

```python
import espulp
import memorymap

shared_mem = memorymap.AddressRange(start=0x50000000, length=1024)
ulp = espulp.ULP()

with open("program.bin", "rb") as f:
    program = f.read()

ulp.run(program)
pin = shared_mem[0]
# ulp.halt()
```

**espulp.get_rtc_gpio_number** *(pin: microcontroller.Pin) → int | None*

Return the RTC GPIO number of the given pin or None if not connected to RTC GPIO.

**class espulp.Architecture**

The ULP architectures available.

*FSM: Architecture*

The ULP Finite State Machine.

*RISCV: Architecture*

The ULP RISC-V Coprocessor.

**class espulp.ULP** *(arch: Architecture = Architecture.FSM)*

The ultra-low-power processor.

Raises an exception if another ULP has been instantiated. This ensures that is is only used by one piece of code at a time.

**Parameters**

*arch (Architecture)* – The ulp arch

*arch: Architecture*

The ulp architecture. (read-only)

**deinit** *(arch: Architecture) → None*

Deinitialises the ULP and releases it for another program.

**__enter__() → ULP**

No-op used by Context Managers.

**__exit__() → None**

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.
run

Loads the program into ULP memory and then runs the program. The given pins are claimed and not reset until `halt()` is called.

The program will continue to run even when the running Python is halted.

halt() → None

Halts the running program and releases the pins given in `run()`.

class espulp.ULPAlarm(ulp: ULP)

Trigger an alarm when the ULP requests wake-up.

Create an alarm that will be triggered when the ULP requests wake-up.

The alarm is not active until it is passed to an `alarm-enabling function, such as `alarm.
light_sleep_until_alarms()` or `alarm.exit_and_deep_sleep_until_alarms()`.

Parameters
ulp (ULP) – Theulp instance

floppyio – Read flux transition information into the buffer.

floppyio.flux_readinto(buffer: circuitpython_typing.WriteableBuffer, data: digitalio.DigitalInOut, index: digitalio.DigitalInOut) → int

The function returns when the buffer has filled, or when the index input indicates that one full revolution of data has been recorded. Due to technical limitations, this process may not be interruptible by KeyboardInterrupt.

Parameters

- buffer – Read data into this buffer. Each element represents the time between successive zero-to-one transitions.
- data – Pin on which the flux data appears
- index – Pin on which the index pulse appears

Returns
The actual number of bytes of read

floppyio.mfm_readinto(buffer: circuitpython_typing.WriteableBuffer, data: digitalio.DigitalInOut, index: digitalio.DigitalInOut) → int

Read mfm blocks into the buffer.

The track is assumed to consist of 512-byte sectors.

The function returns when all sectors have been successfully read, or a number of index pulses have occurred. Due to technical limitations, this process may not be interruptible by KeyboardInterrupt.

Parameters

- buffer – Read data into this buffer. Must be a multiple of 512.
- data – Pin on which the mfm data appears
- index – Pin on which the index pulse appears

Returns
The actual number of sectors read

floppyio.samplerate: int

The approximate sample rate in Hz used by flux_readinto.
fontio – Core font related data structures

Note: This module is intended only for low-level usage. For working with fonts in CircuitPython see the adafruit_bitmap_font library. For information on creating custom fonts for use in CircuitPython, see this Learn guide

class fontio.FontProtocol
    Bases: typing_extensions.Protocol

    A protocol shared by BuiltinFont and classes in adafruit_bitmap_font

    get_bounding_box() → Tuple[int, int] | Tuple[int, int, int, int]
        Retrieve the maximum bounding box of any glyph in the font.

        The four element version is (width, height, x_offset, y_offset). The two element version is (width, height), in which x_offset and y_offset are assumed to be zero.

    get_glyph(codepoint: int) → Glyph | None
        Retrieve the Glyph for a given code point

        If the code point is not present in the font, None is returned.

class fontio.BuiltinFont

    A font built into CircuitPython

    Creation not supported. Available fonts are defined when CircuitPython is built. See the Adafruit_CircuitPython_Bitmap_Font library for dynamically loaded fonts.

    bitmap: displayio.Bitmap
        Bitmap containing all font glyphs starting with ASCII and followed by unicode. Use get_glyph in most cases. This is useful for use with displayio.TileGrid and terminalio.Terminal.

    get_bounding_box() → Tuple[int, int]
        Returns the maximum bounds of all glyphs in the font in a tuple of two values: width, height.

    get_glyph(codepoint: int) → Glyph
        Returns a fontio.Glyph for the given codepoint or None if no glyph is available.


    Storage of glyph info

    Named tuple used to capture a single glyph and its attributes.

    Parameters

    * bitmap – the bitmap including the glyph
    * tile_index – the tile index within the bitmap
    * width – the width of the glyph’s bitmap
    * height – the height of the glyph’s bitmap
    * dx – x adjustment to the bitmap’s position
    * dy – y adjustment to the bitmap’s position
    * shift_x – the x difference to the next glyph
    * shift_y – the y difference to the next glyph
framebufferio – Native framebuffer display driving

The framebufferio module contains classes to manage display output including synchronizing with refresh rates and partial updating. It is used in conjunction with classes from displayio to actually place items on the display; and classes like RGBMatrix to actually drive the display.

```
class framebufferio.FramebufferDisplay(framebuffer: circuitpython_typing.FrameBuffer, *, rotation: int = 0, auto_refresh: bool = True)
```

Manage updating a display with framebuffer in RAM

This initializes a display and connects it into CircuitPython. Unlike other objects in CircuitPython, Display objects live until displayio.release_displays() is called. This is done so that CircuitPython can use the display itself.

Create a Display object with the given framebuffer (a buffer, array, ulab.array, etc)

**Parameters**

- **framebuffer (FrameBuffer)** – The framebuffer that the display is connected to
- **auto_refresh (bool)** – Automatically refresh the screen
- **rotation (int)** – The rotation of the display in degrees clockwise. Must be in 90 degree increments (0, 90, 180, 270)

**auto_refresh**: bool

True when the display is refreshed automatically.

**brightness**: float

The brightness of the display as a float. 0.0 is off and 1.0 is full brightness.

**width**: int

Gets the width of the framebuffer

**height**: int

Gets the height of the framebuffer

**rotation**: int

The rotation of the display as an int in degrees.

**framebuffer**: circuitpython_typing.FrameBuffer

The framebuffer being used by the display

**root_group**: displayio.Group

The root group on the display. If the root group is set to displayio.CIRCUITPYTHON_TERMINAL, the default CircuitPython terminal will be shown. If the root group is set to None, no output will be shown.

**show**(group: displayio.Group) → None

Switches to displaying the given group of layers. When group is None, the default CircuitPython terminal will be shown.

**Parameters**

- **group (Group)** – The group to show.

**refresh**(*, target_frames_per_second: int = 60, minimum_frames_per_second: int = 1) → bool

When auto refresh is off, waits for the target frame rate and then refreshes the display, returning True. If the call has taken too long since the last refresh call for the given target frame rate, then the refresh returns False immediately without updating the screen to hopefully help getting caught up.
If the time since the last successful refresh is below the minimum frame rate, then an exception will be raised. Set minimum_frames_per_second to 0 to disable.

When auto refresh is on, updates the display immediately. (The display will also update without calls to this.)

**Parameters**

- **target_frames_per_second** (*int*) – How many times a second refresh should be called and the screen updated.

- **minimum_frames_per_second** (*int*) – The minimum number of times the screen should be updated per second.

**fill_row**(*y: int, buffer: circuitpython_typing.WriteableBuffer*) → circuitpython_typing.WriteableBuffer

Extract the pixels from a single row

**Parameters**

- **y** (*int*) – The top edge of the area

- **buffer** (*WriteableBuffer*) – The buffer in which to place the pixel data

---

### frequencyio – Support for frequency based protocols

**Warning:** This module is not available in SAMD21 builds. See the Module Support Matrix - Which Modules Are Available on Which Boards for more info.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See *Lifetime and ContextManagers* for more info.

For example:

```python
import time
import frequencyio
import board

frequency = frequencyio.FrequencyIn(board.D11)
frequency.capture_period = 15
time.sleep(0.1)
```

This example will initialize the device, set `capture_period`, and then sleep 0.1 seconds. CircuitPython will automatically turn off FrequencyIn capture when it resets all hardware after program completion. Use `deinit()` or a `with` statement to do it yourself.

**class frequencyio.FrequencyIn**(*pin: microcontroller.Pin, capture_period: int = 10*)

Read a frequency signal

FrequencyIn is used to measure the frequency, in hertz, of a digital signal on an incoming pin. Accuracy has shown to be within 10%, if not better. It is recommended to utilize an average of multiple samples to smooth out readings.

Frequencies below 1KHz are not currently detectable.

FrequencyIn will not determine pulse width (use PulseIn).

Create a FrequencyIn object associated with the given pin.

**Parameters**

---

1.8. Full Table of Contents 209
• **pin (Pin)** – Pin to read frequency from.

• **capture_period (int)** – Keyword argument to set the measurement period, in milliseconds. Default is 10ms; range is 1ms - 500ms.

Read the incoming frequency from a pin:

```python
import frequencyio
import board

frequency = frequencyio.FrequencyIn(board.D11)

# Loop while printing the detected frequency
while True:
    print(frequency.value)

    # Optional clear() will reset the value to zero. Without this, if the incoming signal stops, the last reading will remain as the value.
    frequency.clear()
```

capture_period:  int

The capture measurement period. Lower incoming frequencies will be measured more accurately with longer capture periods. Higher frequencies are more accurate with shorter capture periods.

**Note:** When setting a new capture_period, all previous capture information is cleared with a call to clear().

deinit() → None

Deinitialises the FrequencyIn and releases any hardware resources for reuse.

__enter__() → FrequencyIn

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

pause() → None

Pause frequency capture.

resume() → None

Resumes frequency capture.

clear() → None

Clears the last detected frequency capture value.

__get__(index: int) → int

Returns the value of the last frequency captured.
getpass – Getpass Module

This module provides a way to get input from user without echoing it.

```python
getpass.getpass(prompt: str | None = 'Password: ', stream: io.FileIO | None = None) -> str
```

Prompt the user without echoing.

Parameters

• `prompt` (str) – The user is prompted using the string `prompt`, which defaults to 'Password: '

• `stream` (io.FileIO) – The prompt is written to the file-like object `stream` if provided.

gifio – Access GIF-format images


Construct a GifWriter object

Parameters

• `file` – Either a file open in bytes mode, or the name of a file to open in bytes mode.

• `width` – The width of the image. All frames must have the same width.

• `height` – The height of the image. All frames must have the same height.

• `colorspace` – The colorspace of the image. All frames must have the same colorspace. The supported colorspaces are RGB565, BGR565, RGB565_SWAPPED, BGR565_SWAPPED, and L8 (greyscale)

• `loop` – If True, the GIF is marked for looping playback

• `dither` – If True, and the image is in color, a simple ordered dither is applied.

__enter__ () → GifWriter

No-op used by Context Managers.

__exit__ () → None

Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

definit() → None

Close the underlying file.

add_frame(bitmap: circuitpython_typing.ReadableBuffer, delay: float = 0.1) → None

Add a frame to the GIF.

Parameters

• `bitmap` – The frame data

• `delay` – The frame delay in seconds. The GIF format rounds this to the nearest 1/100 second, and the largest permitted value is 655 seconds.

class gifio.OnDiskGif(file: str)

Loads one frame of a GIF into memory at a time.

The code can be used in cooperation with displayio but this mode is relatively slow:
```python
import board
import gifio
import displayio
import time

display = board.DISPLAY
splash = displayio.Group()
display.root_group = splash

odg = gifio.OnDiskGif('/sample.gif')

start = time.monotonic()
next_delay = odg.next_frame()  # Load the first frame
end = time.monotonic()
overhead = end - start

face = displayio.TileGrid(
    odg.bitmap,
    pixel_shader=displayio.ColorConverter(
        input_colorspace=displayio.Colorspace.RGB565_SWAPPED
    ),
)
splash.append(face)
board.DISPLAY.refresh()

# Display repeatedly.
while True:
    # Sleep for the frame delay specified by the GIF,
    # minus the overhead measured to advance between frames.
    time.sleep(max(0, next_delay - overhead))
    next_delay = odg.next_frame()

The displayio Group and TileGrid layers can be bypassed and the image can be directly blitted to the full screen. This can give a speed-up of ~4x to ~6x depending on the GIF and display. This requires an LCD that uses standard codes to set the update area, and which accepts RGB565_SWAPPED pixel data directly:

```
Create an `OnDiskGif` object with the given file. The GIF frames are decoded into RGB565 big-endian format. `displayio` expects little-endian, so the example above uses `Colorspace.RGB565_SWAPPED`.

Parameters

- `file (file)` – The name of the GIF file.

If the image is too large it will be cropped at the bottom and right when displayed.

Limitations: The image width is limited to 320 pixels at present. `ValueError` will be raised if the image is too wide. The height is not limited but images that are too large will cause a memory exception.

- `width: int`  
  Width of the gif. (read only)

- `height: int`  
  Height of the gif. (read only)

- `bitmap: displayio.Bitmap`  
  The bitmap used to hold the current frame.

- `palette: displayio.Palette | None`  
  The palette for the current frame if it exists.

- `duration: float`  
  Returns the total duration of the GIF in seconds. (read only)

- `frame_count: int`  
  Returns the number of frames in the GIF. (read only)

- `min_delay: float`  
  The minimum delay found between frames. (read only)

- `max_delay: float`  
  The maximum delay found between frames. (read only)

- `__enter__ () → OnDiskGif`  
  No-op used by Context Managers.

- `__exit__ () → None`  
  Automatically deinitializes the GIF when exiting a context. See `Lifetime and ContextManagers` for more info.

- `next_frame () → float`  
  Loads the next frame. Returns expected delay before the next frame in seconds.

- `deinit () → None`  
  Release resources allocated by OnDiskGif.
gnss – Global Navigation Satellite System

The `gnss` module contains classes to control the GNSS and acquire positioning information.

```python
class gnss.GNSS(system: SatelliteSystem | List[SatelliteSystem])
    Get updated positioning information from Global Navigation Satellite System (GNSS)
    Usage:

    import gnss
    import time

    nav = gnss.GNSS([gnss.SatelliteSystem.GPS, gnss.SatelliteSystem.GLONASS])
    last_print = time.monotonic()
    while True:
        nav.update()
        current = time.monotonic()
        if current - last_print >= 1.0:
            last_print = current
            if nav.fix is gnss.PositionFix.INVALID:
                print("Waiting for fix...")
                continue
            print("Latitude: {0:.6f} degrees".format(nav.latitude))
            print("Longitude: {0:.6f} degrees".format(nav.longitude))

    Turn on the GNSS.
    Parameters
    system – satellite system to use

    latitude: float
    Latitude of current position in degrees (float).

    longitude: float
    Longitude of current position in degrees (float).

    altitude: float
    Altitude of current position in meters (float).

    timestamp: time.struct_time
    Time when the position data was updated.

    fix: PositionFix
    Fix mode.

    deinit() → None
    Turn off the GNSS.

    update() → None
    Update GNSS positioning information.
```

```python
class gnss.PositionFix
    Position fix mode
    Enum-like class to define the position fix mode.

    INVALID: PositionFix
    No measurement.
```
FIX_2D: PositionFix
2D fix.

FIX_3D: PositionFix
3D fix.

class gnss.SatelliteSystem
Satellite system type
Enum-like class to define the satellite system type.

GPS: SatelliteSystem
Global Positioning System.

GLONASS: SatelliteSystem
GLObal NAvigation Satellite System.

SBAS: SatelliteSystem
Satellite Based Augmentation System.

QZSS_L1CA: SatelliteSystem
Quasi-Zenith Satellite System L1C/A.

QZSS_L1S: SatelliteSystem
Quasi-Zenith Satellite System L1S.

hashlib – Hashing related functions

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: hashlib.

hashlib.new(name: str, data: bytes = b") → Hash

Returns a Hash object setup for the named algorithm. Raises ValueError when the named algorithm is unsupported.

Return type
hashlib.Hash

class hashlib.Hash
In progress hash algorithm. This object is always created by a hashlib.new(). It has no user-visible constructor.

digest_size: int
Digest size in bytes

update(data: circuitpython_typing.ReadableBuffer) → None
Update the hash with the given bytes.

Parameters
data (ReadableBuffer) – Update the hash from data in this buffer

digest() → bytes
Returns the current digest as bytes() with a length of hashlib.Hash.digest_size.
i2ctarget – Two wire serial protocol target

The *i2ctarget* module contains classes to support an I2C target.

Example emulating a target with 2 addresses (read and write):

```python
import board
from i2ctarget import I2CTarget

regs = [0] * 16
index = 0

with I2CTarget(board.SCL, board.SDA, (0x40, 0x41)) as device:
    while True:
        r = device.request()
        if not r:
            # Maybe do some housekeeping
            continue
        with r:
            # Closes the transfer if necessary by sending a NACK or feeding dummy
            # byte
            if r.address == 0x40:
                if not r.is_read:
                    # Main write which is Selected read
                    b = r.read(1)
                    if not b or b[0] > 15:
                        break
                    index = b[0]
                    b = r.read(1)
                    if b:
                        regs[index] = b[0]
                elif r.is_restart:
                    # Combined transfer: This is the Main read message
                    n = r.write(bytes([regs[index]]))
            # else:
            # A read transfer is not supported in this example
            # If the microcontroller tries, it will get 0xff byte(s) by the ctx

        if r.address == 0x41:
            if not r.is_read:
                b = r.read(1)
                if b and b[0] == 0xde:
                    # do something
                    pass
```

This example sets up an I2C device that can be accessed from Linux like this:

```
$ i2cget -y 1 0x40 0x01
0x00
$ i2cset -y 1 0x40 0x01 0xaa
$ i2cget -y 1 0x40 0x01
0xaa
```

**Warning:** *I2CTarget* makes use of clock stretching in order to slow down the host. Make sure the I2C host supports this.
Raspberry Pi in particular does not support this with its I2C hw block. This can be worked around by using the `i2c-gpio` bit banging driver. Since the RPi firmware uses the hw i2c, it’s not possible to emulate a HAT eeprom.

class `i2ctarget.I2CTarget`(scl: microcontroller.Pin, sda: microcontroller.Pin, addresses: Sequence[int], smbus: bool = False)

Two wire serial protocol target
I2C is a two-wire protocol for communicating between devices. This implements the target (peripheral, sensor, secondary) side.

Parameters
• scl (Pin) – The clock pin
• sda (Pin) – The data pin
• addresses (list[int]) – The I2C addresses to respond to (how many is hardware dependent).
• smbus (bool) – Use SMBUS timings if the hardware supports it

definit() → None
Releases control of the underlying hardware so other classes can use it.

__enter__(...) → I2CTarget
No-op used in Context Managers.

__exit__(...) → None
Automatically deinitializes the hardware on context exit. See *Lifetime and Context Managers* for more info.

request(*, timeout: float = -1) → I2CTargetRequest
Wait for an I2C request.

Parameters
timeout (float) – Timeout in seconds. Zero means wait forever, a negative value means check once

Returns
I2CTargetRequest or None if timeout=-1 and there’s no request

Return type
I2CTargetRequest

class `i2ctarget.I2CTargetRequest`(target: I2CTarget, address: int, is_read: bool, is_restart: bool)
Information about an I2C transfer request This cannot be instantiated directly, but is returned by `I2CTarget.request()`.

Parameters
• target – The I2CTarget object receiving this request
• address – I2C address
• is_read – True if the main target is requesting data
• is_restart – Repeated Start Condition

address: int
The I2C address of the request.
is_read:  bool
The I2C main controller is reading from this target.

is_restart:  bool
Is Repeated Start Condition.

__enter__()  →  I2CTargetRequest
No-op used in Context Managers.

__exit__()  →  None
Close the request.

read(n: int = -1, ack: bool = True)  →  bytearray
Read data. If ack=False, the caller is responsible for calling I2CTargetRequest.ack().

Parameters
• n – Number of bytes to read (negative means all)
• ack – Whether or not to send an ACK after the n’th byte

Returns
Bytes read

write(buffer: circuitpython_typing.ReadableBuffer)  →  int
Write the data contained in buffer.

Parameters
buffer (ReadableBuffer) – Write out the data in this buffer

Returns
Number of bytes written

ack(ack: bool = True)  →  None
Acknowledge or Not Acknowledge last byte received. Use together with I2CTargetRequest.read() ack=False.

Parameters
ack – Whether to send an ACK or NACK

imagecapture – Support for “Parallel capture” interfaces

See also:
Espressif microcontrollers use the espcamera module together.

class imagecapture.ParallelImageCapture(*, data_pins: List[microcontroller.Pin], clock:
microcontroller.Pin, vsync: microcontroller.Pin | None, href: microcontroller.Pin | None)
Capture image frames from a camera with parallel data interface

Create a parallel image capture object

This object is usually used with a camera-specific wrapper library such as adafruit_ov5640.

Parameters
• data_pins (List[microcontroller.Pin]) – The data pins.
• clock (microcontroller.Pin) – The pixel clock input.
• **vsync** *(microcontroller.Pin)* – The vertical sync input, which has a negative-going pulse at the beginning of each frame.

• **href** *(microcontroller.Pin)* – The horizontal reference input, which is high whenever the camera is transmitting valid pixel information.

capture(*buffer: circuitpython_typing.WriteableBuffer*) → circuitpython_typing.WriteableBuffer

Capture a single frame into the given buffer.

This will stop a continuous-mode capture, if one is in progress.

continuous_capture_start(*buffer1: circuitpython_typing.WriteableBuffer, buffer2: circuitpython_typing.WriteableBuffer, /
  ...) → None

Begin capturing into the given buffers in the background.

Call `continuous_capture_get_frame` to get the next available frame, and `continuous_capture_stop` to stop capturing.

Until `continuous_capture_stop` (or `deinit`) is called, the `ParallelImageCapture` object keeps references to `buffer1` and `buffer2`, so the objects will not be garbage collected.

continuous_capture_get_frame() → circuitpython_typing.WriteableBuffer

Return the next available frame, one of the two buffers passed to `continuous_capture_start`

continuous_capture_stop() → None

Stop continuous capture.

Calling this method also causes the object to release its references to the buffers passed to `continuous_capture_start`, potentially allowing the objects to be garbage collected.

deinit() → None

Deinitialize this instance

__enter__() → ParallelImageCapture

No-op used in Context Managers.

__exit__() → None

Automatically deinitializes the hardware on context exit. See *Lifetime and Context Managers* for more info.

**ipaddress**

The `ipaddress` module provides types for IP addresses. It is a subset of CPython’s ipaddress module.

`ipaddress.ip_address(obj: int | str) → IPv4Address`

Return a corresponding IP address object or raise ValueError if not possible.

**class** `ipaddress.IPv4Address(address: int | str | bytes)`

Encapsulates an IPv4 address.

Create a new IPv4Address object encapsulating the address value.

The value itself can either be bytes or a string formatted address.

packed: bytes

The bytes that make up the address (read-only).

version: int

4 for IPv4, 6 for IPv6
Two Address objects are equal if their addresses and address types are equal.

Returns a hash for the IPv4Address data.

**is31fl3741 – Creates an in-memory framebuffer for a IS31FL3741 device.**

class is31fl3741.IS31FL3741_FrameBuffer(is31: IS31FL3741, width: int, height: int, mapping: Tuple[int, Ellipsis], *, framebuffer: circuitpython_typing.WriteableBuffer | None = None, scale: bool = False, gamma: bool = False)

Create a IS31FL3741_FrameBuffer object with the given attributes.

The framebuffer is in “RGB888” format using 4 bytes per pixel. Bits 24-31 are ignored. The format is in RGB order.

If a framebuffer is not passed in, one is allocated and initialized to all black. In any case, the framebuffer can be retrieved by passing the Is31fl3741 object to memoryview().

A Is31fl3741 is often used in conjunction with a framebufferio.FramebufferDisplay.

**Parameters**

- **is31 (is31fl3741.IS31FL3741)** – base IS31FL3741 instance to drive the framebuffer
- **width (int)** – width of the display
- **height (int)** – height of the display
- **mapping (Tuple[int, ...])** – mapping of matrix locations to LEDs
- **framebuffer (Optional[WriteableBuffer])** – Optional buffer to hold the display
- **scale (bool)** – if True display is scaled down by 3 when displayed
- **gamma (bool)** – if True apply gamma correction to all LEDs

**brightness:** float

In the current implementation, 0.0 turns the display off entirely and any other value up to 1.0 turns the display on fully.

**width:** int

The width of the display, in pixels

**height:** int

The height of the display, in pixels

deinit() → None

Free the resources associated with this IS31FL3741 instance. After deinitialization, no further operations may be performed.

refresh() → None

Transmits the color data in the buffer to the pixels so that they are shown.

class is31fl3741.IS31FL3741(i2c: busio.I2C, *, addr: int = 48)

Driver for an IS31FL3741 device.

Create a IS31FL3741 object with the given attributes.

Designed to work low level or passed to and object such as IS31FL3741_FrameBuffer.
Parameters

• `i2c` ([I2C](#)) – I2C bus the IS31FL3741 is on
• `addr` ([int](#)) – device address

`deinit()` → None

Free the resources associated with this IS31FL3741 instance. After deinitialization, no further operations may be performed.

```python
is31fl3741.reset(self) → None
```

Resets the IS31FL3741 chip.

```python
is31fl3741.enable(self) → None
```

Enables the IS31FL3741 chip.

```python
is31fl3741.set_global_current(self, current: int) → None
```

Sets the global current of the IS31FL3741 chip.

Parameters

- `current` ([int](#)) – global current value 0x00 to 0xFF

```python
is31fl3741.set_led(self, led: int, value: int, page: int) → None
```

Resets the IS31FL3741 chip.

Parameters

- `led` ([int](#)) – which LED to set
- `value` ([int](#)) – value to set the LED to 0x00 to 0xFF
- `page` ([int](#)) – page to write to 0 or 2. If the LED is a >= 180 the routine will automatically write to page 1 or 3 (instead of 0 or 2)

```python
is31fl3741.write(mapping: Tuple[int, Ellipsis], buf: circuitpython_typing.ReadableBuffer) → None
```

Write buf out on the I2C bus to the IS31FL3741.

Parameters

- `mapping` (~Tuple[int, ...]) – map the pixels in the buffer to the order addressed by the driver chip
- `buf` (ReadableBuffer) – The bytes to clock out. No assumption is made about color order

### keypad – Support for scanning keys and key matrices

The `keypad` module provides native support to scan sets of keys or buttons, connected independently to individual pins, connected to a shift register, or connected in a row-and-column matrix.

For more information about working with the `keypad` module in CircuitPython, see this [Learn guide](#).

```python
class keypad.Event(key_number: int = 0, pressed: bool = True, timestamp: int | None = None)
```

A key transition event.

Create a key transition event, which reports a key-pressed or key-released transition.

Parameters

- `key_number` ([int](#)) – The key number.
- `pressed` ([bool](#)) – True if the key was pressed; False if it was released.
timestamp (int) – The time in milliseconds that the keypress occurred in the supervisor.ticks_ms time system. If specified as None, the current value of supervisor.ticks_ms is used.

key_number: int
The key number.

pressed: bool
True if the event represents a key down (pressed) transition. The opposite of released.

released: bool
True if the event represents a key up (released) transition. The opposite of pressed.

timestamp: int
The timestamp.

__eq__(other: object) → bool
Two Event objects are equal if their key_number and pressed/released values are equal. Note that this does not compare the event timestamps.

__hash__() → int
Returns a hash for the Event, so it can be used in dictionaries, etc..
Note that as events with different timestamps compare equal, they also hash to the same value.

class keypad.EventQueue
A queue of Event objects, filled by a keypad scanner such as Keys or KeyMatrix.
You cannot create an instance of EventQueue directly. Each scanner creates an instance when it is created.

overflowed: bool
True if an event could not be added to the event queue because it was full. (read-only) Set to False by clear().

get() → Event | None
Return the next key transition event. Return None if no events are pending.
Note that the queue size is limited; see max_events in the constructor of a scanner such as Keys or KeyMatrix. If a new event arrives when the queue is full, the event is discarded, and overflowed is set to True.

Returns
The next queued key transition Event.

Return type
Optional[Event]

get_into(event: Event) → bool
Store the next key transition event in the supplied event, if available, and return True. If there are no queued events, do not touch event and return False.

The advantage of this method over get() is that it does not allocate storage. Instead you can reuse an existing Event object.
Note that the queue size is limited; see max_events in the constructor of a scanner such as Keys or KeyMatrix.

Returns
True if an event was available and stored, False if not.
Return type

    bool

clear() → None
Clear any queued key transition events. Also sets overflowed to False.

__bool__() → bool
True if len() is greater than zero. This is an easy way to check if the queue is empty.

__len__() → int
Return the number of events currently in the queue. Used to implement len().

class keypad.KeyMatrix(row_pins: Sequence[microcontroller.Pin], column_pins: Sequence[microcontroller.Pin], columns_to_anodes: bool = True, interval: float = 0.02, max_events: int = 64)

Manage a 2D matrix of keys with row and column pins.

Create a Keys object that will scan the key matrix attached to the given row and column pins. There should not be any external pull-ups or pull-downs on the matrix: KeyMatrix enables internal pull-ups or pull-downs on the pins as necessary.

The keys are numbered sequentially from zero. A key number can be computed by row * len(column_pins) + column.

An EventQueue is created when this object is created and is available in the events attribute.

Parameters
- row_pins (Sequence[microcontroller.Pin]) – The pins attached to the rows.
- column_pins (Sequence[microcontroller.Pin]) – The pins attached to the columns.
- columns_to_anodes (bool) – Default True. If the matrix uses diodes, the diode anodes are typically connected to the column pins, and the cathodes should be connected to the row pins. If your diodes are reversed, set columns_to_anodes to False.
- interval (float) – Scan keys no more often than interval to allow for debouncing. interval is in float seconds. The default is 0.020 (20 msecs).
- max_events (int) – maximum size of events EventQueue: maximum number of key transition events that are saved. Must be >= 1. If a new event arrives when the queue is full, the oldest event is discarded.

key_count: int
The number of keys that are being scanned. (read-only)

events: EventQueue
    The EventQueue associated with this Keys object. (read-only)

deinit() → None
Stop scanning and release the pins.

__enter__() → KeyMatrix
No-op used by Context Managers.

__exit__() → None
Automatically deinitializes when exiting a context. See Lifetime and ContextManagers for more info.

reset() → None
Reset the internal state of the scanner to assume that all keys are now released. Any key that is already pressed at the time of this call will therefore immediately cause a new key-pressed event to occur.
key_number_to_row_column(key_number: int) → Tuple[int]

Return the row and column for the given key number. The row is key_number // len(column_pins).
The column is key_number % len(column_pins).

Returns
(row, column)

Return type
Tuple[int]

class keypad.Keys(pins: Sequence[microcontroller.Pin], *, value_when_pressed: bool, pull: bool = True, interval: float = 0.02, max_events: int = 64)

Manage a set of independent keys.

Create a Keys object that will scan keys attached to the given sequence of pins. Each key is independent and
attached to its own pin.

An EventQueue is created when this object is created and is available in the events attribute.

Parameters

- pins (Sequence[microcontroller.Pin]) – The pins attached to the keys. The key num-
  bers correspond to indices into this sequence.

- value_when_pressed (bool) – True if the pin reads high when the key is pressed. False
  if the pin reads low (is grounded) when the key is pressed. All the pins must be connected
  in the same way.

- pull (bool) – True if an internal pull-up or pull-down should be enabled on each pin. A
  pull-up will be used if value_when_pressed is False; a pull-down will be used if it is
  True. If an external pull is already provided for all the pins, you can set pull to False.
  However, enabling an internal pull when an external one is already present is not a problem;
  it simply uses slightly more current.

- interval (float) – Scan keys no more often than interval to allow for debouncing.
  interval is in float seconds. The default is 0.020 (20 msecs).

- max_events (int) – maximum size of events EventQueue: maximum number of key
  transition events that are saved. Must be >= 1. If a new event arrives when the queue is full,
  the oldest event is discarded.

key_count: int

The number of keys that are being scanned. (read-only)

events: EventQueue

The EventQueue associated with this Keys object. (read-only)

deinit() → None

Stop scanning and release the pins.

__enter__() → Keys

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes when exiting a context. See Lifetime and ContextManagers for more info.
reset() → None
Reset the internal state of the scanner to assume that all keys are now released. Any key that is already pressed at the time of this call will therefore immediately cause a new key-pressed event to occur.

class keypad.ShiftRegisterKeys(*, clock: microcontroller.Pin, data: microcontroller.Pin, latch: microcontroller.Pin, value_to_latch: bool = True, key_count: int, value_when_pressed: bool, interval: float = 0.02, max_events: int = 64)
Manage a set of keys attached to an incoming shift register.
Create a Keys object that will scan keys attached to a parallel-in serial-out shift register like the 74HC165 or CD4021. Note that you may chain shift registers to load in as many values as you need.
Key number 0 is the first (or more properly, the zero-th) bit read. In the 74HC165, this bit is labeled Q7. Key number 1 will be the value of Q6, etc.
An EventQueue is created when this object is created and is available in the events attribute.
Parameters
- **clock** (microcontroller.Pin) – The shift register clock pin. The shift register should clock on a low-to-high transition.
- **data** (microcontroller.Pin) – The incoming shift register data pin
- **latch** (microcontroller.Pin) – Pin used to latch parallel data going into the shift register.
- **value_to_latch** (bool) – Pin state to latch data being read. True if the data is latched when latch goes high False if the data is latched when latch goes low. The default is True, which is how the 74HC165 operates. The CD4021 latch is the opposite. Once the data is latched, it will be shifted out by toggling the clock pin.
- **key_count** (int) – number of data lines to clock in
- **value_when_pressed** (bool) – True if the pin reads high when the key is pressed. False if the pin reads low (is grounded) when the key is pressed.
- **interval** (float) – Scan keys no more often than interval to allow for debouncing. interval is in float seconds. The default is 0.020 (20 msecs).
- **max_events** (int) – maximum size of events EventQueue: maximum number of key transition events that are saved. Must be >= 1. If a new event arrives when the queue is full, the oldest event is discarded.

**key_count**: int
The number of keys that are being scanned. (read-only)

**events**: EventQueue
The EventQueue associated with this Keys object. (read-only)

definit() → None
Stop scanning and release the pins.

__enter__() → Keys
No-op used by Context Managers.

__exit__() → None
Automatically deinitializes when exiting a context. See Lifetime and ContextManagers for more info.

reset() → None
Reset the internal state of the scanner to assume that all keys are now released. Any key that is already pressed at the time of this call will therefore immediately cause a new key-pressed event to occur.
The `math` module provides some basic mathematical functions for working with floating-point numbers.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: math.

math.e: float
  base of the natural logarithm

math.pi: float
  the ratio of a circle’s circumference to its diameter

math.acos(x: float) → float
  Return the inverse cosine of x.

math.asin(x: float) → float
  Return the inverse sine of x.

math.atan(x: float) → float
  Return the inverse tangent of x.

math.atan2(y: float, x: float) → float
  Return the principal value of the inverse tangent of y/x.

math.ceil(x: float) → int
  Return an integer, being x rounded towards positive infinity.

math.copysign(x: float, y: float) → float
  Return x with the sign of y.

math.cos(x: float) → float
  Return the cosine of x.

math.degrees(x: float) → float
  Return radians x converted to degrees.

math.exp(x: float) → float
  Return the exponential of x.

math.fabs(x: float) → float
  Return the absolute value of x.

math.floor(x: float) → int
  Return an integer, being x rounded towards negative infinity.

math.fmod(x: float, y: float) → int
  Return the remainder of x/y.

math.frexp(x: float) → Tuple[int, int]
  Decomposes a floating-point number into its mantissa and exponent. The returned value is the tuple (m, e) such that x == m * 2**e exactly. If x == 0 then the function returns (0.0, 0), otherwise the relation 0.5 <= abs(m) < 1 holds.

math.isfinite(x: float) → bool
  Return True if x is finite.
math.isinf(x: float) → bool
   Return True if x is infinite.

math.isnan(x: float) → bool
   Return True if x is not-a-number

math.ldexp(x: float, exp: float) → float
   Return x * (2**exp).

math.log(x: float, base: float = e) → float
   Return the logarithm of x to the given base. If base is not specified, returns the natural logarithm (base e) of x

math.modf(x: float) → Tuple[float, float]
   Return a tuple of two floats, being the fractional and integral parts of x. Both return values have the same sign as x.

math.pow(x: float, y: float) → float
   Returns x to the power of y.

math.radians(x: float) → float
   Return degrees x converted to radians.

math.sin(x: float) → float
   Return the sine of x.

math.sqrt(x: float) → float
   Returns the square root of x.

math.tan(x: float) → float
   Return the tangent of x.

math.trunc(x: float) → int
   Return an integer, being x rounded towards 0.

math.expm1(x: float) → float
   Return exp(x) - 1.
   May not be available on some boards.

math.log2(x: float) → float
   Return the base-2 logarithm of x.
   May not be available on some boards.

math.log10(x: float) → float
   Return the base-10 logarithm of x.
   May not be available on some boards.

math.cosh(x: float) → float
   Return the hyperbolic cosine of x.
   May not be available on some boards.

math.sinh(x: float) → float
   Return the hyperbolic sine of x.
   May not be available on some boards.
math.\texttt{tanh}(x: \textit{float}) \to \textit{float}

Return the hyperbolic tangent of \(x\).

May not be available on some boards.

math.\texttt{acosh}(x: \textit{float}) \to \textit{float}

Return the inverse hyperbolic cosine of \(x\).

May not be available on some boards.

math.\texttt{asinh}(x: \textit{float}) \to \textit{float}

Return the inverse hyperbolic sine of \(x\).

May not be available on some boards.

math.\texttt{atanh}(x: \textit{float}) \to \textit{float}

Return the inverse hyperbolic tangent of \(x\).

May not be available on some boards.

math.\texttt{erf}(x: \textit{float}) \to \textit{float}

Return the error function of \(x\).

May not be available on some boards.

math.\texttt{erfc}(x: \textit{float}) \to \textit{float}

Return the complementary error function of \(x\).

May not be available on some boards.

math.\texttt{gamma}(x: \textit{float}) \to \textit{float}

Return the gamma function of \(x\).

May not be available on some boards.

math.\texttt{lgamma}(x: \textit{float}) \to \textit{float}

Return the natural logarithm of the gamma function of \(x\).

May not be available on some boards.

\texttt{mdns} – Multicast Domain Name Service

The \texttt{mdns} module provides basic support for multicast domain name services. Basic use provides hostname resolution under the .local TLD. This module also supports DNS Service Discovery that allows for discovering other hosts that provide a desired service.

\textbf{class} \texttt{mdns.RemoteService}

Encapsulates information about a remote service that was found during a search. This object may only be created by a \texttt{mdns.Server}. It has no user-visible constructor.

Cannot be instantiated directly. Use \texttt{mdns.Server.find}.

\begin{description}
\item[hostname]: \texttt{str}
  The hostname of the device (read-only).
\item[instance\_name]: \texttt{str}
  The human readable instance name for the service. (read-only)
\item[service\_type]: \texttt{str}
  The service type string such as \_http. (read-only)
\end{description}
protocol:  `str`
   The protocol string such as `_tcp`. (read-only)

port:  `int`
   Port number used for the service. (read-only)

ipv4_address:  `ipaddress.IPv4Address | None`
   IP v4 Address of the remote service. None if no A records are found.

__del__( )  →  None
   Deletes the RemoteService object.

class `mdns.Server`(network_interface: `wifi.Radio`)
   The MDNS Server responds to queries for this device’s information and allows for querying other devices.
   Constructs or returns the mdns.Server for the given network_interface. (CircuitPython may already be using it.)
   Only native interfaces are currently supported.

hostname:  `str`
   Hostname resolvable as `<hostname>.local` in addition to `circuitpython.local`. Make sure this is unique across all devices on the network. It defaults to `cpy-#####` where `#####` is the hex digits of the last three bytes of the mac address.

instance_name:  `str`
   Human readable name to describe the device.

definit( )  →  None
   Stops the server

find(service_type: `str`, protocol: `str`, *, timeout: `float` = 1)  →  Tuple[RemoteService]
   Find all locally available remote services with the given service type and protocol.
   This doesn’t allow for direct hostname lookup. To do that, use `socketpool.SocketPool.getaddrinfo()`.

   **Parameters**
   
   •  `service_type (str)` – The service type such as “._http”
   
   •  `protocol (str)` – The service protocol such as “._tcp”
     
   •  `timeout (float/int)` – Time to wait for responses

advertise_service(*, service_type: `str`, protocol: `str`, port: `int` )  →  None
   Respond to queries for the given service with the given port.
   `service_type` and `protocol` can only occur on one port. Any call after the first will update the entry’s port.
   If web workflow is active, the port it uses can’t also be used to advertise a service.

   **Parameters**
   
   •  `service_type (str)` – The service type such as “._http”
     
   •  `protocol (str)` – The service protocol such as “._tcp”
     
   •  `port (int)` – The port used by the service
memorymap – Raw memory map access

The `memorymap` module allows you to read and write memory addresses in the address space seen from the processor running CircuitPython. It is usually the physical address space.

class memorymap.AddressRange(*, start, length)

    Presents a range of addresses as a bytearray.

    The addresses may access memory or memory mapped peripherals.

    Some address ranges may be protected by CircuitPython to prevent errors. An exception will be raised when constructing an AddressRange for an invalid or protected address.

    Multiple AddressRanges may overlap. There is no “claiming” of addresses.

Example usage on ESP32-S2:

```python
import memorymap
rtc_slow_mem = memorymap.AddressRange(start=0x50000000, length=0x2000)
rtc_slow_mem[0:3] = b'\xcc\x10\x00'
```

Constructs an address range starting at `start` and ending at `start + length`. An exception will be raised if any of the addresses are invalid or protected.

```python
__bool__() → bool

__len__() → int

    Return the length. This is used by `(len)`

__getitem__(index: slice) → bytearray

__getitem__(index: int) → int

    Returns the value(s) at the given index.

    1, 2, 4 and 8 byte aligned reads will be done in one transaction. All others may use multiple transactions.

__setitem__(index: slice, value: circuitpython_typing.ReadableBuffer) → None

__setitem__(index: int, value: int) → None

    Set the value(s) at the given index.

    1, 2, 4 and 8 byte aligned writes will be done in one transaction. All others may use multiple transactions.
```

memorymonitor – Memory monitoring helpers

exception memorymonitor.AllocationError

    Bases: Exception

    Catchall exception for allocation related errors.

    Initialize self. See help(type(self)) for accurate signature.

class memorymonitor.AllocationAlarm(*, minimum_block_count: int = 1)

    Throw an exception when an allocation of `minimum_block_count` or more blocks occurs while active.

    Track allocations:
```python
import memorymonitor

aa = memorymonitor.AllocationAlarm(minimum_block_count=2)
x = 2
# Should not allocate any blocks.
with aa:
    x = 5

# Should throw an exception when allocating storage for the 20 bytes.
with aa:
    x = bytearray(20)

ignore(count: int) → AllocationAlarm
Sets the number of applicable allocations to ignore before raising the exception. Automatically set back to zero at context exit.
Use it within a with block:

# Will not alarm because the bytearray allocation will be ignored.
with aa.ignore(2):
    x = bytearray(20)

__enter__() → AllocationAlarm
Enables the alarm.

__exit__(...) → None
Automatically disables the allocation alarm when exiting a context. See Lifetime and ContextManagers for more info.

class memorymonitor.AllocationSize
Tracks the number of allocations in power of two buckets.

It will have 16 16-bit buckets to track allocation counts. It is total allocations meaning frees are ignored. Reallocated memory is counted twice, at allocation and when reallocated with the larger size.

The buckets are measured in terms of blocks which is the finest granularity of the heap. This means bucket 0 will count all allocations less than or equal to the number of bytes per block, typically 16. Bucket 2 will be less than or equal to 4 blocks. See bytes_per_block to convert blocks to bytes.

Multiple AllocationSizes can be used to track different code boundaries.

Track allocations:

```
__enter__() → AllocationSize
Clears counts and resumes tracking.

__exit__() → None
Automatically pauses allocation tracking when exiting a context. See *Lifetime and ContextManagers* for more info.

__len__() → int
Returns the number of allocation buckets.
This allows you to:

```python
mm = memorymonitor.AllocationSize()
print(len(mm))
```

__getitem__(index: int) → int | None
Returns the allocation count for the given bucket.
This allows you to:

```python
mm = memorymonitor.AllocationSize()
print(mm[0])
```

**microcontroller – Pin references and cpu functionality**

The `microcontroller` module defines the pins and other bare-metal hardware from the perspective of the microcontroller. See `board` for board-specific pin mappings.

**microcontroller.cpu:  Processor**

CPU information and control, such as `cpu.temperature` and `cpu.frequency` (clock frequency). This object is an instance of `microcontroller.Processor`.

**microcontroller.cpus:  Processor**

CPU information and control, such as `cpus[0].temperature` and `cpus[1].frequency` (clock frequency) on chips with more than 1 cpu. The index selects which cpu. This object is an instance of `microcontroller.Processor`.

**microcontroller.delay_us(delay: int) → None**

Dedicated delay method used for very short delays. **Do not** do long delays because this stops all other functions from completing. Think of this as an empty while loop that runs for the specified (delay) time. If you have other code or peripherals (e.g. audio recording) that require specific timing or processing while you are waiting, explore a different avenue such as using `time.sleep()`.

**microcontroller.disable_interrupts() → None**

Disable all interrupts. Be very careful, this can stall everything.

**microcontroller.enable_interrupts() → None**

Enable the interrupts that were enabled at the last disable.

**microcontroller.on_next_reset(run_mode: RunMode) → None**

Configure the run mode used the next time the microcontroller is reset but not powered down.

Parameters

**run_mode (RunMode)** – The next run mode
microcontroller.reset() → None
Reset the microcontroller. After reset, the microcontroller will enter the run mode last set by on_next_reset.

**Warning:** This may result in file system corruption when connected to a host computer. Be very careful when calling this! Make sure the device “Safely removed” on Windows or “ejected” on Mac OS X and Linux.

microcontroller.nvm: nvm.ByteArray | None
Available non-volatile memory. This object is the sole instance of nvm.ByteArray when available or None otherwise.

Type
nvm.ByteArray or None

microcontroller.watchdog: watchdog.WatchDogTimer | None
Available watchdog timer. This object is the sole instance of watchdog.WatchDogTimer when available or None otherwise.

class microcontroller.Pin

Identifies an IO pin on the microcontroller.

Identifies an IO pin on the microcontroller. They are fixed by the hardware so they cannot be constructed on demand. Instead, use board or microcontroller.pin to reference the desired pin.

__hash__() → int
Returns a hash for the Pin.

class microcontroller.Processor

Microcontroller CPU information and control

Usage:

```python
import microcontroller
print(microcontroller.cpu.frequency)
print(microcontroller.cpu.temperature)
```

Note that on chips with more than one cpu (such as the RP2040) microcontroller.cpu will return the value for CPU 0. To get values from other CPUs use microcontroller.cpus indexed by the number of the desired cpu. i.e.

```python
print(microcontroller.cpus[0].temperature)
print(microcontroller.cpus[1].frequency)
```

You cannot create an instance of microcontroller.Processor. Use microcontroller.cpu to access the sole instance available.

**frequency:** int
The CPU operating frequency in Hertz.

**Limitations:** On most boards, frequency is read-only. Setting the frequency is possible on RP2040 boards and some i.MX boards.

**Warning:** Overclocking likely voids your warranties and may reduce the lifetime of the chip.
Warning: Changing the frequency may cause issues with other subsystems, such as USB, PWM, and PIO. To minimize issues, set the CPU frequency before initializing other systems.

reset_reason: ResetReason
   The reason the microcontroller started up from reset state.

temperature: float | None
   The on-chip temperature, in Celsius, as a float. (read-only)
   Is None if the temperature is not available.

   Limitations: Not available on ESP32 or ESP32-S3. On small SAMD21 builds without external flash, the reported temperature has reduced accuracy and precision, to save code space.

uid: bytearray
   The unique id (aka serial number) of the chip as a bytearray. (read-only)

voltage: float | None
   The input voltage to the microcontroller, as a float. (read-only)
   Is None if the voltage is not available.

class microcontroller.ResetReason
   The reason the microcontroller was last reset

   POWER_ON: object
      The microcontroller was started from power off.

   BROWNOUT: object
      The microcontroller was reset due to too low a voltage.

   SOFTWARE: object
      The microcontroller was reset from software.

   DEEP_SLEEP_ALARM: object
      The microcontroller was reset for deep sleep and restarted by an alarm.

   RESET_PIN: object
      The microcontroller was reset by a signal on its reset pin. The pin might be connected to a reset button.

   WATCHDOG: object
      The microcontroller was reset by its watchdog timer.

   UNKNOWN: object
      The microcontroller restarted for an unknown reason.

   RESCUE_DEBUG: object
      The microcontroller was reset by the rescue debug port.

class microcontroller.RunMode
   run state of the microcontroller

   Enum-like class to define the run mode of the microcontroller and CircuitPython.

   NORMAL: RunMode
      Run CircuitPython as normal.
SAFE_MODE: RunMode
Run CircuitPython in safe mode. User code will not run and the file system will be writeable over USB.

UF2: RunMode
Run the uf2 bootloader.

BOOTLOADER: RunMode
Run the default bootloader.

msgpack – Pack object in msgpack format

The msgpack format is similar to json, except that the encoded data is binary. See https://msgpack.org for details. The module implements a subset of the cpython module msgpack-python.

Not implemented: 64-bit int, uint, float.

For more information about working with msgpack, see the CPython Library Documentation.

Example 1:

```python
import msgpack
from io import BytesIO

b = BytesIO()
msgpack.pack({'list': [True, False, None, 1, 3.14], 'str': 'blah'}, b)
b.seek(0)
print(msgpack.unpack(b))
```

Example 2: handling objects:

```python
from msgpack import pack, unpack, ExtType
from io import BytesIO

class MyClass:
    def __init__(self, val):
        self.value = val
    def __str__(self):
        return str(self.value)
data = MyClass(b'my_value')

def encoder(obj):
    if isinstance(obj, MyClass):
        return ExtType(1, obj.value)
    return f"no encoder for {obj}"

def decoder(code, data):
    if code == 1:
        return MyClass(data)
    return f"no decoder for type {code}"```

(continues on next page)
buffer = BytesIO()
pack(data, buffer, default=encoder)
buffer.seek(0)
decoded = unpack(buffer, ext_hook=decoder)
print(f"{data} -> {buffer.getvalue()} -> {decoded}")

msgpack.pack(obj: object, stream: circuitpython_typing.ByteStream, *, default: Callable[[object], None] | None = None) → None

Output object to stream in msgpack format.

Parameters

- **obj (object)** – Object to convert to msgpack format.
- **stream (ByteStream)** – stream to write to
- **default (Optional[Callable[[object], None]]):** function called for python objects that do not have a representation in msgpack format.

msgpack.unpack(stream: circuitpython_typing.ByteStream, *, ext_hook: Callable[[int, bytes], object] | None = None, use_list: bool = True) → object

Unpack and return one object from stream.

Parameters

- **stream (ByteStream)** – stream to read from
- **ext_hook (Optional[Callable[[int, bytes], object]]):** function called for objects in msgpack ext format.
- **use_list (Optional[bool]):** return array as list or tuple (use_list=False).

Return object

object read from stream.

class msgpack.ExtType(code: int, data: bytes)

ExtType represents ext type in msgpack.

Constructor

:param int code: type code in range 0~127. :param bytes data: representation.

code: int

The type code, in range 0~127.

data: bytes

Data.

neopixel_write – Low-level neopixel implementation

The **neopixel_write** module contains a helper method to write out bytes in the 800khz neopixel protocol.

For example, to turn off a single neopixel (like the status pixel on Express boards.)

```python
import board
import neopixel_write
import digitalio
```
pin = digitalio.DigitalInOut(board.NEOPIXEL)
pin.direction = digitalio.Direction.OUTPUT
pixel_off = bytearray([0, 0, 0])
neopixel_write.neopixel_write(pin, pixel_off)

Note: This module is typically not used by user level code.
For more information on actually using NeoPixels, refer to the CircuitPython Essentials Learn guide
For a much more thorough guide about using NeoPixels, refer to the Adafruit NeoPixel Überguide.

neopixel_write.neopixel_write(digitalinout: digitalio.DigitalInOut, buf:
circuitpython_typing.ReadableBuffer) → None

Write buf out on the given DigitalInOut.

Parameters

- `digitalinout` (DigitalInOut) – the DigitalInOut to output with
- `buf` (ReadableBuffer) – The bytes to clock out. No assumption is made about color order

**nvm — Non-volatile memory**

The `nvm` module allows you to store whatever raw bytes you wish in a reserved section non-volatile memory.
Note that this module can't be imported and used directly. The sole instance of `ByteArray` is available at `microcontroller.nvm`.

**class nvm.ByteArray**

Presents a stretch of non-volatile memory as a bytearray.

Non-volatile memory is available as a byte array that persists over reloads and power cycles. Each assignment causes an erase and write cycle so its recommended to assign all values to change at once.

Usage:

```py
import microcontroller
microcontroller.nvm[0:3] = b"\xcc\x10\x00"
```

Not currently dynamically supported. Access the sole instance through `microcontroller.nvm`.

- `__bool__()` → bool
- `__len__()` → int
  Return the length. This is used by (`len`)
- `__getitem__`(index: slice) → bytearray
- `__getitem__`(index: int) → int
  Returns the value at the given index.
- `__setitem__`(index: slice, value: circuitpython_typing.ReadableBuffer) → None
- `__setitem__`(index: int, value: int) → None
  Set the value at the given index.
onewireio – Low-level bit primitives for Maxim (formerly Dallas Semi) one-wire protocol.

Protocol definition is here: https://www.analog.com/en/technical-articles/1wire-communication-through-software.html

class onewireio.OneWire(pin: microcontroller.Pin)

Create a OneWire object associated with the given pin.

The object implements the lowest level timing-sensitive bits of the protocol.

Parameters

- **pin** (Pin) – Pin connected to the OneWire bus

Read a short series of pulses:

```python
import onewireio
import board

onewire = onewireio.OneWire(board.D7)
onewire.reset()
onewire.write_bit(True)
onewire.write_bit(False)
print(onewire.read_bit())
```

definit() → None

Deinitialize the OneWire bus and release any hardware resources for reuse.

__enter__() → OneWire

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See Lifetime and ContextManagers for more info.

reset() → bool

Reset the OneWire bus and read presence

Returns

- **False** when at least one device is present

Return type

- bool

read_bit() → bool

Read in a bit

Returns

- bit state read

Return type

- bool

write_bit(value: bool) → None

Write out a bit based on value.
os – functions that an OS normally provides

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: os.

os.uname() → _Uname
   Returns a named tuple of operating specific and CircuitPython port specific information.

class os._Uname
   Bases: NamedTuple
   The type of values that uname() returns
   sysname: str
   nodename: str
   release: str
   version: str
   machine: str

os.chdir(path: str) → None
   Change current directory.

os.getcwd() → str
   Get the current directory.

os.getenv(key: str, default: str | None = None) → str | None
   Get the environment variable value for the given key or return default.
   This may load values from disk so cache the result instead of calling this often.
   On boards that do not support settings.toml reading in the core, this function will raise Not Implemented Error.

os.listdir(dir: str) → str
   With no argument, list the current directory. Otherwise list the given directory.

os.mkdir(path: str) → None
   Create a new directory.

os.remove(path: str) → None
   Remove a file.

os.rmdir(path: str) → None
   Remove a directory.

os.rename(old_path: str, new_path: str) → str
   Rename a file.

os.stat(path: str) → Tuple[int, int, int, int, int, int, int, int, int, int]
   Get the status of a file or directory.
   Returns a tuple with the status of a file or directory in the following order:
   • st_mode – File type, regular or directory
   • st_ino – Set to 0
   • st_dev – Set to 0
• st_nlink – Set to 0
• st_uid – Set to 0
• st_gid – Set to 0
• st_size – Size of the file in bytes
• st_atime – Time of most recent access expressed in seconds
• st_mtime – Time of most recent content modification expressed in seconds.
• st_ctime – Time of most recent content modification expressed in seconds.

Note: On builds without long integers, the number of seconds for contemporary dates will not fit in a small integer. So the time fields return 946684800, which is the number of seconds corresponding to 1999-12-31.

os.statvfs(path: str) → Tuple[int, int, int, int, int, int, int, int, int, int]
Get the status of a filesystem.
Returns a tuple with the filesystem information in the following order:
• f_bsize – file system block size
• f_frsize – fragment size
• f_blocks – size of fs in f_frsize units
• f_bfree – number of free blocks
• f_bavail – number of free blocks for unprivileged users
• f_files – number of inodes
• f_ffree – number of free inodes
• f_favail – number of free inodes for unprivileged users
• f_flag – mount flags
• f_namemax – maximum filename length
Parameters related to inodes: f_files, f_ffree, f_favail and the f_flag parameter may return 0 as they can be unavailable in a port-specific implementation.

os.sync() → None
Sync all filesystems.

os.urandom(size: int) → str
Returns a string of size random bytes based on a hardware True Random Number Generator. When not available, it will raise a NotImplementedError.

Limitations: Not available on SAMD21 due to lack of hardware.

os.utime(path: str, times: Tuple[int, int]) → None
Change the timestamp of a file.

os.sep: str
Separator used to delineate path components such as folder and file names.
CircuitPython Documentation, Release 8.2.9

paralleldisplay – Native helpers for driving parallel displays

class paralleldisplay.ParallelBus(*, data0: microcontroller.Pin, command: microcontroller.Pin,
chip_select: microcontroller.Pin, write: microcontroller.Pin, read: microcontroller.Pin | None, reset: microcontroller.Pin | None = None,
frequency: int = 30000000)

Manage updating a display over 8-bit parallel bus in the background while Python code runs. This protocol may be referred to as 8080-I Series Parallel Interface in datasheets. It doesn’t handle display initialization.

Create a ParallelBus object associated with the given pins. The bus is inferred from data0 by implying the next 7 additional pins on a given GPIO port.

The parallel bus and pins are then in use by the display until displayio.release_displays() is called even after a reload. (It does this so CircuitPython can use the display after your code is done.) So, the first time you initialize a display bus in code.py you should call displayio.release_displays() first, otherwise it will error after the first code.py run.

Parameters

- **data_pins** (microcontroller.Pin) – A list of data pins. Specify exactly one of data_pins or data0.
- **data0** (microcontroller.Pin) – The first data pin. The rest are implied
- **command** (microcontroller.Pin) – Data or command pin
- **chip_select** (microcontroller.Pin) – Chip select pin
- **write** (microcontroller.Pin) – Write pin
- **read** (microcontroller.Pin) – Read pin, optional
- **reset** (microcontroller.Pin) – Reset pin, optional
- **frequency** (int) – The communication frequency in Hz for the display on the bus

**reset()** → None

Performs a hardware reset via the reset pin. Raises an exception if called when no reset pin is available.

**send(command: int, data: circuitpython_typing.ReadableBuffer) → None**

Sends the given command value followed by the full set of data. Display state, such as vertical scroll, set via send may or may not be reset once the code is done.

picodvi – Low-level routines for interacting with PicoDVI Output

class picodvi.Framebuffer(width: int, height: int, *, clk_dp: microcontroller.Pin, clk_dn: microcontroller.Pin,

Create a Framebuffer object with the given dimensions. Memory is allocated outside of onto the heap and then moved outside on VM end.

**Warning:** This will change the system clock speed to match the DVI signal. Make sure to initialize other objects after this one so they account for the changed clock.

This allocates a very large framebuffer and is most likely to succeed the earlier it is attempted.
Each dp and dn pair of pins must be neighboring, such as 19 and 20. They must also be ordered the same way. In other words, dp must be less than dn for all pairs or dp must be greater than dn for all pairs.

The framebuffer pixel format varies depending on color_depth:

- 1 - Each bit is a pixel. Either white (1) or black (0).
- 2 - Each 2 bits is a pixels. Grayscale between white (0x3) and black (0x0).
- 8 - Each byte is a pixels in RGB332 format.
- 16 - Each two bytes are a pixel in RGB565 format.

Two output resolutions are currently supported, 640x480 and 800x480. Monochrome framebuffers (color_depth=1 or 2) must be full resolution. Color framebuffers must be half resolution (320x240 or 400x240) and pixels will be duplicated to create the signal.

A Framebuffer is often used in conjunction with a `framebufferio.FramebufferDisplay`.

Parameters

- `width (int)` – the width of the target display signal. Only 320, 400, 640 or 800 is currently supported depending on color_depth.
- `height (int)` – the height of the target display signal. Only 240 or 480 is currently supported depending on color_depth.
- `clk_dp (Pin)` – the positive clock signal pin
- `clk_dn (Pin)` – the negative clock signal pin
- `red_dp (Pin)` – the positive red signal pin
- `red_dn (Pin)` – the negative red signal pin
- `green_dp (Pin)` – the positive green signal pin
- `green_dn (Pin)` – the negative green signal pin
- `blue_dp (Pin)` – the positive blue signal pin
- `blue_dn (Pin)` – the negative blue signal pin
- `color_depth (int)` – the color depth of the framebuffer in bits. 1, 2 for grayscale and 8 or 16 for color

width: int

The width of the framebuffer, in pixels. It may be doubled for output.

height: int

The height of the framebuffer, in pixels. It may be doubled for output.

deinit() → None

Free the resources (pins, timers, etc.) associated with this `picodvi.Framebuffer` instance. After deinitialization, no further operations may be performed.
**ps2io – Support for PS/2 protocol**

The `ps2io` module contains classes to provide PS/2 communication.

**Warning:** This module is not available in some SAMD21 builds. See the [Module Support Matrix - Which Modules Are Available on Which Boards](https://circuitpython.readthedocs.io/en/latest/_modules/ps2io.html) for more info.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See *Lifetime and ContextManagers* for more info.

```python
class ps2io.Ps2(data_pin: microcontroller.Pin, clock_pin: microcontroller.Pin)

Communicate with a PS/2 keyboard or mouse

Ps2 implements the PS/2 keyboard/mouse serial protocol, used in legacy devices. It is similar to UART but there are only two lines (Data and Clock). PS/2 devices are 5V, so bidirectional level converters must be used to connect the I/O lines to pins of 3.3V boards.

Create a Ps2 object associated with the given pins.

**Parameters**

- `data_pin` ([Pin](https://circuitpython.readthedocs.io/en/latest/_modules/ps2io.html#Ps2)) – Pin tied to data wire.
- `clock_pin` ([Pin](https://circuitpython.readthedocs.io/en/latest/_modules/ps2io.html#Ps2)) – Pin tied to clock wire. This pin must support interrupts.

Read one byte from PS/2 keyboard and turn on Scroll Lock LED:

```python
import ps2io
import board

kbd = ps2io.Ps2(board.D10, board.D11)
while len(kbd) == 0:
    pass

print(kbd.popleft())
print(kbd.sendcmd(0xed))
print(kbd.sendcmd(0x01))
```

**deinit()** → None

Deinitialises the Ps2 and releases any hardware resources for reuse.

**__enter__**() → Ps2

No-op used by Context Managers.

**__exit__**() → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**popleft()** → int

Removes and returns the oldest received byte. When buffer is empty, raises an IndexError exception.

**sendcmd**(byte: `int`) → `int`

Sends a command byte to PS/2. Returns the response byte, typically the general ack value (0xFA). Some commands return additional data which is available through `popleft()`.

Raises a RuntimeError in case of failure. The root cause can be found by calling `clear_errors()`. It is advisable to call `clear_errors()` before `sendcmd()` to flush any previous errors.
Parameters

**byte (int)** – byte value of the command

**clear_errors() → None**

Returns and clears a bitmap with latest recorded communication errors.

Reception errors (arise asynchronously, as data is received):
0x01: start bit not 0
0x02: timeout
0x04: parity bit error
0x08: stop bit not 1
0x10: buffer overflow, newest data discarded

Transmission errors (can only arise in the course of sendcmd()):
0x100: clock pin didn’t go to LO in time
0x200: clock pin didn’t go to HI in time
0x400: data pin didn’t ACK
0x800: clock pin didn’t ACK
0x1000: device didn’t respond to RTS
0x2000: device didn’t send a response byte in time

**__bool__ () → bool**

**__len__ () → int**

Returns the number of received bytes in buffer, available to `popleft()`.

**pulseio – Support for individual pulse based protocols**

The **pulseio** module contains classes to provide access to basic pulse IO. Individual pulses are commonly used in infrared remotes and in DHT temperature sensors.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See **Lifetime and ContextManagers** for more info.


Measure a series of active and idle pulses. This is commonly used in infrared receivers and low cost temperature sensors (DHT). The pulsed signal consists of timed active and idle periods. Unlike PWM, there is no set duration for active and idle pairs.

Create a PulseIn object associated with the given pin. The object acts as a read-only sequence of pulse lengths with a given max length. When it is active, new pulse lengths are added to the end of the list. When there is no more room (`len() == maxlen`) the oldest pulse length is removed to make room.

**Parameters**

- **pin (Pin)** – Pin to read pulses from.
- **maxlen (int)** – Maximum number of pulse durations to store at once
- **idle_state (bool)** – Idle state of the pin. At start and after `resume` the first recorded pulse will the opposite state from idle.

Read a short series of pulses:
```python
import pulseio
import board

pulses = pulseio.PulseIn(board.D7)

# Wait for an active pulse
while len(pulses) == 0:
    pass

# Pause while we do something with the pulses
pulses.pause()

# Print the pulses. pulses[0] is an active pulse unless the length
# reached max length and idle pulses are recorded.
print(pulses)

# Clear the rest
pulses.clear()

# Resume with an 80 microsecond active pulse
pulses.resume(80)
```

### maxlen

**int**

The maximum length of the PulseIn. When len() is equal to maxlen, it is unclear which pulses are active and which are idle.

### paused

**bool**

True when pulse capture is paused as a result of `pause()` or an error during capture such as a signal that is too fast.

### deinit

**None**

Deinitialises the PulseIn and releases any hardware resources for reuse.

### __enter__

**PulseIn**

No-op used by Context Managers.

### __exit__

**None**

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

### pause

**None**

Pause pulse capture

### resume

**None**

Resumes pulse capture after an optional trigger pulse.

**Warning:** Using trigger pulse with a device that drives both high and low signals risks a short. Make sure your device is open drain (only drives low) when using a trigger pulse. You most likely added a “pull-up” resistor to your circuit to do this.

**Parameters**

- `trigger_duration (int)` – trigger pulse duration in microseconds
clear() → None
Clears all captured pulses

popleft() → int
Removes and returns the oldest read pulse.

__bool__() → bool

__len__() → int
Returns the number of pulse durations currently stored.

This allows you to:

```python
pulses = pulseio.PulseIn(pin)
print(len(pulses))
```

__getitem__(index: int) → int | None
Returns the value at the given index or values in slice.

This allows you to:

```python
pulses = pulseio.PulseIn(pin)
print(pulses[0])
```

class pulseio.PulseOut(pin: microcontroller.Pin, *, frequency: int = 38000, duty_cycle: int = 1 << 15)

Pulse PWM “carrier” output on and off. This is commonly used in infrared remotes. The pulsed signal consists of timed on and off periods. Unlike PWM, there is no set duration for on and off pairs.

Create a PulseOut object associated with the given pin.

Parameters

- **pin** (Pin) – Signal output pin
- **frequency** (int) – Carrier signal frequency in Hertz
- **duty_cycle** (int) – 16-bit duty cycle of carrier frequency (0 - 65536)

Send a short series of pulses:

```python
import array
import pulseio
import pwmio
import board

# 50% duty cycle at 38kHz.
pulse = pulseio.PulseOut(board.LED, frequency=38000, duty_cycle=32768)
# on off on off on
pulses = array.array('H', [65000, 1000, 65000, 65000, 1000])
pulse.send(pulses)

# Modify the array of pulses.
pulses[0] = 200
pulse.send(pulses)
```

deinit() → None
Deinitialises the PulseOut and releases any hardware resources for reuse.
__enter__() → PulseOut

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See Lifetime and Context Managers for more info.

send(pulses: circuitpython_typing.ReadableBuffer) → None

Pulse alternating on and off durations in microseconds starting with on. pulses must be an array.array with data type ‘H’ for unsigned halfword (two bytes).

This method waits until the whole array of pulses has been sent and ensures the signal is off afterwards.

Parameters

pulses (array.array) – pulse durations in microseconds

pwmio – Support for PWM based protocols

The pwmio module contains classes to provide access to basic pulse IO.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call deinit() or use a context manager. See Lifetime and Context Managers for more info.

For example:

```python
import time
import pwmio
import board

pwm = pwmio.PWMOut(board.LED)
pwm.duty_cycle = 2 ** 15
time.sleep(0.1)
```

This example will initialize the the device, set duty_cycle, and then sleep 0.1 seconds. CircuitPython will automatically turn off the PWM when it resets all hardware after program completion. Use deinit() or a with statement to do it yourself.

For the essentials of pwmio, see the CircuitPython Essentials Learn guide.

class pwmio.PWMOut(pin: microcontroller.Pin, *, duty_cycle: int = 0, frequency: int = 500, variable_frequency: bool = False)

Output a Pulse Width Modulated signal on a given pin.

Note: The exact frequencies possible depend on the specific microcontroller. If the requested frequency is within the available range, one of the two nearest possible frequencies to the requested one is selected.

If the requested frequency is outside the range, either (A) a ValueError may be raised or (B) the highest or lowest frequency is selected. This behavior is microcontroller-dependent, and may depend on whether it’s the upper or lower bound that is exceeded.

In any case, the actual frequency (rounded to 1Hz) is available in the frequency property after construction.

Note: The frequency is calculated based on a nominal CPU frequency. However, depending on the board, the error between the nominal and actual CPU frequency can be large (several hundred PPM in the case of crystal
Create a PWM object associated with the given pin. This allows you to write PWM signals out on the given pin. Frequency is fixed after init unless variable_frequency is True.

**Note:** When variable_frequency is True, further PWM outputs may be limited because it may take more internal resources to be flexible. So, when outputting both fixed and flexible frequency signals construct the fixed outputs first.

**Parameters**
- **pin (Pin)** – The pin to output to
- **duty_cycle (int)** – The fraction of each pulse which is high. 16-bit
- **frequency (int)** – The target frequency in Hertz (32-bit)
- **variable_frequency (bool)** – True if the frequency will change over time

Simple LED on:

```python
import pwmio
import board

pwm = pwmio.PWMOut(board.LED)

while True:
    pwm.duty_cycle = 2 ** 15 # Cycles the pin with 50% duty cycle (half of 2 ** 16) at the default 500hz
```

PWM LED fade:

```python
import pwmio
import board

pwm = pwmio.PWMOut(board.LED) # output on LED pin with default of 500Hz

while True:
    for cycle in range(0, 65535): # Cycles through the full PWM range from 0 to 65535
        pwm.duty_cycle = cycle # Cycles the LED pin duty cycle through the range of values
    for cycle in range(65534, 0, -1): # Cycles through the PWM range backwards from 65534 to 0
        pwm.duty_cycle = cycle # Cycles the LED pin duty cycle through the range of values
```

PWM at specific frequency (servos and motors):

```python
import pwmio
import board

pwm = pwmio.PWMOut(board.D13, frequency=50)
```

(continues on next page)
Variable frequency (usually tones):

```python
import pwmio
import board
import time

pwm = pwmio.PWMOut(board.D13, duty_cycle=2 ** 15, frequency=440, variable_frequency=True)
time.sleep(0.2)
pwm.frequency = 880
time.sleep(0.1)
```

duty_cycle: int

16 bit value that dictates how much of one cycle is high (1) versus low (0). 0xffff will always be high, 0 will always be low and 0x7fff will be half high and then half low.

Depending on how PWM is implemented on a specific board, the internal representation for duty cycle might have less than 16 bits of resolution. Reading this property will return the value from the internal representation, so it may differ from the value set.

frequency: int

32 bit value that dictates the PWM frequency in Hertz (cycles per second). Only writeable when constructed with variable_frequency=True.

Depending on how PWM is implemented on a specific board, the internal value for the PWM’s duty cycle may need to be recalculated when the frequency changes. In these cases, the duty cycle is automatically recalculated from the original duty cycle value. This should happen without any need to manually re-set the duty cycle. However, an output glitch may occur during the adjustment.

deinit() → None

Deinitialises the PWMOut and releases any hardware resources for reuse.

__enter__() → PWMOut

No-op used by Context Managers.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**qrio – Low-level QR code decoding**

Provides the *QRDecoder* object used for decoding QR codes. For more information about working with QR codes, see this Learn guide.

**Note:** This module only handles decoding QR codes. If you are looking to generate a QR code, use the *adafruit_miniiqr* library.
EVERY_BYTE: `PixelPolicy`

The input buffer to `QRDecoder.decode` consists of greyscale values in every byte.

EVEN_BYTES: `PixelPolicy`

The input buffer to `QRDecoder.decode` consists of greyscale values in positions 0, 2, ..., and ignored bytes in positions 1, 3, .... This can decode directly from YUV images where the even bytes hold the Y (luminance) data.

ODD_BYTES: `PixelPolicy`

The input buffer to `QRDecoder.decode` consists of greyscale values in positions 1, 3, ..., and ignored bytes in positions 0, 2, .... This can decode directly from YUV images where the odd bytes hold the Y (luminance) data.

RGB565_SWAPPED: `PixelPolicy`

The input buffer to `QRDecoder.decode` consists of RGB565 values in byte-swapped order. Most cameras produce data in byte-swapped order. The green component is used.

RGB565: `PixelPolicy`

The input buffer to `QRDecoder.decode` consists of RGB565 values in native order. The green component is used.

class qrio.QRDecoder(width: int, height: int)

Construct a QRDecoder object.

Parameters

- **width** (`int`) – The pixel width of the image to decode
- **height** (`int`) – The pixel height of the image to decode

width: `int`

The width of image the decoder expects

height: `int`

The height of image the decoder expects

`decode(buffer: circuitpython_typing.ReadableBuffer, pixel_policy: PixelPolicy = PixelPolicy.EVERY_BYTE) -> List[QRInfo]`

Decode zero or more QR codes from the given image. The size of the buffer must be at least `length``×``width` bytes for `EVERY_BYTE`, and `2×length``×``width` bytes for `EVEN_BYTES` or `ODD_BYTES`.

class qrio.QRInfo

Information about a decoded QR code

payload: `bytes`

The content of the QR code

data_type: `str | int`

The encoding of the payload as a string (if a standard encoding) or int (if not standard)
rainbowio

rainbowio module.

Provides the `colorwheel()` function.

```python
rainbowio.colorwheel(n: float) → int
```

C implementation of the common colorwheel() function found in many examples. Returns the colorwheel RGB value as an integer value for n (usable in neopixel and dotstar).

**random – pseudo-random numbers and choices**

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: `random`.

Like its CPython cousin, CircuitPython’s random seeds itself on first use with a true random from os.urandom() when available or the uptime otherwise. Once seeded, it will be deterministic, which is why its bad for cryptography.

---

**Warning:** Numbers from this module are not cryptographically strong! Use bytes from `os.urandom` directly for true randomness.

---

random._T

```python
random.seed(seed: int) → None
```

Sets the starting seed of the random number generation. Further calls to `random` will return deterministic results afterwards.

```python
random.getrandbits(k: int) → int
```

Returns an integer with k random bits.

```python
random.randrange(stop: int) → int
```

```python
random.randrange(start: int, stop: int) → int
```

```python
random.randrange(start: int, stop: int, step: int) → int
```

Returns a randomly selected integer from `range(start[, stop[, step]])`.

```python
random.randint(a: int, b: int) → int
```

Returns a randomly selected integer between a and b inclusive. Equivalent to `randrange(a, b + 1, 1)`

```python
random.choice(seq: Sequence[_T]) → _T
```

Returns a randomly selected element from the given sequence. Raises IndexError when the sequence is empty.

```python
random.random() → float
```

Returns a random float between 0 and 1.0.

```python
random.uniform(a: float, b: float) → float
```

Returns a random float between a and b. It may or may not be inclusive depending on float rounding.
entimes text: 72x748

CircuitPython Documentation, Release 8.2.9

**rgbmatrix – Low-level routines for bitbanged LED matrices**

For more information about working with RGB matrix panels in CircuitPython, see the dedicated learn guide.

```python
```

Displays an in-memory framebuffer to a HUB75-style RGB LED matrix.

Create a RGBMatrix object with the given attributes. The height of the display is determined by the number of rgb and address pins and the number of tiles: \[ \text{len(rgb_pins)} \div 3 \times 2^{\text{len(addr_pins)}} \times \text{abs(tile)} \]. With 6 RGB pins, 4 address lines, and a single matrix, the display will be 32 pixels tall. If the optional height parameter is specified and is not 0, it is checked against the calculated height.

Tiled matrices, those with more than one panel, must be laid out in a specific order, as detailed in the guide.

At least 6 RGB pins and 5 address pins are supported, for common panels with up to 64 rows of pixels. Some microcontrollers may support more, up to a soft limit of 30 RGB pins and 8 address pins.

The RGB pins must be within a single “port” and performance and memory usage are best when they are all within “close by” bits of the port. The clock pin must also be on the same port as the RGB pins. See the documentation of the underlying protomatter C library for more information. Generally, Adafruit’s interface boards are designed so that these requirements are met when matched with the intended microcontroller board. For instance, the Feather M4 Express works together with the RGB Matrix Feather.

The framebuffer is in “RGB565” format.

“RGB565” means that it is organized as a series of 16-bit numbers where the highest 5 bits are interpreted as red, the next 6 as green, and the final 5 as blue. The object can be any buffer, but array.array and ulab.ndarray objects are most often useful. To update the content, modify the framebuffer and call refresh.

If a framebuffer is not passed in, one is allocated and initialized to all black. In any case, the framebuffer can be retrieved by passing the RGBMatrix object to memoryview().

If doublebuffer is False, some memory is saved, but the display may flicker during updates.

A RGBMatrix is often used in conjunction with a framebufferio.FramebufferDisplay.

On boards designed for use with RGBMatrix panels, board.MTX_ADDRESS is a tuple of all the address pins, and board.MTX_COMMON is a dictionary with rgb_pins, clock_pin, latch_pin, and output_enable_pin. For panels that use fewer than the maximum number of address pins, “slice” MTX_ADDRESS to get the correct number of address pins. Using these board properties makes calling the constructor simpler and more portable:

```python
matrix = rgbmatrix.RGBMatrix(..., addr_pins=board.MTX_ADDRESS[:4], **board.MTX_˓→COMMON)
```

**Parameters**

- **width (int)** – The overall width of the whole matrix in pixels. For a matrix with multiple panels in row, this is the width of a single panel times the number of panels across.
- **tile (int)** – In a multi-row matrix, the number of rows of panels
- **bit_depth (int)** – The color depth of the matrix. A value of 1 gives 8 colors, a value of 2 gives 64 colors, and so on. Increasing bit depth increases the CPU and RAM usage of the RGBMatrix, and may lower the panel refresh rate. The framebuffer is always in RGB565 format regardless of the bit depth setting.
• **serpentine** (*bool*) – In a multi-row matrix, True when alternate rows of panels are rotated 180°, which can reduce wiring length

• **rgb_pins** (*Sequence[digitalio.DigitalInOut]*) – The matrix’s RGB pins in the order (R1, G1, B1, R2, G2, B2…)

• **addr_pins** (*Sequence[digitalio.DigitalInOut]*) – The matrix’s address pins in the order (A, B, C, D…)

• **clock_pin** (*digitalio.DigitalInOut*) – The matrix’s clock pin

• **latch_pin** (*digitalio.DigitalInOut*) – The matrix’s latch pin

• **output_enable_pin** (*digitalio.DigitalInOut*) – The matrix’s output enable pin

• **doublebuffer** (*bool*) – True if the output is double-buffered

• **framebuffer** (*Optional[WriteableBuffer]*) – A pre-allocated framebuffer to use. If unspecified, a framebuffer is allocated

• **height** (*int*) – The optional overall height of the whole matrix in pixels. This value is not required because it can be calculated as described above.

**brightness** : *float*

In the current implementation, 0.0 turns the display off entirely and any other value up to 1.0 turns the display on fully.

**width** : *int*

The width of the display, in pixels

**height** : *int*

The height of the display, in pixels

**deinit** () → None

Free the resources (pins, timers, etc.) associated with this rgbmatrix instance. After deinitialization, no further operations may be performed.

**refresh** () → None

Transmits the color data in the buffer to the pixels so that they are shown.

### rotaryio – Support for reading rotation sensors

The *rotaryio* module contains classes to read different rotation encoding schemes. See [Wikipedia’s Rotary Encoder page](https://en.wikipedia.org/wiki/Rotary_encoder) for more background.

For more information on working with rotary encoders using this library, see [this Learn Guide](https://learn.adafruit.com/).

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call `deinit()` or use a context manager. See [Lifetime and ContextManagers](https://learn.adafruit.com/lifetime-and-context-managers) for more info.


IncrementalEncoder determines the relative rotational position based on two series of pulses.

Create an IncrementalEncoder object associated with the given pins. It tracks the positional state of an incremental rotary encoder (also known as a quadrature encoder.) Position is relative to the position when the object is constructed.

**Parameters**

• **pin_a** *(Pin)* – First pin to read pulses from.
• **pin_b (Pin)** – Second pin to read pulses from.
• **divisor (int)** – The divisor of the quadrature signal.

For example:

```python
definit() → None
Deinitializes the IncrementalEncoder and releases any hardware resources for reuse.

__enter__() → IncrementalEncoder
No-op used by Context Managers.

__exit__() → None
Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.
```

rp2pio – Hardware interface to RP2 series’ programmable IO (PIO) peripheral.

**Note:** This module is intended to be used with the *adafruit_pioasm library*. For an introduction and guide to working with PIO in CircuitPython, see this *Learn guide*.

```python
rp2pio.pins_are_sequential(pins: List[microcontroller.Pin]) → bool
Return True if the pins have sequential GPIO numbers, False otherwise
```
class rp2pioStateMachine:

A single PIO StateMachine

The programmable I/O peripheral on the RP2 series of microcontrollers is unique. It is a collection of generic state machines that can be used for a variety of protocols. State machines may be independent or coordinated. Program memory and IRQs are shared between the state machines in a particular PIO instance. They are independent otherwise.

This class is designed to facilitate sharing of PIO resources. By default, it is assumed that the state machine is used on its own and can be placed in either PIO. State machines with the same program will be placed in the same PIO if possible.

Construct a StateMachine object on the given pins with the given program.

Parameters

- **program (ReadableBuffer)** – the program to run with the state machine
- **frequency (int)** – the target clock frequency of the state machine. Actual may be less. Use 0 for system clock speed.
- **init (ReadableBuffer)** – a program to run once at start up. This is run after program is started so instructions may be intermingled
- **first_out_pin (Pin)** – the first pin to use with the OUT instruction
- **out_pin_count (int)** – the count of consecutive pins to use with OUT starting at first_out_pin
- **initial_out_pin_state (int)** – the initial output value for out pins starting at first_out_pin
- **initial_out_pin_direction (int)** – the initial output direction for out pins starting at first_out_pin
- **first_in_pin (Pin)** – the first pin to use with the IN instruction
- **in_pin_count (int)** – the count of consecutive pins to use with IN starting at first_in_pin
- **pull_in_pin_up (int)** – a 1-bit in this mask sets pull up on the corresponding pin
- **pull_in_pin_down (int)** – a 1-bit in this mask sets pull down on the corresponding pin. Setting both pulls enables a “bus keep” function, i.e. a weak pull to whatever is current high/low state of GPIO.
- **first_set_pin (Pin)** – the first pin to use with the SET instruction
- **set_pin_count (int)** – the count of consecutive pins to use with SET starting at first_set_pin
- **initial_set_pin_state (int)** – the initial output value for set pins starting at first_set_pin
- **initial_set_pin_direction (int)** – the initial output direction for set pins starting at first_set_pin
- **first_sideset_pin (Pin)** – the first pin to use with the SIDESET instruction
- **sideset_pin_count (int)** – the count of consecutive pins to use with SIDESET starting at first_sideset_pin
- **initial_sideset_pin_state (int)** – the initial output value for sideset pins starting at first_sideset_pin
- **initial_sideset_pin_direction (int)** – the initial output direction for sideset pins starting at first_sideset_pin
- **sideset_enable (bool)** – if true, the sideset function is enabled
- **jmp_pin (Pin)** – the pin to use with the JMP instruction
- **jmp_pin_pull (digitalio.Pull)** – the pull direction for jmp pin
- **exclusive_pin_use (bool)** – if true, pins are used exclusively
- **auto_pull (bool)** – if true, auto pull is enabled
- **pull_threshold (int)** – the threshold for pull direction
- **out_shift_right (bool)** – if true, output direction is shifted right
- **wait_for_txstall (bool)** – if true, wait for txstall
- **auto_push (bool)** – if true, auto push is enabled
- **push_threshold (int)** – the threshold for push direction
- **in_shift_right (bool)** – if true, input direction is shifted right
- **user_interruptible (bool)** – if true, user can interrupt
- **wrap (int)** – the wrap value
- **wrap_target (int)** – the wrap target value
• **set_pin_count** (*int*) – the count of consecutive pins to use with SET starting at
first_set_pin

• **initial_set_pin_state** (*int*) – the initial output value for set pins starting at
first_set_pin

• **initial_set_pin_direction** (*int*) – the initial output direction for set pins starting at
first_set_pin

• **first_sideset_pin** (*Pin*) – the first pin to use with a side set

• **sideset_pin_count** (*int*) – the count of consecutive pins to use with a side set starting at
first_sideset_pin. Does not include sideset enable

• **initial_sideset_pin_state** (*int*) – the initial output value for sideset pins starting at
first_sideset_pin

• **initial_sideset_pin_direction** (*int*) – the initial output direction for sideset pins
starting at first_sideset_pin

• **sideset_enable** (*bool*) – True when the top sideset bit is to enable. This should be used
with the “.side_set # opt” directive

• **jmp_pin** (*Pin*) – the pin which determines the branch taken by JMP PIN instructions

• **jmp_pin_pull** (*Pull*) – The pull value for the jmp pin, default is no pull.

• **exclusive_pin_use** (*bool*) – When True, do not share any pins with other state machines.
Pins are never shared with other peripherals

• **auto_pull** (*bool*) – When True, automatically load data from the tx FIFO into the output
shift register (OSR) when an OUT instruction shifts more than pull_threshold bits

• **pull_threshold** (*int*) – Number of bits to shift before loading a new value into the OSR
from the tx FIFO

• **out_shift_right** (*bool*) – When True, data is shifted out the right side (LSB) of the
OSR. It is shifted out the left (MSB) otherwise. NOTE! This impacts data alignment when
the number of bytes is not a power of two (1, 2 or 4 bytes).

• **wait_for_txstall** (*bool*) – When True, writing data out will block until the TX FIFO
and OSR are empty and an instruction is stalled waiting for more data. When False, data
writes won’t wait for the OSR to empty (only the TX FIFO) so make sure you give enough
time before deiniting or stopping the state machine.

• **auto_push** (*bool*) – When True, automatically save data from input shift register (ISR) into
the rx FIFO when an IN instruction shifts more than push_threshold bits

• **push_threshold** (*int*) – Number of bits to shift before saving the ISR value to the RX
FIFO

• **in_shift_right** (*bool*) – When True, data is shifted into the right side (LSB) of the ISR.
It is shifted into the left (MSB) otherwise. NOTE! This impacts data alignment when
the number of bytes is not a power of two (1, 2 or 4 bytes).

• **user_interruptible** (*bool*) – When True (the default), *write*(), *readinto*(), and
*write_readinto*() can be interrupted by a ctrl-C. This is useful when developing a PIO
program: if there is an error in the program that causes an infinite loop, you will be able to
interrupt the loop. However, if you are writing to a device that can get into a bad state if a
read or write is interrupted, you may want to set this to False after your program has been
vetted.
• **wrap_target** *(int)* – The target instruction number of automatic wrap. Defaults to the first instruction of the program.

• **wrap** *(int)* – The instruction after which to wrap to the *wrap* instruction. As a special case, -1 (the default) indicates the last instruction of the program.

**writing:**  *bool*

Returns True if a background write is in progress.

**pending:**  *int*

Returns the number of pending buffers for background writing.

If the number is 0, then a `StateMachine.background_write` call will not block.

**frequency:**  *int*

The actual state machine frequency. This may not match the frequency requested due to internal limitations.

**txstall:**  *bool*

True when the state machine has stalled due to a full TX FIFO since the last `clear_txstall` call.

**rxstall:**  *bool*

True when the state machine has stalled due to a full RX FIFO since the last `clear_rxfifo` call.

**in_waiting:**  *int*

The number of words available to read into

**deinit()** → None

Turn off the state machine and release its resources.

**__enter__()** → *StateMachine*

No-op used by Context Managers. Provided by context manager helper.

**__exit__()** → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**restart()** → None

Resets this state machine, runs any *init* and enables the clock.

**run**(instructions: *circuitpython_typing.ReadableBuffer*) → None

Runs all given instructions. They will likely be interleaved with in-memory instructions. Make sure this doesn’t wait for input!

This can be used to output internal state to the RX FIFO and then read with `readinto`.

**stop()** → None

Stops the state machine clock. Use *restart* to enable it.

**write**(buffer: *circuitpython_typing.ReadableBuffer*, *, start: int | None = 0, end: int | None = None, swap: bool = False) → None

Write the data contained in *buffer* to the state machine. If the buffer is empty, nothing happens.

Writes to the FIFO will match the input buffer’s element size. For example, bytearray elements will perform 8 bit writes to the PIO FIFO. The RP2040’s memory bus will duplicate the value into the other byte positions. So, pulling more data in the PIO assembly will read the duplicated values.

To perform 16 or 32 bits writes into the FIFO use an *array.array* with a type code of the desired size.

**Parameters**

• **buffer** *(ReadableBuffer)* – Write out the data in this buffer
• **start** (*int*) – Start of the slice of `buffer` to write out: `buffer[start:end]`

• **end** (*int*) – End of the slice; this index is not included. Defaults to `len(buffer)`

• **swap** (*bool*) – For 2- and 4-byte elements, swap (reverse) the byte order

```python
background_write(once: circuitpython_typing.ReadableBuffer | None = None, *, loop: circuitpython_typing.ReadableBuffer | None = None, swap: bool = False) → None
```

Write data to the TX fifo in the background, with optional looping.

First, if any previous `once` or `loop` buffer has not been started, this function blocks until they have been started. This means that any `once` or `loop` buffer will be written at least once. Then the `once` and/or `loop` buffers are queued, and the function returns. The `once` buffer (if specified) will be written just once. Finally, the `loop` buffer (if specified) will continue being looped indefinitely.

Writes to the FIFO will match the input buffer’s element size. For example, bytearray elements will perform 8 bit writes to the PIO FIFO. The RP2040’s memory bus will duplicate the value into the other byte positions. So, pulling more data in the PIO assembly will read the duplicated values.

To perform 16 or 32 bits writes into the FIFO use an `array.array` with a type code of the desired size, or use `memoryview.cast` to change the interpretation of an existing buffer. To send just part of a larger buffer, slice a `memoryview` of it.

If a buffer is modified while it is being written out, the updated values will be used. However, because of interactions between CPU writes, DMA and the PIO FIFO are complex, it is difficult to predict the result of modifying multiple values. Instead, alternate between a pair of buffers.

Having both a `once` and a `loop` parameter is to support a special case in PWM generation where a change in duty cycle requires a special transitional buffer to be used exactly once. Most use cases will probably only use one of `once` or `loop`.

Having neither `once` nor `loop` terminates an existing background looping write after exactly a whole loop. This is in contrast to `stop_background_write`, which interrupts an ongoing DMA operation.

**Parameters**

- **once** (*Optional[circuitpython_typing.ReadableBuffer]*) – Data to be written once

- **loop** (*Optional[circuitpython_typing.ReadableBuffer]*) – Data to be written repeatedly

- **swap** (*bool*) – For 2- and 4-byte elements, swap (reverse) the byte order

```python
stop_background_write() → None
```

Immediately stop a background write, if one is in progress. Any DMA in progress is halted, but items already in the TX FIFO are not affected.

```python
readinto(buffer: circuitpython_typing.WriteableBuffer, *, start: int = 0, end: int | None = None, swap: bool = False) → None
```

Read into `buffer`. If the number of bytes to read is 0, nothing happens. The buffer includes any data added to the fifo even if it was added before this was called.

Reads from the FIFO will match the input buffer’s element size. For example, bytearray elements will perform 8 bit reads from the PIO FIFO. The alignment within the 32 bit value depends on `in_shift_right`. When `in_shift_right` is True, the upper N bits will be read. The lower bits will be read when `in_shift_right` is False.

To perform 16 or 32 bits writes into the FIFO use an `array.array` with a type code of the desired size.

**Parameters**

- **buffer** (*WriteableBuffer*) – Read data into this buffer
• **start** *(int)* – Start of the slice of **buffer** to read into: `buffer[start:end]`
• **end** *(int)* – End of the slice; this index is not included. Defaults to `len(buffer)`
• **swap** *(bool)* – For 2- and 4-byte elements, swap (reverse) the byte order

**write_readinto**(buffer_out: `circuitpython_typing.ReadableBuffer`, buffer_in: `circuitpython_typing.WriteableBuffer`, *, out_start: *int* = 0, out_end: *int | None* = None, in_start: *int* = 0, in_end: *int | None* = None) → None

Write out the data in `buffer_out` while simultaneously reading data into `buffer_in`. The lengths of the slices defined by `buffer_out[out_start:out_end]` and `buffer_in[in_start=in_end]` may be different. The function will return once both are filled. If buffer slice lengths are both 0, nothing happens.

Data transfers to and from the FIFOs will match the corresponding buffer's element size. See `write` and `readinto` for details.

To perform 16 or 32 bits writes into the FIFO use an `array.array` with a type code of the desired size.

**Parameters**
• **buffer_out** *(ReadableBuffer)* – Write out the data in this buffer
• **buffer_in** *(WriteableBuffer)* – Read data into this buffer
• **out_start** *(int)* – Start of the slice of `buffer_out` to write out: `buffer_out[out_start:out_end]`
• **out_end** *(int)* – End of the slice; this index is not included. Defaults to `len(buffer_out)`
• **in_start** *(int)* – Start of the slice of `buffer_in` to read into: `buffer_in[in_start:in_end]`
• **in_end** *(int)* – End of the slice; this index is not included. Defaults to `len(buffer_in)`
• **swap_out** *(bool)* – For 2- and 4-byte elements, swap (reverse) the byte order for the buffer being transmitted (written)
• **swap_in** *(bool)* – For 2- and 4-rx elements, swap (reverse) the byte order for the buffer being received (read)

**clear_rxfifo**() → None
Clears any unread bytes in the rxfifo.

**clear_txstall**() → None
Clears the txstall flag.

**rtc – Real Time Clock**

The **rtc** module provides support for a Real Time Clock. You can access and manage the RTC using **rtc.RTC**. It also backs the `time.time()` and `time.localtime()` functions using the onboard RTC if present.

**rtc.set_time_source**(rtc: **RTC**) → None
Sets the RTC time source used by `time.localtime()`. The default is **rtc.RTC**, but it’s useful to use this to override the time source for testing purposes. For example:

```python
import rtc
import time

class RTC(object):
```

(continues on next page)
@property
def datetime(self):
    return time.struct_time((2018, 3, 17, 21, 1, 47, 0, 0, 0))

r = RTC()
rtc.set_time_source(r)

class rtc.RTC

Real Time Clock

This class represents the onboard Real Time Clock. It is a singleton and will always return the same instance.

datetime: time.struct_time

The current date and time of the RTC as a time.struct_time.

This must be set to the current date and time whenever the board loses power:

```python
import rtc
import time

r = rtc.RTC()
r.datetime = time.struct_time((2019, 5, 29, 15, 14, 15, 0, -1, -1))
```

Once set, the RTC will automatically update this value as time passes. You can read this property to get a snapshot of the current time:

```python
current_time = r.datetime
print(current_time)
# struct_time(tm_year=2019, tm_month=5, ...)
```

calibration: int

The RTC calibration value as an int.

A positive value speeds up the clock and a negative value slows it down.

Limitations: Calibration not supported on SAMD, nRF, RP240, Spresense, and STM.

Range and value is hardware specific, but one step is often approximately 1 ppm:

```python
import rtc
import time

r = rtc.RTC()
r.calibration = 1
```

samd – SAMD implementation settings

class samd.Clock

Identifies a clock on the microcontroller.

They are fixed by the hardware so they cannot be constructed on demand. Instead, use samd.clock to reference the desired clock.
enabled: bool
   Is the clock enabled? (read-only)

parent: Clock | None
   Clock parent. (read-only)

frequency: int
   Clock frequency in Herz. (read-only)

calibration: int
   Clock calibration. Not all clocks can be calibrated.

sdcardio – Interface to an SD card via the SPI bus

class sdcardio.SDCard(bus: busio.SPI, cs: microcontroller.Pin, baudrate: int = 8000000)

SD Card Block Interface

Controls an SD card over SPI. This built-in module has higher read performance than the library adafruit_sdcard, but it is only compatible with busio.SPI, not bitbangio.SPI. Usually an SDCard object is used with storage.VfsFat to allow file I/O to an SD card.

Construct an SPI SD Card object with the given properties

Parameters

• spi (busio.SPI) – The SPI bus

• cs (microcontroller.Pin) – The chip select connected to the card

• baudrate (int) – The SPI data rate to use after card setup

Note that during detection and configuration, a hard-coded low baudrate is used. Data transfers use the specified baudrate (rounded down to one that is supported by the microcontroller)

Important: If the same SPI bus is shared with other peripherals, it is important that the SD card be initialized before accessing any other peripheral on the bus. Failure to do so can prevent the SD card from being recognized until it is powered off or re-inserted.

Example usage:

```python
import os
import board
import sdcardio
import storage

sd = sdcardio.SDCard(board.SPI(), board.SD_CS)
vfs = storage.VfsFat(sd)
storage.mount(vfs, '/sd')
os.listdir('/sd')
```

count() → int

Returns the total number of sectors

Due to technical limitations, this is a function and not a property.
Returns
The number of 512-byte blocks, as a number

definit() → None
Disable permanently.

Returns
None

readblocks(start_block: int, buf: circuitpython_typing.WriteableBuffer) → None
Read one or more blocks from the card

Parameters
• start_block (int) – The block to start reading from
• buf (WriteableBuffer) – The buffer to write into. Length must be multiple of 512.

Returns
None

class sdioio.SDCard(clock: microcontroller.Pin, command: microcontroller.Pin, data: Sequence[microcontroller.Pin], frequency: int)

SD Card Block Interface with SDIO

Controls an SD card over SDIO. SDIO is a parallel protocol designed for SD cards. It uses a clock pin, a command pin, and 1 or 4 data pins. It can be operated at a high frequency such as 25MHz. Usually an SDCard object is used with storage.VfsFat to allow file I/O to an SD card.

Construct an SDIO SD Card object with the given properties

Parameters
• clock (Pin) – the pin to use for the clock.
• command (Pin) – the pin to use for the command.
• data – A sequence of pins to use for data.
• frequency – The frequency of the bus in Hz

Example usage:
import os
import board
import sdioio
import storage

sd = sdioio.SDCard(
    clock=board.SDIO_CLOCK,
    command=board.SDIO_COMMAND,
    data=board.SDIO_DATA,
    frequency=25000000)

vfs = storage.VfsFat(sd)
storage.mount(vfs, '\sd')
os.listdir('sd')

frequency: int

The actual SDIO bus frequency. This may not match the frequency requested due to internal limitations.

width: int

The actual SDIO bus width, in bits

configure(frequency: int = 0, width: int = 0) → None

Configures the SDIO bus.

Parameters

- frequency (int) – the desired clock rate in Hertz. The actual clock rate may be higher or lower due to the granularity of available clock settings. Check the frequency attribute for the actual clock rate.

- width (int) – the number of data lines to use. Must be 1 or 4 and must also not exceed the number of data lines at construction

Note: Leaving a value unspecified or 0 means the current setting is kept

count() → int

Returns the total number of sectors

Due to technical limitations, this is a function and not a property.

Returns

The number of 512-byte blocks, as a number

readblocks(start_block: int, buf: circuitpython_typing.WriteableBuffer) → None

Read one or more blocks from the card

Parameters

- start_block (int) – The block to start reading from

- buf (WriteableBuffer) – The buffer to write into. Length must be multiple of 512.

Returns

None

writeblocks(start_block: int, buf: circuitpython_typing.ReadableBuffer) → None

Write one or more blocks to the card
Parameters

- **start_block** (*int*) – The block to start writing from
- **buf** (*ReadableBuffer*) – The buffer to read from. Length must be multiple of 512.

Returns

None

deinit() → None

Disable permanently.

Returns

None

__enter__() → **SDCard**

No-op used by Context Managers. Provided by context manager helper.

__exit__() → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**sharppdisplay – Support for Sharp Memory Display framebuffers**

For more information about working with Sharp Memory Displays, see this Learn guide.


A framebuffer for a memory-in-pixel display. Sharp makes monochrome displays and JDI used to make 8-color displays.

This initializes a display and connects it into CircuitPython. Unlike other objects in CircuitPython, Display objects live until *displayio.release_displays()* is called. This is done so that CircuitPython can use the display itself.

Create a framebuffer for the memory-in-pixel display.

Parameters

- **spi_bus** (*busio.SPI*) – The SPI bus that the display is connected to
- **chip_select** (*microcontroller.Pin*) – The pin connect to the display’s chip select line
- **width** (*int*) – The width of the display in pixels
- **height** (*int*) – The height of the display in pixels
- **baudrate** (*int*) – The baudrate to communicate with the screen at
- **jdi_display** (*bool*) – When True, work with an 8-color JDI display. Otherwise, a monochrome Sharp display.

deinit() → None

Free the resources (pins, timers, etc.) associated with this SharpMemoryFramebuffer instance. After deinitialization, no further operations may be performed.
socketpool

The `socketpool` module provides sockets through a pool. The pools themselves act like CPython’s `socket` module. For more information about the `socket` module, see the CPython documentation: https://docs.python.org/3/library/socket.html

class `socketpool.Socket`

TCP, UDP and RAW socket. Cannot be created directly. Instead, call `SocketPool.socket()`. Provides a subset of CPython’s `socket.socket` API. It only implements the versions of `recv` that do not allocate bytes objects.

`__hash__()` → int

Returns a hash for the Socket.

`__enter__()` → `Socket`

No-op used by Context Managers.

`__exit__()` → None

Automatically closes the Socket when exiting a context. See Lifetime and ContextManagers for more info.

`accept()` → `Tuple[Socket, Tuple[str, int]]`

Accept a connection on a listening socket of type SOCK_STREAM, creating a new socket of type SOCK_STREAM. Returns a tuple of (new_socket, remote_address)

`bind(address: Tuple[str, int])` → None

Bind a socket to an address

Parameters

`address` (~tuple) – tuple of (remote_address, remote_port)

`close()` → None

Closes this Socket and makes its resources available to its SocketPool.

`connect(address: Tuple[str, int])` → None

Connect a socket to a remote address

Parameters

`address` (~tuple) – tuple of (remote_address, remote_port)

`listen(backlog: int)` → None

Set socket to listen for incoming connections

Parameters

`backlog` (~int) – length of backlog queue for waiting connections

`recvfrom_into(buffer: circuitpython_typing.WriteableBuffer)` → `Tuple[int, Tuple[str, int]]`

Reads some bytes from a remote address.

Returns a tuple containing * the number of bytes received into the given buffer * a remote_address, which is a tuple of ip address and port number

Parameters

`buffer` (object) – buffer to read into

`recv_into(buffer: circuitpython_typing.WriteableBuffer, bufsize: int)` → int

Reads some bytes from the connected remote address, writing into the provided buffer. If bufsize <= len(buffer) is given, a maximum of bufsize bytes will be read into the buffer. If no valid value is given for bufsize, the default is the length of the given buffer.
Suits sockets of type SOCK_STREAM. Returns an int of number of bytes read.

**Parameters**

- **buffer (bytearray)** – buffer to receive into
- **bufsize (int)** – optionally, a maximum number of bytes to read.

**send**(bytes: circuitpython_typing.ReadableBuffer) → int

Send some bytes to the connected remote address. Suits sockets of type SOCK_STREAM

**Parameters**

- **bytes (~bytes)** – some bytes to send

**sendall**(bytes: circuitpython_typing.ReadableBuffer) → None

Send some bytes to the connected remote address. Suits sockets of type SOCK_STREAM

This calls send() repeatedly until all the data is sent or an error occurs. If an error occurs, it’s impossible to tell how much data has been sent.

**Parameters**

- **bytes (~bytes)** – some bytes to send

**sendto**(bytes: circuitpython_typing.ReadableBuffer, address: Tuple[str, int]) → int

Send some bytes to a specific address. Suits sockets of type SOCK_DGRAM

**Parameters**

- **bytes (~bytes)** – some bytes to send
- **address (~tuple)** – tuple of (remote_address, remote_port)

**setblocking**(flag: bool) → int | None

Set the blocking behaviour of this socket.

**Parameters**

- **flag (~bool)** – False means non-blocking, True means block indefinitely.

**setsockopt**(level: int, optname: int, value: int) → None

Sets socket options

**settimeout**(value: int) → None

Set the timeout value for this socket.

**Parameters**

- **value (~int)** – timeout in seconds. 0 means non-blocking. None means block indefinitely.

```python
class socketpool.SocketPool(radio: wifi.Radio)

A pool of socket resources available for the given radio. Only one SocketPool can be created for each radio.

SocketPool should be used in place of CPython’s socket which provides a pool of sockets provided by the underlying OS.

Create a new SocketPool object for the provided radio

**Parameters**

- **radio (wifi.Radio)** – The (connected) network hardware to associate with this SocketPool; currently, this will always be the object returned by wifi.radio

**AF_INET**: int

**AF_INET6**: int
SOCK_STREAM: int
SOCK_DGRAM: int
SOCK_RAW: int
EAI_NONAME: int
TCP_NODELAY: int
IPPROTO_TCP: int
IPPROTO_IP: int
IP_MULTICAST_TTL: int

socket(family: int = AF_INET, type: int = SOCK_STREAM) → Socket

Create a new socket

Parameters

• family (~int) – AF_INET or AF_INET6
• type (~int) – SOCK_STREAM, SOCK_DGRAM or SOCK_RAW

The proto (protocol) and fileno arguments available in socket.socket() in CPython are not supported.

getaddrinfo(host: str, port: int, family: int = 0, type: int = 0, proto: int = 0, flags: int = 0) → Tuple[int, int, str, Tuple[str, int]]

Gets the address information for a hostname and port

Returns the appropriate family, socket type, socket protocol and address information to call socket.socket() and socket.connect() with, as a tuple.

ssl

The ssl module provides SSL contexts to wrap sockets in.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: ssl.

ssl.create_default_context() → SSLContext

Return the default SSLContext.

class ssl.SSLContext

Settings related to SSL that can be applied to a socket by wrapping it. This is useful to provide SSL certificates to specific connections rather than all of them.

check_hostname: bool

Whether to match the peer certificate’s hostname.

load_cert_chain(certfile: str, keyfile: str) → None

Load a private key and the corresponding certificate.

The certfile string must be the path to a single file in PEM format containing the certificate as well as any number of CA certificates needed to establish the certificate’s authenticity. The keyfile string must point to a file containing the private key.
`load_verify_locations` *(cadata: str | None = None) → None*

Load a set of certification authority (CA) certificates used to validate other peers' certificates.

`set_default_verify_paths()` → None

Load a set of default certification authority (CA) certificates.

`wrap_socket` *(sock: socketpool.Socket, *, server_side: bool = False, server_hostname: str | None = None) → SSLSocket*

Wraps the socket into a socket-compatible class that handles SSL negotiation. The socket must be of type `SOCK_STREAM`.

**class** *ssl.SSLSocket*

Implements TLS security on a subset of `socketpool.Socket` functions. Cannot be created directly. Instead, call `wrap_socket` on an existing socket object.

Provides a subset of CPython’s `ssl.SSLSocket` API. It only implements the versions of `recv` that do not allocate bytes objects.

`__hash__` () → int

Returns a hash for the Socket.

`__enter__` () → SSLSocket

No-op used by Context Managers.

`__exit__` () → None

Automatically closes the Socket when exiting a context. See *Lifetime and ContextManagers* for more info.

`accept()` → Tuple[SSLSocket, Tuple[str, int]]

Accept a connection on a listening socket of type `SOCK_STREAM`, creating a new socket of type `SOCK_STREAM`. Returns a tuple of (new_socket, remote_address).

`bind(address: Tuple[str, int]) → None`

Bind a socket to an address

**Parameters**

address (~tuple) – tuple of (remote_address, remote_port)

`close()` → None

Closes this Socket

`connect(address: Tuple[str, int]) → None`

Connect a socket to a remote address

**Parameters**

address (~tuple) – tuple of (remote_address, remote_port)

`listen(backlog: int) → None`

Set socket to listen for incoming connections

**Parameters**

backlog (~int) – length of backlog queue for waiting connections

`recv_into(buffer: circuitpython_typing.WriteableBuffer, bufsize: int) → int`

Reads some bytes from the connected remote address, writing into the provided buffer. If bufsize <= len(buffer) is given, a maximum of bufsize bytes will be read into the buffer. If no valid value is given for bufsize, the default is the length of the given buffer.

Suits sockets of type `SOCK_STREAM` Returns an int of number of bytes read.

**Parameters**
• **buffer** (*bytearray*) – buffer to receive into

• **bufsize** (*int*) – optionally, a maximum number of bytes to read.

```python
send(bytes: circuitpython_typing.ReadableBuffer) → int
```

Send some bytes to the connected remote address. Suits sockets of type SOCK_STREAM

**Parameters**

- **bytes** (*~bytes*) – some bytes to send

```python
settimeout(value: int) → None
```

Set the timeout value for this socket.

**Parameters**

- **value** (*~int*) – timeout in seconds. 0 means non-blocking. None means block indefinitely.

```python
setblocking(flag: bool) → int | None
```

Set the blocking behaviour of this socket.

**Parameters**

- **flag** (*~bool*) – False means non-blocking, True means block indefinitely.

### storage – Storage management

The **storage** provides storage management functionality such as mounting and unmounting which is typically handled by the operating system hosting Python. CircuitPython does not have an OS, so this module provides this functionality directly.

For more information regarding using the **storage** module, refer to the CircuitPython Essentials Learn guide.

```python
storage.mount(filesystem: VfsFat, mount_path: str, *, readonly: bool = False) → None
```

Mounts the given filesystem object at the given path.

This is the CircuitPython analog to the UNIX `mount` command.

**Parameters**

- **filesystem** (*VfsFat*) – The filesystem to mount.
- **mount_path** (*str*) – Where to mount the filesystem.
- **readonly** (*bool*) – True when the filesystem should be readonly to CircuitPython.

```python
storage.umount(mount: str | VfsFat) → None
```

Unmounts the given filesystem object or if `mount` is a path, then unmount the filesystem mounted at that location.

This is the CircuitPython analog to the UNIX `umount` command.

```python
storage.remount(mount_path: str, readonly: bool = False, *, disable_concurrent_write_protection: bool = False) → None
```

Remounts the given path with new parameters.

**Parameters**

- **mount_path** (*str*) – The path to remount.
- **readonly** (*bool*) – True when the filesystem should be readonly to CircuitPython.
- **disable_concurrent_write_protection** (*bool*) – When True, the check that makes sure the underlying filesystem data is written by one computer is disabled. Disabling the protection allows CircuitPython and a host to write to the same filesystem with the risk that the filesystem will be corrupted.
storage.getmount(mount_path: str) → VfsFat

Retrieves the mount object associated with the mount path

storage.erase_filesystem(extended: bool | None = None) → None

Erase and re-create the CIRCUITPY filesystem.

On boards that present USB-visible CIRCUITPY drive (e.g., SAMD21 and SAMD51), then call microcontroller.reset() to restart CircuitPython and have the host computer remount CIRCUITPY.

This function can be called from the REPL when CIRCUITPY has become corrupted.

Parameters

- extended (bool) – On boards that support dualbank module and the extended parameter, the CIRCUITPY storage can be extended by setting this to True. If this isn’t provided or set to None (default), the existing configuration will be used.

Note: New firmware starts with storage extended. In case of an existing filesystem (e.g. uf2 load), the existing extension setting is preserved.

Warning: All the data on CIRCUITPY will be lost, and CircuitPython will restart on certain boards.

storage.disable_usb_drive() → None

Disable presenting CIRCUITPY as a USB mass storage device. By default, the device is enabled and CIRCUITPY is visible. Can be called in boot.py, before USB is connected.

storage.enable_usb_drive() → None

Enabled presenting CIRCUITPY as a USB mass storage device. By default, the device is enabled and CIRCUITPY is visible, so you do not normally need to call this function. Can be called in boot.py, before USB is connected.

If you enable too many devices at once, you will run out of USB endpoints. The number of available endpoints varies by microcontroller. CircuitPython will go into safe mode after running boot.py to inform you if not enough endpoints are available.

class storage.VfsFat(block_device: circuitpython_typing.BlockDevice)

Create a new VfsFat filesystem around the given block device.

Parameters

- block_device – Block device the filesystem lives on

label: str

The filesystem label, up to 11 case-insensitive bytes. Note that this property can only be set when the device is writable by the microcontroller.

readonly: bool

True when the device is mounted as readonly by the microcontroller. This property cannot be changed, use storage.remount instead.

static mkfs(block_device: circuitpython_typing.BlockDevice) → None

Format the block device, deleting any data that may have been there.

Limitations: On SAMD21 builds, mkfs() will raise OSError(22) when attempting to format filesystems larger than 4GB. The extra code to format larger filesystems will not fit on these builds. You can still access larger filesystems, but you will need to format the filesystem on another device.
open(path: str, mode: str) → None
Like builtin open()

ilistdir(path: str) → Iterator[Tuple[AnyStr, int, int, int] | Tuple[AnyStr, int, int]]
Return an iterator whose values describe files and folders within path

mkdir(path: str) → None
Like os.mkdir

rmdir(path: str) → None
Like os.rmdir

stat(path: str) → Tuple[int, int, int, int, int, int, int, int, int, int]
Like os.stat

statvfs(path: int) → Tuple[int, int, int, int, int, int, int, int, int, int]
Like os.statvfs

mount(readonly: bool, mkfs: VfsFat) → None
Don’t call this directly, call storage.mount.

umount() → None
Don’t call this directly, call storage.umount.

**struct – Manipulation of c-style data**

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: struct.

Supported size/byte order prefixes: @, <, >, !.

Supported format codes: b, B, x, h, H, i, l, I, L, q, Q, s, P, f, d (the latter 2 depending on the floating-point support).

struct.calcsize(fmt: str) → int
Return the number of bytes needed to store the given fmt.

struct.pack(fmt: str, *values: Any) → bytes
Pack the values according to the format string fmt. The return value is a bytes object encoding the values.

struct.pack_into(fmt: str, buffer: circuitpython_typing.WriteableBuffer, offset: int, *values: Any) → None
Pack the values according to the format string fmt into a buffer starting at offset. offset may be negative to count from the end of buffer.

struct.unpack(fmt: str, data: circuitpython_typing.ReadableBuffer) → Tuple[Any, Ellipsis]
Unpack from the data according to the format string fmt. The return value is a tuple of the unpacked values. The buffer size must match the size required by the format.

struct.unpack_from(fmt: str, data: circuitpython_typing.ReadableBuffer, offset: int = 0) → Tuple[Any, Ellipsis]
Unpack from the data starting at offset according to the format string fmt. offset may be negative to count from the end of buffer. The return value is a tuple of the unpacked values. The buffer size must be at least as big as the size required by the form.
supervisor – Supervisor settings

supervisor.runtime: `Runtime`
Runtime information, such as `runtime.serial_connected` (USB serial connection status). This object is the sole instance of `supervisor.Runtime`.

supervisor.status_bar: `StatusBar`
The status bar, shown on an attached display, and also sent to an attached terminal via OSC escape codes over the REPL serial connection. The status bar reports the current IP or BLE connection, what file is running, the last exception name and location, and firmware version information. This object is the sole instance of `supervisor.StatusBar`.

supervisor.reload() → None
Reload the main Python code and run it (equivalent to hitting Ctrl-D at the REPL).

Set what file to run on the next vm run.
When not None, the given filename is inserted at the front of the usual ['code.py', 'main.py'] search sequence. The optional keyword arguments specify what happens after the specified file has run:

- `..._on_success` take effect when the program runs to completion or calls `sys.exit()`.
- `..._on_error` take effect when the program exits with an exception, including the `KeyboardInterrupt` caused by CTRL-C.
- `..._on_reload` take effect when the program is interrupted by files being written to the USB drive (auto-reload) or when it calls `supervisor.reload()`.

These settings are stored in RAM, not in persistent memory, and will therefore only affect soft reloads. Powering off or resetting the device will always revert to standard settings.

When called multiple times in the same run, only the last call takes effect, replacing any settings made by previous ones. This is the main use of passing None as a filename: to reset to the standard search sequence.

supervisor.ticks_ms() → int
Return the time in milliseconds since an unspecified reference point, wrapping after 2**29ms.

The value is initialized so that the first overflow occurs about 65 seconds after power-on, making it feasible to check that your program works properly around an overflow.

The wrap value was chosen so that it is always possible to add or subtract two `ticks_ms` values without overflow on a board without long ints (or without allocating any long integer objects, on boards with long ints).

This ticks value comes from a low-accuracy clock internal to the microcontroller, just like `time.monotonic`. Due to its low accuracy and the fact that it “wraps around” every few days, it is intended for working with short term events like advancing an LED animation, not for long term events like counting down the time until a holiday.

Addition, subtraction, and comparison of ticks values can be done with routines like the following:
_TICKS_PERIOD = const(1<<29)
_TICKS_MAX = const(_TICKS_PERIOD-1)
_TICKS_HALFPERIOD = const(_TICKS_PERIOD//2)

def ticks_add(ticks, delta):
    "Add a delta to a base number of ticks, performing wraparound at 2**29ms."
    return (ticks + delta) % _TICKS_PERIOD

def ticks_diff(ticks1, ticks2):
    "Compute the signed difference between two ticks values, assuming that they are within 2**28 ticks"
    diff = (ticks1 - ticks2) & _TICKS_MAX
    diff = ((diff + _TICKS_HALFPERIOD) & _TICKS_MAX) - _TICKS_HALFPERIOD
    return diff

def ticks_less(ticks1, ticks2):
    "Return true iff ticks1 is less than ticks2, assuming that they are within 2**28 ticks"
    return ticks_diff(ticks1, ticks2) < 0

---

supervisor.get_previous_traceback() \rightarrow str | None
If the last vm run ended with an exception (including the KeyboardInterrupt caused by CTRL-C), returns the traceback as a string. Otherwise, returns None.
An exception traceback is only preserved over a soft reload, a hard reset clears it.
Only code (main or boot) runs are considered, not REPL runs.

supervisor.reset_terminal(x_pixels: int, y_pixels: int) \rightarrow None
Reset the CircuitPython serial terminal with new dimensions.

supervisor.set_usb_identification(manufacturer: str | None = None, product: str | None = None, vid: int = -1, pid: int = -1) \rightarrow None
Override identification constants in the USB Device Descriptor.
If passed, manufacturer and product must be ASCII strings (or buffers) of at most 126 characters. Any omitted arguments will be left at their default values.
This method must be called in boot.py to have any effect.
Not available on boards without native USB support.

class supervisor.RunReason
The reason that CircuitPython started running.

STARTUP: object
CircuitPython started the microcontroller started up. See microcontroller.Processor. reset_reason for more detail on why the microcontroller was started.

AUTO_RELOAD: object
CircuitPython restarted due to an external write to the filesystem.

SUPERVISOR_RELOAD: object
CircuitPython restarted due to a call to supervisor.reload().

REPL_RELOAD: object
CircuitPython started due to the user typing CTRL-D in the REPL.
class supervisor.Runtime

Current status of runtime objects.

Usage:

```python
import supervisor
if supervisor.runtime.serial_connected:
    print("Hello World!")
```

You cannot create an instance of `supervisor.Runtime`. Use `supervisor.runtime` to access the sole instance available.

**usb_connected**: `bool`

Returns the USB enumeration status (read-only).

**serial_connected**: `bool`

Returns the USB serial communication status (read-only).

**serial_bytes_available**: `int`

Returns the whether any bytes are available to read on the USB serial input. Allows for polling to see whether to call the built-in input() or wait. (read-only)

**run_reason**: `RunReason`

Why CircuitPython started running this particular time (read-only).

**safe_mode_reason**: `SafeModeReason`

Why CircuitPython went into safe mode this particular time (read-only).

**autoreload**: `bool`

Whether CircuitPython may autoreload based on workflow writes to the filesystem.

**ble_workflow**: `bool`

Enable/Disable ble workflow until a reset. This prevents BLE advertising outside of the VM and the services used for it.

**next_stack_limit**: `int`

The size of the stack for the next vm run. If its too large, the default will be used.

**rgb_status_brightness**: `int`

Set brightness of status RGB LED from 0-255. This will take effect after the current code finishes and the status LED is used to show the finish state.

class supervisor.SafeModeReason

The reason that CircuitPython went into safe mode.

**Limitations**: Class not available on builds that do not implement `safemode.py`.

**NONE**: `object`

CircuitPython is not in safe mode.

**BROWNOUT**: `object`

The microcontroller voltage dropped too low.

**FLASH_WRITE_FAIL**: `object`

Could not write to flash memory.
GC_ALLOC_OUTSIDE_VM: object
CircuitPython tried to allocate storage when its virtual machine was not running.

HARD_FAULT: object
The microcontroller detected a fault, such as an out-of-bounds memory write.

INTERRUPT_ERROR: object
Internal error related to interrupts.

NLR_JUMP_FAIL: object
An error occurred during exception handling, possibly due to memory corruption.

NO_CIRCUITPY: object
The CIRCUITPY drive was not available.

NO_HEAP: object
Heap storage was not present.

PROGRAMMATIC: object
The program entered safe mode using the supervisor module.

SDK_FATAL_ERROR: object
Third party firmware reported a fatal error.

STACK_OVERFLOW: object
The CircuitPython heap was corrupted because the stack was too small.

USB_BOOT_DEVICE_NOT_INTERFACE_ZERO: object
The USB HID boot device was not set up to be the first device, on interface #0.

USB_TOO_MANY_ENDPOINTS: object
USB devices need more endpoints than are available.

USB_TOO_MANY_INTERFACE_NAMES: object
USB devices specify too many interface names.

USER: object
The user pressed one or more buttons to enter safe mode. This safe mode does not cause safemode.py to be run, since its purpose is to prevent all user code from running. This allows errors in safemode.py to be corrected easily.

SAFE_MODE_WATCHDOG: object
An internal watchdog timer expired.

class supervisor.StatusBar
Current status of runtime objects.
Usage:

```python
import supervisor
supervisor.status_bar.console = False
```

You cannot create an instance of supervisor.StatusBar. Use supervisor.status_bar to access the sole instance available.
```python
circuit:

Whether status bar information is sent over the console (REPL) serial connection, using OSC terminal
escape codes that change the terminal’s title. Default is True. If set to False, status bar will be cleared
and then disabled. May be set in boot.py or later. Persists across soft restarts.

display:

Whether status bar information is displayed on the top line of the display. Default is True. If set to False,
status bar will be cleared and then disabled. May be set in boot.py or later. Persists across soft restarts.
Not available if terminalio is not available.

synthio – Support for multi-channel audio synthesis

At least 2 simultaneous notes are supported. samd5x, mimxrt10xx and rp2040 platforms support up to 12 notes.

**synthio.BlockInput**

Blocks and Notes can take any of these types as inputs on certain attributes
A BlockInput can be any of the following types: Math, LFO, builtins.float, None (treated same as 0).

```python
class synthio.Envelope(*, attack_time: float | None = 0.1,
decay_time: float | None = 0.05,
release_time: float | None = 0.2,
attack_level: float | None = 1.0,
sustain_level: float | None = 0.8)
```

Construct an Envelope object

The Envelope defines an ADSR (Attack, Decay, Sustain, Release) envelope with linear amplitude ramping.
A note starts at 0 volume, then increases to attack_volume over attack_time seconds; then it decays to
sustain_level over decay_time seconds. Finally, when the note is released, it decreases to 0 volume over
release_time.

If the sustain_level of an envelope is 0, then the decay and sustain phases of the note are always omitted. The
note is considered to be released as soon as the envelope reaches the end of the attack phase. The decay_time
is ignored. This is similar to how a plucked or struck instrument behaves.

If a note is released before it reaches its sustain phase, it decays with the same slope indicated by
sustain_level/release_time (or attack_level/release_time for plucked envelopes)

**Parameters**

- `attack_time` (float) – The time in seconds it takes to ramp from 0 volume to attack_volume
- `decay_time` (float) – The time in seconds it takes to ramp from attack_volume to sustain_volume
- `release_time` (float) – The time in seconds it takes to ramp from sustain_volume to release_volume. When a note is released before it has reached the sustain phase, the release is done with the same slope indicated by release_time and sustain_level. If the sustain_level is 0.0 then the release slope calculations use the attack_level instead.
- `attack_level` (float) – The level, in the range 0.0 to 1.0 of the peak volume of the attack phase
- `sustain_level` (float) – The level, in the range 0.0 to 1.0 of the volume of the sustain phase relative to the attack level

**attack_time**

The time in seconds it takes to ramp from 0 volume to attack_volume
decay_time: float
The time in seconds it takes to ramp from attack_volume to sustain_volume

release_time: float
The time in seconds it takes to ramp from sustain_volume to release_volume. When a note is released
before it has reached the sustain phase, the release is done with the same slope indicated by release_time
and sustain_level

attack_level: float
The level, in the range 0.0 to 1.0 of the peak volume of the attack phase

sustain_level: float
The level, in the range 0.0 to 1.0 of the volume of the sustain phase relative to the attack level

synthio.from_file(file: BinaryIO, *, sample_rate: int = 11025, waveform: circuitpython_typing.ReadableBuffer
| None = None, envelope: Envelope | None = None) -> MidiTrack
Create an AudioSample from an already opened MIDI file. Currently, only single-track MIDI (type 0) is sup-
ported.

Parameters
- file (BinaryIO) – Already opened MIDI file
- sample_rate (int) – The desired playback sample rate; higher sample rate requires more
  memory
- waveform (ReadableBuffer) – A single-cycle waveform. Default is a 50% duty cycle
  square wave. If specified, must be a ReadableBuffer of type ‘h’ (signed 16 bit)
- envelope (Envelope) – An object that defines the loudness of a note over time. The default
  envelope provides no ramping, voices turn instantly on and off.

Playing a MIDI file from flash:

```python
import audioio
import board
import synthio

data = open("single-track.midi", "rb")
midi = synthio.from_file(data)
a = audioio.AudioOut(board.A0)

print("playing")
a.play(midi)
while a.playing:
    pass
print("stopped")
```

synthio.midi_to_hz(midi_note: float) -> float
Converts the given midi note (60 = middle C, 69 = concert A) to Hz

synthio.voct_to_hz(ctrl: float) -> float
Converts a 1v/octave signal to Hz.

24/12 (2.0) corresponds to middle C, 33/12 (2.75) is concert A.

class synthio.Biquad(b0: float, b1: float, b2: float, a1: float, a2: float)
Construct a normalized biquad filter object.
This implements the “direct form 1” biquad filter, where each coefficient has been pre-divided by a0.

Biquad objects are usually constructed via one of the related methods on a `Synthesizer` object rather than directly from coefficients.

https://github.com/WebAudio/Audio-EQ-Cookbook/blob/main/Audio-EQ-Cookbook.txt

class synthio.LFO(waveform: circuitpython_typing.ReadableBuffer = None, *, rate: BlockInput = 1.0, scale: BlockInput = 1.0, offset: BlockInput = 0.0, phase_offset: BlockInput = 0.0, once=False, interpolate=True)

A low-frequency oscillator block

Every `rate` seconds, the output of the LFO cycles through its `waveform`. The output at any particular moment is `waveform[idx] * scale + offset`.

If `waveform` is None, a triangle waveform is used.

`rate`, `phase_offset`, `offset`, `scale`, and `once` can be changed at run-time. `waveform` may be mutated.

`waveform` must be a `ReadableBuffer` with elements of type 'h' (16-bit signed integer). Internally, the elements of `waveform` are scaled so that the input range [-32768, 32767] maps to [-1.0, 0.99996].

An LFO only updates if it is actually associated with a playing `Synthesizer`, including indirectly via a `Note` or another intermediate LFO.

Using the same LFO as an input to multiple other LFOs or `Notes` is OK, but the result if an LFO is tied to multiple `Synthesizer` objects is undefined.

In the current implementation, LFOs are updated every 256 samples. This should be considered an implementation detail, though it affects how LFOs behave for instance when used to implement an integrator (`1.0 * offset = 1`).

`waveform`: `circuitpython_typing.ReadableBuffer` | None
   The waveform of this LFO. (read-only, but the values in the buffer may be modified dynamically)

`rate`: `BlockInput`
   The rate (in Hz) at which the LFO cycles through its waveform

`offset`: `BlockInput`
   An additive value applied to the LFO’s output

`phase_offset`: `BlockInput`
   An additive value applied to the LFO’s phase

`scale`: `BlockInput`
   An multiplier value applied to the LFO’s output

`once`: `bool`
   True if the waveform should stop when it reaches its last output value, false if it should re-start at the beginning of its waveform

   This applies to the `phase` before the addition of any `phase_offset`

`interpolate`: `bool`
   True if the waveform should perform linear interpolation between values

`phase`: `float`
   The phase of the oscillator, in the range 0 to 1 (read-only)

`value`: `float`
   The value of the oscillator (read-only)
retrigger()

Reset the LFO’s internal index to the start of the waveform. Most useful when it its once property is True.

class synthio.MathOperation
Operation for a Math block

SUM: MathOperation
Computes a+b+c. For 2-input sum, set one argument to 0, 0. To hold a control value for multiple subscribers, set two arguments to 0, 0.

ADD_SUB: MathOperation
Computes a+b–c. For 2-input subtraction, set b to 0, 0.

PRODUCT: MathOperation
Computes a*b*c. For 2-input product, set one argument to 1, 0.

MUL_DIV: MathOperation
Computes a*b/c. If c is zero, the output is 1, 0.

SCALE_OFFSET: MathOperation
Computes (a*b)+c.

OFFSET_SCALE: MathOperation
Computes (a+b)*c. For 2-input multiplication, set b to 0.

LERP: MathOperation
Computes a * (1-c) + b * c.

CONSTRAINED_LERP: MathOperation
Computes a * (1-c') + b * c', where c' is constrained to be between 0, 0 and 1, 0.

DIV_ADD: MathOperation
Computes a/b+c. If b is zero, the output is c.

ADD_DIV: MathOperation
Computes (a+b)/c. For 2-input product, set b to 0, 0.

MID: MathOperation
Returns the middle of the 3 input values.

MAX: MathOperation
Returns the biggest of the 3 input values.

MIN: MathOperation
Returns the smallest of the 3 input values.

ABS: MathOperation
Returns the absolute value of a.

__call__(a: BlockInput, b: BlockInput = 0.0, c: BlockInput = 1.0) → Math
A MathOperation enumeration value can be called to construct a Math block that performs that operation

class synthio.Math(operation: MathOperation, a: BlockInput, b: BlockInput = 0.0, c: BlockInput = 1.0)
An arithmetic block
Performs an arithmetic operation on up to 3 inputs. See the documentation of MathOperation for the specific functions available.
The properties can all be changed at run-time.
An Math only updates if it is actually associated with a playing Synthesizer, including indirectly via a Note or another intermediate Math.

Using the same Math as an input to multiple other Maths or Notes is OK, but the result if an Math is tied to multiple Synthesizer objects is undefined.

In the current implementation, Maths are updated every 256 samples. This should be considered an implementation detail.

a: **BlockInput**
The first input to the operation

b: **BlockInput**
The second input to the operation
c: **BlockInput**
The third input to the operation

operation: **MathOperation**
The function to compute

value: **float**
The value of the oscillator (read-only)

class synthio.MidiTrack(**buffer**: circuitpython_typing.ReadableBuffer, **tempo**: int, *, **sample_rate**: int = 11025, **waveform**: circuitpython_typing.ReadableBuffer | None = None, **envelope**: Envelope | None = None)

Simple MIDI synth

Create a MidiTrack from the given stream of MIDI events. Only “Note On” and “Note Off” events are supported; channel numbers and key velocities are ignored. Up to two notes may be on at the same time.

Parameters

- **buffer** ([ReadableBuffer]) – Stream of MIDI events, as stored in a MIDI file track chunk
- **tempo** (int) – Tempo of the streamed events, in MIDI ticks per second
- **sample_rate** (int) – The desired playback sample rate; higher sample rate requires more memory
- **waveform** ([ReadableBuffer]) – A single-cycle waveform. Default is a 50% duty cycle square wave. If specified, must be a ReadableBuffer of type ‘h’ (signed 16 bit)
- **envelope** ([Envelope]) – An object that defines the loudness of a note over time. The default envelope provides no ramping, voices turn instantly on and off.

Simple melody:

```python
import audioio
import board
import synthio
dac = audioio.AudioOut(board.SPEAKER)
melody = synthio.MidiTrack(b"\0\x90H\0\x80H\0\6\x90J\0\6\x90J\0\6\x90J\0\6\x90\0"
  \x90\0"
  \x80L\0"
  \x80L\0"
  b"\0c\x90H\0T\x80H\0x0c\x90H\0T\x80H\0", tempo=640)
dac.play(melody)
```

(continues on next page)
```python
print("playing")
while dac.playing:
    pass
print("stopped")
```

**sample_rate**: `int`
32 bit value that tells how quickly samples are played in Hertz (cycles per second).

**error_location**: `int | None`
Offset, in bytes within the midi data, of a decoding error

**deinit** () → `None`
Deinitialises the MidiTrack and releases any hardware resources for reuse.

**__enter__** () → `MidiTrack`
No-op used by Context Managers.

**__exit__** () → `None`
Automatically deinitializes the hardware when exiting a context. See *Lifetime and Context Managers* for more info.

```python
class synthio.Note(*, frequency: float, panning: BlockInput = 0.0, waveform: circuitpython_typing.ReadableBuffer | None = None, envelope: Envelope | None = None, amplitude: BlockInput = 0.0, bend: BlockInput = 0.0, filter: Biquad | None = None, ring_frequency: float = 0.0, ring_bend: float = 0.0, ring_waveform: circuitpython_typing.ReadableBuffer | None = 0.0)
```

Construct a Note object, with a frequency in Hz, and optional panning, waveform, envelope, tremolo (volume change) and bend (frequency change).

If waveform or envelope are `None` the synthesizer object's default waveform or envelope are used.

If the same Note object is played on multiple Synthesizer objects, the result is undefined.

**frequency**: `float`
The base frequency of the note, in Hz.

**filter**: `Biquad | None`
If not `None`, the output of this Note is filtered according to the provided coefficients.

Construct an appropriate filter by calling a filter-making method on the *Synthesizer* object where you plan to play the note, as filter coefficients depend on the sample rate

**panning**: `BlockInput`
Defines the channel(s) in which the note appears.

-1 is left channel only, 0 is both channels, and 1 is right channel. For fractional values, the note plays at full amplitude in one channel and partial amplitude in the other channel. For instance -.5 plays at full amplitude in the left channel and 1/2 amplitude in the right channel.

**amplitude**: `BlockInput`
The relative amplitude of the note, from 0 to 1

An amplitude of 0 makes the note inaudible. It is combined multiplicatively with the value from the note’s envelope.

To achieve a tremolo effect, attach an LFO here.
**bend**: *BlockInput*

The pitch bend depth of the note, from -12 to +12

A depth of 0 plays the programmed frequency. A depth of 1 corresponds to a bend of 1 octave. A depth of $(1/12) = 0.0833$ corresponds to a bend of 1 semitone, and a depth of $0.00833$ corresponds to one musical cent.

To achieve a vibrato or sweep effect, attach an LFO here.

**waveform**: *circuitpython_typing.ReadableBuffer | None*

The waveform of this note. Setting the waveform to a buffer of a different size resets the note’s phase.

**envelope**: *Envelope*

The envelope of this note

**ring_frequency**: *float*

The ring frequency of the note, in Hz. Zero disables.

For ring to take effect, both *ring_frequency* and *ring_waveform* must be set.

**ring_bend**: *float*

The pitch bend depth of the note’s ring waveform, from -12 to +12

A depth of 0 plays the programmed frequency. A depth of 1 corresponds to a bend of 1 octave. A depth of $(1/12) = 0.0833$ corresponds to a bend of 1 semitone, and a depth of $0.00833$ corresponds to one musical cent.

To achieve a vibrato or sweep effect on the ring waveform, attach an LFO here.

**ring_waveform**: *circuitpython_typing.ReadableBuffer | None*

The ring waveform of this note. Setting the ring_waveform to a buffer of a different size resets the note’s phase.

For ring to take effect, both *ring_frequency* and *ring_waveform* must be set.

**synthio.NoteSequence**

A sequence of notes, which can each be integer MIDI note numbers or *Note* objects

**synthio.NoteOrNoteSequence**

A note or sequence of notes

**synthio.LFOOrLFOSequence**

An LFO or a sequence of LFOs

**class synthio.Synthesizer(*, sample_rate: int = 11025, channel_count: int = 1, waveform: circuitpython_typing.ReadableBuffer | None = None, envelope: Envelope | None = None)**

Create a synthesizer object.

This API is experimental.

Integer notes use MIDI note numbering, with 60 being C4 or Middle C, approximately 262Hz. Integer notes use the given waveform & envelope, and do not support advanced features like tremolo or vibrato.

**Parameters**

- **sample_rate** (*int*) – The desired playback sample rate; higher sample rate requires more memory
- **channel_count** (*int*) – The number of output channels (1=mono, 2=stereo)
• **waveform** (**ReadableBuffer**) – A single-cycle waveform. Default is a 50% duty cycle square wave. If specified, must be a ReadableBuffer of type ‘h’ (signed 16 bit)

• **envelope** (**Optional[Envelope]**) – An object that defines the loudness of a note over time. The default envelope, **None** provides no ramping, voices turn instantly on and off.

**envelope**:  
*Envelope | None*

The envelope to apply to all notes. **None**, the default envelope, instantly turns notes on and off. The envelope may be changed dynamically, but it affects all notes (even currently playing notes)

**sample_rate**:  
*int*

32 bit value that tells how quickly samples are played in Hertz (cycles per second).

**pressed**:  
*NoteSequence*

A sequence of the currently pressed notes (read-only property)

**blocks**:  
*List[BlockInput]*

A list of blocks to advance whether or not they are associated with a playing note.

This can be used to implement ‘free-running’ LFOs. LFOs associated with playing notes are advanced whether or not they are in this list.

This property is read-only but its contents may be modified by e.g., calling `synth.blocks.append()` or `synth.blocks.remove()`. It is initially an empty list.

**max_polyphony**:  
*int*

Maximum polyphony of the synthesizer (read-only class property)

**press**(*, press=*) → **None**

Turn some notes on.

Pressing a note that was already pressed has no effect.

**Parameters**

**press** (**NoteOrNoteSequence**) – Any sequence of notes.

**release**(*, release=*) → **None**

Turn some notes off.

Releasing a note that was already released has no effect.

**Parameters**

**release** (**NoteOrNoteSequence**) – Any sequence of notes.

**change**(release: **NoteOrNoteSequence** = (), press: **NoteOrNoteSequence** = (), retrigger=LFOOrLFOSequence) → **None**

Start notes, stop them, and/or re-trigger some LFOs.

The changes all happen atomically with respect to output generation.

It is OK to release note that was not actually turned on.

Pressing a note that was already pressed returns it to the attack phase but without resetting its amplitude. Releasing a note and immediately pressing it again returns it to the attack phase with an initial amplitude of 0.

At the same time, the passed LFOs (if any) are retriggered.

**Parameters**

• **release** (**NoteOrNoteSequence**) – Any sequence of notes.

• **press** (**NoteOrNoteSequence**) – Any sequence of notes.
• **retrigger** *(LFO0rLF0Sequence)* – Any sequence of LFOs.

Note: for compatibility, `release_then_press` may be used as an alias for this function. This compatibility name will be removed in 9.0.

**release_all_then_press** (/, press) → None

Turn any currently-playing notes off, then turn on the given notes

Releasing a note and immediately pressing it again returns it to the attack phase with an initial amplitude of 0.

**Parameters**

- **press** *(NoteOrNoteSequence)* – Any sequence of notes.

**release_all** () → None

Turn any currently-playing notes off

**deinit** () → None

Deinitialises the object and releases any memory resources for reuse.

**__enter__** () → *Synthesizer*

No-op used by Context Managers.

**__exit__** () → None

Automatically deinitializes the hardware when exiting a context. See *Lifetime and ContextManagers* for more info.

**low_pass_filter** *(frequency: float, q_factor: float = 0.7071067811865475) → Biquad*

Construct a low-pass filter with the given parameters.

- **frequency**, called f0 in the cookbook, is the corner frequency in Hz of the filter.
- **q_factor**, called Q in the cookbook. Controls how peaked the response will be at the cutoff frequency. A large value makes the response more peaked.

**high_pass_filter** *(frequency: float, q_factor: float = 0.7071067811865475) → Biquad*

Construct a high-pass filter with the given parameters.

- **frequency**, called f0 in the cookbook, is the corner frequency in Hz of the filter.
- **q_factor**, called Q in the cookbook. Controls how peaked the response will be at the cutoff frequency. A large value makes the response more peaked.

**band_pass_filter** *(frequency: float, q_factor: float = 0.7071067811865475) → Biquad*

Construct a band-pass filter with the given parameters.

- **frequency**, called f0 in the cookbook, is the center frequency in Hz of the filter.
- **q_factor**, called Q in the cookbook. Controls how peaked the response will be at the cutoff frequency. A large value makes the response more peaked.

The coefficients are scaled such that the filter has a 0dB peak gain.
terminalio — Displays text in a TileGrid

The `terminalio` module contains classes to display a character stream on a display. The built in font is available as `terminalio.FONT`.

**Note:** This module does not give access to the REPL.

`terminalio.FONT`: `fontio.BuiltinFont`

The built in font

```python

Display a character stream with a TileGrid

ASCII control: *
- \r - Move cursor to column 1
- \n - Move cursor down a row
- \b - Move cursor left one if possible

OSC control sequences: *
- ESC [ 0; <s> ESC \ - Set title bar to <s>

VT100 control sequences: *
- ESC [ K - Clear the remainder of the line
- ESC [ ### D - Move the cursor to the left by ###
- ESC [ 2 J - Erase the entire display
- ESC [ nnnn ; mmmm H - Move the cursor to mmmm, nnnn.

Terminal manages tile indices and cursor position based on VT100 commands. The font should be a `fontio.BuiltinFont` and the TileGrid’s bitmap should match the font’s bitmap.

```write(buf: circuitpython_typing.ReadableBuffer) → int | None

Write the buffer of bytes to the bus.

**Returns**

the number of bytes written

**Return type**

`int` or `None`

time – time and timing related functions

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: `time`.

```time.monotonic() → float

Returns an always increasing value of time with an unknown reference point. Only use it to compare against other values from `time.monotonic()`.

On most boards, `time.monotonic()` converts a 64-bit millisecond tick counter to a float. Floats on most boards are encoded in 30 bits internally, with effectively 22 bits of precision. The float returned by `time.monotonic()` will accurately represent time to millisecond precision only up to 2**22 milliseconds (about 1.165 hours). At that point it will start losing precision, and its value will change only every second millisecond. At 2**23 milliseconds it will change every fourth millisecond, and so forth.

If you need more consistent precision, use `time.monotonic_ns()`, or `supervisor.ticks_ms()`. `time.monotonic_ns()` is not available on boards without long integer support. `supervisor.ticks_ms()` uses intervals of a millisecond, but wraps around, and is not CPython-compatible.

**Returns**

the current monotonic time
Return type

float
time.sleep(seconds: float) → None
Sleep for a given number of seconds.
Parameters
seconds (float) – the time to sleep in fractional seconds
class time.struct_time(time_tuple: Sequence[int])
Structure used to capture a date and time. Can be constructed from a struct_time, tuple, list, or namedtuple with 9 elements.
Parameters
time_tuple (Sequence) – Sequence of time info: (tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec, tm_wday, tm_yday, tm_isdst)
  • tm_year: the year, 2017 for example
  • tm_mon: the month, range [1, 12]
  • tm_mday: the day of the month, range [1, 31]
  • tm_hour: the hour, range [0, 23]
  • tm_min: the minute, range [0, 59]
  • tm_sec: the second, range [0, 61]
  • tm_wday: the day of the week, range [0, 6], Monday is 0
  • tm_yday: the day of the year, range [1, 366], -1 indicates not known
  • tm_isdst: 1 when in daylight savings, 0 when not, -1 if unknown.
time.time() → int
Return the current time in seconds since since Jan 1, 1970.
Returns
the current time
Return type
int
time.monotonic_ns() → int
Return the time of the monotonic clock, which cannot go backward, in nanoseconds. Not available on boards without long integer support.
Returns
the current time
Return type
int
time.localtime(secs: int) → struct_time
Convert a time expressed in seconds since Jan 1, 1970 to a struct_time in local time. If secs is not provided or None, the current time as returned by time() is used. The earliest date for which it can generate a time is Jan 1, 2000.
Returns
the current time
Return type
time.struct_time
time.mktime(t: struct_time) → int

This is the inverse function of localtime(). Its argument is the struct_time or full 9-tuple (since the dst flag is needed; use -1 as the dst flag if it is unknown) which expresses the time in local time, not UTC. The earliest date for which it can generate a time is Jan 1, 2000.

Returns

seconds

Return type

int

touchio – Touch related IO

The touchio module contains classes to provide access to touch IO typically accelerated by hardware on the onboard microcontroller.

All classes change hardware state and should be deinitialized when they are no longer needed if the program continues after use. To do so, either call deinit() or use a context manager. See Lifetime and Context Managers for more info.

For more information about working with the touchio module in CircuitPython, see this Learn guide page.

Example:

```python
import touchio
from board import *

touch_pin = touchio.TouchIn(D6)
print(touch_pin.value)
```

This example will initialize the the device, and print the value.

class touchio.TouchIn(pin: microcontroller.Pin)

Read the state of a capacitive touch sensor

Usage:

```python
import touchio
from board import *

touch = touchio.TouchIn(A1)
while True:
    if touch.value:
        print("touched!")
```

Use the TouchIn on the given pin.

Parameters

pin (Pin) – the pin to read from

value: bool

Whether the touch pad is being touched or not. (read-only)

True when raw_value > threshold.

raw_value: int

The raw touch measurement as an int. (read-only)
threshold: \texttt{int | None} \\
Minimum \texttt{raw_value} needed to detect a touch (and for \texttt{value} to be \texttt{True}).

When the \texttt{TouchIn} object is created, an initial \texttt{raw_value} is read from the pin, and then \texttt{threshold} is set to be 100 + that value.

You can adjust \texttt{threshold} to make the pin more or less sensitive:

\begin{verbatim}
import board 
import touchio

touch = touchio.TouchIn(board.A1)
touch.threshold = 7300
\end{verbatim}

deinit() \rightarrow \texttt{None} \\
Deinitialises the TouchIn and releases any hardware resources for reuse.

\texttt{__enter__()} \rightarrow \texttt{TouchIn} \\
No-op used by Context Managers.

\texttt{__exit__()} \rightarrow \texttt{None} \\
Automatically deinitializes the hardware when exiting a context. See \textit{Lifetime and ContextManagers} for more info.

\textbf{traceback -- Traceback Module}

This module provides a standard interface to print stack traces of programs. This is useful when you want to print stack traces under program control.

This module implements a subset of the corresponding \texttt{CPython module}, as described below. For more information, refer to the original \texttt{CPython documentation}: \texttt{traceback}.

\texttt{traceback.format_exception(\texttt{exc: BaseException | Type[BaseException], /, value: BaseException | None = None, tb: types.TracebackType | None = None, limit: int | None = None, chain: bool | None = True) \rightarrow List[str]}

Format a stack trace and the exception information.

If the exception value is passed in \texttt{exc}, then this exception value and its associated traceback are used. This is compatible with CPython 3.10 and newer.

If the exception value is passed in \texttt{value}, then any value passed in for \texttt{exc} is ignored. \texttt{value} is used as the exception value and the traceback in the \texttt{tb} argument is used. In this case, if \texttt{tb} is None, no traceback will be shown. This is compatible with CPython 3.5 and newer.

The arguments have the same meaning as the corresponding arguments to print\texttt{.exception}(). The return value is a list of strings, each ending in a newline and some containing internal newlines. When these lines are concatenated and printed, exactly the same text is printed as does print\texttt{.exception}().

\textbf{Parameters}

- \texttt{exc} -- The exception. Must be an instance of \texttt{BaseException}. Unused if value is specified.
- \texttt{value} -- If specified, is used in place of \texttt{exc}.
- \texttt{tb} (\texttt{TracebackType}) -- When value is also specified, \texttt{tb} is used in place of the exception's own traceback. If \texttt{None}, the traceback will not be printed.
• **limit** (*int*) – Print up to limit stack trace entries (starting from the caller’s frame) if limit is positive. Otherwise, print the last abs(limit) entries. If limit is omitted or None, all entries are printed.

• **chain** (*bool*) – If True then chained exceptions will be printed.

**traceback.print_exception**(*exc: BaseException | Type[BaseException], ./, value: BaseException | None = None, tb: types.TracebackType | None = None, limit: int | None = None, file: io.FileIO | None = None, chain: bool | None = True) → None

Prints exception information and stack trace entries.

If the exception value is passed in *exc*, then this exception value and its associated traceback are used. This is compatible with CPython 3.10 and newer.

If the exception value is passed in *value*, then any value passed in for *exc* is ignored. value is used as the exception value and the traceback in the *tb* argument is used. In this case, if *tb* is None, no traceback will be shown. This is compatible with CPython 3.5 and newer.

**Parameters**

• **exc** – The exception. Must be an instance of *BaseException*. Unused if value is specified.

• **value** – If specified, is used in place of *exc*.

• **tb** – When value is also specified, *tb* is used in place of the exception’s own traceback. If None, the traceback will not be printed.

• **limit** (*int*) – Print up to limit stack trace entries (starting from the caller’s frame) if limit is positive. Otherwise, print the last abs(limit) entries. If limit is omitted or None, all entries are printed.

• **file** (*io.FileIO*) – If file is omitted or None, the output goes to *sys.stderr*; otherwise it should be an open file or file-like object to receive the output.

• **chain** (*bool*) – If True then chained exceptions will be printed.

**uheap** – Heap size analysis

**uheap.info**(*object: info.object*) → int

Prints memory debugging info for the given object and returns the estimated size.

**ulab** – Manipulate numeric data similar to numpy

**ulab** is a numpy-like module for micropython, meant to simplify and speed up common mathematical operations on arrays. The primary goal was to implement a small subset of numpy that might be useful in the context of a microcontroller. This means low-level data processing of linear (array) and two-dimensional (matrix) data.

**ulab** is adapted from micropython-ulab, and the original project’s documentation can be found at [https://micropython-ulab.readthedocs.io/en/latest/](https://micropython-ulab.readthedocs.io/en/latest/)

**ulab** is modeled after numpy, and aims to be a compatible subset where possible. Numpy’s documentation can be found at [https://docs.scipy.org/doc/numpy/index.html](https://docs.scipy.org/doc/numpy/index.html)
**ulab.np – Numerical approximation methods**

**ulab.np.carray** – Return the real part of the complex argument, which can be either an ndarray, or a scalar.

ulab.np.carray.real(val)

ulab.np.carray.imag(val)

    Return the imaginary part of the complex argument, which can be either an ndarray, or a scalar.

ulab.np.carray.conjugate(val)

    Return the conjugate of the complex argument, which can be either an ndarray, or a scalar.

ulab.np.carray.sort_complex(a: ulab.np.ndarray) → ulab.np.ndarray

    Sort a complex array using the real part first, then the imaginary part. Always returns a sorted complex array, even if the input was real.

ulab.np.carray.abs(a: ulab.np.ndarray) → ulab.np.ndarray

    Return the absolute value of complex ndarray.

**ulab.np.fft – Frequency-domain functions**

ulab.np.fft.fft(r: ulab.np.ndarray, c: ulab.np.ndarray | None = None) → Tuple[ulab.np.ndarray, ulab.np.ndarray]

    Perform a Fast Fourier Transform from the time domain into the frequency domain

    See also **ulab.utils.spectrogram**, which computes the magnitude of the fft, rather than separately returning its real and imaginary parts.

ulab.np.fft.ifft(r: ulab.np.ndarray, c: ulab.np.ndarray | None = None) → Tuple[ulab.np.ndarray, ulab.np.ndarray]

    Perform an Inverse Fast Fourier Transform from the frequency domain into the time domain
ulab.numpy.linalg

ulab.numpy.linalg.cholesky(A: ulab.numpy.ndarray) → ulab.numpy.ndarray

Parameters

A (ndarray) – a positive definite, symmetric square matrix

Return ~ulab.numpy.ndarray L
a square root matrix in the lower triangular form

 Raises

ValueError – If the input does not fulfill the necessary conditions

The returned matrix satisfies the equation m=LL*

ulab.numpy.linalg.det(m: ulab.numpy.ndarray) → float

Param

m, a square matrix

Return float

The determinant of the matrix

Computes the eigenvalues and eigenvectors of a square matrix

ulab.numpy.linalg.eig(m: ulab.numpy.ndarray) → Tuple[ulab.numpy.ndarray, ulab.numpy.ndarray]

Parameters

m – a square matrix

Return tuple (eigenvectors, eigenvalues)

Computes the eigenvalues and eigenvectors of a square matrix

ulab.numpy.linalg.inv(m: ulab.numpy.ndarray) → ulab.numpy.ndarray

Parameters

m (ndarray) – a square matrix

Returns

The inverse of the matrix, if it exists

 Raises

ValueError – if the matrix is not invertible

Computes the inverse of a square matrix

ulab.numpy.linalg.norm(x: ulab.numpy.ndarray) → float

Parameters

x (ndarray) – a vector or a matrix

Computes the 2-norm of a vector or a matrix, i.e., sqrt(sum(x*x)), however, without the RAM overhead.

ulab.numpy.linalg.qr(m: ulab.numpy.ndarray) → Tuple[ulab.numpy.ndarray, ulab.numpy.ndarray]

Parameters

m – a matrix

Return tuple (Q, R)

Factor the matrix a as QR, where Q is orthonormal and R is upper-triangular.
ulab.numpy.interp(x: ndarray, xp: ndarray, fp: ndarray, *, left: _float | None = None, right: _float | None = None) → ndarray

Parameters

- x (ulab.numpy.ndarray) – The x-coordinates at which to evaluate the interpolated values.
- xp (ulab.numpy.ndarray) – The x-coordinates of the data points, must be increasing
- fp (ulab.numpy.ndarray) – The y-coordinates of the data points, same length as xp
- left – Value to return for $x < xp[0]$, default is $fp[0]$.
- right – Value to return for $x > xp[-1]$, default is $fp[-1]$.

Returns the one-dimensional piecewise linear interpolant to a function with given discrete data points (xp, fp), evaluated at x.

ulab.numpy.trapz(y: ndarray, x: ndarray | None = None, dx: _float = 1.0) → _float

Parameters

- y (1D ulab.numpy.ndarray) – the values of the dependent variable
- x (1D ulab.numpy.ndarray) – optional, the coordinates of the independent variable. Defaults to uniformly spaced values.
- dx (float) – the spacing between sample points, if x=None

Returns the integral of y(x) using the trapezoidal rule.

ulab.numpy.arange(stop: _float, step: _float = 1, *, dtype: _DType = ulab.numpy.float) → ndarray
ulab.numpy.arange(start: _float, stop: _float, step: _float = 1, *, dtype: _DType = ulab.numpy.float) → ndarray

Return a new 1-D array with elements ranging from start to stop, with step size step.

ulab.numpy.concatenate(arrays: Tuple[ndarray], *, axis: int = 0) → ndarray

Join a sequence of arrays along an existing axis.

ulab.numpy.diag(a: ndarray, *, k: int = 0) → ndarray

Return specified diagonals.

ulab.numpy.empty(shape: int | Tuple[int, Ellipsis], *, dtype: _DType = ulab.numpy.float) → ndarray

Return a new array of the given shape with all elements set to 0. An alias for numpy.zeros.

ulab.numpy.eye(size: int, *, M: int | None = None, k: int = 0, dtype: _DType = ulab.numpy.float) → ndarray

Return a new square array of size, with the diagonal elements set to 1 and the other elements set to 0. If k is given, the diagonal is shifted by the specified amount.

ulab.numpy.full(shape: int | Tuple[int, Ellipsis], fill_value: _float | _bool, *, dtype: _DType = ulab.numpy.float) → ndarray

Return a new array of the given shape with all elements set to 0.


Return a new 1-D array with num elements ranging from start to stop linearly.


Return a new 1-D array with num evenly spaced elements on a log scale. The sequence starts at base ** start, and ends with base ** stop.
**ulab.numpy.ones**

Return a new array of the given shape with all elements set to 1.

**ulab.numpy.zeros**

Return a new array of the given shape with all elements set to 0.

**ulab.numpy._ArrayLike**

**ulab.numpy._DType**

- **ulab.numpy.int8**
  - Type code for signed integers in the range -128 .. 127 inclusive, like the ‘b’ typecode of `array.array`

- **ulab.numpy.uint8**
  - Type code for unsigned integers in the range 0 .. 255 inclusive, like the ‘H’ typecode of `array.array`

- **ulab.numpy.int16**
  - Type code for signed integers in the range -32768 .. 32767 inclusive, like the ‘h’ typecode of `array.array`

- **ulab.numpy.uint16**
  - Type code for unsigned integers in the range 0 .. 65535 inclusive, like the ‘h’ typecode of `array.array`

- **ulab.numpy.float**
  - Type code for floating point values, like the ‘f’ typecode of `array.array`

- **ulab.numpy.bool**
  - Type code for boolean values

**ulab.numpy.argmax**

Return the index of the maximum element of the 1D array

**ulab.numpy.argmin**

Return the index of the minimum element of the 1D array

**ulab.numpy.argsort**

Returns an array which gives indices into the input array from least to greatest.

**ulab.numpy.cross**

Return the cross product of two vectors of length 3

**ulab.numpy.diff**

Return the numerical derivative of successive elements of the array, as an array. axis=None is not supported.

**ulab.numpy.flip**

Returns a new array that reverses the order of the elements along the given axis, or along all axes if axis is None.

**ulab.numpy.max**

Return the maximum element of the 1D array

**ulab.numpy.mean**

Return the mean element of the 1D array, as a number if axis is None, otherwise as an array.

**ulab.numpy.median**

Find the median value in an array along the given axis, or along all axes if axis is None.
ulab.numpy.min(array: _ArrayLike, *, axis: int | None = None) → float
Return the minimum element of the 1D array

ulab.numpy.roll(array: ndarray, distance: int, *, axis: int | None = None) → None
Shift the content of a vector by the positions given as the second argument. If the axis keyword is supplied, the shift is applied to the given axis. The array is modified in place.

ulab.numpy.sort(array: ndarray, *, axis: int = -1) → ndarray
Sort the array along the given axis, or along all axes if axis is None. The array is modified in place.

ulab.numpy.std(array: _ArrayLike, *, axis: int | None = None, ddof: int = 0) → float
Return the standard deviation of the array, as a number if axis is None, otherwise as an array.

ulab.numpy.sum(array: _ArrayLike, *, axis: int | None = None) → float | int | ndarray
Return the sum of the array, as a number if axis is None, otherwise as an array.

class ulab.numpy.ndarray

ulab.numpy.get_printoptions() → Dict[str, int]
Get printing options

ulab.numpy.set_printoptions(threshold: int | None = None, edgeitems: int | None = None) → None
Set printing options

ulab.numpy.ndinfo(array: ndarray) → None

ulab.numpy.array(values: ndarray | Iterable[float | bool | Iterable[Any]], *, dtype: _DType = ulab.numpy.float) → ndarray
alternate constructor function for ulab.numpy.ndarray. Mirrors numpy.array

ulab.numpy.trace(m: ndarray) → float
Parameters
m – a square matrix
Compute the trace of the matrix, the sum of its diagonal elements.

ulab.numpy.dot(m1: ndarray, m2: ndarray) → ndarray | float
Parameters
• m1 (ndarray) – a matrix, or a vector
• m2 (ndarray) – a matrix, or a vector
Computes the product of two matrices, or two vectors. In the letter case, the inner product is returned.

ulab.numpy.acos(a: _ArrayLike) → ndarray
Computes the inverse cosine function

ulab.numpy.acosh(a: _ArrayLike) → ndarray
Computes the inverse hyperbolic cosine function

ulab.numpy.asin(a: _ArrayLike) → ndarray
Computes the inverse sine function

ulab.numpy.asinh(a: _ArrayLike) → ndarray
Computes the inverse hyperbolic sine function
ulab.numpy.around\(a: \_\text{ArrayLike}, *, \text{decimals: int} = 0\) → ndarray

Returns a new float array in which each element is rounded to decimals places.

ulab.numpy.atan\(a: \_\text{ArrayLike}\) → ndarray

Computes the inverse tangent function; the return values are in the range \([-\pi/2, \pi/2]\).

ulab.numpy.arctan2\(ya: \_\text{ArrayLike}, xa: \_\text{ArrayLike}\) → ndarray

Computes the inverse tangent function of \(y/x\); the return values are in the range \([-\pi, \pi]\).

ulab.numpy.atanh\(a: \_\text{ArrayLike}\) → ndarray

Computes the inverse hyperbolic tangent function.

ulab.numpy.ceil\(a: \_\text{ArrayLike}\) → ndarray

Rounds numbers up to the next whole number.

ulab.numpy.cos\(a: \_\text{ArrayLike}\) → ndarray

Computes the cosine function.

ulab.numpy.cosh\(a: \_\text{ArrayLike}\) → ndarray

Computes the hyperbolic cosine function.

ulab.numpy.degrees\(a: \_\text{ArrayLike}\) → ndarray

Converts angles from radians to degrees.

ulab.numpy.erf\(a: \_\text{ArrayLike}\) → ndarray

Computes the error function, which has applications in statistics.

ulab.numpy.erfc\(a: \_\text{ArrayLike}\) → ndarray

Computes the complementary error function, which has applications in statistics.

ulab.numpy.exp\(a: \_\text{ArrayLike}\) → ndarray

Computes the exponent function.

ulab.numpy.expm1\(a: \_\text{ArrayLike}\) → ndarray

Computes \(e^x-1\). In certain applications, using this function preserves numeric accuracy better than the \(\text{exp}\) function.

ulab.numpy.floor\(a: \_\text{ArrayLike}\) → ndarray

Rounds numbers up to the next whole number.

ulab.numpy.gamma\(a: \_\text{ArrayLike}\) → ndarray

Computes the gamma function.

ulab.numpy.lgamma\(a: \_\text{ArrayLike}\) → ndarray

Computes the natural log of the gamma function.

ulab.numpy.log\(a: \_\text{ArrayLike}\) → ndarray

Computes the natural log.

ulab.numpy.log10\(a: \_\text{ArrayLike}\) → ndarray

Computes the log base 10.

ulab.numpy.log2\(a: \_\text{ArrayLike}\) → ndarray

Computes the log base 2.

ulab.numpy.radians\(a: \_\text{ArrayLike}\) → ndarray

Converts angles from degrees to radians.
ulab.numpy.sin(a: _ArrayLike) → ndarray
Computes the sine function

ulab.numpy.sinc(a: _ArrayLike) → ndarray
Computes the normalized sinc function

ulab.numpy.sinh(a: _ArrayLike) → ndarray
Computes the hyperbolic sine

ulab.numpy.sqrt(a: _ArrayLike) → ndarray
Computes the square root

ulab.numpy.tan(a: _ArrayLike) → ndarray
Computes the tangent

ulab.numpy.tanh(a: _ArrayLike) → ndarray
Computes the hyperbolic tangent

ulab.numpy.vectorize(f: Callable[[int], float] | Callable[[float], float], *, otypes: _DType | None = None) → Callable[[_ArrayLike], ndarray]

Parameters

• f (callable) – The function to wrap
• otypes – List of array types that may be returned by the function. None is interpreted to mean the return value is float.

Wrap a Python function f so that it can be applied to arrays. The callable must return only values of the types specified by otypes, or the result is undefined.

ulab.scipy – Compatibility layer for scipy

ulab.scipy.linalg

ulab.scipy.linalg.solve_triangular(A: ulab.numpy.ndarray, b: ulab.numpy.ndarray, lower: bool) → ulab.numpy.ndarray

Parameters

• A (ndarray) – a matrix
• b (ndarray) – a vector
• lower (~bool) – if true, use only data contained in lower triangle of A, else use upper triangle of A

Returns
solution to the system A x = b. Shape of return matches b

Raises
• TypeError – if A and b are not of type ndarray and are not dense
• ValueError – if A is a singular matrix

Solve the equation A x = b for x, assuming A is a triangular matrix
ulab.scipy.linalg.cho_solve(L: ulab.numpy.ndarray, b: ulab.numpy.ndarray) → ulab.numpy.ndarray

**Parameters**
- L (ndarray) – the lower triangular, Cholesky factorization of A
- b (ndarray) – right-hand-side vector b

**Returns**
- solution to the system A x = b. Shape of return matches b

**Raises**
- TypeError – if L and b are not of type ndarray and are not dense

Solve the linear equations A x = b, given the Cholesky factorization of A as input

ulab.scipy.optimize

ulab.scipy.optimize.bisect(fun: Callable[[float], float], a: float, b: float, *, xtol: float = 2.4e-07, maxiter: int = 100) → float

**Parameters**
- f (callable) – The function to bisect
- a (float) – The left side of the interval
- b (float) – The right side of the interval
- xtol (float) – The tolerance value
- maxiter (float) – The maximum number of iterations to perform

Find a solution (zero) of the function f(x) on the interval (a..``b``) using the bisection method. The result is accurate to within xtol unless more than maxiter steps are required.

ulab.scipy.optimize.fmin(fun: Callable[[float], float], x0: float, *, xatol: float = 2.4e-07, fatol: float = 2.4e-07, maxiter: int = 200) → float

**Parameters**
- f (callable) – The function to bisect
- x0 (float) – The initial x value
- xatol (float) – The absolute tolerance value
- fatol (float) – The relative tolerance value

Find a minimum of the function f(x) using the downhill simplex method. The located x is within fxtol of the actual minimum, and f(x) is within fatol of the actual minimum unless more than maxiter steps are required.

ulab.scipy.optimize.newton(fun: Callable[[float], float], x0: float, *, xtol: float = 2.4e-07, rtol: float = 0.0, maxiter: int = 50) → float

**Parameters**
- f (callable) – The function to bisect
- x0 (float) – The initial x value
- xtol (float) – The absolute tolerance value
- rtol (float) – The relative tolerance value
• **maxiter** (*float*) – The maximum number of iterations to perform

Find a solution (zero) of the function \( f(x) \) using Newton’s Method. The result is accurate to within \( \text{xtol} \times \text{rtol} \times |f(x)| \) unless more than \( \text{maxiter} \) steps are required.

**ulab.user** – This module should hold arbitrary user-defined functions.

**ulab.utils**

**ulab.utils.spectrogram**(*r: ulab.numpy.ndarray*) \( \rightarrow \) *ulab.numpy.ndarray*

**Parameters**

• **r** (*ulab.numpy.ndarray*) – A 1-dimension array of values whose size is a power of 2

Computes the spectrum of the input signal. This is the absolute value of the (complex-valued) fft of the signal. This function is similar to scipy's `scipy.signal.welch` [https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.welch.html).

**usb** – PyUSB-compatible USB host API

The **usb** is a subset of PyUSB that allows you to communicate to USB devices.

**usb.core** – USB Core

This is a subset of the PyUSB core module.

**exception** **usb.core.USBErr**

**exception** **usb.core.USBT**

**exception** **usb.core.USBT**

**usb.core.find**(*find_all: bool = False, *, idVendor: int | None = None, idProduct: int | None = None*) \( \rightarrow \) **Device**

Find the first device that matches the given requirements or, if **find_all** is True, return a generator of all matching devices.

Returns None if no device matches.

**class** **usb.core.Device**

User code cannot create Device objects. Instead, get them from **usb.core.find**.

**idVendor**: int

The USB vendor ID of the device

**idProduct**: int

The USB product ID of the device
serial_number: str
The USB device’s serial number string.

product: str
The USB device’s product string.

manufacturer: str
The USB device’s manufacturer string.

write(endpoint: int, data: circuitpython_typing.ReadableBuffer, timeout: int | None = None) → int
Write data to a specific endpoint on the device.

Parameters

• endpoint (int) – the bEndpointAddress you want to communicate with.
• data (ReadableBuffer) – the data to send
• timeout (int) – Time to wait specified in milliseconds. (Different from most Circuit-Python!)

Returns
the number of bytes written

read(endpoint: int, size_or_buffer: array.array, timeout: int | None = None) → int
Read data from the endpoint.

Parameters

• endpoint (int) – the bEndpointAddress you want to communicate with.
• size_or_buffer (array.array) – the array to read data into. PyUSB also allows size but CircuitPython only support array to force deliberate memory use.
• timeout (int) – Time to wait specified in milliseconds. (Different from most Circuit-Python!)

Returns
the number of bytes read

ctrl_transfer(bmRequestType: int, bRequest: int, wValue: int = 0, wIndex: int = 0, data_or_wLength: array.array | None = None, timeout: int | None = None) → int
Do a control transfer on the endpoint 0. The parameters bmRequestType, bRequest, wValue and wIndex are the same of the USB Standard Control Request format.

Control requests may or may not have a data payload to write/read. In cases which it has, the direction bit of the bmRequestType field is used to infer the desired request direction.

For host to device requests (OUT), data_or_wLength parameter is the data payload to send, and it must be a sequence type convertible to an array object. In this case, the return value is the number of bytes written in the data payload.

For device to host requests (IN), data_or_wLength is an array object which the data will be read to, and the return value is the number of bytes read.

is_kernel_driver_active(interface: int) → bool
Determine if CircuitPython is using the interface. If it is, the object will be unable to perform I/O.

Parameters

• interface (int) – the device interface number to check
**detach_kernel_driver** (*interface: int*) → None

Stop CircuitPython from using the interface. If successful, you will then be able to perform I/O. CircuitPython will automatically re-start using the interface on reload.

**Parameters**

- **interface** (*int*) – the device interface number to stop CircuitPython on

**attach_kernel_driver** (*interface: int*) → None

Allow CircuitPython to use the interface if it wants to.

**Parameters**

- **interface** (*int*) – the device interface number to allow CircuitPython to use

---

**usb_cdc – USB CDC Serial streams**

The `usb_cdc` module allows access to USB CDC (serial) communications.

On Windows, each `Serial` is visible as a separate COM port. The ports will often be assigned consecutively, `console` first, but this is not always true.

On Linux, the ports are typically `/dev/ttyACM0` and `/dev/ttyACM1`. The `console` port will usually be first.

On MacOS, the ports are typically `/dev/cu.usbmodem<something>`. The something varies based on the USB bus and port used. The `console` port will usually be first.

**usb_cdc.console**: `Serial` | `None`

The `console` `Serial` object is used for the REPL, and for `sys.stdin` and `sys.stdout`. `console` is `None` if disabled.

However, note that `sys.stdin` and `sys.stdout` are text-based streams, and the `console` object is a binary stream. You do not normally need to write to `console` unless you want to write binary data.

**usb_cdc.data**: `Serial` | `None`

A `Serial` object that can be used to send and receive binary data to and from the host. Note that `data is disabled` by default. `data` is `None` if disabled.

**usb_cdc.disable()** → `None`

Do not present any USB CDC device to the host. Can be called in `boot.py`, before USB is connected. Equivalent to `usb_cdc.enable(console=False, data=False)`.

**usb_cdc.enable(*, console: bool = True, data: bool = False)** → `None`

Enable or disable each CDC device. Can be called in `boot.py`, before USB is connected.

**Parameters**

- **bool (data)** – Enable or disable the `console` USB serial device. True to enable; False to disable. Enabled by default.
- **bool** – Enable or disable the `data` USB serial device. True to enable; False to disable. `Disabled` by default.

If you enable too many devices at once, you will run out of USB endpoints. The number of available endpoints varies by microcontroller. CircuitPython will go into safe mode after running `boot.py` to inform you if not enough endpoints are available.

**class** `usb_cdc.Serial`

Receives cdc commands over USB

You cannot create an instance of `usb_cdc.Serial`. The available instances are in the `usb_cdc.serials` tuple.
connected:  bool
    True if this Serial is connected to a host. (read-only)

Note: The host is considered to be connected if it is asserting DTR (Data Terminal Ready). Most terminal
programs and pyserial assert DTR when opening a serial connection. However, the C# SerialPort
API does not. You must set SerialPort.DtrEnable.

in_waiting:  int
    Returns the number of bytes waiting to be read on the USB serial input. (read-only)

out_waiting:  int
    Returns the number of bytes waiting to be written on the USB serial output. (read-only)

timeout:  float | None
    The initial value of timeout is None. If None, wait indefinitely to satisfy the conditions of a read operation.
    If 0, do not wait. If > 0, wait only timeout seconds.

write_timeout:  float | None
    The initial value of write_timeout is None. If None, wait indefinitely to finish writing all the bytes passed
to write(). If 0, do not wait. If > 0, wait only write_timeout seconds.

read(size:  int = 1) → bytes
    Read at most size bytes. If size exceeds the internal buffer size only the bytes in the buffer will be read.
    If timeout is > 0 or None, and fewer than size bytes are available, keep waiting until the timeout expires
    or size bytes are available.

    Returns
    Data read

    Return type
    bytes

readinto(buf:  circuitpython_typing.WriteableBuffer) → int
    Read bytes into the buf. Read at most len(buf) bytes. If timeout is > 0 or None, keep waiting until the
    timeout expires or len(buf) bytes are available.

    Returns
    number of bytes read and stored into buf

    Return type
    int

readline(size:  int = -1) → bytes | None
    Read a line ending in a newline character ("\n"), including the newline. Return everything readable if no
    newline is found and timeout is 0. Return None in case of error.

    This is a binary stream: the newline character “\n” cannot be changed. If the host computer transmits “\r”
    it will also be included as part of the line.

    Parameters
    size (int) – maximum number of characters to read. -1 means as many as possible.

    Returns
    the line read

    Return type
    bytes or None
readlines() → List[bytes | None]
Read multiple lines as a list, using readline().

**Warning:** If timeout is None, readlines() will never return, because there is no way to indicate end of stream.

Returns
a list of the line read

Return type
list

write(buf: circuitpython_typing.ReadableBuffer) → int
Write as many bytes as possible from the buffer of bytes.

Returns
the number of bytes written

Return type
int

flush() → None
Force out any unwritten bytes, waiting until they are written.

reset_input_buffer() → None
Clears any unread bytes.

reset_output_buffer() → None
Clears any unwritten bytes.

usb_hid – USB Human Interface Device

The `usb_hid` module allows you to output data as a HID device.

usb_hid.devices: Tuple[Device, Ellipsis]
Tuple of all active HID device interfaces. The default set of devices is Device.KEYBOARD, Device.MOUSE, Device.CONSUMER_CONTROL, On boards where `usb_hid` is disabled by default, devices is an empty tuple.

If a boot device is enabled by `usb_hid.enable()`, and the host has requested a boot device, the devices tuple is replaced when `code.py` starts with a single-element tuple containing a `Device` that describes the boot device chosen (keyboard or mouse). The request for a boot device overrides any other HID devices.

usb_hid.disable() → None
Do not present any USB HID devices to the host computer. Can be called in boot.py, before USB is connected. The HID composite device is normally enabled by default, but on some boards with limited endpoints, including STM32F4, it is disabled by default. You must turn off another USB device such as `usb_cdc` or `storage` to free up endpoints for use by `usb_hid`.

usb_hid.enable(devices: Sequence[Device] | None, boot_device: int = 0) → None
Specify which USB HID devices that will be available. Can be called in boot.py, before USB is connected.

Parameters
- **devices (Sequence) − Device objects**. If `devices` is empty, HID is disabled. The order of the Devices may matter to the host. For instance, for MacOS, put the mouse device before any Gamepad or Digitizer HID device or else it will not work.
• **boot_device** *(int)* – If non-zero, inform the host that support for a a boot HID device is available. If `boot_device=1`, a boot keyboard is available. If `boot_device=2`, a boot mouse is available. No other values are allowed. See below.

If you enable too many devices at once, you will run out of USB endpoints. The number of available endpoints varies by microcontroller. CircuitPython will go into safe mode after running `boot.py` to inform you if not enough endpoints are available.

**Boot Devices**

Boot devices implement a fixed, predefined report descriptor, defined in https://www.usb.org/sites/default/files/hid1_12.pdf, Appendix B. A USB host can request to use the boot device if the USB device says it is available. Usually only a BIOS or other kind of limited-functionality host needs boot keyboard support.

For example, to make a boot keyboard available, you can use this code:

```python
usb_hid.enable((Device.KEYBOARD), boot_device=1)  # 1 for a keyboard
```

If the host requests the boot keyboard, the report descriptor provided by `Device.KEYBOARD` will be ignored, and the predefined report descriptor will be used. But if the host does not request the boot keyboard, the descriptor provided by `Device.KEYBOARD` will be used.

The HID boot device must usually be the first or only device presented by CircuitPython. The HID device will be USB interface number 0. To make sure it is the first device, disable other USB devices, including CDC and MSC (CIRCUITPY). If you specify a non-zero `boot_device`, and it is not the first device, CircuitPython will enter safe mode to report this error.

```python
usb_hid.get_boot_device() → int
```

Returns

the boot device requested by the host, if any. Returns 0 if the host did not request a boot device, or if `usb_hid.enable()` was called with `boot_device=0`, the default, which disables boot device support. If the host did request a boot device, returns the value of `boot_device` set in `usb_hid.enable()`: 1 for a boot keyboard, or 2 for boot mouse. However, the standard devices provided by CircuitPython, `Device.KEYBOARD` and `Device.MOUSE`, describe reports that match the boot device reports, so you don’t need to check this if you are using those devices.

**Rtype int**


HID Device specification

Create a description of a USB HID device. The actual device is created when you pass a `Device` to `usb_hid.enable()`.

**Parameters**

• **report_descriptor** *(ReadableBuffer)* – The USB HID Report descriptor bytes. The descriptor is not verified for correctness; it is up to you to make sure it is not malformed.

• **usage_page** *(int)* – The Usage Page value from the descriptor. Must match what is in the descriptor.

• **usage** *(int)* – The Usage value from the descriptor. Must match what is in the descriptor.

• **report_ids** *(Sequence[int])* – Sequence of report ids used by the descriptor. If the `report_descriptor` does not specify any report IDs, use *(0,)*.
• **in_report_lengths** *(Sequence[int]*) – Sequence of sizes in bytes of the HID reports sent to the host. The sizes are in order of the report_ids. Use a size of 0 for a report that is not an IN report. “IN” is with respect to the host.

• **out_report_lengths** *(int)* – Sequence of sizes in bytes of the HID reports received from the host. The sizes are in order of the report_ids. Use a size of 0 for a report that is not an OUT report. “OUT” is with respect to the host.

report_ids, in_report_lengths, and out_report_lengths must all have the same number of elements.

Here is an example of a `Device` with a descriptor that specifies two report IDs, 3 and 4. Report ID 3 sends an IN report of length 5, and receives an OUT report of length 6. Report ID 4 sends an IN report of length 2, and does not receive an OUT report:

```python
device = usb_hid.Device(
    descriptor=b"...",  # Omitted for brevity.
    report_ids=(3, 4),
    in_report_lengths=(5, 2),
    out_report_lengths=(6, 0),
)
```

**KEYBOARD:** *Device*
Standard keyboard device supporting keycodes 0x00-0xDD, modifiers 0xE-0xE7, and five LED indicators. Uses Report ID 1 for its IN and OUT reports.

**MOUSE:** *Device*
Standard mouse device supporting five mouse buttons, X and Y relative movements from -127 to 127 in each report, and a relative mouse wheel change from -127 to 127 in each report. Uses Report ID 2 for its IN report.

**CONSUMER_CONTROL:** *Device*
Consumer Control device supporting sent values from 1-652, with no rollover. Uses Report ID 3 for its IN report.

**usage_page:** *int*
The device usage page identifier, which designates a category of device. (read-only)

**usage:** *int*
The device usage identifier, which designates a specific kind of device. (read-only)

For example, Keyboard is 0x06 within the generic desktop usage page 0x01. Mouse is 0x02 within the same usage page.

**send_report**(report: `circuitpython_typing.ReadableBuffer`, report_id: *int | None = None*) → *None*
Send an HID report. If the device descriptor specifies zero or one report id’s, you can supply `None` (the default) as the value of report_id. Otherwise you must specify which report id to use when sending the report.

**get_last_received_report**(report_id: *int | None = None*) → *bytes | None*
Get the last received HID OUT or feature report for the given report ID. The report ID may be omitted if there is no report ID, or only one report ID. Return `None` if nothing received. After returning a report, subsequent calls will return `None` until next report is received.
usb_host – USB Host

The `usb_host` module allows you to manage USB host ports. To communicate with devices use the `usb` module that is a subset of PyUSB’s API.

```python
class usb_host.Port(dp: microcontroller.Pin, dm: microcontroller.Pin)
    USB host port. Also known as a root hub port.

    Create a USB host port on the given pins. Access attached devices through the `usb` module. Keep this object referenced while interacting with devices, otherwise they will be disconnected.

    Parameters
    • `dp` (Pin) – The data plus pin
    • `dm` (Pin) – The data minus pin

definit() → None
    Turn off the USB host port and release the pins for other use.

__enter__() → Port
    No-op used by Context Managers.

__exit__() → None
    Automatically deinitializes the hardware when exiting a context. See `Lifetime and ContextManagers` for more info.
```

usb_midi – MIDI over USB

The `usb_midi` module contains classes to transmit and receive MIDI messages over USB.

```python
usb_midi.ports: Tuple[PortIn | PortOut, Ellipsis]
    Tuple of all MIDI ports. Each item is ether `PortIn` or `PortOut`.

usb_midi.disable() → None
    Disable presenting a USB MIDI device to the host. The device is normally enabled by default, but on some boards with limited endpoints including ESP32-S2 and certain STM boards, it is disabled by default. Can be called in `boot.py`, before USB is connected.

usb_midi.enable() → None
    Enable presenting a USB MIDI device to the host. The device is enabled by default, so you do not normally need to call this function. Can be called in `boot.py`, before USB is connected.

    If you enable too many devices at once, you will run out of USB endpoints. The number of available endpoints varies by microcontroller. CircuitPython will go into safe mode after running boot.py to inform you if not enough endpoints are available.

class usb_midi.PortIn
    Receives midi commands over USB

    You cannot create an instance of `usb_midi.PortIn`.

    PortIn objects are constructed for every corresponding entry in the USB descriptor and added to the `usb_midi.ports` tuple.

    `read(nbytes: int | None = None) → bytes | None`
        Read characters. If `nbytes` is specified then read at most that many bytes. Otherwise, read everything that arrives until the connection times out. Providing the number of bytes expected is highly recommended because it will be faster.
```
Returns
Data read

Return type
bytes or None

readinto(buf: circuitpython_typing.WriteableBuffer, nbytes: int | None = None) → bytes | None
Read bytes into the buf. If nbytes is specified then read at most that many bytes. Otherwise, read at most len(buf) bytes.

Returns
number of bytes read and stored into buf

Return type
bytes or None

class usb_midi.PortOut
Sends midi messages to a computer over USB
You cannot create an instance of usb_midi.PortOut.
PortOut objects are constructed for every corresponding entry in the USB descriptor and added to the usb_midi.ports tuple.

write(buf: circuitpython_typing.ReadableBuffer) → int | None
Write the buffer of bytes to the bus.

Returns
the number of bytes written

Return type
int or None

ustack – Stack information and analysis

ustack.max_stack_usage() → int
Return the maximum excursion of the stack so far.

ustack.stack_size() → int
Return the size of the entire stack. Same as in micropython.mem_info(), but returns a value instead of just printing it.

ustack.stack_usage() → int
Return how much stack is currently in use. Same as micropython.stack_use(); duplicated here for convenience.

vectorio – Lightweight 2D shapes for displays

The vectorio module provide simple filled drawing primitives for use with displayio.

group = displayio.Group()

palette = displayio.Palette(1)
palette[0] = 0x125690
circle = vectorio.Circle(pixel_shader=palette, radius=25, x=70, y=40)
group.append(circle)

(continues on next page)
rectangle = vectorio.Rectangle(pixel_shader=palette, width=40, height=30, x=55, y=45)
group.append(rectangle)

points=[(5, 5), (100, 20), (20, 20), (20, 100)]
polygon = vectorio.Polygon(pixel_shader=palette, points=points, x=0, y=0)
group.append(polygon)


Circle is positioned on screen by its center point.

Parameters

- **pixel_shader** (`Union[ColorConverter, Palette]`) – The pixel shader that produces colors from values
- **radius** (`int`) – The radius of the circle in pixels
- **x** (`int`) – Initial x position of the axis.
- **y** (`int`) – Initial y position of the axis.
- **color_index** (`int`) – Initial color_index to use when selecting color from the palette.

radius: `int`

The radius of the circle in pixels.

color_index: `int`

The color_index of the circle as 0 based index of the palette.

x: `int`

X position of the center point of the circle in the parent.

y: `int`

Y position of the center point of the circle in the parent.

hidden: `bool`

Hide the circle or not.

location: `Tuple[int, int]`

(X,Y) position of the center point of the circle in the parent.

pixel_shader: `displayio.ColorConverter | displayio.Palette`

The pixel shader of the circle.

class vectorio.Polygon(pixel_shader: displayio.ColorConverter | displayio.Palette, points: List[Tuple[int, int]], x: int, y: int)

Represents a closed shape by ordered vertices. The path will be treated as ‘closed’, the last point will connect to the first point.

Parameters

- **pixel_shader** (`Union[ColorConverter, Palette]`) – The pixel shader that produces colors from values
- **points** (`List[Tuple[int, int]]`) – Vertices for the polygon
- **x** (`int`) – Initial screen x position of the 0,0 origin in the points list.
- **y** (`int`) – Initial screen y position of the 0,0 origin in the points list.
• **color_index (int)** – Initial color_index to use when selecting color from the palette.

**points**: `List[Tuple[int, int]]`

Vertices for the polygon.

**color_index**: `int`

The color_index of the polygon as 0 based index of the palette.

**x**: `int`

X position of the 0,0 origin in the points list.

**y**: `int`

Y position of the 0,0 origin in the points list.

**hidden**: `bool`

Hide the polygon or not.

**location**: `Tuple[int, int]`

(X,Y) position of the 0,0 origin in the points list.

**pixel_shader**: `displayio.ColorConverter | displayio.Palette`

The pixel shader of the polygon.

**vectorio.Rectangle**

```python
```

Represents a rectangle by defining its bounds

**Parameters**

• **pixel_shader** (*Union[ColorConverter, Palette]*) – The pixel shader that produces colors from values

• **width (int)** – The number of pixels wide

• **height (int)** – The number of pixels high

• **x (int)** – Initial x position of the top left corner.

• **y (int)** – Initial y position of the top left corner.

• **color_index (int)** – Initial color_index to use when selecting color from the palette.

**width**: `int`

The width of the rectangle in pixels.

**height**: `int`

The height of the rectangle in pixels.

**color_index**: `int`

The color_index of the rectangle in 1 based index of the palette.

**x**: `int`

X position of the top left corner of the rectangle in the parent.

**y**: `int`

Y position of the top left corner of the rectangle in the parent.

**hidden**: `bool`

Hide the rectangle or not.
location: Tuple[int, int]
(X,Y) position of the top left corner of the rectangle in the parent.

pixel_shader: displayio.ColorConverter | displayio.Palette
The pixel shader of the rectangle.

**watchdog – Watchdog Timer**

The `watchdog` module provides support for a Watchdog Timer. This timer will reset the device if it hasn’t been fed after a specified amount of time. This is useful to ensure the board has not crashed or locked up. Note that on some platforms the watchdog timer cannot be disabled once it has been enabled.

The `WatchDogTimer` is used to restart the system when the application crashes and ends up into a non recoverable state. Once started it cannot be stopped or reconfigured in any way. After enabling, the application must “feed” the watchdog periodically to prevent it from expiring and resetting the system.

Example usage:

```python
from microcontroller import watchdog as w
from watchdog import WatchDogMode
w.timeout=2.5 # Set a timeout of 2.5 seconds
w.mode = WatchDogMode.RAISE
w.feed()
```

**exception watchdog.WatchDogTimeout**

Bases: Exception

Exception raised when the watchdog timer is set to `WatchDogMode.RAISE` and expires.

Example:

```python
import microcontroller
import watchdog
import time

wdt = microcontroller.watchdog
wdt.timeout = 5

while True:
    wdt.mode = watchdog.WatchDogMode.RAISE
    print("Starting loop -- should exit after five seconds")
    try:
        while True:
            time.sleep(10) # Also works with pass
    except watchdog.WatchDogTimeout as e:
        print("Watchdog expired")
    except Exception as e:
        print("Other exception")

print("Exited loop")
```

Initialize self. See help(type(self)) for accurate signature.

**class watchdog.WatchDogMode**

run state of the watchdog timer
Enum-like class to define the run mode of the watchdog timer.

**RAISE:** `WatchDogMode`

Raise an exception when the WatchDogTimer expires.

**Limitations:** RAISE mode is not supported on SAMD or RP2040.

**RESET:** `WatchDogMode`

Reset the system if the WatchDogTimer expires.

class `watchdog.WatchDogTimer`

Timer that is used to detect code lock ups and automatically reset the microcontroller when one is detected.

A lock up is detected when the watchdog hasn’t been fed after a given duration. So, make sure to call `feed` within the timeout.

Not currently dynamically supported. Access the sole instance through `microcontroller.watchdog`.

**timeout:** `float`

The maximum number of seconds that can elapse between calls to `feed()`

**mode:** `WatchDogMode`

The current operating mode of the WatchDogTimer `watchdog.WatchDogMode`.

Setting a WatchDogMode activates the WatchDog:

```python
import microcontroller
import watchdog

w = microcontroller.watchdog
w.timeout = 5
w.mode = watchdog.WatchDogMode.RAISE
```

Once set, the WatchDogTimer will perform the specified action if the timer expires.

**feed()** → `None`

Feed the watchdog timer. This must be called regularly, otherwise the timer will expire.

**deinit()** → `None`

Stop the watchdog timer. This may raise an error if the watchdog timer cannot be disabled on this platform.

**wifi**

The `wifi` module provides necessary low-level functionality for managing wifi connections. Use `socketpool` for communicating over the network.

**wifi.radio:** `Radio`

Wifi radio used to manage both station and AP modes. This object is the sole instance of `wifi.Radio`.

class `wifi.AuthMode`

The authentication protocols used by WiFi.

**OPEN:** `object`

Open network. No authentication required.
WEP: object
Wired Equivalent Privacy.

WPA: object
Wireless Protected Access.

WPA2: object
Wireless Protected Access 2.

WPA3: object
Wireless Protected Access 3.

PSK: object
Pre-shared Key. (password)

ENTERPRISE: object
Each user has a unique credential.

```python
class wifi.Monitor (channel: int | None = 1, queue: int | None = 128)
```
For monitoring WiFi packets.
Initialize `wifi.Monitor` singleton.

Parameters
- `channel`: `int` - The WiFi channel to scan.
- `queue`: `int` - The queue size for buffering the packet.

```python
channel: int
```
The WiFi channel to scan.

```python
queue: int
```
The queue size for buffering the packet.

```python
deinit () → None
```
De-initialize `wifi.Monitor` singleton.

```python
lost () → int
```
Returns the packet loss count. The counter resets after each poll.

```python
queued () → int
```
Returns the packet queued count.

```python
packet () → dict
```
Returns the monitor packet.

```python
class wifi.Network
```
A wifi network provided by a nearby access point.
You cannot create an instance of `wifi.Network`. They are returned by `wifi.Radio.start_scanning_networks`.

```python
ssid: str
```
String id of the network

```python
bssid: bytes
```
BSSID of the network (usually the AP’s MAC address)
rssi:  int
   Signal strength of the network

channel:  int
   Channel number the network is operating on

country:  str
   String id of the country code

authmode:  str
   String id of the authmode

class wifi.Packet
   The packet parameters.
   
   CH:  object
      The packet’s channel.

   LEN:  object
      The packet’s length.

   RAW:  object
      The packet’s payload.

   RSSI: object
      The packet’s rssi.

class wifi.Radio
   Native wifi radio.

   This class manages the station and access point functionality of the native Wifi radio.

   You cannot create an instance of wifi.Radio. Use wifi.radio to access the sole instance available.

   enabled:  bool
      True when the wifi radio is enabled. If you set the value to False, any open sockets will be closed.

   hostname:  str | circuitpython_typing.ReadableBuffer
      Hostname for wifi interface. When the hostname is altered after interface started/connected the changes would only be reflected once the interface restarts/reconnects.

   mac_address:  circuitpython_typing.ReadableBuffer
      MAC address for the station. When the address is altered after interface is connected the changes would only be reflected once the interface reconnects.

      Limitations: Not settable on RP2040 CYW43 boards, such as Pi Pico W.

   tx_power:  float
      Wifi transmission power, in dBm.

   mac_address_ap:  circuitpython_typing.ReadableBuffer
      MAC address for the AP. When the address is altered after interface is started the changes would only be reflected once the interface restarts.

      Limitations: Not settable on RP2040 CYW43 boards, such as Pi Pico W.

   ap_active:  bool
      True if running as an access point. (read-only)
connected:  bool
True if connected to an access point (read-only).

ipv4_gateway:  ipaddress.IPv4Address | None
IP v4 Address of the station gateway when connected to an access point. None otherwise. (read-only)

ipv4_gateway_ap:  ipaddress.IPv4Address | None
IP v4 Address of the access point gateway, when enabled. None otherwise. (read-only)

ipv4_subnet:  ipaddress.IPv4Address | None
IP v4 Address of the station subnet when connected to an access point. None otherwise. (read-only)

ipv4_subnet_ap:  ipaddress.IPv4Address | None
IP v4 Address of the access point subnet, when enabled. None otherwise. (read-only)

ipv4_address:  ipaddress.IPv4Address | None
IP v4 Address of the station when connected to an access point. None otherwise. (read-only)

ipv4_address_ap:  ipaddress.IPv4Address | None
IP v4 Address of the access point, when enabled. None otherwise. (read-only)

ipv4_dns:  ipaddress.IPv4Address
IP v4 Address of the DNS server to be used.

ap_info:  Network | None
Network object containing BSSID, SSID, authmode, channel, country and RSSI when connected to an access point. None otherwise.

start_scanning_networks(*, start_channel: int = 1, stop_channel: int = 11) → Iterable[Network]
Scans for available wifi networks over the given channel range. Make sure the channels are allowed in your country.

Note: In the raspberry pi port (RP2040 CYW43), start_channel and stop_channel are ignored.

stop_scanning_networks() → None
Stop scanning for Wifi networks and free any resources used to do it.

start_station() → None
Starts a Station.

stop_station() → None
Stops the Station.

start_ap(ssid: str | circuitpython_typing.ReadableBuffer, password: str | circuitpython_typing.ReadableBuffer = b'', *, channel: int = 1, authmode: Iterable[AuthMode] = (), max_connections: int | None = 4) → None
Starts running an access point with the specified ssid and password.

If channel is given, the access point will use that channel unless a station is already operating on a different channel.

If authmode is not None, the access point will use the given authentication modes. If a non-empty password is given, authmode must not include OPEN. If authmode is not given or is an empty iterable, (wifi.AuthMode.OPEN,) will be used when the password is the empty string, otherwise authmode will be (wifi.AuthMode.WPA, wifi.AuthMode.WPA2, wifi.AuthMode.PSK).
**Limitations:** On Espressif, `authmode` with a non-empty password must include `wifi.AuthMode.PSK`, and one or both of `wifi.AuthMode.WPA` and `wifi.AuthMode.WPA2`. On Pi Pico W, `authmode` is ignored; it is always `(wifi.AuthMode.WPA2, wifi.AuthMode.PSK)` with a non-empty password, or `(wifi.AuthMode.OPEN,)` when no password is given.

The length of `password` must be 8-63 characters if it is ASCII, or exactly 64 hexadecimal characters if it is the hex form of the 256-bit key.

If `max_connections` is given, the access point will allow up to that number of stations to connect.

```python
def stop_ap() -> None:
    Stop the access point.

def connect(ssid: str | circuitpython_typing.ReadableBuffer, password: str | circuitpython_typing.ReadableBuffer = b'', *, channel: int = 0, bssid: str | circuitpython_typing.ReadableBuffer | None = None, timeout: float | None = None) -> None:
    Connects to the given ssid and waits for an ip address. Reconnections are handled automatically once one connection succeeds.
    
    The length of `password` must be 0 if there is no password, 8-63 characters if it is ASCII, or exactly 64 hexadecimal characters if it is the hex form of the 256-bit key.
    
    By default, this will scan all channels and connect to the access point (AP) with the given ssid and greatest signal strength (rssi).
    
    If `channel` is non-zero, the scan will begin with the given channel and connect to the first AP with the given ssid. This can speed up the connection time significantly because a full scan doesn’t occur.
    
    If `bssid` is given and not None, the scan will start at the first channel or the one given and connect to the AP with the given `bssid` and `ssid`.

def set_ipv4_address(*, ipv4: ipaddress.IPv4Address, netmask: ipaddress.IPv4Address, gateway: ipaddress.IPv4Address, ipv4_dns: ipaddress.IPv4Address | None) -> None:
    Sets the IP v4 address of the station. Must include the netmask and gateway. DNS address is optional. Setting the address manually will stop the DHCP client.

    **Note:** In the raspberrypi port (RP2040 CYW43), the access point needs to be started before the IP v4 address can be set.

def set_ipv4_address_ap(*, ipv4: ipaddress.IPv4Address, netmask: ipaddress.IPv4Address, gateway: ipaddress.IPv4Address) -> None:
    Sets the IP v4 address of the access point. Must include the netmask and gateway.

def start_dhcp() -> None:
    Starts the station DHCP client.

def stop_dhcp() -> None:
    Stops the station DHCP client. Needed to assign a static IP address.

def start_dhcp_ap() -> None:
    Starts the access point DHCP server.

def stop_dhcp_ap() -> None:
    Stops the access point DHCP server. Needed to assign a static IP address.

def ping(ip: ipaddress.IPv4Address, *, timeout: float | None = 0.5) -> float | None:
    Ping an IP to test connectivity. Returns echo time in seconds. Returns None when it times out.
class wifi.ScannedNetworks

Iterates over all wifi.Network objects found while scanning. This object is always created by a wifi.Radio: it has no user-visible constructor.

Cannot be instantiated directly. Use wifi.Radio.start_scanning_networks.

__iter__ () → Iterator[Network]

Returns itself since it is the iterator.

__next__ () → Network

Returns the next wifi.Network. Raises StopIteration if scanning is finished and no other results are available.

zlib – zlib decompression functionality

The zlib module allows limited functionality similar to the CPython zlib library. This module allows to decompress binary data compressed with DEFLATE algorithm (commonly used in zlib library and gzip archiver). Compression is not yet implemented.

zlib.decompress(data: bytes, wbits: int | None = 0, bufsize: int | None = 0) → bytes

Return decompressed data as bytes. wbits is DEFLATE dictionary window size used during compression (8-15, the dictionary size is power of 2 of that value). Additionally, if value is positive, data is assumed to be zlib stream (with zlib header). Otherwise, if it’s negative, it’s assumed to be raw DEFLATE stream.

The wbits parameter controls the size of the history buffer (or “window size”), and what header and trailer format is expected.

Common wbits values:

• To decompress deflate format, use wbits = -15
• To decompress zlib format, use wbits = 15
• To decompress gzip format, use wbits = 31

Parameters

• data(bytes) – data to be decompressed
• wbits(int) – DEFLATE dictionary window size used during compression. See above.
• bufsize(int) – ignored for compatibility with CPython only

help() – Built-in method to provide helpful information

help(object=None)

Prints a help method about the given object. When object is none, prints general port information.
1.8.2 Supported Ports

CircuitPython supports a number of microcontroller families. Support quality for each varies depending on the active contributors for each port.

Adafruit sponsored developers are actively contributing to atmel-samd, mimxrt10xx, nrf and stm ports. They also maintain the other ports in order to ensure the boards build. Additional testing is limited.

SAMD21 and SAMD51

This port supports many development boards that utilize SAMD21 and SAMD51 chips. See https://circuitpython.org/downloads for all supported boards.

Building

For build instructions see this guide: https://learn.adafruit.com/building-circuitpython/

Debugging

For debugging instructions see this guide: https://learn.adafruit.com/debugging-the-samd21-with-gdb

Port Specific modules

CircuitPython port to Spresense

This directory contains the port of CircuitPython to Spresense. It is a compact development board based on Sony’s power-efficient multicore microcontroller CXD5602.

Board features:

- Integrated GPS
  - The embedded GNSS with support for GPS, QZSS and GLONASS enables applications where tracking is required.
- Hi-res audio output and multi mic inputs
  - Advanced 192kHz/24 bit audio codec and amplifier for audio output, and support for up to 8 mic input channels.
- Multicore microcontroller
  - Spresense is powered by Sony’s CXD5602 microcontroller (ARM® Cortex®-M4F × 6 cores), with a clock speed of 156 MHz.

Currently, Spresense port does not support Audio and Multicore.

Refer to developer.sony.com/develop/spresense/ for further information about this board.
Prerequisites

Linux

Add user to dialout group:

$ sudo usermod -a -G dialout <user-name>

Windows

Download and install USB serial driver
- CP210x USB to serial driver for Windows 7/8/8.1
- CP210x USB to serial driver for Windows 10

macOS

Download and install USB serial driver
- CP210x USB to serial driver for Mac OS X

Build instructions

Pull all submodules into your clone:

$ git submodule update --init --recursive

Build the MicroPython cross-compiler:

$ make -C mpy-cross

Change directory to cxd56:

$ cd ports/cxd56

To build circuitpython image run:

$ make BOARD=spresense

USB connection

Connect the Spresense main board to the PC via the USB cable.
Flash the bootloader

The correct bootloader is required for the Spresense board to function. 

Bootloader information:

- The bootloader has to be flashed the very first time the board is used.
- You have to accept the End User License Agreement to be able to download and use the Spresense bootloader binary.

Download the spresense binaries zip archive from: Spresense firmware v3-0-0

Extract spresense binaries in your PC to ports/spresense/spresense-exported-sdk/firmware/

To flash the bootloader run the command:

```
$ make BOARD=spresense flash-bootloader
```

Flash the circuitpython image

To flash the firmware run the command:

```
$ make BOARD=spresense flash
```

Accessing the board

Connect the Spresense extension board to the PC via the USB cable.

Once built and deployed, access the CircuitPython REPL (the Python prompt) via USB. You can run:

```
$ screen /dev/ttyACM0 115200
```

CircuitPython on Espressif SoCs

This port adds the Espressif line of SoCs to CircuitPython.

Support Status:

<table>
<thead>
<tr>
<th>SoC</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP</td>
<td>beta</td>
</tr>
<tr>
<td>ESP32-C3</td>
<td>beta</td>
</tr>
<tr>
<td>ESP32-S2</td>
<td>stable</td>
</tr>
<tr>
<td>ESP32-S3</td>
<td>beta</td>
</tr>
</tbody>
</table>
How this port is organized:

- **bindings/** contains some required bindings to the ESP-IDF for exceptions and memory.
- **boards/** contains the configuration files for each development board and breakout available on the port.
- **common-hal/** contains the port-specific module implementations, used by shared-module and shared-bindings.
- **esp-idf/** contains the Espressif IoT Development Framework installation, including all the drivers for the port.
- **peripherals/** contains peripheral setup files and peripheral mapping information, sorted by family and sub-variant. Most files in this directory can be generated with the python scripts in **tools/**.
- **supervisor/** contains port-specific implementations of internal flash, serial and USB, as well as the **port.c** file, which initializes the port at startup.
- **tools/** includes useful Python scripts for debugging and other purposes.

At the root level, refer to **mpconfigboard.h** and **mpconfigport.mk** for port specific settings and a list of enabled CircuitPython modules.

Connecting to the ESP32

The ESP32 chip itself has no USB support. On many boards there is a USB-serial adapter chip, such as a CP2102N, CP2104 or CH9102F, usually connected to the ESP32 TXD0 (GPIO1) and RXD0 (GPIO3) pins, for access to the bootloader. CircuitPython also uses this serial channel for the REPL.

Connecting to the ESP32-C3

**USB Connection:**

On ESP32-C3 REV3 chips, a USB Serial/JTAG Controller is available. Note: This USB connection cannot be used for a CIRCUITPY drive.

Depending on the board you have, the USB port may or may not be connected to native USB.

The following connections need to be made if native USB isn’t available on the USB port:

<table>
<thead>
<tr>
<th>GPIO</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>D+ (green)</td>
</tr>
<tr>
<td>18</td>
<td>D- (white)</td>
</tr>
<tr>
<td>GND</td>
<td>GND (black)</td>
</tr>
<tr>
<td>5V</td>
<td>5V (red)</td>
</tr>
</tbody>
</table>

Connect these pins using a USB adapter or breakout cable.

**UART Connection:**

A USB to UART converter can be used for connecting to ESP32-C3 to get access to the serial console and REPL and for flashing CircuitPython.

The following connections need to be made in this case:

1.8. Full Table of Contents
BLE Connection:

This feature is not yet available and currently under development.

**Connecting to the ESP32-S2**

**USB Connection:**

Depending on the board you have, the USB port may or may not be connected to native USB.

The following connections need to be made if native USB isn’t available on the USB port:

<table>
<thead>
<tr>
<th>GPIO</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>D+ (green)</td>
</tr>
<tr>
<td>19</td>
<td>D- (white)</td>
</tr>
<tr>
<td>GND</td>
<td>GND (black)</td>
</tr>
<tr>
<td>5V</td>
<td>5V (red)</td>
</tr>
</tbody>
</table>

Connect these pins using a USB adapter or breakout cable to access the CircuitPython drive.

**UART Connection:**

A USB to UART converter can be used for connecting to ESP32-S2 to get access to the serial console and REPL and for flashing CircuitPython.

The following connections need to be made in this case:

<table>
<thead>
<tr>
<th>GPIO</th>
<th>UART</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>RX</td>
</tr>
<tr>
<td>44</td>
<td>TX</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>5V</td>
<td>5V</td>
</tr>
</tbody>
</table>

**BLE Connection:**

This feature isn’t available on ESP32-S2.
Connecting to the ESP32-S3

**USB Connection:**
Depending on the board you have, the USB port may or may not be connected to native USB.
The following connections need to be made if native USB isn’t available on the USB port:

<table>
<thead>
<tr>
<th>GPIO</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>D+ (green)</td>
</tr>
<tr>
<td>19</td>
<td>D- (white)</td>
</tr>
<tr>
<td>GND</td>
<td>GND (black)</td>
</tr>
<tr>
<td>5V</td>
<td>5V (red)</td>
</tr>
</tbody>
</table>

Connect these pins using a [USB adapter](#) or [breakout cable](#) to access the CircuitPython drive.

**UART Connection:**
A [USB to UART converter](#) can be used for connecting to ESP32-S3 to get access to the serial console and REPL and for flashing CircuitPython.
The following connections need to be made in this case:

<table>
<thead>
<tr>
<th>GPIO</th>
<th>UART</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>RX</td>
</tr>
<tr>
<td>44</td>
<td>TX</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>5V</td>
<td>5V</td>
</tr>
</tbody>
</table>

**BLE Connection:**
This feature is not yet available and currently under development.

**Building and flashing**
Before building or flashing the, you must install the ESP-IDF.
Note: This must be re-done every time the ESP-IDF is updated, but not every time you build.
Run `cd ports/espressif` from `circuitpython/` to move to the espressif port root, and run:

```
./esp-idf/install.sh
```

After this initial installation, you must add the ESP-IDF tools to your path.
Note: This must be re-done every time you open a new shell environment for building or flashing.
Run `cd ports/espressif` from `circuitpython/` to move to the espressif port root, and run:

```
./esp-idf/export.sh
```

When CircuitPython updates the ESP-IDF to a new release, you may need to run this installation process again. The exact commands used may also vary based on your shell environment.

Building boards is typically done through `make BOARD=board_id`. The default port is `tty.SLAB_USBtoUART`, which will only work on certain Mac setups. On most machines, both Mac and Linux, you will need to set the port yourself.
by running `ls /dev/tty.usb*` and selecting the one that only appears when your development board is plugged in. An example make command with the port setting is as follows:

```
make BOARD=board_id PORT=/dev/tty.usbserial-1421120 flash
```

board_id is the unique board identifier in CircuitPython. It is the same as the name of the board in the boards directory.

**Debugging**

TODO: Add documentation for ESP32-C3/S3 JTAG feature.

The ESP32-S2 supports JTAG debugging over OpenOCD using a JLink or other probe hardware. The official tutorials can be found on the Espressif website [here](#), but they are mostly for the ESP32-S2 Kaluga, which has built-in debugging.

OpenOCD is automatically installed and added to your bash environment during the ESP-IDF installation and setup process. You can double check that it is installed by using `openocd --version`, as per the tutorial. Attach the JTAG probe pins according to the instructions for JTAG debugging on boards that do not contain an integrated debugger.

Once the debugger is connected physically, you must run OpenOCD with attached configuration files specifying the interface (your debugger probe) and either a target or a board (targets are for SoCs only, and can be used when a full board configuration file doesn’t exist). You can find the location of these files by checking the `OPENOCD_SCRIPTS` environmental variable by running `echo $OPENOCD_SCRIPTS`. Interfaces will be in the `interface/` directory, and targets and boards in the `target/` and `board/` directories, respectively.

Note: Unfortunately, there are no board files for the esp32-s2 other than the Kaluga, and the included `target/esp32s2.cfg` target file will not work by default on the JLink for boards like the Saola 1, as the default speed is incorrect. In addition, these files are covered under the GPL and cannot be included in CircuitPython. Thus, you must make a copy of the `esp32s2.cfg` file yourself and add the following line manually, under `transport select jtag` at the start of the file:

```
adapter_khz 1000
```

Once this is complete, your final OpenOCD command may look something like this:

```
openocd -f interface/jlink.cfg -f SOMEPATH/copied-esp32s2-saola-1.cfg
```

Where `SOMEPATH` is the location of your copied configuration file (this can be placed in the `port/boards` directory with a prefix to ignore it with `.gitignore`, for instance). Interface, target and board config files sourced from Espressif only need their paths from the `OPENOCD_SCRIPTS` location, you don’t need to include their full path. Once OpenOCD is running, connect to GDB with:

```
xtena-esp32s2-elf-gdb build-espressif_saola_1_wrover/firmware.elf
```

And follow the Espressif GDB tutorial instructions for connecting, or add them to your `gdbinit`:

```
target remote :3333
set remote hardware-watchpoint-limit 2
mon reset halt
flushregs
thb app_main
```

Chapter 1. CircuitPython
**LiteX (FPGA)**

LiteX is a Python-based System on a Chip (SoC) designer for open source supported Field Programmable Gate Array (FPGA) chips. This means that the CPU core(s) and peripherals are not defined by the physical chip. Instead, they are loaded as separate “gateware”. Once this gateware is loaded, CircuitPython can be loaded on top of it to work as expected.

**Installation**

You'll need dfu-util to install CircuitPython on the Fomu.

Make sure the foboot bootloader is updated. Instructions are here: [https://github.com/im-tomu/fomu-workshop/blob/master/docs/bootloader.rst](https://github.com/im-tomu/fomu-workshop/blob/master/docs/bootloader.rst)

Once you’ve updated the bootloader, you should know how to use dfu-util. It’s pretty easy!

To install CircuitPython do:

```bash
```

It will install and then restart. CIRCUITPY should appear as it usually does and work the same.

**CircuitPython Port To The NXP i.MX RT10xx Series**

This is a port of CircuitPython to the i.MX RT10xx series of chips.

**CircuitPython Port To The Nordic Semiconductor nRF52 Series**

This is a port of CircuitPython to the Nordic Semiconductor nRF52 series of chips.

**NOTE**: There are board-specific READMEs that may be more up to date than the generic board-neutral documentation below.

**Flash**

Some boards have UF2 bootloaders and can simply be flashed in the normal way, by copying firmware.uf2 to the BOOT drive.

For some boards, you can use the `flash` target:

```bash
make BOARD= pca10056 flash
```

**Segger Targets**

Install the necessary tools to flash and debug using Segger:

- JLink Download
- nrfjprog linux-32bit Download
- nrfjprog linux-64bit Download
- nrfjprog osx Download
nrfjprog win32 Download

note: On Linux it might be required to link SEGGER’s `libjlinkarm.so` inside nrfjprog’s folder.

**DFU Targets**

run follow command to install adafruit-nrfutil from PyPi

```
$ pip3 install --user adafruit-nrfutil
```

**make flash** and **make sd** will not work with DFU targets. Hence, **dfu-gen** and **dfu-flash** must be used instead.

- **dfu-gen**: Generates a Firmware zip to be used by the DFU flash application.
- **dfu-flash**: Triggers the DFU flash application to upload the firmware from the generated Firmware zip file.

When enabled you have different options to test it:

- NUS Console for Linux (recommended)
- WebBluetooth REPL (experimental)

**RP2040**

This port supports many development boards that utilize RP2040 chips. See [https://circuitpython.org/downloads](https://circuitpython.org/downloads) for all supported boards.

**Building**

For build instructions see this guide: [https://learn.adafruit.com/building-circuitpython/](https://learn.adafruit.com/building-circuitpython/)

**Port Specific modules**

**Circuitpython on STM32**

This port brings the ST Microelectronics STM32 series of MCUs to Circuitpython. STM32 chips have a wide range of capability, from <$1 low power STM32F0s to dual-core STM32H7s running at 400+ MHz. Currently, only the F4, F7, and H7 families are supported, powered by the ARM Cortex M4 and M7 processors.

Refer to the ST Microelectronics website for more information on features sorted by family and individual chip lines: [st.com/en/microcontrollers-microprocessors/stm32-high-performance-mcus.html](st.com/en/microcontrollers-microprocessors/stm32-high-performance-mcus.html)

STM32 SoCs vary product-by-product in clock speed, peripheral capability, pin assignments, and their support within this port. Refer to `mpconfigport.mk` for a full list of enabled modules sorted by family.
How this port is organized:

- **boards/** contains the configuration files for each development board and breakout available on the port, as well as system files and both shared and SoC-specific linker files. Board configuration includes a pin mapping of the board, oscillator information, board-specific build flags, and setup for OLED or TFT screens where applicable.

- **common-hal/** contains the port-specific module implementations, used by shared-module and shared-bindings.

- **packages/** contains package-specific pin bindings (LQFP100, BGA216, etc)

- **peripherals/** contains peripheral setup files and peripheral mapping information, sorted by family and sub-variant. Most files in this directory can be generated with the python scripts in **tools/**.

- **st-driver/** submodule for ST HAL and LL files generated via CubeMX. Shared with TinyUSB.

- **supervisor/** contains port-specific implementations of internal flash, serial and USB, as well as the **port.c** file, which initializes the port at startup.

- **tools/** python scripts for generating peripheral and pin mapping files in **peripherals/** and **board/**.

At the root level, refer to **mpconfigboard.h** and **mpconfigport.mk** for port specific settings and a list of enabled modules.

Build instructions

Ensure your clone of CircuitPython is ready to build by following the guide on the Adafruit Website. This includes installing the toolchain, synchronizing submodules, and running **mpy-cross**.

Once the one-time build tasks are complete, you can build at any time by navigating to the port directory:

```bash
$ cd ports/stm
```

To build for a specific circuitpython board, run:

```bash
$ make BOARD=feather_stm32f405_express
```

You may also build with certain flags available in the makefile, depending on your board and development goals. The following flags would enable debug information and correct flash locations for a pre-flashed UF2 bootloader:

```bash
$ make BOARD=feather_stm32f405_express DEBUG=1 UF2_BOOTLOADER=1
```

USB connection

Connect your development board of choice to the host PC via the USB cable. Note that for most ST development boards such as the Nucleo and Discovery series, you must use a secondary OTG USB connector to access circuitpython, as the primary USB connector will be connected to a built-in ST-Link debugger rather than the chip itself.

In many cases, this ST-Link USB connector will **still need to be connected to power** for the chip to turn on - refer to your specific product manual for details.
Flash the bootloader

Most ST development boards come with a built-in STLink programming and debugging probe accessible via USB. This programmer may show up as an Mbed drive on the host PC, enabling simple drag and drop programming with a .bin file, or they may require a tool like OpenOCD or StLink-org/stlink to run flashing and debugging commands.

Many hobbyist and 3rd party development boards also expose SWD pins. These can be used with a cheap stlink debugger or other common programmers.

For non-ST products or users without a debugger, all STM32 boards in the high performance families (F4, F7 and H7) include a built-in DFU bootloader stored in ROM. This bootloader is accessed by ensuring the BOOT0 pin is held to a logic 1 and the BOOT1 pin is held to a logic 0 when the chip is reset (ST Appnote AN2606). Most chips hold BOOT low by default, so this can usually be achieved by running a jumper wire from 3.3V power to the BOOT0 pin, if it is exposed, or by flipping the appropriate switch or button as the chip is reset. Once the chip is started in DFU mode, BOOT0 no longer needs to be held high and can be released. An example is available in the Feather STM32F405 guide.

Windows users will need to install stm32cubeprog, while Mac and Linux users will need to install dfu-util with brew install dfu-util or sudo apt-get install dfu-util. More details are available in the Feather F405 guide.

Flashing the circuitpython image with DFU-Util

Ensure the board is in dfu mode by following the steps in the previous section. Then run:

```
$ make BOARD=feather_stm32F405_express flash
```

Alternatively, you can navigate to the build directory and run the raw dfu-util command:

```
dfu-util -a 0 --dfuse-address 0x08000000 -D firmware.bin
```

Accessing the board

Connecting the board to the PC via the USB cable will allow code to be uploaded to the CIRCUITPY volume.

Circuitpython exposes a CDC virtual serial connection for REPL access and debugging. Connecting to it from OSX will look something like this:

```
screen /dev/tty.usbmodem14111201 115200
```

You may also use a program like mu to assist with REPL access.

1.8.3 Troubleshooting

From time to time, an error occurs when working with CircuitPython. Here are a variety of errors that can happen, what they mean and how to fix them.
File system issues

If your host computer starts complaining that your CIRCUITPY drive is corrupted or files cannot be overwritten or deleted, then you will have to erase it completely. When CircuitPython restarts it will create a fresh empty CIRCUITPY filesystem.

Corruption often happens on Windows when the CIRCUITPY disk is not safely ejected before being reset by the button or being disconnected from USB. This can also happen on Linux and Mac OSX but it’s less likely.

Caution: To erase and re-create CIRCUITPY (for example, to correct a corrupted filesystem), follow one of the procedures below. It’s important to note that any files stored on the CIRCUITPY drive will be erased. Back up your code if possible before continuing!

REPL Erase Method

This is the recommended method of erasing your board. If you are having trouble accessing the CIRCUITPY drive or the REPL, consider first putting your board into safe mode.

To erase any board if you have access to the REPL:

1. Connect to the CircuitPython REPL using a terminal program.
2. Type `import storage` into the REPL.
3. Then, type `storage.erase_filesystem()` into the REPL.
4. The CIRCUITPY drive will be erased and the board will restart with an empty CIRCUITPY drive.

Erase File Method

If you do not have access to the REPL, you may still have options to erase your board.

The Erase CIRCUITPY Without Access to the REPL section of the Troubleshooting page in the Welcome to CircuitPython guide covers the non-REPL erase process for various boards. Visit the guide, find the process that applies to your board, and follow the instructions to erase your board.

ValueError: Incompatible .mpy file.

This error occurs when importing a module that is stored as a .mpy binary file (rather than a .py text file) that was generated by a different version of CircuitPython than the one it’s being loaded into. Most versions are compatible but, rarely they aren’t. In particular, the .mpy binary format changed between CircuitPython versions 1.x and 2.x, 2.x and 3.x, and will change again between 6.x and 7.x.

So, for instance, if you just upgraded to CircuitPython 7.x from 6.x you’ll need to download a newer version of the library that triggered the error on `import`. They are all available in the Adafruit bundle and the Community bundle. Make sure to download a version with 7.0.0 or higher in the filename.
1.8.4 Adafruit CircuitPython Libraries


1.8.5 CircuitPython Library Bundles

Many Python libraries, including device drivers, have been written for use with CircuitPython. They are maintained in separate GitHub repos, one per library.

Libraries are packaged in bundles, which are ZIP files that are snapshots in time of a group of libraries.

Adafruit sponsors and maintains several hundred libraries, packaged in the Adafruit Library Bundle. Adafruit-sponsored libraries are also available on <https://pypi.org>.

Yet other libraries are maintained by members of the CircuitPython community, and are packaged in the CircuitPython Community Library Bundle.

The Adafruit bundles are available on GitHub: <https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases>. The Community bundles are available at: <https://github.com/adafruit/CircuitPython_Community_Bundle/releases>.

More detailed information about the bundles, and download links for the latest bundles are at <https://circuitpython.org/libraries>.


Documentation for Community Libraries is not available on ReadTheDocs at this time. See the GitHub repository for each library for any included documentation.

1.8.6 Workflows

Workflows are the process used to 1) manipulate files on the CircuitPython device and 2) interact with the serial connection to CircuitPython. The serial connection is usually used to access the REPL.

Starting with CircuitPython 3.x we moved to a USB-only workflow. Prior to that, we used the serial connection alone to do the whole workflow. In CircuitPython 7.x, a BLE workflow was added with the advantage of working with mobile devices. CircuitPython 8.x added a web workflow that works over the local network (usually Wi-Fi) and a web browser. Other clients can also use the Web REST API. Boards should clearly document which workflows are supported.

Code for workflows lives in supervisor/shared.

The workflow APIs are documented here.

USB

These USB interfaces are enabled by default on boards with USB support. They are usable once the device has been plugged into a host.
CIRCUITPY drive

CircuitPython exposes a standard mass storage (MSC) interface to enable file manipulation over a standard interface. This interface works underneath the file system at the block level so using it excludes other types of workflows from manipulating the file system at the same time.

CDC serial

CircuitPython exposes one CDC USB interface for CircuitPython serial. This is a standard serial USB interface. TODO: Document how it designates itself from the user CDC.

Setting baudrate 1200 and disconnecting will reboot into a bootloader. (Used by Arduino to trigger a reset into bootloader.)

BLE

The BLE workflow is enabled for nRF boards. By default, to prevent malicious access, it is disabled. To connect to the BLE workflow, press the reset button while the status led blinks blue quickly after the safe mode blinks. The board will restart and broadcast the file transfer service UUID (0xfebb) along with the board’s Creation IDs. This public broadcast is done at a lower transmit level so the devices must be closer. On connection, the device will need to pair and bond. Once bonded, the device will broadcast whenever disconnected using a rotating key rather than a static one. Non-bonded devices won’t be able to resolve it. After connection, the central device can discover two default services. One for file transfer and one for CircuitPython specifically that includes serial characteristics.

To change the default BLE advertising name without (or before) running user code, the desired name can be put in the settings.toml file. The key is CIRCUITPY_BLE_NAME. It’s limited to approximately 30 characters depending on the port’s settings and will be truncated if longer.

File Transfer API

CircuitPython uses an open File Transfer API to enable file system access.

CircuitPython Service

The base UUID for the CircuitPython service is ADAFXXXX-4369-7263-7569-7450794686e. The XXXX is replaced by the four specific digits below. The service itself is 0001.

TX - 0002 / RX - 0003

These characteristic work just like the Nordic Uart Service (NUS) but have different UUIDs to prevent conflicts with user created NUS services.
**Version - 0100**

Read-only characteristic that returns the UTF-8 encoded version string.

**Web**

The web workflow is depends on adding Wi-Fi credentials into the settings.toml file. The keys are `CIRCUITPY_WIFI_SSID` and `CIRCUITPY_WIFI_PASSWORD`. Once these are defined, CircuitPython will automatically connect to the network and start the webserver used for the workflow. The webserver is on port 80 unless overridden by `CIRCUITPY_WEB_API_PORT`. It also enables MDNS. The name of the board as advertised to the network can be overridden by `CIRCUITPY_WEB_INSTANCE_NAME`.

Here is an example `/settings.toml`:

```toml
# To auto-connect to Wi-Fi
CIRCUITPY_WIFI_SSID="scottswifi"
CIRCUITPY_WIFI_PASSWORD="secretpassword"

# To enable modifying files from the web. Change this too!
# Leave the User field blank in the browser.
CIRCUITPY_WEB_API_PASSWORD="passw0rd"

CIRCUITPY_WEB_API_PORT=80
CIRCUITPY_WEB_INSTANCE_NAME=""
```

MDNS is used to resolve `circuitpython.local` to a device specific hostname of the form `cpy-XXXXXX.local`. The `XXXXXX` is based on network MAC address. The device also provides the MDNS service with service type `_circuitpython` and protocol `_tcp`.

Since port 80 (or the port assigned to `CIRCUITPY_WEB_API_PORT`) is used for web workflow, the `mdns` module can’t advertise an additional service on that port.

**HTTP**

The web server is HTTP 1.1 and may use chunked responses so that it doesn’t need to precompute content length.

The API generally consists of an HTTP method such as GET or PUT and a path. Requests and responses also have headers. Responses will contain a status code and status text such as `404 Not Found`. This API tries to use standard status codes to encode the status of the various operations. The Mozilla Developer Network HTTP docs are a great reference.

**Examples**

The examples use `curl`, a common command line program for issuing HTTP requests. The examples below use `circuitpython.local` as the easiest way to work. If you have multiple active devices, you’ll want to use the specific `cpy-XXXXXX.local` version.

The examples also use `password` as the password placeholder. Replace it with your password before running the example.
The root welcome page links to the file system page and also displays other CircuitPython devices found using MDNS service discovery. This allows web browsers to find other devices from one. (All devices will respond to circuitpython.local so the device redirected to may vary.)

CORS

The web server will allow requests from cpy-XXXXXX.local, 127.0.0.1, the device’s IP and code.circuitpython.org. (circuitpython.local requests will be redirected to cpy-XXXXXX.local.)

File REST API

All file system related APIs are protected by HTTP basic authentication. It is NOT secure but will hopefully prevent some griefing in shared settings. The password is sent unencrypted so do not reuse a password with something important. The user field is left blank.

The password is taken from settings.toml with the key CIRCUITPY_WEB_API_PASSWORD. If this is unset, the server will respond with 403 Forbidden. When a password is set, but not provided in a request, it will respond 401 Unauthorized.

/fs/

The /fs/ page will respond with a directory browsing HTML once authenticated. This page is always gzipped. If the Accept: application/json header is provided, then the JSON representation of the root will be returned.

OPTIONS

When requested with the OPTIONS method, the server will respond with CORS related headers. Most aren’t needed for API use. They are there for the web browser.

- Access-Control-Allow-Methods - Varies with USB state. GET, OPTIONS when USB is active. GET, OPTIONS, PUT, DELETE, MOVE otherwise.

Example:

curl -v -u :password -X OPTIONS -L --location-trusted http://circuitpython.local/fs/

/fs/<directory path>/

Directory paths must end with a /. Otherwise, the path is assumed to be a file.
GET

Returns a JSON representation of the directory.

- **200 OK** - Directory exists and JSON returned
- **401 Unauthorized** - Incorrect password
- **403 Forbidden** - No CIRCUITPY_WEB_API_PASSWORD set
- **404 Not Found** - Missing directory

Returns information about each file in the directory:

- **name** - File name. No trailing / on directory names
- **directory** - true when a directory, false otherwise
- **modified_ns** - File modification time in nanoseconds since January 1st, 1970. May not use full resolution
- **file_size** - File size in bytes. 0 for directories

Example:

```
```

```json
[
   {
      "name": "world.txt",
      "directory": false,
      "modified_ns": 946934328000000000,
      "file_size": 12
   }
]
```

PUT

Tries to make a directory at the given path. Request body is ignored. The custom X-Timestamp header can provide a timestamp in milliseconds since January 1st, 1970 (to match JavaScript’s file time resolution) used for the directories modification time. The RTC time will used otherwise.

Returns:

- **204 No Content** - Directory exists
- **201 Created** - Directory created
- **401 Unauthorized** - Incorrect password
- **403 Forbidden** - No CIRCUITPY_WEB_API_PASSWORD set
- **409 Conflict** - USB is active and preventing file system modification
- **404 Not Found** - Missing parent directory
- **500 Server Error** - Other, unhandled error

Example:
curl -v -u :passw0rd -X PUT -L --location-trusted http://circuitpython.local/fs/lib/...hello/world/

**Move**

Moves the directory at the given path to `X-Destination`. Also known as rename.

The custom `X-Destination` header stores the destination path of the directory.

- 201 Created - Directory renamed
- 401 Unauthorized - Incorrect password
- 403 Forbidden - No CIRCUITPY_WEB_API_PASSWORD set
- 404 Not Found - Source directory not found or destination path is missing
- 409 Conflict - USB is active and preventing file system modification
- 412 Precondition Failed - The destination path is already in use

Example:

```
curl -v -u :passw0rd -X MOVE -H "X-Destination: /fs/lib/hello2/" -L --location-trusted...
http://circuitpython.local/fs/lib/hello/
```

**DELETE**

Deletes the directory and all of its contents.

- 204 No Content - Directory and its contents deleted
- 401 Unauthorized - Incorrect password
- 403 Forbidden - No CIRCUITPY_WEB_API_PASSWORD set
- 404 Not Found - No directory
- 409 Conflict - USB is active and preventing file system modification

Example:

```
curl -v -u :passw0rd -X DELETE -L --location-trusted http://circuitpython.local/fs/lib/
...hello2/world/
```

`/fs/<file path>`

**PUT**

Stores the provided content to the file path.

The custom `X-Timestamp` header can provide a timestamp in milliseconds since January 1st, 1970 (to match JavaScript’s file time resolution) used for the directories modification time. The RTC time will used otherwise.

Returns:

- 201 Created - File created and saved
**GET**

Returns the raw file contents. Content-Type will be set based on extension:

- `text/plain` - `.py`, `.txt`
- `text/javascript` - `.js`
- `text/html` - `.html`
- `application/json` - `.json`
- `application/octet-stream` - Everything else

Will return:

- `200 OK` - File exists and file returned
- `401 Unauthorized` - Incorrect password
- `403 Forbidden` - No CIRCUITPY_WEB_API_PASSWORD set
- `404 Not Found` - Missing file

Example:

```
curl -v -u :passw0rd -L --location-trusted http://circuitpython.local/fs/lib/hello/world.txt
```
**Move**

Moves the file at the given path to the X-Destination. Also known as rename. The custom X-Destination header stores the destination path of the file.

- **201 Created** - File renamed
- **401 Unauthorized** - Incorrect password
- **403 Forbidden** - No CIRCUITPY_WEB_API_PASSWORD set
- **404 Not Found** - Source file not found or destination path is missing
- **409 Conflict** - USB is active and preventing file system modification
- **412 Precondition Failed** - The destination path is already in use

Example:

```
```

**DELETE**

Deletes the file.

- **204 No Content** - File existed and deleted
- **401 Unauthorized** - Incorrect password
- **403 Forbidden** - No CIRCUITPY_WEB_API_PASSWORD set
- **404 Not Found** - File not found
- **409 Conflict** - USB is active and preventing file system modification

Example:

```
curl -v -u :passw0rd -X DELETE -L --location-trusted http://circuitpython.local/fs/lib/hello/world2.txt
```

/\cp/

/\cp/ serves basic info about the CircuitPython device and others discovered through MDNS. It is not protected by basic auth in case the device is someone else's.

Only GET requests are supported and will return 405 Method Not Allowed otherwise.
/cp/devices.json

Returns information about other devices found on the network using MDNS.

- **total**: Total MDNS response count. May be more than in `devices` if internal limits were hit.
- **devices**: List of discovered devices.
  - **hostname**: MDNS hostname
  - **instance_name**: MDNS instance name. Defaults to human readable board name.
  - **port**: Port of CircuitPython Web API
  - **ip**: IP address

Example:

```
curl -v -L http://circuitpython.local/cp/devices.json
```

```json
{
    "total": 1,
    "devices": [
        {
            "hostname": "cpy-951032",
            "instance_name": "Adafruit Feather ESP32-S2 TFT",
            "port": 80,
            "ip": "192.168.1.235"
        }
    ]
}
```

/cp/serial/

Serves a basic serial terminal program when a GET request is received without the `Upgrade: websocket` header. Otherwise the socket is upgraded to a WebSocket. See WebSockets below for more detail.

This is an authenticated endpoint in both modes.

/cp/version.json

Returns information about the device.

- **web_api_version**: Always 1. This versions the rest of the API and new versions may not be backwards compatible.
- **version**: CircuitPython build version.
- **build_date**: CircuitPython build date.
- **board_name**: Human readable name of the board.
- **mcu_name**: Human readable name of the microcontroller.
- **board_id**: Board id used in code and on circuitpython.org.
- **creator_id**: Creator ID for the board.
- **creation_id**: Creation ID for the board, set by the creator.
hostname: MDNS hostname.

port: Port of CircuitPython Web Service.

ip: IP address of the device.

Example:

curl -v -L http://circuitpython.local/cp/version.json

{
    "web_api_version": 1,
    "version": "8.0.0-alpha.1-20-geld4518a9-dirty",
    "build_date": "2022-06-24",
    "board_name": "ESP32-S3-USB-OTG-N8",
    "mcu_name": "ESP32S3",
    "board_id": "espressif.esp32s3_usb_otg_n8",
    "creator_id": 12346,
    "creation_id": 28683,
    "hostname": "cpy-f57ce8",
    "port": 80,
    "ip": "192.168.1.94"
}

/code/

The /code/ page returns a small static html page that will pull in and load the full code editor from code.circuitpython.org for a full code editor experience. Because most of the resources reside online instead of the device, an active internet connection is required.

Static files

- /favicon.ico - Blinka
- /directory.js - JavaScript for /fs/
- /welcome.js - JavaScript for /

WebSocket

The CircuitPython serial interactions are available over a WebSocket. A WebSocket begins as a special HTTP request that gets upgraded to a WebSocket. Authentication happens before upgrading.

WebSockets are not bare sockets once upgraded. Instead they have their own framing format for data. CircuitPython can handle PING and CLOSE opcodes. All others are treated as TEXT. Data to CircuitPython is expected to be masked UTF-8, as the spec requires. Data from CircuitPython to the client is unmasked. It is also unbuffered so the client will get a variety of frame sizes.

Only one WebSocket at a time is supported.
1.8.7 Environment Variables

CircuitPython 8.0.0 introduces support for environment variables. Environment variables are commonly used to store “secrets” such as Wi-Fi passwords and API keys. This method *does not* make them secure. It only separates them from the code.

CircuitPython uses a file called `settings.toml` at the drive root (no folder) as the environment. User code can access the values from the file using `os.getenv()`. It is recommended to save any values used repeatedly in a variable because `os.getenv()` will parse the `settings.toml` file contents on every access.

CircuitPython only supports a subset of the full toml specification, see below for more details. The subset is very “Python-like”, which is a key reason we selected the format.

Due to technical limitations it probably also accepts some files that are not valid TOML files; bugs of this nature are subject to change (i.e., be fixed) without the usual deprecation period for incompatible changes.

File format example:

```toml
str_key="Hello world" # with trailing comment
int_key = 7
unicode_key="œuvre"
unicode_key2="\u0153uvre" # same as above
unicode_key3="\U00000153uvre" # same as above
escape_codes="supported, including \r\n\"\\" # comment
[subtable]
subvalue="cannot retrieve this using getenv"
```

**Details of the toml language subset**

- The content is required to be in UTF-8 encoding
- The supported data types are string and integer
- Only basic strings are supported, not triple-quoted strings
- Only integers supported by strtol. (no 0o, no 0b, no underscores 1_000, 011 is 9, not 11)
- Only bare keys are supported
- Duplicate keys are not diagnosed.
- Comments are supported
- Only values from the “root table” can be retrieved
- due to technical limitations, the content of multi-line strings can erroneously be parsed as a value.

**CircuitPython behavior**

CircuitPython will also read the environment to configure its behavior. Other keys are ignored by CircuitPython. Here are the keys it uses:
**CIRCUITPY_BLE_NAME**

Default BLE name the board advertises as, including for the BLE workflow.

**CIRCUITPY_PYSTACK_SIZE**

Sets the size of the python stack. Must be a multiple of 4. The default value is currently 1536. Increasing the stack reduces the size of the heap available to python code. Used to avoid “Pystack exhausted” errors when the code can’t be reworked to avoid it.

**CIRCUITPY_RESERVED_PSRAM**

On boards with Espressif microcontrollers with PSRAM (also called SPIRAM), permanently reserve a portion of PSRAM for use by esp-idf. This storage is removed from the CircuitPython “heap” and is available for allocation by esp-idf routines in the core instead. Generally, only set this to a non-zero value when it is required by a specific core module.

**CIRCUITPY_WEB_API_PASSWORD**

Password required to make modifications to the board from the Web Workflow.

**CIRCUITPY_WEB_API_PORT**

TCP port number used for the web HTTP API. Defaults to 80 when omitted.

**CIRCUITPY_WEB_INSTANCE_NAME**

Name the board advertises as for the WEB workflow. Defaults to human readable board name if omitted.

**CIRCUITPY_WIFI_PASSWORD**

Wi-Fi password used to auto connect to CIRCUITPY_WIFI_SSID.

**CIRCUITPY_WIFI_SSID**

Wi-Fi SSID to auto-connect to even if user code is not running.
1.8.8 Design Guide

This guide covers a variety of development practices for CircuitPython core and library APIs. These APIs are both built-into CircuitPython and those that are distributed on GitHub and in the Adafruit and Community bundles. Consistency with these practices ensures that beginners can learn a pattern once and apply it throughout the CircuitPython ecosystem.

Start libraries with the cookiecutter

Cookiecutter is a tool that lets you bootstrap a new repo based on another repo. We’ve made one here for CircuitPython libraries that include configs for Travis CI and ReadTheDocs along with a setup.py, license, code of conduct, readme among other files.

Cookiecutter will provide a series of prompts relating to the library and then create a new directory with all of the files. See the CircuitPython cookiecutter README for more details.

Module Naming

Adafruit funded libraries should be under the adafruit organization and have the format Adafruit_CircuitPython_<name> and have a corresponding adafruit_<name> directory (aka package) or adafruit_<name>.py file (aka module).

If the name would normally have a space, such as “Thermal Printer”, use an underscore instead (“Thermal_Printer”). This underscore will be used everywhere even when the separation between “adafruit” and “circuitpython” is done with a -. Use the underscore in the cookiecutter prompts.

Community created libraries should have the repo format CircuitPython_<name> and not have the adafruit_module or package prefix.

Both should have the CircuitPython repository topic on GitHub.

Terminology

As our Code of Conduct states, we strive to use “welcoming and inclusive language.” Whether it is in documentation or in code, the words we use matter. This means we disfavor language that due to historical and social context can make community members and potential community members feel unwelcome.

There are specific terms to avoid except where technical limitations require it. While specific cases may call for other terms, consider using these suggested terms first:

<table>
<thead>
<tr>
<th>Preferred</th>
<th>Deprecated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main (device)</td>
<td>Master</td>
</tr>
<tr>
<td>Peripheral</td>
<td>Slave</td>
</tr>
<tr>
<td>Sensor</td>
<td></td>
</tr>
<tr>
<td>Secondary (device)</td>
<td></td>
</tr>
<tr>
<td>Denylist</td>
<td>Blacklist</td>
</tr>
<tr>
<td>Allowlist</td>
<td>Whitelist</td>
</tr>
</tbody>
</table>

Note that “technical limitations” refers e.g., to the situation where an upstream library or URL has to contain those substrings in order to work. However, when it comes to documentation and the names of parameters and properties in CircuitPython, we will use alternate terms even if this breaks tradition with past practice.
**Lifetime and ContextManagers**

A driver should be initialized and ready to use after construction. If the device requires deinitialization, then provide it through `deinit()` and also provide `__enter__` and `__exit__` to create a context manager usable with `with`.

For example, a user can then use `deinit()`:

```python
import digitalio
import board
import time

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

for i in range(10):
    led.value = True
    time.sleep(0.5)

    led.value = False
    time.sleep(0.5)

led.deinit()
```

This will deinit the underlying hardware at the end of the program as long as no exceptions occur.

Alternatively, using a `with` statement ensures that the hardware is deinitialized:

```python
import digitalio
import board
import time

with digitalio.DigitalInOut(board.LED) as led:
    led.direction = digitalio.Direction.OUTPUT

    for i in range(10):
        led.value = True
        time.sleep(0.5)

        led.value = False
        time.sleep(0.5)
```

Python’s `with` statement ensures that the deinit code is run regardless of whether the code within the `with` statement executes without exceptions.

For small programs like the examples this isn’t a major concern because all user usable hardware is reset after programs are run or the REPL is run. However, for more complex programs that may use hardware intermittently and may also handle exceptions on their own, deinitializing the hardware using a `with` statement will ensure hardware isn’t enabled longer than needed.
Verify your device

Whenever possible, make sure the device you are talking to is the device you expect. If not, raise a RuntimeError. Beware that I2C addresses can be identical on different devices so read registers you know to make sure they match your expectation. Validating this upfront will help catch mistakes.

Getters/Setters

When designing a driver for a device, use properties for device state and use methods for sequences of abstract actions that the device performs. State is a property of the device as a whole that exists regardless of what the code is doing. This includes things like temperature, time, sound, light and the state of a switch. For a more complete list see the sensor properties bullet below.

Another way to separate state from actions is that state is usually something the user can sense themselves by sight or feel for example. Actions are something the user can watch. The device does this and then this.

Making this separation clear to the user will help beginners understand when to use what.

Here is more info on properties from Python.

Exceptions and asserts

Raise an appropriate Exception, along with a useful message, whenever a critical test or other condition fails.

Example:

```python
if not 0 <= pin <= 7:
    raise ValueError("Pin number must be 0-7.")
```

If memory is constrained and a more compact method is needed, use The assert statement instead.

Example:

```python
assert 0 <= pin <= 7, "Pin number must be 0-7."
```

Design for compatibility with CPython

CircuitPython is aimed to be one’s first experience with code. It will be the first step into the world of hardware and software. To ease one’s exploration out from this first step, make sure that functionality shared with CPython shares the same API. It doesn't need to be the full API it can be a subset. However, do not add non-CPython APIs to the same modules. Instead, use separate non-CPython modules to add extra functionality. By distinguishing API boundaries at modules you increase the likelihood that incorrect expectations are found on import and not randomly during runtime.

When adding a new module for additional functionality related to a CPython module do NOT simply prefix it with u. This is not a large enough differentiation from CPython. This is the MicroPython convention and they use u* modules interchangeably with the CPython name. This is confusing. Instead, think up a new name that is related to the extra functionality you are adding.

For example, storage mounting and unmounting related functions were moved from uos into a new storage module. These names better match their functionality and do not conflict with CPython names. Make sure to check that you don't conflict with CPython libraries too. That way we can port the API to CPython in the future.
Example

When adding extra functionality to CircuitPython to mimic what a normal operating system would do, either copy an existing CPython API (for example file writing) or create a separate module to achieve what you want. For example, mounting and unmounting drives is not a part of CPython so it should be done in a module, such as a new storage module, that is only available in CircuitPython. That way when someone moves the code to CPython they know what parts need to be adapted.

Document inline

Whenever possible, document your code right next to the code that implements it. This makes it more likely to stay up to date with the implementation itself. Use Sphinx’s automodule to format these all nicely in ReadTheDocs. The cookiecutter helps set these up.

Use Sphinx flavor rST for markup.

Lots of documentation is a good thing but it can take a lot of space. To minimize the space used on disk and on load, distribute the library as both .py and .mpy, MicroPython and CircuitPython’s bytecode format that omits comments.

Module description

After the license comment:

```
"
`<module name>`

================================================================

<Longer description>

* Author(s):

Implementation Notes

---------------------

**Hardware:**

* `Adafruit Device Description
  <hyperlink>_ (Product ID: <Product Number>)`

**Software and Dependencies:**

* Adafruit CircuitPython firmware for the supported boards:
  https://circuitpython.org/downloads

* Adafruit's Bus Device library:
  https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

* Adafruit's Register library:
  https://github.com/adafruit/Adafruit_CircuitPython_Register

"
```
Version description

After the import statements:

```
__version__ = "0.0.0+auto.0"
__repo__ = "<repo github link>"
```

Class description

At the class level document what class does and how to initialize it:

```python
class DS3231:
    """DS3231 real-time clock.
    :
    :param ~busio.I2C i2c_bus: The I2C bus the DS3231 is connected to.
    :param int address: The I2C address of the device. Defaults to :const:`0x40`
    """

    def __init__(self, i2c_bus, address=0x40):
        self._i2c = i2c_bus
```

Renders as:

```python
class DS3231(i2c_bus, address=64)
    DS3231 real-time clock.

    Parameters
    ----------
    i2c_bus (I2C) -- The I2C bus the DS3231 is connected to.
    address (int) -- The I2C address of the device. Defaults to 0x40
```

Documenting Parameters

Although there are different ways to document class and functions definitions in Python, the following is the prevalent method of documenting parameters for CircuitPython libraries. When documenting class parameters you should use the following structure:

```python
:param param_type param_name: Parameter_description
```

`param_type`

The type of the parameter. This could be, among others, int, float, str, bool, etc. To document an object in the CircuitPython domain, you need to include a ~ before the definition as shown in the following example:

```python
:param ~busio.I2C i2c_bus: The I2C bus the DS3231 is connected to.
```

To include references to CircuitPython modules, cookiecutter creates an entry in the intersphinx_mapping section in the conf.py file located within the docs directory. To add different types outside CircuitPython you need to include them in the intersphinx_mapping:
The intersphinx_mapping above includes references to Python, BusDevice and CircuitPython Documentation.

When the parameter have two different types, you should reference them as follows:

```python
class Character_LCD:
    """Base class for character LCD
    :param ~digitalio.DigitalInOut rs: The reset data line
    :param ~pwmio.PWMOut,~digitalio.DigitalInOut blue: Blue RGB Anode
    ""

def __init__(self, rs, blue):
    self._rc = rs
    self.blue = blue
```

Renders as:

```python
class Character_LCD(rs, blue)
    Base class for character LCD

    Parameters
    • rs (DigitalInOut) – The reset data line
    • blue (PWMOut,DigitalInOut) – Blue RGB Anode
```

**param_name**

Parameter name used in the class or method definition

**Parameter_description**

Parameter description. When the parameter defaults to a particular value, it is good practice to include the default:

```python
:param int pitch: Pitch value for the servo. Defaults to :const:`4500`
```
Attributes

Attributes are state on objects. (See Getters/Setters above for more discussion about when to use them.) They can be
defined internally in a number of different ways. Each approach is enumerated below with an explanation of where the
comment goes.

Regardless of how the attribute is implemented, it should have a short description of what state it represents including
the type, possible values and/or units. It should be marked as (read-only) or (write-only) at the end of the first
line for attributes that are not both readable and writable.

Instance attributes

Comment comes from after the assignment:

```python
def __init__(self, drive_mode):
    self.drive_mode = drive_mode
    
    """
    The pin drive mode. One of:
    - `digitalio.DriveMode.PUSH_PULL`
    - `digitalio.DriveMode.OPEN_DRAIN`
    """
```

Renders as:

**drive_mode**

The pin drive mode. One of:

- `digitalio.DriveMode.PUSH_PULL`
- `digitalio.DriveMode.OPEN_DRAIN`

Property description

Comment comes from the getter:

```python
@property
def datetime(self):
    """The current date and time as a `time.struct_time`."""
    return self.datetime_register

@datetime.setter
def datetime(self, value):
    pass
```

Renders as:

**datetime**

The current date and time as a `time.struct_time`.

Read-only example:
@property
def temperature(self):
    
    """
    The current temperature in degrees Celsius. (read-only)
    The device may require calibration to get accurate readings.
    """
    return self._read(TEMPERATURE)

Renders as:

```
temperature
    The current temperature in degrees Celsius. (read-only)
    The device may require calibration to get accurate readings.
```

Data descriptor description

Comment is after the definition:

```
lost_power = i2c_bit.RWBit(0x0f, 7)
""""""""True if the device has lost power since the time was set."""
```

Renders as:

```
lost_power
    True if the device has lost power since the time was set.
```

Method description

First line after the method definition:

```
def turn_right(self, degrees):
    """
    Turns the bot `degrees` right.
    
    :param float degrees: Degrees to turn right
    """
```

Renders as:

```
turn_right(degrees)
    Turns the bot degrees right.

Parameters

    degrees (float) – Degrees to turn right
```
Documentation References to other Libraries

When you need to make references to documentation in other libraries you should refer the class using single back-ticks `class: ~adafruit_motor.servo.Servo`. You must also add the reference in the `conf.py` file in the `intersphinx_mapping` section by adding a new entry:

```
```

Use `adafruit_register` when possible

`Register` is a foundational library that manages packing and unpacking data from I2C device registers. There is also `RegisterSPI` for SPI devices. When possible, use one of these libraries for unpacking and packing registers. This ensures the packing code is shared amongst all registers (even across drivers). Furthermore, it simplifies device definitions by making them declarative (only data.)

Values with non-consecutive bits in a register or that represent FIFO endpoints may not map well to existing register classes. In unique cases like these, it is ok to read and write the register directly.

**Do not** add all registers from a datasheet upfront. Instead, only add the ones necessary for the functionality the driver exposes. Adding them all will lead to unnecessary file size and API clutter. See this video about outside-in design from @tannewt.

I2C Example

```python
from adafruit_register import i2c_bit
from adafruit_bus_device import i2c_device

class HelloWorldDevice:
    """Device with two bits to control when the words 'hello' and 'world' are lit.""

    hello = i2c_bit.RWBit(0x0, 0x0)
    """Bit to indicate if hello is lit.""

    world = i2c_bit.RWBit(0x1, 0x0)
    """Bit to indicate if world is lit.""

    def __init__(self, i2c, device_address=0x0):
        self.i2c_device = i2c_device.I2CDevice(i2c, device_address)
```

Use `BusDevice`

`BusDevice` is an awesome foundational library that manages talking on a shared I2C or SPI device for you. The devices manage locking which ensures that a transfer is done as a single unit despite CircuitPython internals and, in the future, other Python threads. For I2C, the device also manages the device address. The SPI device, manages baudrate settings, chip select line and extra post-transaction clock cycles.
I2C Example

```python
from adafruit_bus_device import i2c_device

DEVICE_DEFAULT_I2C_ADDR = 0x42

class Widget:
    """A generic widget."""
    def __init__(self, i2c, address=DEVICE_DEFAULT_I2C_ADDR):
        self.i2c_device = i2c_device.I2CDevice(i2c, address)
        self.buf = bytearray(1)

    @property
    def register(self):
        """Widget's one register."""
        with self.i2c_device as i2c:
            i2c.writeto(b'0x00')
            i2c.readfrom_into(self.buf)
        return self.buf[0]
```

SPI Example

```python
from adafruit_bus_device import spi_device

class SPIWidget:
    """A generic widget with a weird baudrate."""
    def __init__(self, spi, chip_select):
        self.spi_device = spi_device.SPIDevice(spi, chip_select, baudrate=12345)
        self.buf = bytearray(1)

    @property
    def register(self):
        """Widget's one register."""
        with self.spi_device as spi:
            spi.write(b'0x00')
            spi.readinto(self.buf)
        return self.buf[0]
```
Class documentation example template

When documenting classes, you should use the following template to illustrate basic usage. It is similar with the simplest example, however this will display the information in the Read The Docs documentation. The advantage of using this template is it makes the documentation consistent across the libraries.

This is an example for a AHT20 temperature sensor. Include the following after the class parameter:

```python
**Quickstart: Importing and using the AHT10/AHT20 temperature sensor**

Here is an example of using the :class:`AHTx0` class. First you will need to import the libraries to use the sensor

```code
import board
import adafruit_ahtx0
```

Once this is done you can define your `board.I2C` object and define your sensor object

```code
i2c = board.I2C() # uses board.SCL and board.SDA
aht = adafruit_ahtx0.AHTx0(i2c)
```

Now you have access to the temperature and humidity using the :attr:`temperature` and :attr:`relative_humidity` attributes

```code
temperature = aht.temperature
relative_humidity = aht.relative_humidity
```

Use composition

When writing a driver, take in objects that provide the functionality you need rather than taking their arguments and constructing them yourself or subclassing a parent class with functionality. This technique is known as composition and leads to code that is more flexible and testable than traditional inheritance.

See also:

Wikipedia has more information on “dependency inversion”.

For example, if you are writing a driver for an I2C device, then take in an I2C object instead of the pins themselves. This allows the calling code to provide any object with the appropriate methods such as an I2C expansion board.

Another example is to expect a DigitalInOut for a pin to toggle instead of a Pin from board. Taking in the Pin object alone would limit the driver to pins on the actual microcontroller instead of pins provided by another driver such as an IO expander.
Lots of small modules

CircuitPython boards tend to have a small amount of internal flash and a small amount of ram but large amounts of external flash for the file system. So, create many small libraries that can be loaded as needed instead of one large file that does everything.

Speed second

Speed isn’t as important as API clarity and code size. So, prefer simple APIs like properties for state even if it sacrifices a bit of speed.

Avoid allocations in drivers

Although Python doesn’t require managing memory, it’s still a good practice for library writers to think about memory allocations. Avoid them in drivers if you can because you never know how much something will be called. Fewer allocations means less time spent cleaning up. So, where you can, prefer bytearray buffers that are created in __init__ and used throughout the object with methods that read or write into the buffer instead of creating new objects. Unified hardware API classes such as busio.SPI are designed to read and write to subsections of buffers.

It’s ok to allocate an object to return to the user. Just beware of causing more than one allocation per call due to internal logic.

However, this is a memory tradeoff so do not do it for large or rarely used buffers.

Examples

struct.pack

Use struct.pack_into instead of struct.pack.

Use of MicroPython const()

The MicroPython const() feature, as discussed in this forum post, and in this issue thread, provides some optimizations that can be useful on smaller, memory constrained devices. However, when using const(), keep in mind these general guidelines:

- Always use via an import, ex: from micropython import const
- Limit use to global (module level) variables only.
- Only used when the user will not need access to variable and prefix name with a leading underscore, ex: _SOME_CONST.
Example

```python
from adafruit_bus_device import i2c_device
from micropython import const

_DEFAULT_I2C_ADDR = const(0x42)

class Widget:
    """A generic widget."""

    def __init__(self, i2c, address=_DEFAULT_I2C_ADDR):
        self.i2c_device = i2c_device.I2CDevice(i2c, address)
```

Libraries Examples

When adding examples, cookiecutter will add a `<name>_simpletest.py` file in the examples directory for you. Be sure to include code with the library minimal functionalities to work on a device. You could other examples if needed featuring different functionalities of the library. If you add additional examples, be sure to include them in the `examples.rst`. Naming of the examples files should use the name of the library followed by a description, using underscore to separate them.

Sensor properties and units

The Adafruit Unified Sensor Driver Arduino library has a great list of measurements and their units. Use the same ones including the property name itself so that drivers can be used interchangeably when they have the same properties.
<table>
<thead>
<tr>
<th>Property name</th>
<th>Python type</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceleration</td>
<td>(float, float)</td>
<td>x, y, z meter per second per second</td>
</tr>
<tr>
<td>magnetic</td>
<td>(float, float)</td>
<td>x, y, z micro-Tesla (uT)</td>
</tr>
<tr>
<td>orientation</td>
<td>(float, float)</td>
<td>x, y, z degrees</td>
</tr>
<tr>
<td>gyro</td>
<td>(float, float)</td>
<td>x, y, z radians per second</td>
</tr>
<tr>
<td>temperature</td>
<td>float</td>
<td>degrees Celsius</td>
</tr>
<tr>
<td>CO2</td>
<td>float</td>
<td>measured CO2 in ppm</td>
</tr>
<tr>
<td>eCO2</td>
<td>float</td>
<td>equivalent/estimated CO2 in ppm (estimated from some other measurement)</td>
</tr>
<tr>
<td>TVOC</td>
<td>float</td>
<td>Total Volatile Organic Compounds in ppb</td>
</tr>
<tr>
<td>distance</td>
<td>float</td>
<td>centimeters (cm)</td>
</tr>
<tr>
<td>proximity</td>
<td>int</td>
<td>non-unit-specific proximity values</td>
</tr>
<tr>
<td>light</td>
<td>float</td>
<td>non-unit-specific light levels (should be monotonic but not lux)</td>
</tr>
<tr>
<td>lux</td>
<td>float</td>
<td>SI lux</td>
</tr>
<tr>
<td>pressure</td>
<td>float</td>
<td>hectopascal (hPa)</td>
</tr>
<tr>
<td>relative_humidity</td>
<td>float</td>
<td>percent</td>
</tr>
<tr>
<td>current</td>
<td>float</td>
<td>milliamps (mA)</td>
</tr>
<tr>
<td>voltage</td>
<td>float</td>
<td>volts (V)</td>
</tr>
<tr>
<td>color</td>
<td>int</td>
<td>RGB, eight bits per channel (0xff0000 is red)</td>
</tr>
<tr>
<td>alarm</td>
<td>(time.struct, str)</td>
<td>Sample alarm time and string to characterize frequency such as “hourly”</td>
</tr>
<tr>
<td>datetime</td>
<td>time.struct</td>
<td>date and time</td>
</tr>
<tr>
<td>duty_cycle</td>
<td>int</td>
<td>16-bit PWM duty cycle (regardless of output resolution)</td>
</tr>
<tr>
<td>frequency</td>
<td>int</td>
<td>Hertz (Hz)</td>
</tr>
<tr>
<td>value</td>
<td>bool</td>
<td>Digital logic</td>
</tr>
<tr>
<td>value</td>
<td>int</td>
<td>16-bit Analog value, unit-less</td>
</tr>
<tr>
<td>weight</td>
<td>float</td>
<td>grams (g)</td>
</tr>
<tr>
<td>sound_level</td>
<td>float</td>
<td>non-unit-specific sound level (monotonic but not actual decibels)</td>
</tr>
</tbody>
</table>

**Driver constant naming**

When adding variables for constant values for a driver. Do not include the device’s name in the variable name. For example, in `adafruit_fancy123.py`, variables should not start with `FANCY123_`. Adding this prefix increases RAM usage and .mpy file size because variable names are preserved. User code should refer to these constants as `adafruit_fancy123.HELLO_WORLD` for clarity. `adafruit_fancy123.FANCY123 HELLO WORLD` would be overly verbose.

**Adding native modules**

The Python API for a new module should be defined and documented in `shared-bindings` and define an underlying C API. If the implementation is port-agnostic or relies on underlying APIs of another module, the code should live in `shared-module`. If it is port specific then it should live in `common-hal` within the port’s folder. In either case, the file and folder structure should mimic the structure in `shared-bindings`.

To test your native modules or core enhancements, follow these Adafruit Learning Guides for building local firmware to flash onto your device(s):

Build CircuitPython
MicroPython compatibility

Keeping compatibility with MicroPython isn’t a high priority. It should be done when it’s not in conflict with any of the above goals.

We love CircuitPython and would love to see it come to more microcontroller platforms. Since 3.0 we’ve reworked CircuitPython to make it easier than ever to add support. While there are some major differences between ports, this page covers the similarities that make CircuitPython what it is and how that core fits into a variety of microcontrollers.

1.8.9 Architecture

There are three core pieces to CircuitPython:

The first is the Python VM that the awesome MicroPython devs have created. These VMs are written to be portable so there is not much needed when moving to a different microcontroller, especially if it is ARM based.

The second is the infrastructure around those VMs which provides super basic operating system functionality such as initializing hardware, running USB, prepping file systems and automatically running user code on boot. In CircuitPython we’ve dubbed this component the supervisor because it monitors and facilitates the VMs which run user Python code. Porting involves the supervisor because many of the tasks it does while interfacing with the hardware. Once complete, the REPL works and debugging can migrate to a Python based approach rather than C.

The third core piece is the plethora of low level APIs that CircuitPython provides as the foundation for higher level libraries including device drivers. These APIs are called from within the running VMs through the Python interfaces defined in shared-bindings. These bindings rely on the underlying common_hal C API to implement the functionality needed for the Python API. By splitting the two, we work to ensure standard functionality across which means that libraries and examples apply across ports with minimal changes.

1.8.10 Porting

Step 1: Getting building

The first step to porting to a new microcontroller is getting a build running. The primary goal of it should be to get main.c compiling with the assistance of the supervisor/supervisor.mk file. Port specific code should be isolated to the port’s directory (in the top level until the ports directory is present). This includes the Makefile and any C library resources. Make sure these resources are compatible with the MIT License of the rest of the code!

Circuitpython has a number of modules enabled by default in py/circuitpy_mpconfig.mk. Most of these modules will need to be disabled in mpconfigboard.mk during the early stages of a port in order for it to compile. As the port progresses in module support, this list can be pruned down as a natural “TODO” list. An example minimal build list is shown below:

```
# These modules are implemented in ports/<port>/common-hal:

# Typically the first module to create
CIRCUITPY_MICROCONTROLLER = 0
# Typically the second module to create
CIRCUITPY_DIGITALIO = 0
# Other modules:
CIRCUITPY_ANALOGIO = 0
CIRCUITPY_BUSIO = 0
CIRCUITPY_COUNTIO = 0
CIRCUITPY_NEOPIXEL_WRITE = 0
CIRCUITPY_PULSEIO = 0
```

(continues on next page)
CIRCUITPY_OS = 0
CIRCUITPY_NVM = 0
CIRCUITPY_AUDIOBUSIO = 0
CIRCUITPY_AUDIOIO = 0
CIRCUITPY_ROTARYIO = 0
CIRCUITPY_RTC = 0
CIRCUITPY_SDCARDIO = 0
CIRCUITPY_FRAMEBUFFERIO = 0
CIRCUITPY_FREQUENCYIO = 0
CIRCUITPY_I2CTARGET = 0
# Requires SPI, PulseIO (stub ok):
CIRCUITPY_DISPLAYIO = 0

# These modules are implemented in shared-module/ - they can be included in
# any port once their prerequisites in common-hal are complete.
# Requires DigitalIO:
CIRCUITPY_BITBANGIO = 0
# Requires neopixel_write or SPI (dotstar)
CIRCUITPY_PIXELBUF = 0
# Requires OS
CIRCUITPY_RANDOM = 0
# Requires OS, filesystem
CIRCUITPY_STORAGE = 0
# Requires Microcontroller
CIRCUITPY_TOUCHIO = 0
# Requires USB
CIRCUITPY_USB_HID = 0
CIRCUITPY_USB_MIDI = 0
# Does nothing without I2C
CIRCUITPY_REQUIRE_I2C_PULLUPS = 0
# No requirements, but takes extra flash
CIRCUITPY_ULAB = 0

### Step 2: Init

Once your build is setup, the next step should be to get your clocks going as you expect from the supervisor. The
supervisor calls `port_init` to allow for initialization at the beginning of main. This function also has the ability to
request a safe mode state which prevents the supervisor from running user code while still allowing access to the REPL
and other resources.

The core port initialization and reset methods are defined in `supervisor/port.c` and should be the first to be imple-
mented. It’s required that they be implemented in the `supervisor` directory within the port directory. That way, they
are always in the expected place.

The supervisor also uses three linker variables, `_ezero`, `_estack` and `_ebss` to determine memory layout for stack
overflow checking.
Step 3: REPL

Getting the REPL going is a huge step. It involves a bunch of initialization to be done correctly and is a good sign you are well on your porting way. To get the REPL going you must implement the functions and definitions from supervisor/serial.h with a corresponding supervisor/serial.c in the port directory. This involves sending and receiving characters over some sort of serial connection. It could be UART or USB for example.

1.8.11 Adding *io support to other ports

digitalio provides a well-defined, cross-port hardware abstraction layer built to support different devices and their drivers. It’s backed by the Common HAL, a C api suitable for supporting different hardware in a similar manner. By sharing this C api, developers can support new hardware easily and cross-port functionality to the new hardware.

These instructions also apply to analogio, busio, pulseio and touchio. Most drivers depend on analogio, digitalio and busio so start with those.

File layout

Common HAL related files are found in these locations:

- **shared-bindings** Shared home for the Python <-> C bindings which includes inline RST documentation for the created interfaces. The common hal functions are defined in the .h files of the corresponding C files.

- **shared-module** Shared home for C code built on the Common HAL and used by all ports. This code only uses common_hal methods defined in shared-bindings.

- `<port>/common-hal` Port-specific implementation of the Common HAL.

Each folder has the substructure of / and they should match 1:1. __init__.c is used for module globals that are not classes (similar to __init__.py).

Adding support

Modifying the build

The first step is to hook the shared-bindings into your build for the modules you wish to support. Here’s an example of this step for the atmel-samd/Makefile:

```
SRC_BINDINGS = \n    board/__init__.c \n    microcontroller/__init__.c \n    microcontroller/Pad.c \n    analogio/__init__.c \n    analogio/AnalogIn.c \n    analogio/AnalogOut.c \n    digitalio/__init__.c \n    digitalio/DigitalInOut.c \n    pulseio/__init__.c \n    pulseio/PulseIn.c \n    pulseio/PulseOut.c \n    pulseio/PWMOut.c \n    busio/__init__.c \n    busio/I2C.c \n```

(continues on next page)
busio/SPI.c \  
busio/UART.c \  
neopixel_write/__init__.c \  
time/__init__.c \  
usb_hid/__init__.c \  
usb_hid/Device.c

SRC_BINDINGS_EXPANDED = $(addprefix shared-bindings/, $(SRC_BINDINGS)) \  
$(addprefix common-hal/, $(SRC_BINDINGS))

# Add the resulting objects to the full list
OBJ += $(addprefix $(BUILD)/, $(SRC_BINDINGS_EXPANDED:.c=.o))

# Add the sources for QSTR generation
SRC_QSTR += $(SRC_C) $(SRC_BINDINGS_EXPANDED) $(STM_SRC_C)

The Makefile defines the modules to build and adds the sources to include the shared-bindings version and the common-hal version within the port specific directory. You may comment out certain subfolders to reduce the number of modules to add but don’t comment out individual classes. It won’t compile then.

### Hooking the modules in

Built in modules are typically defined in mpconfigport.h. To add support you should have something like:

```c
extern const struct _mp_obj_module_t microcontroller_module;
extern const struct _mp_obj_module_t analogio_module;
extern const struct _mp_obj_module_t digitalio_module;
extern const struct _mp_obj_module_t pulseio_module;
extern const struct _mp_obj_module_t busio_module;
extern const struct _mp_obj_module_t board_module;
extern const struct _mp_obj_module_t time_module;
extern const struct _mp_obj_module_t neopixel_write_module;

#define MICROPY_PORT_BUILTIN_MODULES
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_microcontroller), (mp_obj_t)&microcontroller_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_analogio), (mp_obj_t)&analogio_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_digitalio), (mp_obj_t)&digitalio_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_pulseio), (mp_obj_t)&pulseio_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_busio), (mp_obj_t)&busio_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_board), (mp_obj_t)&board_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_time), (mp_obj_t)&time_module },, \  
  \{ MP_OBJ_NEW_QSTR(MP_QSTR_neopixel_write),(mp_obj_t)&neopixel_write_module } \
```

1.8. Full Table of Contents 357
Implementing the Common HAL

At this point in the port, nothing will compile yet, because there’s still work to be done to fix missing sources, compile issues, and link issues. I suggest start with a common-hal directory from another port that implements it such as atmel-samd or esp8266, deleting the function contents and stubbing out any return statements. Once that is done, you should be able to compile cleanly and import the modules, but nothing will work (though you are getting closer).

The last step is actually implementing each function in a port specific way. I can’t help you with this. :-) If you have any questions how a Common HAL function should work then see the corresponding .h file in shared-bindings.

Testing

Woohoo! You are almost done. After you implement everything, lots of drivers and sample code should just work. There are a number of drivers and examples written for Adafruit’s Feather ecosystem. Here are places to start:

- Adafruit repos with CircuitPython topic
- Adafruit driver bundle

1.8.12 MicroPython libraries

Python standard libraries and micro-libraries

The libraries below are inherited from MicroPython. They are similar to the standard Python libraries with the same name. They implement a subset of or a variant of the corresponding standard Python library.

CircuitPython’s long-term goal is that code written in CircuitPython using Python standard libraries will be runnable on CPython without changes.

These libraries are not enabled on CircuitPython builds with limited flash memory, usually on non-Express builds: binascii, errno, json, re.

These libraries are not currently enabled in any CircuitPython build, but may be in the future: ctypes

builtins – builtin functions and exceptions

All builtin functions and exceptions are described here. They are also available via the builtins module.

For more information about built-ins, see the following CPython documentation:

- Builtin CPython Functions
- Builtin CPython Exceptions
- Builtin CPython Constants

Note: Not all of these functions, types, exceptions, and constants are turned on in all CircuitPython ports, for space reasons.
Functions and types

```
builtins.abs()
builtins.all()
builtins.any()
builtins.bin()
class builtins.bool
class builtins.bytestring
class builtins.bytes
    See CPython documentation: bytes.
builtins.callable()
builtins.chr()
builtins.classmethod()
builtins.compile()
class builtins.complex
builtins.delattr(obj, name)
    The argument name should be a string, and this function deletes the named attribute from the object given by obj.
class builtins.dict
builtins.dir()
builtins.divmod()
builtins.enumerate()
builtins.eval()
builtins.exec()
builtins.filter()
class builtins.float
class builtins.frozenset
    frozenset() is not enabled on non-Express CircuitPython boards.
builtins.getattr()
builtins.globals()
builtins.hasattr()
builtins.hash()
builtins.hex()
```
builtins.id()
builtins.input()
class builtins.int
    classmethod from_bytes(bytes, byteorder)
        In CircuitPython, byteorder parameter must be positional (this is compatible with CPython).
    to_bytes(size, byteorder)
        In CircuitPython, byteorder parameter must be positional (this is compatible with CPython).
builtins.isinstance()
builtins.issubclass()
builtins.iter()
builtins.len()
class builtins.list
builtins.locals()
builtins.map()
builtins.max()
class builtins.memoryview
builtins.min()
builtins.next()
class builtins.object
builtins.oct()
builtins.open()
builtins.ord()
builtins.pow()
builtins.print()
builtins.property()
builtins.range()
builtins.repr()
builtins.reversed()
    reversed() is not enabled on non-Express CircuitPython boards.
builtins.round()
class builtins.set
builtinssetattr()
class builtins.slice
    The slice builtin is the type that slice objects have.
builtins.sorted()
builtins.staticmethod()
class builtins.str
builtins.sum()
builtins.super()
class builtins.tuple
builtins.type()
builtins.zip()

Exceptions

exception builtins.ArithmeticError
exception builtins.AssertionError
exception builtins.AttributeError
exception builtins.BaseException
exception builtins.BrokenPipeError
exception builtins.ConnectionError
exception builtins.EOFError
exception builtins.Exception
exception builtins.ImportError
exception builtins_indentation.Error
exception builtins.IndexError
exception builtins.KeyError
exception builtins.LookupError
exception builtins.MemoryError
exception builtins.NameError
exception builtins.NotImplementedError
exception builtins.OperError
exception builtins.OverflowError
exception builtins.RuntimeError

exception builtins.ReloadException

ReloadException is used internally to deal with soft restarts.

Not a part of the CPython standard library

exception builtins.StopAsyncIteration

exception builtins.StopIteration

exception builtins_SyntaxError

exception builtins.SystemExit

exception builtins.TimeoutError

exception builtins.TypeError

exception builtins.UnicodeError

exception builtins.ValueError

exception builtins.ZeroDivisionError

Constants

builtins.Ellipsis

builtins.NotImplemented

heapq – heap queue algorithm

Warning: Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: heapq.

This module implements the min heap queue algorithm.

A heap queue is essentially a list that has its elements stored in such a way that the first item of the list is always the smallest.
Functions

heapq.heappush(heap, item)
Push the item onto the heap.

heapq.heappop(heap)
Pop the first item from the heap, and return it. Raise IndexError if heap is empty.
The returned item will be the smallest item in the heap.

heapq.heapify(x)
Convert the list x into a heap. This is an in-place operation.

array – arrays of numeric data

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: array.

Supported format codes: b, B, h, i, I, l, L, q, Q, f, d (the latter 2 depending on the floating-point support).

Classes

class array.array(typecode[, iterable])
Create array with elements of given type. Initial contents of the array are given by an iterable. If it is not provided, an empty array is created.

append(val)
Append new element val to the end of array, growing it.

extend(iterable)
Append new elements as contained in iterable to the end of array, growing it.

binascii – binary/ASCII conversions

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: binascii.

This module implements conversions between binary data and various encodings of it in ASCII form (in both directions).

Functions

binascii.hexlify(data[, sep])
Convert the bytes in the data object to a hexadecimal representation. Returns a bytes object.
If the additional argument sep is supplied it is used as a separator between hexadecimal values.

binascii.unhexlify(data)
Convert hexadecimal data to binary representation. Returns bytes string. (i.e. inverse of hexlify)
binascii.a2b_base64(data)

Decode base64-encoded data, ignoring invalid characters in the input. Conforms to RFC 2045 s.6.8. Returns a bytes object.

binascii.b2a_base64(data)

Encode binary data in base64 format, as in RFC 3548. Returns the encoded data followed by a newline character, as a bytes object.

binascii.crc32(data, value=0, /)

Compute CRC-32, the 32-bit checksum of the bytes in data starting with an initial CRC of value. The default initial CRC is 0. The algorithm is consistent with the ZIP file checksum.

collections – collection and container types

Warning: Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: collections.

This module implements advanced collection and container types to hold/accumulate various objects.

Classes

collections.deque(iterable, maxlen[, flags])

Deques (double-ended queues) are a list-like container that support O(1) appends and pops from either side of the deque. New deques are created using the following arguments:

• iterable must be the empty tuple, and the new deque is created empty.
• maxlen must be specified and the deque will be bounded to this maximum length. Once the deque is full, any new items added will discard items from the opposite end.
• The optional flags can be 1 to check for overflow when adding items.

As well as supporting bool and len, deque objects have the following methods:

deqe.append(x)

Add x to the right side of the deque. Raises IndexError if overflow checking is enabled and there is no more room left.

deqe.popleft()

Remove and return an item from the left side of the deque. Raises IndexError if no items are present.

collections.namedtuple(name, fields)

This is factory function to create a new namedtuple type with a specific name and set of fields. A namedtuple is a subclass of tuple which allows to access its fields not just by numeric index, but also with an attribute access syntax using symbolic field names. Fields is a sequence of strings specifying field names. For compatibility with CPython it can also be a a string with space-separated field named (but this is less efficient). Example of use:
from collections import namedtuple

MyTuple = namedtuple("MyTuple", ("id", "name"))
t1 = MyTuple(1, "foo")
t2 = MyTuple(2, "bar")
print(t1.name)
assert t2.name == t2[1]

collections.OrderedDict(...)

dict type subclass which remembers and preserves the order of keys added. When ordered dict is iterated over,
keys/items are returned in the order they were added:

from collections import OrderedDict

# To make benefit of ordered keys, OrderedDict should be initialized
# from sequence of (key, value) pairs.
d = OrderedDict([("z", 1), ("a", 2)])
# More items can be added as usual
d["w"] = 5
d["b"] = 3
for k, v in d.items():
    print(k, v)

Output:

z 1
a 2
w 5
b 3

erro — system error codes

This module implements a subset of the corresponding CPython module, as described below. For more information,
refer to the original CPython documentation: errno.

This module provides access to symbolic error codes for OSError exception. A particular inventory of codes depends
on MicroPython port.

Constants

EEXIST, EAGAIN, etc.

Error codes, based on ANSI C/POSIX standard. All error codes start with “E”. As mentioned above, inventory
of the codes depends on MicroPython port. Errors are usually accessible as exc.args[0] where exc is an
instance of OSError. Usage example:

try:
    os.mkdir("my_dir")
except OSError as exc:
    if exc.args[0] == errno.EEXIST:
        print("Directory already exists")
**errno.errorcode**

Dictionary mapping numeric error codes to strings with symbolic error code (see above):

```python
>>> print(errno.errorcode[errno.EEXIST])
EEXIST
```

**gc – control the garbage collector**

**Warning:** Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: gc.

**Functions**

**gc.enable()**

Enable automatic garbage collection.

**gc.disable()**

Disable automatic garbage collection. Heap memory can still be allocated, and garbage collection can still be initiated manually using `gc.collect()`.

**gc.collect()**

Run a garbage collection.

**gc.mem_alloc()**

Return the number of bytes of heap RAM that are allocated.

**Difference to CPython**

This function is a MicroPython extension.

**gc.mem_free()**

Return the number of bytes of available heap RAM, or -1 if this amount is not known.

**Difference to CPython**

This function is a MicroPython extension.

**gc.threshold([amount])**

Set or query the additional GC allocation threshold. Normally, a collection is triggered only when a new allocation cannot be satisfied, i.e. on an out-of-memory (OOM) condition. If this function is called, in addition to OOM, a collection will be triggered each time after `amount` bytes have been allocated (in total, since the previous time such an amount of bytes have been allocated). `amount` is usually specified as less than the full heap size, with the intention to trigger a collection earlier than when the heap becomes exhausted, and in the hope that an
early collection will prevent excessive memory fragmentation. This is a heuristic measure, the effect of which will vary from application to application, as well as the optimal value of the *amount* parameter.

Calling the function without argument will return the current value of the threshold. A value of -1 means a disabled allocation threshold.

### Difference to CPython

This function is a MicroPython extension. CPython has a similar function - `set_threshold()`, but due to different GC implementations, its signature and semantics are different.

### io – input/output streams

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: `io`.

This module contains additional types of stream (file-like) objects and helper functions.

### Conceptual hierarchy

### Difference to CPython

Conceptual hierarchy of stream base classes is simplified in MicroPython, as described in this section.

(Abstract) base stream classes, which serve as a foundation for behavior of all the concrete classes, adhere to few dichotomies (pair-wise classifications) in CPython. In MicroPython, they are somewhat simplified and made implicit to achieve higher efficiencies and save resources.

An important dichotomy in CPython is unbuffered vs buffered streams. In MicroPython, all streams are currently unbuffered. This is because all modern OSES, and even many RTOSes and filesystem drivers already perform buffering on their side. Adding another layer of buffering is counter-productive (an issue known as “bufferbloat”) and takes precious memory. Note that there still cases where buffering may be useful, so we may introduce optional buffering support at a later time.

But in CPython, another important dichotomy is tied with “bufferedness” - it’s whether a stream may incur short read/writes or not. A short read is when a user asks e.g. 10 bytes from a stream, but gets less, similarly for writes. In CPython, unbuffered streams are automatically short operation susceptible, while buffered are guaranteed against them.

The no short read/writes is an important trait, as it allows to develop more concise and efficient programs - something which is highly desirable for MicroPython. So, while MicroPython doesn’t support buffered streams, it still provides for no-short-operations streams. Whether there will be short operations or not depends on each particular class’ needs, but developers are strongly advised to favor no-short-operations behavior for the reasons stated above. For example, MicroPython sockets are guaranteed to avoid short read/writes. Actually, at this time, there is no example of a short-operations stream class in the core, and one would be a port-specific class, where such a need is governed by hardware peculiarities.

The no-short-operations behavior gets tricky in case of non-blocking streams, blocking vs non-blocking behavior being another CPython dichotomy, fully supported by MicroPython. Non-blocking streams never wait for data either to arrive or be written - they read/write whatever possible, or signal lack of data (or ability to write data). Clearly, this conflicts with “no-short-operations” policy, and indeed, a case of non-blocking buffered (and this no-short-ops) streams is convoluted in CPython - in some places, such combination is prohibited, in some it’s undefined or just not documented, in some cases it raises verbose exceptions. The matter is much simpler in MicroPython: non-blocking stream are important for efficient asynchronous operations, so this property prevails on the “no-short-ops” one. So,
while blocking streams will avoid short reads/writes whenever possible (the only case to get a short read is if end of
file is reached, or in case of error (but errors don’t return short data, but raise exceptions)), non-blocking streams may
produce short data to avoid blocking the operation.

The final dichotomy is binary vs text streams. MicroPython of course supports these, but while in CPYthon text streams
are inherently buffered, they aren’t in MicroPython. (Indeed, that’s one of the cases for which we may introduce
buffering support.)

Note that for efficiency, MicroPython doesn’t provide abstract base classes corresponding to the hierarchy above, and
it’s not possible to implement, or subclass, a stream class in pure Python.

**Functions**

```python
io.open(name, mode='r', **kwargs)
```

Open a file. Built-in `open()` function is aliased to this function. All ports (which provide access to file system)
are required to support `mode` parameter, but support for other arguments vary by port.

**Classes**

```python
class io.FileIO(...)
```

This is type of a file open in binary mode, e.g. using `open(name, "rb")`. You should not instantiate this class
directly.

```python
class io.TextIOWrapper(...)
```

This is type of a file open in text mode, e.g. using `open(name, "rt")`. You should not instantiate this class
directly.

```python
class io.StringIO([string])
```

```python
class io.BytesIO([string])
```

In-memory file-like objects for input/output. `StringIO` is used for text-mode I/O (similar to a normal file opened
with “t” modifier). `BytesIO` is used for binary-mode I/O (similar to a normal file opened with “b” modifier).

Initial contents of file-like objects can be specified with `string` parameter (should be normal string for `StringIO`
or bytes object for `BytesIO`). All the usual file methods like `read()`, `write()`, `seek()`, `flush()`, `close()`
are available on these objects, and additionally, a following method:

```python
getvalue()
```

Get the current contents of the underlying buffer which holds data.

**json – JSON encoding and decoding**

This module implements a subset of the corresponding CPYthon module, as described below. For more information,
refer to the original CPYthon documentation: `json`.

This module allows to convert between Python objects and the JSON data format.
Functions

json.\texttt{dump}(\texttt{obj}, \texttt{stream})

Serialise \texttt{obj} to a JSON string, writing it to the given \texttt{stream}.

json.\texttt{dumps}(\texttt{obj})

Return \texttt{obj} represented as a JSON string.

json.\texttt{load}(\texttt{stream})

Parse the given \texttt{stream}, interpreting it as a JSON string and deserialising the data to a Python object. The resulting object is returned.

Parsing continues until end-of-file is encountered. A \texttt{ValueError} is raised if the data in \texttt{stream} is not correctly formed.

json.\texttt{loads}(\texttt{str})

Parse the JSON \texttt{str} and return an object. Raises \texttt{ValueError} if the string is not correctly formed.

\texttt{re} -- simple regular expressions

This module implements a subset of the corresponding \texttt{CPython} module, as described below. For more information, refer to the original \texttt{CPython} documentation: \texttt{re}.

This module implements regular expression operations. Regular expression syntax supported is a subset of \texttt{CPython} \texttt{re} module (and actually is a subset of \texttt{POSIX} extended regular expressions).

Supported operators and special sequences are:

\begin{itemize}
  \item \texttt{.}
    Match any character.
  \item \texttt{[... ]}
    Match set of characters. Individual characters and ranges are supported, including negated sets (e.g. \texttt{[^a-c]}).
  \item \texttt{^}
    Match the start of the string.
  \item \texttt{$}
    Match the end of the string.
  \item \texttt{?}
    Match zero or one of the previous sub-pattern.
  \item \texttt{*}
    Match zero or more of the previous sub-pattern.
  \item \texttt{+}
    Match one or more of the previous sub-pattern.
  \item \texttt{??}
    Non-greedy version of ?, match zero or one, with the preference for zero.
  \item \texttt{*?}
    Non-greedy version of *, match zero or more, with the preference for the shortest match.
  \item \texttt{+?}
    Non-greedy version of +, match one or more, with the preference for the shortest match.
  \item \texttt{|}
    Match either the left-hand side or the right-hand side sub-patterns of this operator.
\end{itemize}
Grouping. Each group is capturing (a substring it captures can be accessed with `match.group()` method).

\d
Matches digit. Equivalent to [0-9].

\D
Matches non-digit. Equivalent to [^0-9].

\s
Matches whitespace. Equivalent to [ \t-\r].

\S
Matches non-whitespace. Equivalent to [^ \t-\r].

\w
Matches “word characters” (ASCII only). Equivalent to [A-Za-z0-9_].

\W
Matches non “word characters” (ASCII only). Equivalent to [^A-Za-z0-9_].

\ Escape character. Any other character following the backslash, except for those listed above, is taken literally. For example, \* is equivalent to literal * (not treated as the * operator). Note that \r, \n, etc. are not handled specially, and will be equivalent to literal letters r, n, etc. Due to this, it’s not recommended to use raw Python strings (r“””) for regular expressions. For example, r“\r\n” when used as the regular expression is equivalent to “rn”. To match CR character followed by LF, use “\r\n”.

NOT SUPPORTED:

• counted repetitions ({m,n})
• named groups ((?P<name>…))
• non-capturing groups ((?:…))
• more advanced assertions (\b, \B)
• special character escapes like \r, \n - use Python’s own escaping instead
• etc.

Example:

```python
import re

# As re doesn't support escapes itself, use of r'' strings is not # recommended.
regex = re.compile("\r\n")

regex.split("line1\rline2\nline3\r\n")

# Result:
# ['line1', 'line2', 'line3', '', '']
```
Functions

```python
re.compile(regex_str[, flags])
```
Compile regular expression, return regex object.

```python
re.match(regex_str, string)
```
Compile regex_str and match against string. Match always happens from starting position in a string.

```python
re.search(regex_str, string)
```
Compile regex_str and search it in a string. Unlike `match`, this will search string for first position which matches regex (which still may be 0 if regex is anchored).

```python
re.sub(regex_str, replace, string[, count=0, flags=0, /])
```
Compile regex_str and search for it in string, replacing all matches with replace, and returning the new string.

replace can be a string or a function. If it is a string then escape sequences of the form \<number> and \g<number> can be used to expand to the corresponding group (or an empty string for unmatched groups). If replace is a function then it must take a single argument (the match) and should return a replacement string.

If `count` is specified and non-zero then substitution will stop after this many substitutions are made. The `flags` argument is ignored.

Note: availability of this function depends on MicroPython port.

```python
re.DEBUG
```
Flag value, display debug information about compiled expression. (Availability depends on MicroPython port.)

Regex objects

Compiled regular expression. Instances of this class are created using `re.compile()`.

```python
regex.match(string)
regex.search(string)
regex.sub(replace, string[, count=0, flags=0, /])
```
Similar to the module-level functions `match()`, `search()` and `sub()`. Using methods is (much) more efficient if the same regex is applied to multiple strings.

```python
regex.split(string[, max_split=-1, /])
```
Split a string using regex. If `max_split` is given, it specifies maximum number of splits to perform. Returns list of strings (there may be up to `max_split+1` elements if it’s specified).

Match objects

Match objects as returned by `match()` and `search()` methods, and passed to the replacement function in `sub()`.

```python
match.group(index)
```
Return matching (sub)string. `index` is 0 for entire match, 1 and above for each capturing group. Only numeric groups are supported.

```python
match.groups()
```
Return a tuple containing all the substrings of the groups of the match.

Note: availability of this method depends on MicroPython port.

```python
match.start([index])
```
match.end([index])

Return the index in the original string of the start or end of the substring group that was matched. index defaults to the entire group, otherwise it will select a group.

Note: availability of these methods depends on MicroPython port.

match.span([index])

Returns the 2-tuple (match.start(index), match.end(index)).

Note: availability of this method depends on MicroPython port.

sys – system specific functions

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: sys.

Functions

sys.exit([retval=0, /])

Terminate current program with a given exit code. Underlyingly, this function raise as SystemExit exception. If an argument is given, its value given as an argument to SystemExit.

Constants

sys.argv

A mutable list of arguments the current program was started with.

sys.byteorder

The byte order of the system (“little” or “big”).

sys.implementation

Object with information about the current Python implementation. For CircuitPython, it has following attributes:

- name - string “circuitpython”
- version - tuple (major, minor, micro), e.g. (1, 7, 0)

This object is the recommended way to distinguish CircuitPython from other Python implementations (note that it still may not exist in the very minimal ports).

Difference to CPython

CPython mandates more attributes for this object, but the actual useful bare minimum is implemented in CircuitPython.
sys.maxsize

Maximum value which a native integer type can hold on the current platform, or maximum value representable by CircuitPython integer type, if it’s smaller than platform max value (that is the case for CircuitPython ports without long int support).

This attribute is useful for detecting “bitness” of a platform (32-bit vs 64-bit, etc.). It’s recommended to not compare this attribute to some value directly, but instead count number of bits in it:

```python
bits = 0
v = sys.maxsize
while v:
    bits += 1
    v >>= 1
if bits > 32:
    # 64-bit (or more) platform
    ...
else:
    # 32-bit (or less) platform
    # Note that on 32-bit platform, value of bits may be less than 32
    # (e.g. 31) due to peculiarities described above, so use "-> 16",
    # "-> 32", "-> 64" style of comparisons.
```

sys.modules

Dictionary of loaded modules. On some ports, it may not include built-in modules.

sys.path

A mutable list of directories to search for imported modules.

---

**Difference to CPython**

On MicroPython, an entry with the value ".frozen" will indicate that import should search frozen modules at that point in the search. If no frozen module is found then search will not look for a directory called .frozen, instead it will continue with the next entry in sys.path.

sys.platform

The platform that CircuitPython is running on. For OS/RTOS ports, this is usually an identifier of the OS, e.g. "linux". For baremetal ports it is an identifier of the chip on a board, e.g. "Microchip SAMD51". It thus can be used to distinguish one board from another. If you need to check whether your program runs on CircuitPython (vs other Python implementation), use sys.implementation instead.

sys.stderr

Standard error stream.

sys.stdin

Standard input stream.

sys.stdout

Standard output stream.

sys.version

Python language version that this implementation conforms to, as a string.

sys.version_info

Python language version that this implementation conforms to, as a tuple of ints.
Difference to CPython

Only the first three version numbers (major, minor, micro) are supported and they can be referenced only by index, not by name.

uctypes – access binary data in a structured way

This module implements “foreign data interface” for MicroPython. The idea behind it is similar to CPython’s `ctypes` modules, but the actual API is different, streamlined and optimized for small size. The basic idea of the module is to define data structure layout with about the same power as the C language allows, and then access it using familiar dot-syntax to reference sub-fields.

**Warning:** `uctypes` module allows access to arbitrary memory addresses of the machine (including I/O and control registers). Uncareful usage of it may lead to crashes, data loss, and even hardware malfunction.

See also:

Module `struct`

Standard Python way to access binary data structures (doesn’t scale well to large and complex structures).

Usage examples:

```python
import uctypes

# Example 1: Subset of ELF file header
# https://en.wikipedia.org/wiki/Executable_and_Linkable_Format#File_header
ELF_HEADER = {
    "EI_MAG": (0x0 | uctypes.ARRAY, 4 | uctypes.UINT8),
    "EI_DATA": 0x5 | uctypes.UINT8,
    "e_machine": 0x12 | uctypes.UINT16,
}

# "f" is an ELF file opened in binary mode
buf = f.read(uctypes.sizeof(ELF_HEADER, uctypes.LITTLE_ENDIAN))
header = uctypes.struct(uctypes.addressof(buf), ELF_HEADER, uctypes.LITTLE_ENDIAN)
assert header.EI_MAG == b"\x7fELF"
assert header.EI_DATA == 1, "Oops, wrong endianness. Could retry with uctypes.BIG_ENDIAN."
print("machine: ", hex(header.e_machine))

# Example 2: In-memory data structure, with pointers
COORD = {
    "x": 0 | uctypes.FLOAT32,
    "y": 4 | uctypes.FLOAT32,
}

STRUCT1 = {
    "data1": 0 | uctypes.UINT8,
    "data2": 4 | uctypes.UINT32,
    "ptr": (8 | uctypes.PTR, COORD),
}
```

(continues on next page)
# Suppose you have address of a structure of type STRUCT1 in "addr"
# uctypes.NATIVE is optional (used by default)
struct1 = uctypes.struct(addr, STRUCT1, uctypes.NATIVE)
print("x: ", struct1.ptr[0].x)

# Example 3: Access to CPU registers. Subset of STM32F4xx WWDG block
WWDG_LAYOUT = {
    "WWDG CR": (0, {
        # BFUINT32 here means size of the WWDG CR register
        "WDGA": 7 << uctypes.BF_POS | 1 << uctypes.BF_LEN | uctypes.BFUINT32,  
        "T": 0 << uctypes.BF_POS | 7 << uctypes.BF_LEN | uctypes.BFUINT32,  
        }),
    "WWDG CFR": (4, {
        "EWI": 9 << uctypes.BF_POS | 1 << uctypes.BF_LEN | uctypes.BFUINT32,  
        "WDGTB": 7 << uctypes.BF_POS | 2 << uctypes.BF_LEN | uctypes.BFUINT32,  
        "W": 0 << uctypes.BF_POS | 7 << uctypes.BF_LEN | uctypes.BFUINT32,  
        }),
    }
}

WWDG = uctypes.struct(0x40002c00, WWDG_LAYOUT)
WWDG.WWDG CFR.WDGTB = 0b10
WWDG.WWDG CR.WDGA = 1
print("Current counter:", WWDG.WWDG CR.T)

## Defining structure layout

Structure layout is defined by a “descriptor” - a Python dictionary which encodes field names as keys and other properties required to access them as associated values:

```python
{
    "field1": <properties>,
    "field2": <properties>,
    ...
}
```

Currently, uctypes requires explicit specification of offsets for each field. Offset are given in bytes from the structure start.

Following are encoding examples for various field types:

- **Scalar types:**
  ```python
  "field_name": offset | uctypesUINT32
  ```
  in other words, the value is a scalar type identifier ORed with a field offset (in bytes) from the start of the structure.

- **Recursive structures:**
"sub": (offset, {
    "b0": 0 | uctypes.UINT8,
    "b1": 1 | uctypes.UINT8,
})

i.e. value is a 2-tuple, first element of which is an offset, and second is a structure descriptor dictionary (note: offsets in recursive descriptors are relative to the structure it defines). Of course, recursive structures can be specified not just by a literal dictionary, but by referring to a structure descriptor dictionary (defined earlier) by name.

- Arrays of primitive types:

"arr": (offset | uctypes.ARRAY, size | uctypes.UINT8),

i.e. value is a 2-tuple, first element of which is ARRAY flag ORed with offset, and second is scalar element type ORed number of elements in the array.

- Arrays of aggregate types:

"arr2": (offset | uctypes.ARRAY, size, {"b": 0 | uctypes.UINT8}),

i.e. value is a 3-tuple, first element of which is ARRAY flag ORed with offset, second is a number of elements in the array, and third is a descriptor of element type.

- Pointer to a primitive type:

"ptr": (offset | uctypes.PTR, uctypes.UINT8),

i.e. value is a 2-tuple, first element of which is PTR flag ORed with offset, and second is a scalar element type.

- Pointer to an aggregate type:

"ptr2": (offset | uctypes.PTR, {"b": 0 | uctypes.UINT8}),

i.e. value is a 2-tuple, first element of which is PTR flag ORed with offset, second is a descriptor of type pointed to.

- Bitfields:

"bitf0": offset | uctypes.BFUINT16 | lsbit << uctypes.BF_POS | bitsize << uctypes.BF_LEN,

i.e. value is a type of scalar value containing given bitfield (typenames are similar to scalar types, but prefixes with BF), ORed with offset for scalar value containing the bitfield, and further ORed with values for bit position and bit length of the bitfield within the scalar value, shifted by BF_POS and BF_LEN bits, respectively. A bitfield position is counted from the least significant bit of the scalar (having position of 0), and is the number of right-most bit of a field (in other words, it’s a number of bits a scalar needs to be shifted right to extract the bitfield).

In the example above, first a UINT16 value will be extracted at offset 0 (this detail may be important when accessing hardware registers, where particular access size and alignment are required), and then bitfield whose rightmost bit is lsbit bit of this UINT16, and length is bitsize bits, will be extracted. For example, if lsbit is 0 and bitsize is 8, then effectively it will access least-significant byte of UINT16.

Note that bitfield operations are independent of target byte endianness, in particular, example above will access least-significant byte of UINT16 in both little- and big-endian structures. But it depends on the least significant bit being numbered 0. Some targets may use different numbering in their native ABI, but uctypes always uses the normalized numbering described above.
Module contents

class uctypes.struct(addr, descriptor, layout_type=NATIVE, /)
    Instantiate a “foreign data structure” object based on structure address in memory, descriptor (encoded as a
dictionary), and layout type (see below).

uctypes.LITTLE_ENDIAN
    Layout type for a little-endian packed structure. (Packed means that every field occupies exactly as many bytes
as defined in the descriptor, i.e. the alignment is 1).

uctypes.BIG_ENDIAN
    Layout type for a big-endian packed structure.

uctypes.NATIVE
    Layout type for a native structure - with data endianness and alignment conforming to the ABI of the system on
which MicroPython runs.

uctypes.sizeof(struct, layout_type=NATIVE, /)
    Return size of data structure in bytes. The struct argument can be either a structure class or a specific instantiated
structure object (or its aggregate field).

uctypes.addressof(obj)
    Return address of an object. Argument should be bytes, bytearray or other object supporting buffer protocol (and
address of this buffer is what actually returned).

uctypes.bytes_at(addr, size)
    Capture memory at the given address and size as bytes object. As bytes object is immutable, memory is actually
duplicated and copied into bytes object, so if memory contents change later, created object retains original value.

uctypes.bytearray_at(addr, size)
    Capture memory at the given address and size as bytearray object. Unlike bytes_at() function above, memory
is captured by reference, so it can be both written too, and you will access current value at the given memory
address.

uctypes.UINT8
uctypes.INT8
uctypes.UINT16
uctypes.INT16
uctypes.UINT32
uctypes.INT32
uctypes.UINT64
uctypes.INT64
    Integer types for structure descriptors. Constants for 8, 16, 32, and 64 bit types are provided, both signed and
unsigned.

uctypes.FLOAT32
uctypes.FLOAT64
    Floating-point types for structure descriptors.

uctypes.VOID
    VOID is an alias for UINT8, and is provided to conveniently define C’s void pointers: (uctypes.PTR, uctypes.
VOID).

uctypes.PTR
uctypes.ARRAY

Type constants for pointers and arrays. Note that there is no explicit constant for structures, it’s implicit: an aggregate type without PTR or ARRAY flags is a structure.

Structure descriptors and instantiating structure objects

Given a structure descriptor dictionary and its layout type, you can instantiate a specific structure instance at a given memory address using uctypes.struct() constructor. Memory address usually comes from following sources:

- Predefined address, when accessing hardware registers on a baremetal system. Lookup these addresses in datasheet for a particular MCU/SoC.
- As a return value from a call to some FFI (Foreign Function Interface) function.
- From uctypes.addressof(), when you want to pass arguments to an FFI function, or alternatively, to access some data for I/O (for example, data read from a file or network socket).

Structure objects

Structure objects allow accessing individual fields using standard dot notation: my_struct.substruct1.field1. If a field is of scalar type, getting it will produce a primitive value (Python integer or float) corresponding to the value contained in a field. A scalar field can also be assigned to.

If a field is an array, its individual elements can be accessed with the standard subscript operator [] - both read and assigned to.

If a field is a pointer, it can be dereferenced using [0] syntax (corresponding to C * operator, though [0] works in C too). Subscripting a pointer with other integer values but 0 are also supported, with the same semantics as in C.

Summing up, accessing structure fields generally follows the C syntax, except for pointer dereference, when you need to use [0] operator instead of *.

Limitations

1. Accessing non-scalar fields leads to allocation of intermediate objects to represent them. This means that special care should be taken to layout a structure which needs to be accessed when memory allocation is disabled (e.g. from an interrupt). The recommendations are:

   - Avoid accessing nested structures. For example, instead of mcu_registers.peripheral_a.register1, define separate layout descriptors for each peripheral, to be accessed as peripheral_a.register1. Or just cache a particular peripheral: peripheral_a = mcu_registers.peripheral_a. If a register consists of multiple bitfields, you would need to cache references to a particular register: reg_a = mcu_registers.peripheral_a.reg_a.

   - Avoid other non-scalar data, like arrays. For example, instead of peripheral_a.register[0] use peripheral_a.register0. Again, an alternative is to cache intermediate values, e.g. register0 = peripheral_a.register[0].

2. Range of offsets supported by the uctypes module is limited. The exact range supported is considered an implementation detail, and the general suggestion is to split structure definitions to cover from a few kilobytes to a few dozen of kilobytes maximum. In most cases, this is a natural situation anyway, e.g. it doesn’t make sense to define all registers of an MCU (spread over 32-bit address space) in one structure, but rather a peripheral block by peripheral block. In some extreme cases, you may need to split a structure in several parts artificially (e.g. if accessing native data structure with multi-megabyte array in the middle, though that would be a very synthetic case).
select – wait for events on a set of streams

**Warning:** Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

This module implements a subset of the corresponding CPython module, as described below. For more information, refer to the original CPython documentation: select.

This module provides functions to efficiently wait for events on multiple stream objects (select streams which are ready for operations).

**Functions**

`select.poll()`  
Create an instance of the Poll class.

`select.select(rlist, wlist, xlist[, timeout])`  
Wait for activity on a set of objects.

This function is provided by some MicroPython ports for compatibility and is not efficient. Usage of Poll is recommended instead.

**class Poll**

**Methods**

`poll.register(obj[, eventmask])`  
Register stream obj for polling. eventmask is logical OR of:

- `select.POLLIN` - data available for reading
- `select.POLLOUT` - more data can be written

Note that flags like `select.POLLHUP` and `select.POLLERR` are not valid as input eventmask (these are unsolicited events which will be returned from `poll()` regardless of whether they are asked for). This semantics is per POSIX.

`eventmask` defaults to `select.POLLIN | select.POLLOUT`.

It is OK to call this function multiple times for the same `obj`. Successive calls will update `obj`’s eventmask to the value of `eventmask` (i.e. will behave as `modify()`).

`poll.unregister(obj)`  
Unregister `obj` from polling.

`poll.modify(obj, eventmask)`  
Modify the `eventmask` for `obj`. If `obj` is not registered, `OSError` is raised with error of ENOENT.
poll \( \text{poll}(\text{timeout}=-1,/) \)

Wait for at least one of the registered objects to become ready or have an exceptional condition, with optional timeout in milliseconds (if \text{timeout} arg is not specified or -1, there is no timeout).

Returns list of (obj, event, ...) tuples. There may be other elements in tuple, depending on a platform and version, so don’t assume that its size is 2. The event element specifies which events happened with a stream and is a combination of select.POLL\* constants described above. Note that flags select.POLLHUP and select.POLLERR can be returned at any time (even if were not asked for), and must be acted on accordingly (the corresponding stream unregistered from poll and likely closed), because otherwise all further invocations of \text{poll()} may return immediately with these flags set for this stream again.

In case of timeout, an empty list is returned.

### Difference to CPython

Tuples returned may contain more than 2 elements as described above.

poll \( \text{ipoll}(\text{timeout}=-1,\text{flags}=0,/) \)

Like \text{poll.poll()}, but instead returns an iterator which yields a callee-owned tuples. This function provides efficient, allocation-free way to poll on streams.

If \text{flags} is 1, one-shot behaviour for events is employed: streams for which events happened will have their event masks automatically reset (equivalent to \text{poll.modify(obj, 0)}), so new events for such a stream won’t be processed until new mask is set with \text{poll.modify()}. This behaviour is useful for asynchronous I/O schedulers.

### Difference to CPython

This function is a MicroPython extension.

__Omitted functions in the string library__

A few string operations are not enabled on small builds (usually non-Express), due to limited flash memory: \text{string.center()}, \text{string.partition()}, \text{string.splitlines()}, \text{string.reversed()}.

### CircuitPython/MicroPython-specific libraries

Functionality specific to the CircuitPython/MicroPython implementation is available in the following libraries. These libraries may change significantly or be removed in future versions of CircuitPython.

**btree – simple BTree database**

**Warning:** Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

The \text{btree} module implements a simple key-value database using external storage (disk files, or in general case, a random-access stream). Keys are stored sorted in the database, and besides efficient retrieval by a key value, a database also supports efficient ordered range scans (retrieval of values with the keys in a given range). On the application
interface side, BTree database work as close a possible to a way standard \texttt{dict} type works, one notable difference is that both keys and values must be \texttt{bytes} objects (so, if you want to store objects of other types, you need to serialize them to \texttt{bytes} first).

The module is based on the well-known BerkelyDB library, version 1.xx.

Example:

```python
import btree

# First, we need to open a stream which holds a database
# This is usually a file, but can be in-memory database
# using \texttt{io.BytesIO}, a raw flash partition, etc.
# Oftentimes, you want to create a database file if it doesn't
# exist and open if it exists. Idiom below takes care of this.
# DO NOT open database with "a+b" access mode.
try:
    f = open("mydb", "r+b")
except IOError:
    f = open("mydb", "w+b")

# Now open a database itself
db = btree.open(f)

# The keys you add will be sorted internally in the database
db[b"3"] = b"three"
db[b"1"] = b"one"
db[b"2"] = b"two"

# Assume that any changes are cached in memory unless
# explicitly flushed (or database closed). Flush database
# at the end of each "transaction".
db.flush()

# Prints b'two'
print(db[b"2"])

# Iterate over sorted keys in the database, starting from b"2"
# until the end of the database, returning only values.
# Mind that arguments passed to values() method are *key* values.
# Prints:
# b'two'
# b'three'
for word in db.values(b"2"):
    print(word)

del db[b"2"]

# No longer true, prints False
print(b"2" in db)
```

(continues on next page)
for key in db:
    print(key)

db.close()

# Don't forget to close the underlying stream!
f.close()
btree.values([start_key, end_key[, flags]])
btree.items([start_key, end_key[, flags]])

These methods are similar to standard dictionary methods, but also can take optional parameters to iterate over a key sub-range, instead of the entire database. Note that for all 3 methods, start_key and end_key arguments represent key values. For example, values() method will iterate over values corresponding to the key range given. None values for start_key means “from the first key”, no end_key or its value of None means “until the end of database”. By default, range is inclusive of start_key and exclusive of end_key, you can include end_key in iteration by passing flags of btree.INCL. You can iterate in descending key direction by passing flags of btree.DESC. The flags values can be ORed together.

Constants

btree.INCL
A flag for keys(), values(), items() methods to specify that scanning should be inclusive of the end key.

btree.DESC
A flag for keys(), values(), items() methods to specify that scanning should be in descending direction of keys.

framebuf — frame buffer manipulation

Warning: Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

This module provides a general frame buffer which can be used to create bitmap images, which can then be sent to a display.

class FrameBuffer

The FrameBuffer class provides a pixel buffer which can be drawn upon with pixels, lines, rectangles, text and even other FrameBuffer’s. It is useful when generating output for displays.

For example:

```python
import framebuf

# FrameBuffer needs 2 bytes for every RGB565 pixel
fbuf = framebuf.FrameBuffer(bytearray(100 * 10 * 2), 100, 10, framebuf.RGB565)

fbuf.fill(0)
fbuf.text('MicroPython!', 0, 0, 0xffff)
fbuf.hline(0, 9, 96, 0xffff)
```
Constructors

```python
class framebuffer.FrameBuffer(buffer, width, height, format, stride=width, /)
```

Construct a FrameBuffer object. The parameters are:

- `buffer` is an object with a buffer protocol which must be large enough to contain every pixel defined by the width, height and format of the FrameBuffer.
- `width` is the width of the FrameBuffer in pixels
- `height` is the height of the FrameBuffer in pixels
- `format` specifies the type of pixel used in the FrameBuffer; permissible values are listed under Constants below. These set the number of bits used to encode a color value and the layout of these bits in `buffer`. Where a color value `c` is passed to a method, `c` is a small integer with an encoding that is dependent on the format of the FrameBuffer.
- `stride` is the number of pixels between each horizontal line of pixels in the FrameBuffer. This defaults to `width` but may need adjustments when implementing a FrameBuffer within another larger FrameBuffer or screen. The `buffer` size must accommodate an increased step size.

One must specify valid `buffer`, `width`, `height`, `format` and optionally `stride`. Invalid `buffer` size or dimensions may lead to unexpected errors.

Drawing primitive shapes

The following methods draw shapes onto the FrameBuffer.

- `FrameBuffer.fill(c)`
  Fill the entire FrameBuffer with the specified color.

- `FrameBuffer.pixel(x, y[, c])`
  If `c` is not given, get the color value of the specified pixel. If `c` is given, set the specified pixel to the given color.

- `FrameBuffer.hline(x, y, w, c)`
- `FrameBuffer.vline(x, y, h, c)`
- `FrameBuffer.line(x1, y1, x2, y2, c)`
  Draw a line from a set of coordinates using the given color and a thickness of 1 pixel. The `line` method draws the line up to a second set of coordinates whereas the `hline` and `vline` methods draw horizontal and vertical lines respectively up to a given length.

- `FrameBuffer.rect(x, y, w, h, c)`
- `FrameBuffer.fill_rect(x, y, w, h, c)`
  Draw a rectangle at the given location, size and color. The `rect` method draws only a 1 pixel outline whereas the `fill_rect` method draws both the outline and interior.
Drawing text

FrameBuffer.text(s, x, y[, c])

Write text to the FrameBuffer using the coordinates as the upper-left corner of the text. The color of the text can be defined by the optional argument but is otherwise a default value of 1. All characters have dimensions of 8x8 pixels and there is currently no way to change the font.

Other methods

FrameBuffer.scroll(xstep, ystep)

Shift the contents of the FrameBuffer by the given vector. This may leave a footprint of the previous colors in the FrameBuffer.

FrameBuffer.blit(buf, x, y, key=-1, palette=None)

Draw another FrameBuffer on top of the current one at the given coordinates. If key is specified then it should be a color integer and the corresponding color will be considered transparent: all pixels with that color value will not be drawn.

The palette argument enables blitting between FrameBuffers with differing formats. Typical usage is to render a monochrome or grayscale glyph/icon to a color display. The palette is a FrameBuffer instance whose format is that of the current FrameBuffer. The palette height is one pixel and its pixel width is the number of colors in the source FrameBuffer. The palette for an N-bit source needs 2**N pixels; the palette for a monochrome source would have 2 pixels representing background and foreground colors. The application assigns a color to each pixel in the palette. The color of the current pixel will be that of that palette pixel whose x position is the color of the corresponding source pixel.

Constants

framebuf.MONO_VLSB

Monochrome (1-bit) color format This defines a mapping where the bits in a byte are vertically mapped with bit 0 being nearest the top of the screen. Consequently each byte occupies 8 vertical pixels. Subsequent bytes appear at successive horizontal locations until the rightmost edge is reached. Further bytes are rendered at locations starting at the leftmost edge, 8 pixels lower.

framebuf.MONO_HLSB

Monochrome (1-bit) color format This defines a mapping where the bits in a byte are horizontally mapped. Each byte occupies 8 horizontal pixels with bit 7 being the leftmost. Subsequent bytes appear at successive horizontal locations until the rightmost edge is reached. Further bytes are rendered on the next row, one pixel lower.

framebuf.MONO_HMSB

Monochrome (1-bit) color format This defines a mapping where the bits in a byte are horizontally mapped. Each byte occupies 8 horizontal pixels with bit 0 being the leftmost. Subsequent bytes appear at successive horizontal locations until the rightmost edge is reached. Further bytes are rendered on the next row, one pixel lower.

framebuf.RGB565

Red Green Blue (16-bit, 5+6+5) color format

framebuf.GS2_HMSB

Grayscale (2-bit) color format

framebuf.GS4_HMSB

Grayscale (4-bit) color format
framebuf.GS8

Grayscale (8-bit) color format

micropython — access and control MicroPython internals

**Warning:** Though this MicroPython-based library may be available for use in some builds of CircuitPython, it is unsupported and its functionality may change in the future, perhaps significantly. As CircuitPython continues to develop, it may be changed to comply more closely with the corresponding standard Python library. You will likely need to change your code later if you rely on any non-standard functionality it currently provides.

### Functions

**micropython.const**(expr)

Used to declare that the expression is a constant so that the compile can optimise it. The use of this function should be as follows:

```python
from micropython import const

CONST_X = const(123)
CONST_Y = const(2 * CONST_X + 1)
```

Constants declared this way are still accessible as global variables from outside the module they are declared in. On the other hand, if a constant begins with an underscore then it is hidden, it is not available as a global variable, and does not take up any memory during execution.

This `const` function is recognised directly by the MicroPython parser and is provided as part of the `micropython` module mainly so that scripts can be written which run under both CPython and MicroPython, by following the above pattern.

**micropython.opt_level**(level)

If `level` is given then this function sets the optimisation level for subsequent compilation of scripts, and returns `None`. Otherwise it returns the current optimisation level.

The optimisation level controls the following compilation features:

- **Assertions:** at level 0 assertion statements are enabled and compiled into the bytecode; at levels 1 and higher assertions are not compiled.
- **Built-in __debug__ variable:** at level 0 this variable expands to `True`; at levels 1 and higher it expands to `False`.
- **Source-code line numbers:** at levels 0, 1 and 2 source-code line number are stored along with the bytecode so that exceptions can report the line number they occurred at; at levels 3 and higher line numbers are not stored.

The default optimisation level is usually level 0.

**micropython.mem_info**(verbose)

Print information about currently used memory. If the `verbose` argument is given then extra information is printed. The information that is printed is implementation dependent, but currently includes the amount of stack and heap used. In verbose mode it prints out the entire heap indicating which blocks are used and which are free.
micropython.qstr_info([verbose])

Print information about currently interned strings. If the `verbose` argument is given then extra information is printed.

The information that is printed is implementation dependent, but currently includes the number of interned strings and the amount of RAM they use. In verbose mode it prints out the names of all RAM-interned strings.

micropython.stack_use()

Return an integer representing the current amount of stack that is being used. The absolute value of this is not particularly useful, rather it should be used to compute differences in stack usage at different points.

micropython.heap_lock()

micropython.heap_unlock()

micropython.heap_locked()

Lock or unlock the heap. When locked no memory allocation can occur and a MemoryError will be raised if any heap allocation is attempted. `heap_locked()` returns a true value if the heap is currently locked.

These functions can be nested, i.e., `heap_lock()` can be called multiple times in a row and the lock-depth will increase, and then `heap_unlock()` must be called the same number of times to make the heap available again.

Both `heap_unlock()` and `heap_locked()` return the current lock depth (after unlocking for the former) as a non-negative integer, with 0 meaning the heap is not locked.

If the REPL becomes active with the heap locked then it will be forcefully unlocked.

Note: `heap_locked()` is not enabled on most ports by default, requires MICROPY_PY_MICROPYTHON_HEAP_LOCKED.

micropython.kbd_intr(chr)

Set the character that will raise a `KeyboardInterrupt` exception. By default this is set to 3 during script execution, corresponding to Ctrl-C. Passing -1 to this function will disable capture of Ctrl-C, and passing 3 will restore it.

This function can be used to prevent the capturing of Ctrl-C on the incoming stream of characters that is usually used for the REPL, in case that stream is used for other purposes.

micropython.schedule(func, arg)

Schedule the function `func` to be executed “very soon”. The function is passed the value `arg` as its single argument. “Very soon” means that the MicroPython runtime will do its best to execute the function at the earliest possible time, given that it is also trying to be efficient, and that the following conditions hold:

- A scheduled function will never preempt another scheduled function.
- Scheduled functions are always executed “between opcodes” which means that all fundamental Python operations (such as appending to a list) are guaranteed to be atomic.
- A given port may define “critical regions” within which scheduled functions will never be executed. Functions may be scheduled within a critical region but they will not be executed until that region is exited. An example of a critical region is a preempting interrupt handler (an IRQ).

A use for this function is to schedule a callback from a preempting IRQ. Such an IRQ puts restrictions on the code that runs in the IRQ (for example the heap may be locked) and scheduling a function to call later will lift those restrictions.

Note: If `schedule()` is called from a preempting IRQ, when memory allocation is not allowed and the callback to be passed to `schedule()` is a bound method, passing this directly will fail. This is because creating a reference to a bound method causes memory allocation. A solution is to create a reference to the method in the class constructor and to pass that reference to `schedule()`.
There is a finite queue to hold the scheduled functions and `schedule()` will raise a `RuntimeError` if the queue is full.

1.8.13 Glossary

**baremetal**
A system without a (full-fledged) operating system, for example an **MCU**-based system. When running on a baremetal system, MicroPython effectively functions like a small operating system, running user programs and providing a command interpreter (**REPL**).

**buffer protocol**
Any Python object that can be automatically converted into bytes, such as `bytes`, `bytearray`, `memoryview` and `str` objects, which all implement the “buffer protocol”.

**board**
Typically this refers to a printed circuit board (PCB) containing a **microcontroller** and supporting components. MicroPython firmware is typically provided per-board, as the firmware contains both MCU-specific functionality but also board-level functionality such as drivers or pin names.

**bytecode**
A compact representation of a Python program that generated by compiling the Python source code. This is what the VM actually executes. Bytecode is typically generated automatically at runtime and is invisible to the user. Note that while **CPython** and MicroPython both use bytecode, the format is different. You can also pre-compile source code offline using the **cross-compiler**.

**callee-owned tuple**
This is a MicroPython-specific construct where, for efficiency reasons, some built-in functions or methods may re-use the same underlying tuple object to return data. This avoids having to allocate a new tuple for every call, and reduces heap fragmentation. Programs should not hold references to callee-owned tuples and instead only extract data from them (or make a copy).

**CircuitPython**
A variant of MicroPython developed by Adafruit Industries.

**CPython**
CPython is the reference implementation of the Python programming language, and the most well-known one. It is, however, one of many implementations (including Python, IronPython, PyPy, and MicroPython). While MicroPython’s implementation differs substantially from CPython, it aims to maintain as much compatibility as possible.

**cross-compiler**
Also known as **mpy-cross**. This tool runs on your PC and converts a `.py` file containing MicroPython code into a `.mpy` file containing MicroPython bytecode. This means it loads faster (the board doesn’t have to compile the code), and uses less space on flash (the bytecode is more space efficient).

**driver**
A MicroPython library that implements support for a particular component, such as a sensor or display.

**FFI**
Acronym for Foreign Function Interface. A mechanism used by the MicroPython Unix port to access operating system functionality. This is not available on baremetal ports.

**filesystem**
Most MicroPython ports and boards provide a filesystem stored in flash that is available to user code via the standard Python file APIs such as `open()`. Some boards also make this internal filesystem accessible to the host via USB mass-storage.
frozen module
A Python module that has been cross compiled and bundled into the firmware image. This reduces RAM requirements as the code is executed directly from flash.

Garbage Collector
A background process that runs in Python (and MicroPython) to reclaim unused memory in the heap.

GPIO
General-purpose input/output. The simplest means to control electrical signals (commonly referred to as “pins”) on a microcontroller. GPIO typically allows pins to be either input or output, and to set or get their digital value (logical “0” or “1”). MicroPython abstracts GPIO access using the machine.Pin and machine.Signal classes.

GPIO port
A group of GPIO pins, usually based on hardware properties of these pins (e.g. controllable by the same register).

heap
A region of RAM where MicroPython stores dynamic data. It is managed automatically by the Garbage Collector. Different MCUs and boards have vastly different amounts of RAM available for the heap, so this will affect how complex your program can be.

interned string
An optimisation used by MicroPython to improve the efficiency of working with strings. An interned string is referenced by its (unique) identity rather than its address and can therefore be quickly compared just by its identifier. It also means that identical strings can be de-duplicated in memory. String interning is almost always invisible to the user.

MCU
Microcontroller. Microcontrollers usually have much less resources than a desktop, laptop, or phone, but are smaller, cheaper and require much less power. MicroPython is designed to be small and optimized enough to run on an average modern microcontroller.

MicroPython port
MicroPython supports different boards, RTOSe, and OSes, and can be relatively easily adapted to new systems. MicroPython with support for a particular system is called a “port” to that system. Different ports may have widely different functionality. This documentation is intended to be a reference of the generic APIs available across different ports (“MicroPython core”). Note that some ports may still omit some APIs described here (e.g. due to resource constraints). Any such differences, and port-specific extensions beyond the MicroPython core functionality, would be described in the separate port-specific documentation.

MicroPython Unix port
The unix port is one of the major MicroPython ports. It is intended to run on POSIX-compatible operating systems, like Linux, MacOS, FreeBSD, Solaris, etc. It also serves as the basis of Windows port. The Unix port is very useful for quick development and testing of the MicroPython language and machine-independent features. It can also function in a similar way to CPython’s python executable.

.mp4 file
The output of the cross-compiler. A compiled form of a .py file that contains MicroPython bytecode instead of Python source code.

native
Usually refers to “native code”, i.e. machine code for the target microcontroller (such as ARM Thumb, Xtensa, x86/x64). The @native decorator can be applied to a MicroPython function to generate native code instead of bytecode for that function, which will likely be faster but use more RAM.

port
Usually short for MicroPython port, but could also refer to GPIO port.

.py file
A file containing Python source code.
REPL
An acronym for “Read, Eval, Print, Loop”. This is the interactive Python prompt, useful for debugging or testing short snippets of code. Most MicroPython boards make a REPL available over a UART, and this is typically accessible on a host PC via USB.

stream
Also known as a “file-like object”. An Python object which provides sequential read-write access to the underlying data. A stream object implements a corresponding interface, which consists of methods like `read()`, `write()`, `readinto()`, `seek()`, `flush()`, `close()`, etc. A stream is an important concept in MicroPython; many I/O objects implement the stream interface, and thus can be used consistently and interchangeably in different contexts. For more information on streams in MicroPython, see the `io` module.

UART
Acronym for “Universal Asynchronous Receiver/Transmitter”. This is a peripheral that sends data over a pair of pins (TX & RX). Many boards include a way to make at least one of the UARTs available to a host PC as a serial port over USB.

1.8.14 CircuitPython

CircuitPython is a beginner friendly, open source version of Python for tiny, inexpensive computers called microcontrollers. Microcontrollers are the brains of many electronics including a wide variety of development boards used to build hobby projects and prototypes. CircuitPython in electronics is one of the best ways to learn to code because it connects code to reality. Simply install CircuitPython on a supported USB board usually via drag and drop and then edit a code.py file on the CIRCUITPY drive. The code will automatically reload. No software installs are needed besides a text editor (we recommend Mu for beginners.)

Starting with CircuitPython 7.0.0, some boards may only be connectable over Bluetooth Low Energy (BLE). Those boards provide serial and file access over BLE instead of USB using open protocols. (Some boards may use both USB and BLE.) BLE access can be done from a variety of apps including code.circuitpython.org.

CircuitPython features unified Python core APIs and a growing list of 300+ device libraries and drivers that work with it. These libraries also work on single board computers with regular Python via the Adafruit Blinka Library.

CircuitPython is based on MicroPython. See below for differences. Most, but not all, CircuitPython development is sponsored by Adafruit and is available on their educational development boards. Please support both MicroPython and Adafruit.
Get CircuitPython

Official binaries for all supported boards are available through circuitpython.org/downloads. The site includes stable, unstable and continuous builds. Full release notes are available through GitHub releases as well.

Documentation

Guides and videos are available through the Adafruit Learning System under the CircuitPython category. An API reference is also available on Read the Docs. A collection of awesome resources can be found at Awesome CircuitPython. Specifically useful documentation when starting out:

- Welcome to CircuitPython
- CircuitPython Essentials
- Example Code

Code Search

GitHub doesn’t currently support code search on forks. Therefore, CircuitPython doesn’t have code search through GitHub because it is a fork of MicroPython. Luckily, SourceGraph has free code search for public repos like CircuitPython. So, visit sourcegraph.com/github.com/adafruit/circuitpython to search the CircuitPython codebase online.

Contributing

See CONTRIBUTING.md for full guidelines but please be aware that by contributing to this project you are agreeing to the Code of Conduct. Contributors who follow the Code of Conduct are welcome to submit pull requests and they will be promptly reviewed by project admins. Please join the Discord too.

Branding

While we are happy to see CircuitPython forked and modified, we’d appreciate it if forked releases not use the name “CircuitPython” or the Blinka logo. “CircuitPython” means something special to us and those who learn about it. As a result, we’d like to make sure products referring to it meet a common set of requirements.

If you’d like to use the term “CircuitPython” and Blinka for your product here is what we ask:

- Your product is supported by the primary “adafruit/circuitpython” repo. This way we can update any custom code as we update the CircuitPython internals.
- Your product is listed on circuitpython.org (source here). This is to ensure that a user of your product can always download the latest version of CircuitPython from the standard place.
- Your product supports at least one standard “Workflow” for serial and file access:
  - With a user accessible USB plug which appears as a CIRCUITPY drive when plugged in.
  - With file and serial access over Bluetooth Low Energy using the BLE Workflow.
  - With file access over WiFi using the WiFi Workflow with serial access over USB and/or WebSocket.
- Boards that do not support the USB Workflow should be clearly marked.

If you choose not to meet these requirements, then we ask you call your version of CircuitPython something else (for example, SuperDuperPython) and not use the Blinka logo. You can say it is “CircuitPython-compatible” if most CircuitPython drivers will work with it.
Differences from MicroPython

CircuitPython:

- Supports native USB on most boards and BLE otherwise, allowing file editing without special tools.
- Floats (aka decimals) are enabled for all builds.
- Error messages are translated into 10+ languages.
- Concurrency within Python is not well supported. Interrupts and threading are disabled. async/await keywords are available on some boards for cooperative multitasking. Some concurrency is achieved with native modules for tasks that require it such as audio file playback.

Behavior

- The order that files are run and the state that is shared between them. CircuitPython’s goal is to clarify the role of each file and make each file independent from each other.
  - boot.py runs only once on start up before workflows are initialized. This lays the groundwork for configuring USB at startup rather than it being fixed. Since serial is not available, output is written to boot.out.txt.
  - code.py (or main.py) is run after every reload until it finishes or is interrupted. After it is done running, the vm and hardware is reinitialized. This means you cannot read state from code.py in the REPL anymore, as the REPL is a fresh vm. CircuitPython’s goal for this change includes reducing confusion about pins and memory being used.
  - After the main code is finished the REPL can be entered by pressing any key.
  - Autoreload state will be maintained across reload.
- Adds a safe mode that does not run user code after a hard crash or brown out. This makes it possible to fix code that causes nasty crashes by making it available through mass storage after the crash. A reset (the button) is needed after it’s fixed to get back into normal mode.
- Safe mode may be handled programmatically by providing a safemode.py. safemode.py is run if the board has reset due to entering safe mode, unless the safe mode initiated by the user by pressing button(s). USB is not available so nothing can be printed. safemode.py can determine why the safe mode occurred using supervisor.runtime.safe_mode_reason, and take appropriate action. For instance, if a hard crash occurred, safemode.py may do a microcontroller.reset() to automatically restart despite the crash. If the battery is low, but is being charged, safemode.py may put the board in deep sleep for a while. Or it may simply reset, and have code.py check the voltage and do the sleep.
- RGB status LED indicating CircuitPython state. - One green flash - code completed without error. - Two red flashes - code ended due to an exception. - Three yellow flashes - safe mode. May be due to CircuitPython internal error.
- Re-runs code.py or other main file after file system writes by a workflow. (Disable with supervisor.disable_autoreload())
- Autoreload is disabled while the REPL is active.
- code.py may also be named `code.txt`, main.py, or main.txt.
- boot.py may also be named boot.txt.
- safemode.py may also be named safemode.txt.
API

- Unified hardware APIs. Documented on ReadTheDocs.
- API docs are Python stubs within the C files in shared-bindings.
- No machine API.

Modules

- No module aliasing. (uos and utime are not available as os and time respectively.) Instead os, time, and random are CPython compatible.
- New storage module which manages file system mounts. (Functionality from uos in MicroPython.)
- Modules with a CPython counterpart, such as time, os and random, are strict subsets of their CPython version. Therefore, code from CircuitPython is runnable on CPython but not necessarily the reverse.
- tick count is available as time.monotonic()

Project Structure

Here is an overview of the top-level source code directories.

Core

The core code of MicroPython is shared amongst ports including CircuitPython:

- docs High level user documentation in Sphinx reStructuredText format.
- drivers External device drivers written in Python.
- examples A few example Python scripts.
- extmod Shared C code used in multiple ports’ modules.
- lib Shared core C code including externally developed libraries such as FATFS.
- logo The CircuitPython logo.
- mpy-cross A cross compiler that converts Python files to byte code prior to being run in MicroPython. Useful for reducing library size.
- py Core Python implementation, including compiler, runtime, and core library.
- shared-bindings Shared definition of Python modules, their docs and backing C APIs. Ports must implement the C API to support the corresponding module.
- shared-module Shared implementation of Python modules that may be based on common-hal.
- tests Test framework and test scripts.
- tools Various tools, including the pyboard.py module.
Ports

Ports include the code unique to a microcontroller line.

<table>
<thead>
<tr>
<th>Supported</th>
<th>Support status</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmel-samd</td>
<td>SAMD21 stable</td>
</tr>
<tr>
<td>cxd56</td>
<td>stable</td>
</tr>
<tr>
<td>espressif</td>
<td>ESP32 beta</td>
</tr>
<tr>
<td>litex</td>
<td>alpha</td>
</tr>
<tr>
<td>mimxrt10xx</td>
<td>alpha</td>
</tr>
<tr>
<td>nrf</td>
<td>stable</td>
</tr>
<tr>
<td>raspberrypi</td>
<td>stable</td>
</tr>
<tr>
<td>efr32</td>
<td>alpha</td>
</tr>
<tr>
<td>stm</td>
<td>F4 stable</td>
</tr>
<tr>
<td>unix</td>
<td>alpha</td>
</tr>
</tbody>
</table>

- **stable** Highly unlikely to have bugs or missing functionality.
- **beta** Being actively improved but may be missing functionality and have bugs.
- **alpha** Will have bugs and missing functionality.

Boards

- Each port has a boards directory containing boards which belong to a specific microcontroller line.
- A list of native modules supported by a particular board can be found [here](#).

Back to Top

1.8.15 Contributing

Please note that this project is released with a [Contributor Code of Conduct](#). By participating in this project you agree to abide by its terms. Participation covers any forum used to converse about CircuitPython including unofficial and official spaces. Failure to do so will result in corrective actions such as time out or ban from the project.

Licensing

By contributing to this repository you are certifying that you have all necessary permissions to license the code under an MIT License. You still retain the copyright but are granting many permissions under the MIT License.

If you have an employment contract with your employer please make sure that they don’t automatically own your work product. Make sure to get any necessary approvals before contributing. Another term for this contribution off-hours is moonlighting.
Ways to contribute

As CircuitPython grows, there are more and more ways to contribute. Here are some ideas:

- Build a project with CircuitPython and share how to do it online.
- Test the latest libraries and CircuitPython versions with your projects and file issues for any bugs you find.
- Contribute Python code to CircuitPython libraries that support new devices or features of an existing device.
- Contribute C code to CircuitPython which fixes an open issue or adds a new feature.

Getting started with C

CircuitPython developer Dan Halbert (@dhalbert) has written up build instructions using native build tools here. For SAMD21 debugging workflow tips check out this learn guide from Scott (@tannewt).

Developer contacts

Scott Shawcroft (@tannewt) is the lead developer of CircuitPython and is sponsored by Adafruit Industries LLC. Scott is usually available during US West Coast working hours. Dan Halbert (@dhalbert) and Kattni Rembor (@kattni) are also sponsored by Adafruit Industries LLC and are usually available during US East Coast daytime hours including some weekends.

They are all reachable on Discord, GitHub issues and the Adafruit support forum.

Code guidelines

We aim to keep our code and commit style compatible with MicroPython upstream. Please review their code conventions to do so. Familiarity with their design philosophy is also useful though not always applicable to CircuitPython.

Furthermore, CircuitPython has a design guide that covers a variety of different topics. Please read it as well.

1.8.16 Building CircuitPython

Welcome to CircuitPython!

This document is a quick-start guide only.

Detailed guides on how to build CircuitPython can be found in the Adafruit Learn system at https://learn.adafruit.com/building-circuitpython/

Setup

Please ensure you set up your build environment appropriately, as per the guide. You will need:

- Linux: https://learn.adafruit.com/building-circuitpython/linux
- MacOS: https://learn.adafruit.com/building-circuitpython/macos
Submodules

This project has a bunch of git submodules. You will need to update them regularly.

In the root folder of the CircuitPython repository, execute the following:

```
make fetch-all-submodules
```

Or, in the ports directory for the particular port you are building, do:

```
make fetch-port-submodules
```

Required Python Packages

Failing to install these will prevent from properly building.

```
pip3 install -r requirements-dev.txt
```

If you run into an error installing minify_html, you may need to install rust.

mpy-cross

As part of the build process, mpy-cross is needed to compile .py files into .mpy files. To compile (or recompile) mpy-cross:

```
make -C mpy-cross
```

1.8.17 Building

There a number of ports of CircuitPython! To build for your board, change to the appropriate ports directory and build.

Examples:

```
cd ports/atmel-samd
make BOARD=circuitplayground_express

cd ports/nrf
make BOARD=circuitplayground_bluefruit
```

If you aren't sure what boards exist, have a peek in the boards subdirectory of your port. If you have a fast computer with many cores, consider adding -j to your build flags, such as -j17 on a 6-core 12-thread machine.
1.8.18 Testing

If you are working on changes to the core language, you might find it useful to run the test suite. The test suite in the top level tests directory. It needs the unix port to run.

```
cd ports/unix
make axtls
make micropython
```

Then you can run the test suite:

```
cd ../../tests
./run-tests.py
```

A successful run will say something like

```
676 tests performed (19129 individual testcases)
676 tests passed
30 tests skipped: buffered_writer builtin_help builtin_range_binop class_delattr_setattr...
  cmd_parsetree extra_coverage framebuf1 framebuf16 framebuf2 framebuf4 framebuf8...
  framebuffer subclass mpy_invalid namedtuple_asdict non_compliant resource_stream schedule...
  sys_getsizeof urandom_extra ure_groups ure_span ure_sub ure_sub_unmatched vks_basic...
  vfs_fat_fileio1 vfs_fat_fileio2 vfs_fat_more vfs_fat_oldproto vfs_fat_ramdisk vfs_
  userfs
```

1.8.19 Debugging

The easiest way to debug CircuitPython on hardware is with a JLink device, JLinkGDBServer, and an appropriate GDB. Instructions can be found at https://learn.adafruit.com/debugging-the-samd21-with-gdb

If using JLink, you’ll need both the JLinkGDBServer and arm-none-eabi-gdb running.

Example:

```
JLinkGDBServer -if SWD -device ATSAMD51J19
arm-none-eabi-gdb build-metro_m4_express/firmware.elf -iex "target extended-remote :2331"
```

If your port/build includes arm-none-eabi-gdb-py, consider using it instead, as it can be used for better register debugging with https://github.com/bnahill/PyCortexMDebug

1.8.20 Code Quality Checks

We apply code quality checks using pre-commit. Install pre-commit once per system with

```
python3 -mpip install pre-commit
```

Activate it once per git clone with

```
pre-commit install
```

Pre-commit also requires some additional programs to be installed through your package manager:

- Standard Unix tools such as make, find, etc
- The gettext package, any modern version
• uncrustify version 0.71 (0.72 is also tested and OK; 0.75 is not OK)

Each time you create a git commit, the pre-commit quality checks will be run. You can also run them e.g., with `pre-commit run foo.c` or `pre-commit run --all` to run on all files whether modified or not.

Some pre-commit quality checks require your active attention to resolve, others (such as the formatting checks of uncrustify) are made automatically and must simply be incorporated into your code changes by committing them.

1.8.21 Adafruit Community Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and leaders pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level or type of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

We are committed to providing a friendly, safe and welcoming environment for all.

Examples of behavior that contributes to creating a positive environment include:

• Be kind and courteous to others
• Using welcoming and inclusive language
• Being respectful of differing viewpoints and experiences
• Collaborating with other community members
• Gracefully accepting constructive criticism
• Focusing on what is best for the community
• Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

• The use of sexualized language or imagery and sexual attention or advances
• The use of inappropriate images, including in a community member’s avatar
• The use of inappropriate language, including in a community member’s nickname
• Any spamming, flaming, baiting or other attention-stealing behavior
• Excessive or unwelcome helping; answering outside the scope of the question asked
• Trolling, insulting/derogatory comments, and personal or political attacks
• Promoting or spreading disinformation, lies, or conspiracy theories against a person, group, organisation, project, or community
• Public or private harassment
• Publishing others’ private information, such as a physical or electronic address, without explicit permission
• Other conduct which could reasonably be considered inappropriate
The goal of the standards and moderation guidelines outlined here is to build and maintain a respectful community. We ask that you don’t just aim to be “technically unimpeachable”, but rather try to be your best self.

We value many things beyond technical expertise, including collaboration and supporting others within our community. Providing a positive experience for other community members can have a much more significant impact than simply providing the correct answer.

**Our Responsibilities**

Project leaders are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project leaders have the right and responsibility to remove, edit, or reject messages, comments, commits, code, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any community member for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

**Moderation**

Instances of behaviors that violate the Adafruit Community Code of Conduct may be reported by any member of the community. Community members are encouraged to report these situations, including situations they witness involving other community members.

You may report in the following ways:

In any situation, you may send an email to support@adafruit.com.

On the Adafruit Discord, you may send an open message from any channel to all Community Moderators by tagging @community moderators. You may also send an open message from any channel, or a direct message to @kattni#1507, @tannewt#4653, @dan#1614, @cater#2442, @sommersoft#0222, @Mr. Certainly#0472 or @Andon#8175.

Email and direct message reports will be kept confidential.

In situations on Discord where the issue is particularly egregious, possibly illegal, requires immediate action, or violates the Discord terms of service, you should also report the message directly to Discord.

These are the steps for upholding our community’s standards of conduct.

1. Any member of the community may report any situation that violates the Adafruit Community Code of Conduct. All reports will be reviewed and investigated.
2. If the behavior is an egregious violation, the community member who committed the violation may be banned immediately, without warning.
3. Otherwise, moderators will first respond to such behavior with a warning.
4. Moderators follow a soft “three strikes” policy - the community member may be given another chance, if they are receptive to the warning and change their behavior.
5. If the community member is un receptive or unreasonable when warned by a moderator, or the warning goes unheeded, they may be banned for a first or second offense. Repeated offenses will result in the community member being banned.
Scope

This Code of Conduct and the enforcement policies listed above apply to all Adafruit Community venues. This includes but is not limited to any community spaces (both public and private), the entire Adafruit Discord server, and Adafruit GitHub repositories. Examples of Adafruit Community spaces include but are not limited to meet-ups, audio chats on the Adafruit Discord, or interaction at a conference.

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. As a community member, you are representing our community, and are expected to behave accordingly.

Attribution


For other projects adopting the Adafruit Community Code of Conduct, please contact the maintainers of those projects for enforcement. If you wish to use this code of conduct for your own project, consider explicitly mentioning your moderation policy or making a copy with your own moderation policy so as to avoid confusion.

1.8.22 MicroPython & CircuitPython License

MIT License

Copyright (c) 2013-2022 Damien P. George and others

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.8.23 WebUSB Serial Support

To date, this has only been tested on one port (espressif), on one board (espressif_kaluga_1).
What it does

If you have ever used CircuitPython on a platform with a graphical LCD display, you have probably already seen multiple “consoles” in use (although the LCD console is “output only”).

New compile-time option CIRCUITPY_USB_VENDOR enables an additional “console” that can be used in parallel with the original (CDC) serial console.

Web pages that support the WebUSB standard can connect to the “vendor” interface and activate this WebUSB serial console at any time.

You can type into either console, and CircuitPython output is sent to all active consoles.

One example of a web page you can use to test drive this feature can be found at:
https://adafruit.github.io/Adafruit_TinyUSB_Arduino/examples/webusb-serial/index.html

How to enable

Update your platform’s mpconfigboard.mk file to enable and disable specific types of USB interfaces.

CIRCUITPY_USB_HID = xxx CIRCUITPY_USB_MIDI = xxx CIRCUITPY_USB_VENDOR = xxx

On at least some of the hardware platforms, the maximum number of USB endpoints is fixed. For example, on the ESP32-S2, you must pick only one of the above 3 interfaces to be enabled.

Original espressif_kaluga_1 mpconfigboard.mk settings:

CIRCUITPY_USB_HID = 1 CIRCUITPY_USB_MIDI = 0 CIRCUITPY_USB_VENDOR = 0

Settings to enable WebUSB instead:

CIRCUITPY_USB_HID = 0 CIRCUITPY_USB_MIDI = 0 CIRCUITPY_USB_VENDOR = 1

Notice that to enable VENDOR on ESP32-S2, we had to give up HID. There may be platforms that can have both, or even all three.

Implementation Notes

CircuitPython uses the tinyusb library.

The tinyusb library already has support for WebUSB serial. The tinyusb examples already include a “WebUSB serial” example.

Sidenote - The use of the term "vendor" instead of "WebUSB" was done to match tinyusb.

Basically, this feature was ported into CircuitPython by pulling code snippets out of the tinyusb example, and putting them where they best belonged in the CircuitPython codebase.
TODO: This needs to be reworked for dynamic USB descriptors.
INDICES AND TABLES

• genindex
• modindex
• search
__
_bleio, 106
_eve, 118
_pew, 126
_pixmap, 127
_stage, 128

a
adafruit_bus_device, 129
adafruit_bus_device.i2c_device, 129
adafruit_bus_device.spi_device, 131
adafruit_pixelbuf, 131
aesio, 133
alarm, 134
alarm.pin, 134
alarm.time, 135
alarm.touch, 135
analogbufio, 137
analogio, 138
array, 363
atexit, 140
audiobusio, 140
audiocore, 144
audioio, 146
audiomixer, 148
audiompg3, 150
audiopwmio, 151

b
binascii, 363
bitbangio, 153
bitmaptools, 157
bitops, 162
board, 163
btree, 380
builtins, 358
busio, 163

camera, 170
canio, 171
collections, 364

countio, 176
cyw43, 177
digitalio, 178
displayio, 180
dualbank, 195
eerrno, 365
espcamera, 196
espidf, 201
espnw, 202
espulp, 205
floppyio, 206
fontio, 207
framebuf, 383
framebufferio, 208
frequencyio, 209
gc, 366
getpass, 211
gifio, 211
gnss, 214
hashlib, 215
heapq, 362

i2c, 216
imagecapture, 218
io, 367
ipaddress, 219
is31fl3741, 220
json, 368
INDEX

Symbols

.py file, 389
.mpypy file, 389
_ArrayLike (in module ulab.numpy), 293
_DType (in module ulab.numpy), 293
_DisplayBus (in module displayio), 183
_EVE (class in _eve), 118
_T (in module random), 251
_Uname (class in os), 239
__bool__ (alarm.SleepMemory method), 137
__bool__ (displayio.Group method), 190
__bool__ (displayio.Palette method), 192
__bool__ (espnow.ESPNow method), 203
__bool__ (keypad.EventQueue method), 223
__bool__ (memorymap.AddressRange method), 230
__bool__ (nvram.ByteArray method), 237
__bool__ (ps2io.Ps2 method), 244
__bool__ (pulseio.PulseIn method), 246
__call__ (synthio.MathOperation method), 279
__contains__ (bitbufio.BufferedIn method), 382
__contains__ (displayio.Groups method), 190
__del__ (mdns.RemoteService method), 229
__delitem__ (bufio.buffer method), 382
__delitem__ (displayio.Group method), 190
__enter__ (adafruit_bus_device.i2c_device.I2CDevice method), 130
__enter__ (adafruit_bus_device.spi_device.SPIDevice method), 131
__enter__ (analogbufio.BufferedIn method), 138
__enter__ (analogio.AnalogIn method), 139
__enter__ (analogio.AnalogOut method), 140
__enter__ (audiobusio.I2SOut method), 142
__enter__ (audiobusio.PDMIn method), 143
__enter__ (audiocore.RawSample method), 144
__enter__ (audiocore.WaveFile method), 145
__enter__ (audioio.AudioOut method), 147
__enter__ (audiomixer.Mixer method), 149
__enter__ (audiompg3.MPG3Decoder method), 151
__enter__ (audiopwmio.PWMAudioOut method), 152
__enter__ (bitbangio.I2C method), 154
__enter__ (bitbangio.SPI method), 156
__enter__ (busio.I2C method), 164
__enter__ (busio.I2C method), 166
__enter__ (busio.SPI method), 169
__enter__ (canio.CAN method), 174
__enter__ (canio.Listener method), 174
__enter__ (countio.Counter method), 177
__enter__ (digitalio.DigitalInOut method), 179
__enter__ (espcamera.Camera method), 201
__enter__ (espnow.ESPNow method), 203
__enter__ (espulp.ULP method), 205
__enter__ (frequencyio.FrequencyIn method), 210
__enter__ (gifio.GifWriter method), 211
__enter__ (gifio.OnDiskGif method), 213
__enter__ (i2c.target.I2CDevice method), 217
__enter__ (i2c.target.I2CDevice method), 218
__enter__ (imagecapture.ParallelImageCapture method), 219
__enter__ (keypad.KeyMatrix method), 223
__enter__ (keypad.Keys method), 224
__enter__ (keypad.ShiftRegisterKeys method), 225
__enter__ (memorymonitor.AllocationAlarm method), 231
__enter__ (memorymonitor.AllocationSize method), 231
__enter__ (onedewire.OneWire method), 238
__enter__ (ps2io.Ps2 method), 243
__enter__ (pulseio.PulseIn method), 245
__enter__ (pulseio.PulseOut method), 246
__enter__ (pwmio.PWMAudioOut method), 249
__enter__ (rotaryio.IncrementalEncoder method), 254
__enter__ (rp2pio.StateMachine method), 257
__enter__ (sdioio.SDCard method), 264
__enter__ (socketpool.Socket method), 265
__enter__ (ssl.SSLSocket method), 268
__enter__ (synthio.MidiTrack method), 281
__enter__ (synthio.Synthesizer method), 284
__enter__ (touchio.Touchln method), 288
__enter__ (usb_host.Port method), 305
__eq__ (_bleio.Address method), 110
__eq__ (_bleio.UID method), 117
__eq__ (ipaddress.IPv4Address method), 219
Index
__getitem__ () (memorymap.AddressRange method), 230
__getitem__ () (nvm.ByteArray method), 237
_bleio module, 106
_eve module, 106
_pwm module, 118
_pixelmap module, 127
_stage module, 128

A
a (synthio.Math attribute), 280
a2b_base64() (in module binascii), 363
ABS (synthio.MathOperation attribute), 279
abs() (in module buildins), 359
abs() (in module ulab.numpy.array), 290
accept() (socketpool.Socket method), 265
accept() (ssl.SSLSocket method), 268
ack() (i2ctarget.I2CTargetRequest method), 218
acos() (in module math), 226
acos() (in module ulab.numpy.array), 294
acosh() (in module math), 228
acosh() (in module ulab.numpy.array), 294
adafruit_bus_device module, 129
adafruit_bus_device.i2c_device module, 129
adafruit_bus_device.spi_device module, 131
adafruit_pixelbuf module, 131
Adapter (class in _bleio), 107
adapter (in module _bleio), 106
ADD_DIV (synthio.MathOperation attribute), 279
add_frame() (gifio.GifWriter method), 211
ADD_SUB (synthio.MathOperation attribute), 279
add_to_characteristic() (_bleio.Descriptor class method), 114
add_to_service() (_bleio.Characteristic method), 111
address (_bleio.Adapter attribute), 107
address (_bleio.ScanEntry attribute), 116
Address (class in _bleio), 109
address (espccamera.Camera attribute), 200
address (i2ctarget.I2CTargetRequest attribute), 217
address_bytes (_bleio.Address attribute), 109
addressof() (in module uctypes), 377
AddressRange (class in memorymap), 230
advertise_service() (mdns.Server method), 229
advertisement_bytes (_bleio.ScanEntry attribute), 116

advertising (_bleio.Adapter attribute), 107
ae_level (espccamera.Camera attribute), 200
aec2 (espccamera.Camera attribute), 200
aec_value (espccamera.Camera attribute), 200
AES (class in aesio), 133
aesio module, 133
AF_INET (socketpool.SocketPool attribute), 266
AF_INET6 (socketpool.SocketPool attribute), 266
agc_gain (espccamera.Camera attribute), 200
alarm module, 134
alarm.pin module, 134
alarm.time module, 135
alarm.touch module, 135
all() (in module bigints), 359
AllocationAlarm (class in memorymonitor), 230
AllocationError, 230
AllocationSize (class in memorymonitor), 231
alphablend() (in module bitmaptools), 158
AlphaFunc (_eve._EVE method), 118
altitude (gnss.GNSS attribute), 214
amplitude (synthio.Note attribute), 281
analogbufio module, 137
AnalogIn (class in analogio), 139
analogio module, 138
AnalogOut (class in analogio), 139
any() (in module bigints), 359
ap_active (wifi.Radio attribute), 312
ap_info (wifi.Radio attribute), 313
append() (array.array method), 363
append() (collections.deque method), 364
append() (displayio.Group method), 190
append() (espnow.Peers method), 204
arange() (in module ulab.numpy.array), 292
arch (espulp.ULP attribute), 205
Architecture (class in espulp), 205
arctan2() (in module ulab.numpy.array), 295
argmax() (in module ulab.numpy.array), 293
argmin() (in module ulab.numpy.array), 293
argsort() (in module ulab.numpy.array), 293
cg (in module array), 372
ArithmeticError, 361
around() (in module ulab.numpy.array), 294
array module, 363
array (class in array), 363
ARRAY (in module uctypes), 377
array() (in module ulab.numpy.array), 294
<table>
<thead>
<tr>
<th>Function/Module/Attribute</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrayblit()</td>
<td>160</td>
</tr>
<tr>
<td>atan()</td>
<td>226</td>
</tr>
<tr>
<td>atan2()</td>
<td>226</td>
</tr>
<tr>
<td>asin()</td>
<td>229</td>
</tr>
<tr>
<td>asinh()</td>
<td>229</td>
</tr>
<tr>
<td>asin()</td>
<td>294</td>
</tr>
<tr>
<td>asin()</td>
<td>294</td>
</tr>
<tr>
<td>atexit</td>
<td>140</td>
</tr>
<tr>
<td>attack_time</td>
<td>277</td>
</tr>
<tr>
<td>attack_time</td>
<td>276</td>
</tr>
<tr>
<td>AttackTime()</td>
<td>110</td>
</tr>
<tr>
<td>AttributeError</td>
<td>361</td>
</tr>
<tr>
<td>audiobusio</td>
<td>140</td>
</tr>
<tr>
<td>audiocore</td>
<td>144</td>
</tr>
<tr>
<td>audioio</td>
<td>146</td>
</tr>
<tr>
<td>audiomixer</td>
<td>148</td>
</tr>
<tr>
<td>audiomp3</td>
<td>150</td>
</tr>
<tr>
<td>AudioOut</td>
<td>146</td>
</tr>
<tr>
<td>AuthMode()</td>
<td>310</td>
</tr>
<tr>
<td>authmode()</td>
<td>312</td>
</tr>
<tr>
<td>auto_refresh</td>
<td>185</td>
</tr>
<tr>
<td>auto_refresh (displayio.BitmapBufferDisplay attribute)</td>
<td>208</td>
</tr>
<tr>
<td>AUTO_RELOAD</td>
<td>273</td>
</tr>
<tr>
<td>auto_restart</td>
<td>173</td>
</tr>
<tr>
<td>auto_write</td>
<td>127</td>
</tr>
<tr>
<td>auto_write (adafruit_pixelbuf.ImagingBuffer attribute)</td>
<td>132</td>
</tr>
<tr>
<td>autoreload</td>
<td>274</td>
</tr>
<tr>
<td>awb_gain</td>
<td>200</td>
</tr>
<tr>
<td>b (synthio.Math attribute)</td>
<td>280</td>
</tr>
<tr>
<td>b2a_base64()</td>
<td>364</td>
</tr>
<tr>
<td>background_write()</td>
<td>258</td>
</tr>
<tr>
<td>band_pass_filter()</td>
<td>284</td>
</tr>
<tr>
<td>baremetal</td>
<td>388</td>
</tr>
<tr>
<td>BaseException</td>
<td>361</td>
</tr>
<tr>
<td>baudrate (busio.UART attribute)</td>
<td>169</td>
</tr>
<tr>
<td>baudrate (canio.CAN attribute)</td>
<td>173</td>
</tr>
<tr>
<td>Begin() (eve._EVE method)</td>
<td>118</td>
</tr>
<tr>
<td>bend (synthio.Note attribute)</td>
<td>281</td>
</tr>
<tr>
<td>BIG_ENDIAN (in module uctypes)</td>
<td>377</td>
</tr>
<tr>
<td>bin() (in module builtins)</td>
<td>359</td>
</tr>
<tr>
<td>binascii</td>
<td>363</td>
</tr>
<tr>
<td>bind() (socketpool.Socket method)</td>
<td>265</td>
</tr>
<tr>
<td>bind() (ssl.SSLContext method)</td>
<td>268</td>
</tr>
<tr>
<td>Biquad (class in synthio)</td>
<td>277</td>
</tr>
<tr>
<td>bisect() (in module numpy)</td>
<td>297</td>
</tr>
<tr>
<td>bit_transpose()</td>
<td>162</td>
</tr>
<tr>
<td>bitbangio</td>
<td>153</td>
</tr>
<tr>
<td>Bitmap (in displayio)</td>
<td>181</td>
</tr>
<tr>
<td>bitmap (displayio.TileGrid attribute)</td>
<td>194</td>
</tr>
<tr>
<td>bitmap (fontio.BuiltinFont attribute)</td>
<td>207</td>
</tr>
<tr>
<td>bitmap (gifio.OnDiskGif attribute)</td>
<td>213</td>
</tr>
<tr>
<td>bitmapExtFormat()</td>
<td>118</td>
</tr>
<tr>
<td>bitmapHandle()</td>
<td>118</td>
</tr>
<tr>
<td>bitmapLayout()</td>
<td>119</td>
</tr>
<tr>
<td>bitmapLayoutH()</td>
<td>119</td>
</tr>
<tr>
<td>bitmapSize()</td>
<td>119</td>
</tr>
<tr>
<td>bitmapSizeH()</td>
<td>119</td>
</tr>
<tr>
<td>bitmapSource()</td>
<td>119</td>
</tr>
<tr>
<td>bitmapSwizzle()</td>
<td>119</td>
</tr>
<tr>
<td>bitmaptools</td>
<td>157</td>
</tr>
<tr>
<td>BitmapTransformA()</td>
<td>120</td>
</tr>
<tr>
<td>BitmapTransformB()</td>
<td>120</td>
</tr>
<tr>
<td>BitmapTransformC()</td>
<td>120</td>
</tr>
<tr>
<td>BitmapTransformD()</td>
<td>120</td>
</tr>
<tr>
<td>BitmapTransformE()</td>
<td>121</td>
</tr>
<tr>
<td>BitmapTransformF()</td>
<td>121</td>
</tr>
<tr>
<td>bitops</td>
<td>162</td>
</tr>
<tr>
<td>bits_per_sample (audiocore.WaveFile attribute)</td>
<td>145</td>
</tr>
<tr>
<td>bits_per_sample (audiomp3.MP3Decoder attribute)</td>
<td>151</td>
</tr>
<tr>
<td>ble_workflow (supervisor.Runtime attribute)</td>
<td>274</td>
</tr>
<tr>
<td>BlendFunc() (eve._EVE method)</td>
<td>121</td>
</tr>
<tr>
<td>blit() (displayio.Bitmap method)</td>
<td>182</td>
</tr>
<tr>
<td>blit() (framebuf.FrameBuffer method)</td>
<td>385</td>
</tr>
<tr>
<td>BlockInput (in module synthio)</td>
<td>276</td>
</tr>
<tr>
<td>blocks (synthio.Synthesizer attribute)</td>
<td>283</td>
</tr>
<tr>
<td>BluetoothError</td>
<td>106</td>
</tr>
<tr>
<td>board</td>
<td>388</td>
</tr>
<tr>
<td>board_id (in module board)</td>
<td>163</td>
</tr>
<tr>
<td>bool (class in builtins)</td>
<td>359</td>
</tr>
<tr>
<td>bool (in module numpy)</td>
<td>293</td>
</tr>
<tr>
<td>BOOTLOADER (microcontroller.RunMode attribute)</td>
<td>235</td>
</tr>
<tr>
<td>boundary_fill()</td>
<td>159</td>
</tr>
</tbody>
</table>
Index
Code (msgpack.ExtType attribute), 236
collect() (in module gc), 366
collections
   module, 364
color_index (vectorio.Circle attribute), 307
color_index (vectorio.Polygon attribute), 308
color_index (vectorio.Rectangle attribute), 308
ColorA() (_eve._EVE method), 122
colorbar (espncamera.Camera attribute), 199
ColorConverter (class in displayio), 182
ColorMask() (_eve._EVE method), 122
ColorRGB() (_eve._EVE method), 123
colorspace (class in displayio), 180
colorwheel() (in module rainbowio), 251
compile() (in module builtins), 359
concatenate() (in module re), 371
color_index (in class builtins), 359
connect() (in module ulab.numpy), 292
circular() (bitbangio.SPI method), 361
circular() (busio.SPI method), 367
circular() (bsdio.SPI method), 263
conjugate() (in module ulab.numpy, curray), 290
connect() (_bleio.Adapter method), 109
socketpool.Socket method), 265
circular() (ssl.SSLSocket method), 265
connect() (wifi.Radio method), 314
connectable (_bleio.ScanEntry attribute), 116
connected (_bleio.Adapter attribute), 107
connected (_bleio.Connection attribute), 113
connected (usb_cdc.Serial attribute), 300
connectable (wifi.Radio attribute), 312
Connection (class in _bleio), 113
connection_interval (_bleio.Connection attribute), 113
ConnectionError, 361
connections (_bleio.Adapter attribute), 107
console in module usb_cdc, 300
console (supervisor.StatusBar attribute), 275
const() (in module micropython), 386
CONstrained_LERP (synthio.MathOperation attribute), 279
CONSUMER_CONTROL (usb_hid.Device attribute), 304
contains() (displayio.TileGrid method), 194
continuous_capture_get_frame() (imagecapture.ParallelImageCapture method), 219
continuous_capture_start() (imagecapture.ParallelImageCapture method), 219
continuous_capture_stop() (imagecapture.ParallelImageCapture method), 219
contrast (espncamera.Camera attribute), 199
convert() (displayio.ColorConverter method), 183
copysign() (in module math), 226
cos() (in module math), 226
cos() (in module ulab.numpy), 295
cosh() (in module math), 227
cosh() (in module ulab.numpy), 295
count (countio.Counter attribute), 176
count() (sdcardio.SDCard method), 261
count() (sdioio.SDCard method), 263
Counter (class in countio), 176
countio
   module, 176
country (wifi.Network attribute), 312
cpu (in module microcontroller), 232
cpus (in module microcontroller), 232
CPython, 388
crc32() (in module binascii), 364
create_default_context() (in module ssl), 267
cross() (in module ulab.numpy), 293
cross-compiler, 388
csr_transfer() (usb.core.Device method), 299
cyW43
   module, 177
cyWPin (class in cyW43), 177
D
data (canio.Message attribute), 175
data (in module usb_cdc), 300
data (msgpack.ExtType attribute), 236
data_type (grio.QRInfo attribute), 250
datetime (rtc.RTC attribute), 260
dcw (espncamera.Camera attribute), 200
DEBUG (in module re), 371
decay_time (synthio.Envelope attribute), 276
decode() (grio.QRDecoder method), 250
decompress() (in module zlib), 315
decrypt_into() (aesio.AES method), 134
DEEP_SLEEP_ALARM (microcontroller.ResetReason attribute), 234
degrees() (in module math), 226
degrees() (in module ulab.numpy), 295
deinit() (_bleio.CharacteristicBuffer method), 113
deinit() (_bleio.PacketBuffer method), 116
deinit() (analogbufio.Bufferedln method), 138
deinit() (analogio.AnalogIn method), 139
deinit() (analogio.AnalogOut method), 140
deinit() (audiobusio.I2SOut method), 142
deinit() (audiobusio.PDMIn method), 143
deinit() (audiocore.RawSample method), 144
deinit() (audiocore.WaveFile method), 145
deinit() (audiomixer.Mixer method), 148
deinit() (audiomisc.MP3Decoder method), 151
deinit() (audioio.AudioOut method), 147
deinit() (audioio.AudioOut method), 147
deinit() (audioio.Mixer method), 149
deinit() (audioio.MP3Decoder method), 151
deinit() (audioio.PWMAudioOut method), 152
deinit() (bitbangio.I2C method), 154
deinit() (bitbangio.SPI method), 156
deinit() (busio.I2C method), 164
deinit() (busio.SPI method), 166
deinit() (busio_UART method), 169
deinit() (camera.Camera method), 171
deinit() (canio_CAN method), 174
deinit() (canio_Listen method), 174
deinit() (countio.Counter method), 176
deinit() (digitalio.DigitalInOut method), 179
deinit() (displayio.Bitmap method), 182
deinit() (espncamera.Camera method), 201
deinit() (espnow.ESPNow method), 203
deinit() (espulp.ULP method), 205
deinit() (frequencyio.FrequencyIn method), 210
deinit() (gifio.GifWriter method), 211
deinit() (gifio.OnDiskGif method), 213
deinit() (gnss.GNSS method), 214
deinit() (i2cdevice.I2CDevice method), 217
deinit() (imagecapture.ParallelImageCapture method), 219
deinit() (is31fl3741.IS31FL3741 method), 221
deinit() (is31fl3741.IS31FL3741_FrameBuffer method), 220
deinit() (keypad.KeyMatrix method), 223
deinit() (keypad.Keys method), 224
deinit() (keypad.ShiftRegisterKeys method), 225
deinit() (mdns.Server method), 229
deinit() (onednewire.OneWire method), 238
deinit() (picodvi.Framebuffer method), 242
deinit() (ps2io.Ps2 method), 243
deinit() (pulseio.PulseIn method), 245
deinit() (pulseio.PulseOut method), 246
deinit() (pspio.PWMOut method), 249
deinit() (rgbmatrix.RGBMatrix method), 253
deinit() (rotaryio.IncrementalEncoder method), 254
deinit() (rp2pio.StateMachine method), 257
deinit() (sdcardio.SDCard method), 262
deinit() (sdioio.SDCard method), 264
deinit() (sharpdisplay.SharpMemoryFramebuffer method), 264
deinit() (synthio.MidiTrack method), 281
deinit() (synthio.Synthesizer method), 284
deinit() (touchio.TouchIn method), 288
deinit() (usb_host.Port method), 305
deinit() (watchdog.WatchDogTimer method), 310
deinit() (wifi.Monitor method), 311
delattr() (in module builtins), 359
delay_us() (in module microcontroller), 359
denoise (espncamera.Camera attribute), 199
deg() (in module collections), 364
desc (in module btree), 383
descendants (in module btree), 383
descerator (in module _bleio), 114
descriptors (_bleio.Characteristic attribute), 114
det() (in module ubinum.linalg), 291
detach_kernel_driver() (usb.core.Device method), 299
Device (in module usb.core), 298

Device (class in usb_hid), 303
devices (in module usb_hid), 302
diag() (in module ulab.numpy), 292
dict (in builtins), 359
diff() (in module ulab.numpy), 293
digest() (hashlib.Hash method), 215
digest_size (hashlib.Hash attribute), 215
digitalInOut (class in digitalio), 178
digitalio
   module, 178
dir() (in module builtins), 359
Direction (class in digitalio), 180
direction (digitalio.DigitalInOut attribute), 179
dirty() (displayio.Bitmap method), 182
disable() (in module gc), 366
disable() (in module usb_cdc), 300
disable() (in module usb_hid), 302
disable() (in moduleusb_digital), 305
disable_interrupts() (in module microcontroller), 232
disable_usb_drive() (in module storage), 270
disconnect() (_bleio.Connection method), 114
discover_remote_services() (_bleio.Connection method), 114
Display (in module displayio), 183
display (supervisor.StatusBar attribute), 276
Display() (_eve._EVE method), 123
displayio
   module, 180
dither (displayio.ColorConverter attribute), 183
dither (displayio.Palette attribute), 192
dither() (in module bitmaptools), 161
DitherAlgorithm (class in bitmaptools), 161
DIV_ADD (synthio.MathOperation attribute), 279
divisor (rotaryio.IncrementalEncoder attribute), 254
divmod() (in module builtins), 359
dot() (in module ubinum.numpy), 294
DOWN (digitalio.Pull attribute), 180
draw_circle() (in module bitmaptools), 162
draw_line() (in module bitmaptools), 159
draw_polygon() (in module bitmaptools), 159
drive_mode (digitalio.DigitalInOut attribute), 179
DriveMode (class in digitalio), 178
driver, 388
dualbank
   module, 195
dump() (in module json), 369
dumps() (in module json), 369
duration (gifio.OnDiskGif attribute), 213
duty_cycle (pspio.PWMOut attribute), 249

E
e (in module math), 226
EAT_NONAME (socketpool.SocketPool attribute), 267
Edge (class in countio), 176

eig() (in module ulab.numpy.linalg), 291

Ellipsis (in module builtins), 362

empty() (in module ulab.numpy), 292

enable() (in module gc), 366

enable() (in module is31fl3741), 221

enable() (in module usb_cdc), 300

enable() (in module usb_hid), 302

enable() (in module usb_mid), 305

enable_interrupts() (in module microcontroller), 322

enable_usb_drive() (in module storage), 270

enabled (_bleio.Adapter attribute), 107

enabled (samd.Clock attribute), 260

enabled (wifi.Radio attribute), 312

encrypt_into() (aesio.AES method), 133

ENCRYPT_NO_MITM (_bleio.Attribute attribute), 110

ENCRYPT_WITH_MITM (_bleio.Attribute attribute), 110

crypt_type (espnow.Peer attribute), 204
End() (_eve._EVE method), 123

end() (re.Match method), 371

ENTERPRISE (wifi.AuthMode attribute), 311

enumerate() (in module builtins), 359

Envelope (class in synthio), 276

envelope (synthio.Note attribute), 282

envelope (synthio.Synthesizer attribute), 283

EOFError, 361

EPaperDisplay (class in displayio), 186

erase_bonding() (_bleio.Adapter method), 109

erase_filesystem() (in module storage), 270

erase_nvss() (in module espidf), 201

erf() (in module math), 228

erf() (in module ulab.numpy), 295

erfc() (in module math), 228

erfc() (in module ulab.numpy), 295

errno

module, 365

ERROR_ACTIVE (canio.BusState attribute), 172

error_location (synthio.MidiTrack attribute), 281

ERROR_PASSIVE (canio.BusState attribute), 172

ERROR_WARNING (canio.BusState attribute), 172

errorcode (in module errno), 365

espcamera

module, 196

espidf

module, 201

espnow

module, 202

ESPNow (class in espnow), 202

ESPNowPacket (class in espnow), 204

espulp

module, 205

eval() (in module builtins), 359

EVEN (busio.Parity attribute), 170

EVENgetBytes (qrio.PixelPolicy attribute), 250

Event (class in keypad), 221

EventQueue (class in keypad), 222

events (keypad.KeyMatrix attribute), 223

events (keypad.Keys attribute), 224

events (keypad.ShiftRegisterKeys attribute), 225

EVERY_BYTE (qrio.PixelPolicy attribute), 249

Exception, 361

eexec() (in module builtins), 359

exit() (in module sys), 372

exit_and_deep_sleep_until_alarms() (in module alarm), 135

exp() (in module math), 226

exp() (in module ulab.numpy), 295

expml() (in module math), 227

expml() (in module ulab.numpy), 295

exposure_ctrl (espcamera.Camera attribute), 199

extend() (array.array method), 363

textended (canio.Match attribute), 175

extended (canio.Message attribute), 175

extended (canio.RemoteTransmissionRequest attribute), 175

ExtType (class in msgpack), 236

eye() (in module ulab.numpy), 292

F

fabs() (in module math), 226

FALL (countio.Edge attribute), 176

feed() (watchdog.WatchDogTimer method), 310

FFI, 388

fft() (in module ulab.numpy.fft), 290

FHD (espcamera.FrameSize attribute), 197

file (audiomp3.MP3Decoder attribute), 150

FileIO (class in io), 368

filesystem, 388

fill1() (_pixelmap.PixelMap method), 127

fill1() (adafruit_pixelbuf.PixelBuf method), 132

canio.Message attribute), 175

fill() (displayio.Bitmap method), 182

fill() (framebuf.FrameBuffer method), 384

canio.Message attribute), 175

fill_rect() (framebuf.FrameBuffer method), 384

fill_region() (in module bitmaptools), 158

fill_row() (displayio.Display method), 186

fill_row() (framebufferio.FramebufferDisplay method), 209

filter (synthio.Note attribute), 281

filter() (in module builtins), 359

find() (in module usb.core), 298

find() (mdns.Server method), 229

fix (gnss.GNSSattribute), 214

FIX_2D (gnss.PositionFix attribute), 214

FIX_3D (gnss.PositionFix attribute), 215

flash() (in module dualbank), 195

FLASH_WRITE_FAIL (supervisor.SafeModeReason attribute), 274
Index
module, 211
getpass() (in module getpass), 211
getrandbits() (in module random), 251
getvalue() (io.BytesIO method), 368
gifio
module, 211
GifWriter (class in gifio), 211
globals() (in module builtins), 359
GLONASS (gnss.SatelliteSystem attribute), 215
Glyph (class in fontio), 207
gnss
module, 214
GNSS (class in gnss), 214
GPIO, 389
GPIO port, 389
GPS (gnss.SatelliteSystem attribute), 215
grab_mode (espcamera.Camera attribute), 201
GrabMode (class in espcamera), 196
GRAYSCALE (espcameraPixelFormat attribute), 196
Group (class in displayio), 189
group() (re.match method), 371
groups() (re.match method), 371

Halt() (espulp.ULP method), 206
HARD_FAULT (supervisor.SafeModeReason attribute), 275
hasattr() (in module builtins), 359
Hash (class in hashlib), 215
hash() (in module builtins), 359
hashlib
module, 215
HD (espcamera_FRAME_SIZE attribute), 197
heap, 389
heap_caps_get_free_size() (in module espidf), 201
heap_caps_get_largest_free_block() (in module espidf), 201
heap_caps_get_total_size() (in module espidf), 201
heap_lock() (in module micropython), 387
heap_locked() (in module micropython), 387
heap_unlock() (in module micropython), 387
heapify() (in module heapq), 363
heappop() (in module heapq), 363
heappush() (in module heapq), 363
heapq
module, 362
height (displayio.Bitmap attribute), 181
height (displayio.Display attribute), 185
height (displayio.EPaperDisplay attribute), 188
height (displayio.OnDiskDisplay attribute), 192
height (displayio.TileGrid attribute), 194
height (espcamera.Camera attribute), 200
height (framebufferio.FramebufferDisplay attribute), 208

height (gifio.OnDiskGif attribute), 213
height (is31fl3741.IS31FL3741_FrameBuffer attribute), 220
height (picodvi.Framebuffer attribute), 242
height (qrio.QRDecoder attribute), 250
height (rgbmatrix.RGBMatrix attribute), 253
height (vectorio.Rectangle attribute), 308
help()
built-in function, 315
hex() (in module builtins), 359
hexlify() (in module binascii), 363
hidden (displayio.Group attribute), 189
hidden (displayio.TileGrid attribute), 193
hidden (vectorio.Circle attribute), 307
hidden (vectorio.Polygon attribute), 308
hidden (vectorio.Rectangle attribute), 308
high_pass_filter() (synthio.Synthesizer method), 284
hline() (framebuf.FrameBuffer method), 384
hmirror (espcamera.Camera attribute), 199
hostname (mdns.RemoteService attribute), 228
hostname (mdns.Server attribute), 229
hostname (wifi.Radio attribute), 312
HqVGA (espcamera.FRAMESIZE attribute), 196
HVGA (espcamera.FRAMESIZE attribute), 196

I
I2C (class in bitbangio), 153
I2C (class in busio), 164
I2C() (in module busio), 163
I2DDevice (class in adafruit_bus_device.i2c_device), 129
I2DDevice (class in displayio), 190
i2ctarget
module, 216
I2Cetarget (class in i2ctarget), 217
I2CetargetRequest (class in i2ctarget), 217
I2SOut (class in audiobusio), 140
id (canio.Match attribute), 175
id (canio.Message attribute), 175
id (canio.RemoteTransmissionRequest attribute), 175
id() (in module builtins), 359
IDError, 201
idProduct (usb.core.Device attribute), 298
idVendor (usb.core.Device attribute), 298
ifft() (in module ulab.numpy.fft), 290
ignore() (memorymonitor.AllocationAlarm method), 231
ilistdir() (storage.VfsFat method), 271
imag() (in module ulab.numpy, carray), 290
imagecapture
module, 218
ImageFormat (class in camera), 171
implementation (in module sys), 372
LESC_ENCRYPT_WITH_MITH (_bleio.Attribute attribute), 110
level (audiomixer.MixerVoice attribute), 149
LFO (class in synthio), 278
LFO0rLFOSequence (in module synthio), 282
lgamma() (in module math), 228
lgamma() (in module ulab.numpy), 295
light_sleep_until_alarms() (in module alarm), 135
line() (framebuf.FrameBuffer method), 384
LineWidth() (_eve._EVE method), 125
linspace() (in module ulab.numpy), 292
list (class in builtins), 360
listdir() (in module os), 239
listen() (canio.CAN method), 173
listen() (socketpool.Socket method), 265
listen() (ssl.SSLSocket method), 268
Listener (class in canio), 174
LITTLE_ENDIAN (in module ucrypts), 377
lmk (esphome.Peer attribute), 204
load() (in module json), 369
load_cert_chain() (ssl.SSLContext method), 267
load_verify_locations() (ssl.SSLContext method), 267
loads() (in module json), 369
locals() (in module builtins), 360
localtime() (in module time), 286
location (vectorio.Circle attribute), 307
location (vectorio.Polygon attribute), 308
location (vectorio.Rectangle attribute), 308
log() (in module math), 227
log() (in module ulab.numpy), 295
log10() (in module math), 227
log10() (in module ulab.numpy), 295
log2() (in module math), 227
log2() (in module ulab.numpy), 295
logspace() (in module ulab.numpy), 292
longitude (gnss.GNSS attribute), 214
LookupError, 361
loopback (canio.CAN attribute), 173
lost() (wifi.Monitor method), 311
low_pass_filter() (synthio.Synthesizer method), 284
make_transparent() (displayio.Palette method), 192
manufacturer (usb.core.Device attribute), 299
map() (in module builtins), 360
mask (canio.Match attribute), 175
Match (class in canio), 174
match() (in module re), 371
match() (re.regex method), 371
matches() (_bleio.ScanEntry method), 116
math
    module, 226
Math (class in synthio), 279
MathOperation (class in synthio), 279
MAX (synthio.MathOperation attribute), 279
max() (in module builtins), 360
max() (in module ulab.numpy), 293
max_delay (gifio.OnDiskGif attribute), 213
max_frame_size (espcamera.Camera attribute), 200
max_length (_bleio.Characteristic attribute), 111
max_packet_length (_bleio.Connection attribute), 113
max_polyphony (synthio.Synthesizer attribute), 283
max_stack_usage() (in module ustack), 306
maxlen (pulsio.PulseIn attribute), 245
maxsize (in module sys), 372
MCU, 389
mdns
    module, 228
mean() (in module ulab.numpy), 293
median() (in module ulab.numpy), 293
mem_alloc() (in module gc), 366
mem_free() (in module gc), 366
mem_info() (in module micropython), 386
MemoryError, 201, 361
memorymap
    module, 230
memorymonitor
    module, 230
memoryview (class in builtins), 360
Message (class in canio), 175
mfm_readinto() (in module floppyio), 206
microcontroller
    module, 232
micropython
    module, 386
MicroPython port, 389
MicroPython Unix port, 389
MID (synthio.MathOperation attribute), 279
midi_to_hz() (in module synthio), 277
MidiTrack (class in synthio), 280
MIN (synthio.MathOperation attribute), 279
min() (in module builtins), 360
min() (in module ulab.numpy), 293
min_delay (gifio.OnDiskGif attribute), 213
Mixer (class in audiomixer), 148
MixerVoice (class in audiomixer), 149
mkdir() (in module os), 239
mkdir() (storage.VfsFat method), 271
mkfs() (storage.VfsFat static method), 270
mktimes() (in module time), 286
mode (watchdog.WatchDogTimer attribute), 310
MODE_CBC (in module aesio), 133
MODE_CTR (in module aesio), 133
MODE_ECB (in module aesio), 133
modf() (in module math), 227
modify() (select.poll method), 379
module
  _bleio, 106
  _eve, 118
  _pew, 126
  _pixelmap, 127
  _stage, 128
  adafruit_bus_device, 129
  adafruit_bus_device.i2c_device, 129
  adafruit_bus_device.spi_device, 131
  adafruit_pixelbuf, 131
  aesio, 133
  alarm, 134
  alarm.pin, 134
  alarm.time, 135
  alarm.touch, 135
  analogbufio, 137
  analogio, 138
  array, 363
  atexit, 140
  audiobusio, 140
  audiocore, 144
  audioio, 146
  audiomixer, 148
  audiomp3, 150
  audiopwmio, 151
  binascii, 363
  bitbangio, 153
  bitmaptools, 157
  bitops, 162
  board, 163
  btree, 380
  builtins, 358
  busio, 163
  camera, 170
  canio, 171
  collections, 364
  countio, 176
cyw43, 177
digitalio, 178
displayio, 180
dualbank, 195
erorno, 365
espcamera, 196
espidf, 201
espnw, 202
espulp, 205
floppyio, 206
fontio, 207
framebuf, 383
framebufferio, 208
frequencyio, 209
gc, 366
getpass, 211
gifio, 211
gnss, 214
hashlib, 215
heapq, 362
i2ctarget, 216
imagecapture, 218
io, 367
ipaddress, 219
is31fl3741, 220
json, 368
keypad, 221
math, 226
mdns, 228
memorymap, 230
memorymonitor, 230
microcontroller, 232
micropython, 386
msgpack, 35
neopixel_write, 236
nvm, 237
onewireio, 238
os, 239
paralleldisplay, 241
picodv, 241
ps2io, 243
pulseio, 244
pwmi, 247
qrio, 249
rainbowio, 251
random, 251
re, 369
rgbmatrix, 252
rotaryio, 253
rp2pio, 254
rtc, 259
samd, 260
sdcardio, 261
sdioio, 262
select, 379
sharpdisplay, 264
socketpool, 265
ssl, 267
storage, 269
struct, 271
supervisor, 272

Index
ord() (in module builtins), 360
OrderedDict() (in module collections), 365
os
module, 239
OSError, 361
out_waiting (usb_cdc.Serial attribute), 301
outgoing_packet_length (_bleio.PacketBuffer attribute), 115
OUTPUT (digitalio.Direction attribute), 180
overflowed (keypad.EventQueue attribute), 222
OverflowError, 361
P
P_3MP (espcamera.FrameSize attribute), 197
P_FHD (espcamera.FrameSize attribute), 197
P_HD (espcamera.FrameSize attribute), 197
pack() (in module msgpack), 236
pack() (in module struct), 271
pack_into() (_bleio.UUID method), 117
pack_into() (in module struct), 271
packed (ipaddress.IPv4Address attribute), 219
Packet (class in wifi), 312
packet() (wifi.Monitor method), 311
PacketBuffer (class in _bleio), 115
pair() (_bleio.Connection method), 114
paired (_bleio.Connection attribute), 113
Palette (class in displayio), 192
palette (gifio.OnDiskGif attribute), 213
PaletteSource() (_eve._EVE method), 123
panning (synthio.Note attribute), 281
ParallelBus (class in paralleldisplay), 241
paralleldisplay
module, 241
ParallelImageCapture (class in imagecapture), 218
parent (samd.Clock attribute), 261
Parity (class in busio), 170
path (in module sys), 373
pause() (audioio.AudioOut method), 142
pause() (audioio.AudioOut method), 147
pause() (audiopwmio.PWMAudioOut method), 153
pause() (frequencyio.FrequencyIn method), 210
pause() (pulseio.PulseIn method), 245
paused (audioio.AudioOut method), 142
audioio.AudioOut attribute), 147
audioio.AudioOut and PWMAudioOut attribute), 152
audioio.AudioOut and PWMAudioOut attribute), 245
payload (qrio.QRInfo attribute), 250
PDMIn (class in audiobusio), 142
Peer (class in espnow), 204
Peers (class in espnow), 204
peers (espnow.ESPNow attribute), 203
pending (rp2pio.ESPNow attribute), 257
PewPew (class in _pew), 126
phase (synthio.LFO attribute), 278
phase_offset (synthio.LFO attribute), 278
phy_rate (espnow.ESPNow attribute), 203
pi (in module math), 226
picodvi
module, 241
pin (alarm.pin.PinAlarm attribute), 134
pin (espnow.TouchAlarm attribute), 135
Pin (class in microcontroller), 233
PinAlarm (class in alarm.pin), 134
ping() (wifi.Radio method), 314
pins_are_sequential() (in module rp2pio), 254
pixel() (framebuf.FrameBuffer method), 384
pixel_format (espcamera.GreyscaleCamera attribute), 198
pixel_shader (displayio.TileGrid attribute), 192
pixel_shader (displayio.TileGrid attribute), 194
pixel_shader (vectorio.Circle attribute), 307
pixel_shader (vectorio.Polygon attribute), 308
pixel_shader (vectorio.Rectangle attribute), 309
PixelBuf (class in adafruit_pixelbuf), 132
PixelFormat (class in espnow), 196
PixelMap (class in pixelmap), 127
PixelPolicy (class in qrio), 249
PixelReturnSequence (in module _pixelmap), 127
PixelReturnSequence (in module adafruit_pixelbuf), 131
PixelReturnType (in module _pixelmap), 127
PixelReturnType (in module adafruit_pixelbuf), 131
PixelSequence (in module _pixelmap), 127
PixelSequence (in module adafruit_pixelbuf), 132
PixelType (in module _pixelmap), 127
PixelType (in module adafruit_pixelbuf), 131
platform (in module sys), 373
play() (audiobusio.I2SOut method), 142
play() (audioio.AudioOut method), 147
play() (audiomixer.Mixer method), 149
play() (audiomixer.MixerVoice method), 149
play() (audiopwmio.PWMAudioOutOut method), 153
playing (audioio.AudioOut method), 141
playing (audioio.AudioOut method), 147
playing (audiomixer.Mixer attribute), 148
playing (audiomixer.MixerVoice attribute), 149
playing (audiopwmio.PWMAudioOutOut method), 152
PM_AGGRESSIVE (in module cyw43), 177
PM_DISABLED (in module cyw43), 177
PM_PERFORMANCE (in module cyw43), 177
PM_STANDARD (in module cyw43), 177
points (vectorio.Polygon attribute), 308
PointSize() (_eve._EVE method), 125
poll() (in module select), 379
poll() (select.poll method), 379
Polygon (class in vectorio), 307
pop() (displayio.Group method), 190
popleft() (collections.deque method), 364
popleft() (ps2io.Ps2 method), 243
popleft() (pulseio.PulseIn method), 246
port, 389
Port (class in usb_host), 305
port (mdns.RemoteService attribute), 229
PortIn (class in usb_mid), 305
PortOut (class in usb_mid), 306
ports (in module usb_mid), 305
position (rotaryio.IncrementalEncoder attribute), 254
PositionFix (class in gns), 214
pow() (in module builtins), 360
pow() (in module math), 227
POWER_ON (microcontroller.ResetReason attribute), 234
press() (synthio.Synthesizer method), 283
pressed (keypad.Event attribute), 222
pressed (synthio.Synthesizer attribute), 283
print() (in module builtins), 360
print_exception() (in module traceback), 289
Processor (class in microcontroller), 233
PRODUCT (synthio.MathOperation attribute), 279
product (usb.core.Device attribute), 299
PROGRAMMATIC (supervisor.SafeModeReason attribute), 275
properties (_bleio.Characteristic attribute), 110
property() (in module builtins), 360
protocol (mdns.RemoteService attribute), 228
Ps2 (class in ps2io), 243
ps2io
    module, 243
PSK (wifi.AuthMode attribute), 311
PTR (in module uctypes), 377
PUBLIC (_bleio.Address attribute), 110
Pull (class in digitalio), 180
pull (digitalio.DigitalInOut attribute), 179
PulseIn (class in pulseio), 244
pulseio
    module, 244
PulseOut (class in pulseio), 246
PUSH_PULL (digitalio.DriveMode attribute), 178
PWMAudioOut (class in audiopwmio), 151
pwmio
    module, 247
PWMOut (class in pwmio), 247
Q
QCIF (espcamera.FrameSize attribute), 196
QHD (espcamera.FrameSize attribute), 197
QQVGA (espcamera.FrameSize attribute), 196
qr() (in module ulab.numpy.linalg), 291
QRDecoder (class in qrio), 250
QRInfo (class in qrio), 250
qrio
    module, 249
qstr_info() (in module micropython), 386
QSXGA (espcamera.FrameSize attribute), 197
quality (espcamera.Camera attribute), 199
queue (wifi.Monitor attribute), 311
queued() (wifi.Monitor method), 311
QVGA (espcamera.FrameSize attribute), 196
QXGA (espcamera.FrameSize attribute), 197
QZSS_L1CA (gnss.SatelliteSystem attribute), 215
QZSS_L1S (gnss.SatelliteSystem attribute), 215
R
R240X240 (espcamera.FrameSize attribute), 196
R96X96 (espcamera.FrameSize attribute), 196
radians() (in module math), 227
radians() (in module ulab.numpy), 295
Radio (class in wifi), 312
radio (in module wifi), 310
radius (vectorio.Circle attribute), 307
rainbowio
    module, 251
RAISE (watchdog.WatchDogMode attribute), 310
randint() (in module random), 251
random
    module, 251
random() (in module random), 251
RANDOM_PRIVATE_NON_RESOLVABLE (_bleio.Address attribute), 110
RANDOM_PRIVATE_RESOLVABLE (_bleio.Address attribute), 110
RANDOM_STATIC (_bleio.Address attribute), 110
randrange() (in module random), 251
range() (in module builtins), 360
rate (synthio.LFO attribute), 278
RAW (wifi.Packet attribute), 312
raw_gma (espcamera.Camera attribute), 200
raw_value (touchio.TouchIn attribute), 287
RawSample (class in audiocore), 144
re
    module, 369
READ (_bleio.Characteristic attribute), 111
read() (_bleioCharacteristicBuffer method), 112
read() (busio.UART method), 169
read() (espnow.ESPNow method), 203
read() (i2c.target.I2CRequestRequest method), 218
read() (usb.core.Device method), 299
read() (usb_cdc.Serial method), 301
read() (usb_midi.PortIn method), 305
read_bit() (onewireio.OneWire method), 238
read_failure (espnow.ESPNow attribute), 203
read_success (espnow.ESPNow attribute), 203
readblocks() (sdcardio.SDCard method), 262
readblocks() (sdio.SDCard method), 263
readfrom_into() (bitbangio.I2C method), 154
readfrom_into() (busio.I2C method), 165
readinto() (_bleio.CharacteristicBuffer method), 112
readinto() (_bleio.PacketBuffer method), 115
ring·bend (synthio.Note attribute), 282
ring·frequency (synthio.Note attribute), 282
ring·waveform (synthio.Note attribute), 282
RISCV (espulp. Architecture attribute), 205
RISE (countio. Edge attribute), 176
RISE_AND_FALL (countio. Edge attribute), 176
rmdir() (in module os), 239
rmdir() (storage.VfsFat method), 271
rms·level (audiomp3.MP3Decoder attribute), 151
RoleError, 107
roll() (in module ulab.numpy), 294
root·group (displayio. Display attribute), 185
root·group (displayio. EPaperDisplay attribute), 188
root·group (framebufferio. FramebufferDisplay attribute), 208

rotaryio
module, 253
rotation (displayio. Display attribute), 185
rotation (displayio. EPaperDisplay attribute), 188
rotation (framebufferio. FramebufferDisplay attribute), 208
rotozoom() (in module bitmaptools), 157
round() (in module builtins), 360
row·column_to_key_number() (keypad. KeyMatrix method), 224

rp2pio
module, 254
rssi (_bleio.ScanEntry attribute), 116
rssi (espnow.ESPNowPacket attribute), 204
rssi (wifi. Network attribute), 311
RSSI (wifi. Packet attribute), 312


rtc
module, 259
RTC (class in rtc), 260
run() (espulp. ULP method), 205
run() (rp2pio.StateMachine method), 257
run_reason (supervisor. Runtime attribute), 274
RunMode (class in microcontroller), 234
RunReason (class in supervisor), 273
Runtime (class in microcontroller), 273
runtime (in module supervisor), 272
RuntimeError, 361
rxstall (rp2pio.StateMachine attribute), 257

S
SAFE_MODE (microcontroller. RunMode attribute), 234
safe_mode_reason (supervisor. Runtime attribute), 274
SAFE_MODE_WATCHDOG (supervisor. SafeModeReason attribute), 275
SafeModeReason (class in supervisor), 274
samd
module, 260
sample_rate (audiobusio.PDMIn attribute), 143
sample_rate (audiocore.RawSample attribute), 144
sample_rate (audiocore.WaveFile attribute), 145
sample_rate (audiomixer.Mixer attribute), 148
sample_rate (audiomp3.MP3Decoder attribute), 151
sample_rate (synthio.MidiTrack attribute), 281
sample_rate (synthio. Synthesizer attribute), 283
sample_rate (in module floppyio), 206
samples_decoded (audiomp3.MP3Decoder attribute), 151
SatelliteSystem (class in gnss), 215
saturation (espccamera.Camera attribute), 199
SaveContext() (_eve.EVE method), 123
SBAS (gnss. SatelliteSystem attribute), 215
scale (displayio. Group attribute), 189
scale (synthio. LFO attribute), 278
SCALE_OFFSET (synthio. MathOperation attribute), 279
scan() (bitbangio.I2C method), 154
scan() (busio.I2C method), 164
scan_response (_bleio.ScanEntry attribute), 116
ScanEntry (class in _bleio), 116
ScannedNetworks (class in wifi), 314
ScanResults (class in _bleio), 116
schedule() (in module micropython), 387
ScissorSize() (_eve.EVE method), 123
ScissorXY() (_eve.EVE method), 124
scroll() (framebuf. FrameBuffer method), 385
SDCard (class in sdcardio), 261
SDCard (class in sidioio), 262
sdcardio
module, 261
sdioio
module, 262
SDK_FATAL_ERROR (supervisor. SafeModeReason attribute), 275
search() (in module re), 371
search() (re. regex method), 371
secondary (_bleio. Service attribute), 117
SecurityError, 107
seed() (in module random), 251
select
module, 379
select() (in module select), 379
send() (canio.CAN method), 173
send() (displayio. FourWire method), 189
send() (displayio.I2CDisplay method), 191
send() (espnow. ESPNow method), 203
send() (paralleldisplay. ParallelBus method), 241
send() (pulseio. PulseOut method), 247
send() (socketpool. Socket method), 266
send() (ssl. SSLSocket method), 269
send_failure (espnow.ESPNow attribute), 203
send_report() (usb_hid. Device method), 304
send_success (espnow.ESPNow attribute), 203
sendall() (socketpool. Socket method), 266
sendcmd() (ps2io. Ps2 method), 243

424

Index
StateMachine (class in rp2pio), 254
statistic() (in module builtins), 361
status_bar (in module supervisor), 272
StatusBar (class in supervisor), 275
statvfs() (in module os), 240
statvfs() (storage.VfsFat method), 271
std() (in module ulab.numpy), 294
stderr (in module sys), 373
stdin (in module sys), 373
stdout (in module sys), 373
QDebug( _eve._EVE method), 124
StencilFunc() (_eve._EVE method), 124
StencilMask() (_eve._EVE method), 124
StencilOp() (_eve._EVE method), 124
stop() (audiobusio.I2SOut method), 142
stop() (audioio.AudioOut method), 147
stop() (audiomixer.MixerVoice method), 149
stop() (audiopwmio.PWMAudioOut method), 153
stop() (rp2pio.StateMachine method), 257
stop() (radio.Radio method), 314
stop_background_write() (rp2pio.StateMachine method), 258
stop_dhcp() (radio.Radio method), 314
stop_dhcp() (radio.Radio method), 314
stop() (_bleio.Adapter method), 109
stop() (wifi.WLAN method), 313
stop_scanning() (wifi.WLAN method), 313
stop_voice() (audiomixer.Mixer method), 149
StopAsyncIteration, 362
StopIteration, 362
storage
    module, 269
str (class in builtins), 361
stream, 390
StringIO (class in io), 368
struct
    module, 271
struct (class in uctypes), 377
struct_time (class in time), 286
sub() (in module re), 371
sub() (re.regex method), 371
SUM (synthio.MathOperation attribute), 279
sum() (in module builtins), 361
sum() (in module ulab.numpy), 294
super() (in module builtins), 361
supervisor
    module, 272
SUPERVISOR_RELOAD (supervisor.RunReason attribute), 273
supports_jpeg (espcamera.Camera attribute), 200
sustain_level (synthio.Envelope attribute), 277
SVG (espcamera.FrameSize attribute), 196
switch() (in module dualbank), 195
switch_to_input() (digitalio.DigitalInOut method), 179
switch_to_output() (digitalio.DigitalInOut method), 179
SXGA (espcamera.FrameSize attribute), 197
sync() (in module os), 240
sync() (sdcard.SDCard method), 262
SyntaxError, 362
Synthesizer (class in synthio), 282
synthio
    module, 276
sys
    module, 372
sysname (os._Uname attribute), 239
SystemExit, 362
T
Tag() (_eve._EVE method), 125
TagMask() (_eve._EVE method), 125
take() (espcamera.Camera method), 201
take_picture() (camera.Camera method), 171
tan() (in module math), 227
tanh() (in module math), 227
tanh() (in module ulab.numpy), 296
TCP_NODELAY (socketpool.SocketPool attribute), 267
temperature (microcontroller.Processor attribute), 234
Terminal (class in terminalio), 285
terminalio
    module, 285
Text (class in _stage), 129
text() (framebuf.FrameBuffer method), 385
TextIOWrapper (class in io), 368
threshold (touchio.TouchIn attribute), 287
threshold() (in module gc), 366
ticks_ms() (in module supervisor), 272
tile_height (displayio.TileGrid attribute), 194
tile_width (displayio.TileGrid attribute), 194
TileGrid (class in displayio), 193
time
    module, 285
time() (in module time), 286
time_to_refresh (displayio.EPaperDisplay attribute), 188
TimeAlarm (class in alarm.time), 135
timeout (busio.UART attribute), 169
timeout (canio.Listener attribute), 174
timeout (usb_cdc.Serial attribute), 301
timeout (watchdog.WatchDogTimer attribute), 310
TimeoutError, 362
timestamp (gnss.GNSS attribute), 214
timestamp (keypad.Event attribute), 222
to_bytes() (builtins.int method), 360
TouchAlarm (class in alarm.touch), 135
TouchIn (class in touchio), 287
touchio module, 287
trace() (in module ulab.numpy), 294
traceback module, 288
transmit_error_count (canio.CAN attribute), 173
transpose_xy (displayio.TileGrid attribute), 194
trapz() (in module ulab.numpy), 292
trunc() (in module math), 227
try_lock() (bitbangio.I2C method), 154
try_lock() (bitbangio.SPI method), 156
try_lock() (busio.I2C method), 164
try_lock() (busio.SPI method), 167
tuple (class in builtins), 361
tx_power (wifi.Radio attribute), 312
txstall (rp2pio.StateMachine attribute), 257
type (_bleio.Address attribute), 109
type() (in module builtins), 361
TypeError, 362
UART, 390
UART (class in busio), 168
UART() (in module board), 163
u8types module, 374
UF2 (microcontroller.RunMode attribute), 235
uheap module, 289
uid (microcontroller.Processor attribute), 234
UINT16 (in module uctypes), 377
uint16 (in module ulab.numpy), 293
UINT32 (in module uctypes), 377
UINT64 (in module uctypes), 377
UINT8 (in module uctypes), 377
uint8 (in module ulab.numpy), 293
ulab module, 289
ulab.numpy module, 290
ulab.numpy.carray module, 290
ulab.numpy.fft module, 290
ulab.numpy.linalg module, 291
ulab.scipy module, 296
ulab.scipy.linalg module, 296
ulab.scipy.optimize module, 297
ulab.user module, 298
ulab.utils module, 298
ULP (class in espulp), 205
ULPAlarm (class in espulp), 206
umount() (in module storage), 269
umount() (storage.VfsFat method), 271
uname() (in module os), 239
unhexlify() (in module binascii), 363
UnicodeError, 362
uniform() (in module random), 251
UNKNOWN (microcontroller.ResetReason attribute), 234
unlock() (bitbangio.I2C method), 154
unlock() (bitbangio.SPI method), 156
unlock() (busio.I2C method), 165
unlock() (busio.SPI method), 167
unpack() (in module msgpack), 236
unpack() (in module struct), 271
unpack_from() (in module struct), 271
unregister() (in module atexit), 140
unregister() (select.poll method), 379
UP (digitalio.Pull attribute), 180
update() (gnss.GNSS method), 214
update() (hashlib.Hash method), 215
update_refresh_mode() (displayio.EPaperDisplay method), 188
urandom() (in module os), 240
usage (usb_hid.Device attribute), 304
usage_page (usb_hid.Device attribute), 304
usb module, 298
usb.core module, 298
USB_BOOT_DEVICE_NOT_INTERFACE_ZERO (supervisor.SafeModeReason attribute), 275
usb_cdc module, 300
usb_control (supervisor.Runtime attribute), 274
usb_hid module, 302
usb_host module, 305
usb_midi module, 305
USB_TOO_MANY_ENDPOINTS (supervisor.SafeModeReason attribute), 275
USB_TOO_MANY_INTERFACE_NAMES (supervisor.SafeModeReason attribute), 275
USBError, 298
USBTimeoutError, 298
USER (supervisor.SafeModeReason attribute), 275
ustack module, 306
utime() (in module os), 240
uuid (_bleio.Characteristic attribute), 111
uuid (_bleio.Descriptor attribute), 114
uuid (_bleio.Service attribute), 117
UUID (class in _bleio), 117
uuid128 (_bleio.UUID attribute), 117
uuid16 (_bleio.UUID attribute), 117
UXGA (espcamera.FrameSize attribute), 197

V
value (_bleio.Characteristic attribute), 111
value (_bleio.Descriptor attribute), 114
value (alarm.pin.PinAlarm attribute), 135
value (analogio.AnalogIn attribute), 139
value (analogio.AnalogOut attribute), 139
value (digitalio.DigitalIn attribute), 179
value (synthio.LFO attribute), 278
value (synthio.Math attribute), 280
value (touchio.TouchIn attribute), 287
ValueError, 362
values() (btree.btree method), 382
vectorio
    module, 306
vectorize() (in module ublab.numpy), 296
version (in module sys), 373
version (ipaddress.IPv4Address attribute), 219
version (os._Uname attribute), 239
version_info (in module sys), 373
Vertex2f() (_eve._EVE method), 125
Vertex2ii() (_eve._EVE method), 125
VertexFormat() (_eve._EVE method), 126
VertexTranslateX() (_eve._EVE method), 125
VertexTranslateY() (_eve._EVE method), 126
vflip (espcamera.Camera attribute), 199
VfsFat (class in storage), 270
VGA (espcamera.FrameSize attribute), 196
vline() (framebuf.FrameBuffer method), 384
voct_to_hz() (in module synthio), 277
voice (audiomixer.Mixer attribute), 148
VOID (in module actypes), 377
voltage (microcontroller.Processor attribute), 234

W
wake_alarm (in module alarm), 135
watchdog
    module, 309
watchdog (in module microcontroller), 233
WATCHDOG (microcontroller.RestReason attribute), 234
WatchDogMode (class in watchdog), 309
WatchDogTimeout, 309
WatchDogTimer (class in watchdog), 310
WaveFile (class in audiocore), 144
waveform (synthio.LFO attribute), 278
waveform (synthio.Note attribute), 282
wb_mode (espcamera.Camera attribute), 200
WEP (wifi.AuthMode attribute), 310
WHEN_EMPTY (espcamera.GrabMode attribute), 196
whitebal (espcamera.Camera attribute), 199
width (displayio.Bitmap attribute), 181
width (displayio.Display attribute), 185
width (displayio.EPaperDisplay attribute), 188
width (displayio.OnDiskBitmap attribute), 192
width (displayio.TileGrid attribute), 194
width (espcamera.Camera attribute), 200
width (framebufferio.FramebufferDisplay attribute), 208
width (gifio.OnDiskGif attribute), 213
width (is31fl3741.IS31FL3741_FrameBuffer attribute), 220
width (picodvi.Framebuffer attribute), 242
width (qrio.QRDecoder attribute), 250
width (rgbmatrix.RGBMatrix attribute), 253
width (sdioio.SDCard attribute), 263
width (vectorio.Rectangle attribute), 308
wifi
    module, 310
WPA (wifi.AuthMode attribute), 311
WPA2 (wifi.AuthMode attribute), 311
WPA3 (wifi.AuthMode attribute), 311
wpc (espcamera.Camera attribute), 200
WQXGA (espcamera.FrameSize attribute), 197
wrap_socket() (ssl.SSLContext method), 268
WRITE (_bleio.Characteristic attribute), 111
write() (_bleio.PacketBuffer method), 116
write() (adafruit_bus_device.i2c_device.I2CDevice method), 130
write() (bitbangio.SPI method), 156
write() (busio.SPI method), 167
write() (busio.UART method), 170
write() (i2ctarget.I2CDeviceRequest method), 218
write() (in module is31fl3741), 221
write() (rp2pio.StateMachine method), 257
write() (terminalio.Terminal method), 285
write() (usb.core.Device method), 299
write() (usb_cdc.Serial method), 302
write() (usb_midi.PortOut method), 306
write_bit() (onewireio.OneWire method), 238
WRITE_NO_RESPONSE (_bleio.Characteristic attribute), 111
write_readinto() (bitbangio.SPI method), 157
write_readinto() (busio.SPI method), 168
write_readinto() (rp2pio.StateMachine method), 259
write_then_readinto()
    (adafruit_bus_device.i2c_device.I2CDevice method), 130
write_timeout (usb_cdc.Serial attribute), 301
writeblocks() (sdcardio.SDCard method), 262
writeblocks() (sdioio.SDCard method), 263
writeln() (bitbangio.I2C method), 154
writeto() (busio.I2C method), 165
writeto_then_readfrom() (bitbangio.I2C method), 155
writeto_then_readfrom() (busio.I2C method), 165
writing (rp2pio.StateMachine attribute), 257

X
x (displayio.Group attribute), 189
x (displayio.TileGrid attribute), 193
x (vectorio.Circle attribute), 307
x (vectorio.Polygon attribute), 308
x (vectorio.Rectangle attribute), 308
XGA (espcamera.FrameSize attribute), 197

Y
y (displayio.Group attribute), 190
y (displayio.TileGrid attribute), 194
y (vectorio.Circle attribute), 307
y (vectorio.Polygon attribute), 308
y (vectorio.Rectangle attribute), 308

Z
ZeroDivisionError, 362
zeros() (in module ulab.numpy), 293
zip() (in module builtins), 361
zlib
   module, 315