MicroPython libraries

Warning

Important summary of this section

- MicroPython implements a subset of Python functionality for each module.
- To ease extensibility, MicroPython versions of standard Python modules usually have `u` (“micro”) prefix.
- Any particular MicroPython variant or port may miss any feature/function described in this general documentation (due to resource constraints or other limitations).

This chapter describes modules (function and class libraries) which are built into MicroPython. There are a few categories of such modules:

- Modules which implement a subset of standard Python functionality and are not intended to be extended by the user.
- Modules which implement a subset of Python functionality, with a provision for extension by the user (via Python code).
- Modules which implement MicroPython extensions to the Python standard libraries.
- Modules specific to a particular `MicroPython port` and thus not portable.

Note about the availability of the modules and their contents: This documentation in general aspires to describe all modules and functions/classes which are implemented in MicroPython project. However, MicroPython is highly configurable, and each port to a particular board/embedded system makes available only a subset of MicroPython libraries. For officially supported ports, there is an effort to either filter out non-applicable items, or mark individual descriptions with “Availability:” clauses describing which ports provide a given feature.

With that in mind, please still be warned that some functions/classes in a module (or even the entire module) described in this documentation **may be unavailable** in a particular build of MicroPython on a particular system. The best place to find general information of the availability/non-availability of a particular feature is the “General Information” section which contains information pertaining to a specific `MicroPython port`.

On some ports you are able to discover the available, built-in libraries that can be imported by entering the following at the REPL:

```
help('modules')
```

Beyond the built-in libraries described in this documentation, many more modules from the Python standard library, as well as further MicroPython extensions to it, can be found in `micropython-lib`.

Python standard libraries and micro-libraries

The following standard Python libraries have been "micro-ified" to fit in with the philosophy of MicroPython. They provide the core functionality of that module and are intended to be a drop-in replacement for the standard Python library. Some modules below use a standard Python name, but prefixed with "u", e.g. `ujson` instead of `json`. This is to signify that such a module is micro-library, i.e. implements only a subset of CPython module functionality. By naming them differently, a user has a choice to write a Python-level module to extend functionality for better compatibility with CPython (indeed, this is what done by the `micropython-lib` project mentioned above).

On some embedded platforms, where it may be cumbersome to add Python-level wrapper modules to achieve naming compatibility with CPython, micro-modules are available both by their u-name, and also by their non-u-name. The non-u-name can be overridden by a file of that name in your library path (`sys.path`). For example, `import json` will first search for a file `json.py` (or package directory `json`) and load that module if it is found. If nothing is found, it will fallback to loading the built-in `ujson` module.

- `uzlib` – zlib decompression
- `_thread` – multithreading support

MicroPython-specific libraries

Functionality specific to the MicroPython implementation is available in the following libraries.

- `btree` – simple BTree database
- `framebuf` — Frame buffer manipulation
- `machine` — functions related to the hardware
- `micropython` – access and control MicroPython internals
- `network` — network configuration
- `ucryptolib` – cryptographic ciphers
- `uctypes` – access binary data in a structured way

Libraries specific to the pyboard

The following libraries are specific to the pyboard.

- `pyb` — functions related to the board
  - Time related functions
  - Reset related functions
  - Interrupt related functions
  - Power related functions
  - Miscellaneous functions
  - Classes
- `lcd160cr` — control of LCD160CR display
  - class LCD160CR
  - Constructors
  - Static methods
  - Instance members
  - Setup commands
  - Pixel access methods
  - Drawing text
  - Drawing primitive shapes
  - Touch screen methods
  - Advanced commands
  - Constants

Libraries specific to the WiPy

The following libraries and classes are specific to the WiPy.

Libraries specific to the ESP8266 and ESP32

The following libraries are specific to the ESP8266 and ESP32.