# AdafruitOneWire Library Documentation

***Release 1.0***

**Carter Nelson**

**Apr 10, 2020**

# Contents

Classes for use in communicating with devices on a 1-Wire bus.

CHAPTER 1

# Dependencies

This driver depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

# Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-onewire
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-onewire
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-onewire
```

# CHAPTER 3

## Usage Example

```
import board
from adafruit_onewire.bus import OneWireBus
ow_bus = OneWireBus(board.D2)
devices = ow_bus.scan()
for d in devices:
    print("ROM={}\tFamily=0x{:02x}".format(d.rom, d.family_code))
```

CHAPTER 4

# Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

# Documentation

For information on building library documentation, please check out this guide.

# CHAPTER 6

# Table of Contents

## 6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/onewire_simpletest.py

```
import board
from adafruit_onewire.bus import OneWireBus

# Create the 1-Wire Bus
# Use whatever pin you've connected to on your board
ow_bus = OneWireBus(board.D2)

# Reset and check for presence pulse.
# This is basically - "is there anything out there?"
print("Resetting bus...", end="")
if ow_bus.reset():
    print("OK.")
else:
    raise RuntimeError("Nothing found on bus.")

# Run a scan to get all of the device ROM values
print("Scanning for devices...", end="")
devices = ow_bus.scan()
print("OK.")
print("Found {} device(s).".format(len(devices)))

# For each device found, print out some info
for i, d in enumerate(devices):
    print("Device {:>3}".format(i))
    print("\tSerial Number = ", end="")
    for byte in d.serial_number:
        print("0x{:02x} ".format(byte), end="")
```

(continues on next page)

(continued from previous page)

```
    print("\n\tFamily = 0x{:02x}".format(d.family_code))

# Usage beyond this is device specific. See a CircuitPython library for a 1-Wire
# device for examples and how OneWireDevice is used.
```

## 6.2 adafruit_onewire.bus

Provide access to a 1-Wire bus.

- Author(s): Carter Nelson

**class** adafruit_onewire.bus.**OneWireAddress**(*rom*)
A class to represent a 1-Wire address.

**crc**
The 8 bit CRC.

**family_code**
The 8 bit family code.

**rom**
The unique 64 bit ROM code.

**serial_number**
The 48 bit serial number.

**class** adafruit_onewire.bus.**OneWireBus**(*pin*)
A class to represent a 1-Wire bus.

**static crc8**(*data*)
Perform the 1-Wire CRC check on the provided data.

**Parameters** **data** (*bytearray*) – 8 byte array representing 64 bit ROM code

**maximum_devices**
The maximum number of devices the bus will scan for. Valid range is 1 to 255. It is an error to have more devices on the bus than this number. Having less is OK.

**readinto**(*buf*, *, *start=0*, *end=None*)
Read into `buf` from the device. The number of bytes read will be the length of `buf`.

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

**Parameters**

- **buf** (*bytearray*) – buffer to write into
- **start** (*int*) – Index to start writing at
- **end** (*int*) – Index to write up to but not include

**reset**(*required=False*)
Perform a reset and check for presence pulse.

**Parameters** **required** (*bool*) – require presence pulse

**scan**()
Scan for devices on the bus and return a list of addresses.

**write**(*buf*, ***, *start=0*, *end=None*)
Write the bytes from `buf` to the device.

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

**Parameters**

- **buf** (*bytearray*) – buffer containing the bytes to write
- **start** (*int*) – Index to start writing from
- **end** (*int*) – Index to read up to but not include

**exception** adafruit_onewire.bus.**OneWireError**
A class to represent a 1-Wire exception.

## 6.3 adafruit_onewire.device

Provides access to a single device on the 1-Wire bus.

- Author(s): Carter Nelson

**class** adafruit_onewire.device.**OneWireDevice**(*bus*, *address*)
A class to represent a single device on the 1-Wire bus.

**readinto**(*buf*, ***, *start=0*, *end=None*)
Read into `buf` from the device. The number of bytes read will be the length of `buf`.

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

**Parameters**

- **buf** (*bytearray*) – buffer to write into
- **start** (*int*) – Index to start writing at
- **end** (*int*) – Index to write up to but not include

**write**(*buf*, ***, *start=0*, *end=None*)
Write the bytes from `buf` to the device.

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

**Parameters**

- **buf** (*bytearray*) – buffer containing the bytes to write
- **start** (*int*) – Index to start writing from
- **end** (*int*) – Index to read up to but not include

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## a

# Index