aardvark_py Documentation

Release 5.30.2

Flying Camp Design

Mar 20, 2018

Contents

1 Total Phase Release Versions **3**

2 OS Support **5**

3 Python 3 Support **7**

4 Installation **9**

5 Usage **11**

6 Development **13**

7 License **15**

8 API Documentation **17**
8.1 aardvark_py package 17

9 Indices and tables **23**

Python Module Index **25**

This package provides the Total Phase Aardvark Python API packaged for easy distribution and installation via PyPI.

CHAPTER 1

Total Phase Release Versions

The "major.minor" component of the PyPI release version (major.minor.micro) matches the v[major.minor] version of the aardvark API release zip file. For example, the aardvark-py 5.15.0 release on PyPI is guaranteed to be API compatible with aardvark-api-[platform]-[arch]-v5.15.zip. The "micro" component is an additional version specifier that is incremented whenever a new release of this package is published on PyPI.

CHAPTER 2

OS Support

The Python API from Total Phase officially supports the following OS versions:

- Windows 7, 8, 8.1, 10
- Mac OS 10.5 - 10.12
- Ubuntu 12.04 LTS, 14.04 LTS, 16.04 LTS

Since this package is derived directly from Total Phase API releases, it is subject to the same compatibility restrictions.

CHAPTER 3

Python 3 Support

API version 5.30 adds support for Python 3 (previous versions only supported Python 2).

CHAPTER 4

Installation

The `aardvark_py` package can be installed from PyPI using `pip`:

```
$ pip install aardvark_py
```

CHAPTER 5

Usage

Once installed, the `aardvark_py` package is a drop-in replacement for the `aardvark_py.py` language module distributed in the Aardvark API release from Total Phase.

```
$ python
Python 2.7.10 (default, Dec  3 2015, 13:28:10)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from aardvark_py import *
>>> aa_find_devices(1)
(1, array('H', [0]))
>>> handle = aa_open(0)
>>> aa_features(handle)
27
>>> aa_close(handle)
1
```

CHAPTER 6

Development

```
$ git clone https://github.com/FlyingCampDesign/aardvark_py.git
$ cd aardvark_py
$ make dev-install
$ make
```

Open `docs/_build/html/index.html` in a browser to view the generated documentation.

CHAPTER 7

License

Permission to modify and redistribute the Python language modules and associated shared object files has been granted explicitly by Total Phase, Inc. for use in this package. Distribution and use of this package is subject to the license agreement provided in the `LICENSE.txt` file distributed with this package.

CHAPTER 8

API Documentation

8.1 aardvark_py package

8.1.1 Subpackages

aardvark_py.aardvark package

Module contents

Under the hood, the 'aardvark' submodule is actually a precompiled python extension module provided by Total Phase in their API release:

`aardvark.so` – Linux/Mac shared object

`aardvark.dll/pyd` – Windows dynamic link library

The official API documentation can be found at https://www.totalphase.com/support/articles/200468316-Aardvark-I2C-SPI-Host-Adapter-User/#s5

8.1.2 Module contents

class aardvark_py.**AardvarkExt**

class aardvark_py.**AardvarkVersion**

aardvark_py.**aa_async_poll**(*aardvark*, *timeout*)
usage: int return = aa_async_poll(Aardvark aardvark, int timeout)

aardvark_py.**aa_close**(*aardvark*)
usage: int return = aa_close(Aardvark aardvark)

aardvark_py.**aa_configure**(*aardvark*, *config*)
usage: int return = aa_configure(Aardvark aardvark, AardvarkConfig config)

aardvark_py.**aa_features**(*aardvark*)
usage: int return = aa_features(Aardvark aardvark)

aardvark_py.**aa_find_devices**(*devices*)
usage: (int return, u16[] devices) = aa_find_devices(u16[] devices)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_find_devices_ext**(*devices*, *unique_ids*)
usage: (int return, u16[] devices, u32[] unique_ids) = aa_find_devices_ext(u16[] devices, u32[] unique_ids)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_gpio_change**(*aardvark*, *timeout*)
usage: int return = aa_gpio_change(Aardvark aardvark, u16 timeout)

aardvark_py.**aa_gpio_direction**(*aardvark*, *direction_mask*)
usage: int return = aa_gpio_direction(Aardvark aardvark, u08 direction_mask)

aardvark_py.**aa_gpio_get**(*aardvark*)
usage: int return = aa_gpio_get(Aardvark aardvark)

aardvark_py.**aa_gpio_pullup**(*aardvark*, *pullup_mask*)
usage: int return = aa_gpio_pullup(Aardvark aardvark, u08 pullup_mask)

aardvark_py.**aa_gpio_set**(*aardvark*, *value*)
usage: int return = aa_gpio_set(Aardvark aardvark, u08 value)

aardvark_py.**aa_i2c_bitrate**(*aardvark*, *bitrate_khz*)
usage: int return = aa_i2c_bitrate(Aardvark aardvark, int bitrate_khz)

aardvark_py.**aa_i2c_bus_timeout**(*aardvark*, *timeout_ms*)
usage: int return = aa_i2c_bus_timeout(Aardvark aardvark, u16 timeout_ms)

aardvark_py.**aa_i2c_free_bus**(*aardvark*)
usage: int return = aa_i2c_free_bus(Aardvark aardvark)

aardvark_py.**aa_i2c_monitor_disable**(*aardvark*)
usage: int return = aa_i2c_monitor_disable(Aardvark aardvark)

aardvark_py.**aa_i2c_monitor_enable**(*aardvark*)
usage: int return = aa_i2c_monitor_enable(Aardvark aardvark)

aardvark_py.**aa_i2c_monitor_read**(*aardvark*, *data*)
usage: (int return, u16[] data) = aa_i2c_monitor_read(Aardvark aardvark, u16[] data)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to

the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_i2c_pullup**(*aardvark*, *pullup_mask*)
usage: int return = aa_i2c_pullup(Aardvark aardvark, u08 pullup_mask)

aardvark_py.**aa_i2c_read**(*aardvark*, *slave_addr*, *flags*, *data_in*)
usage: (int return, u08[] data_in) = aa_i2c_read(Aardvark aardvark, u16 slave_addr, AardvarkI2cFlags flags, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_i2c_read_ext**(*aardvark*, *slave_addr*, *flags*, *data_in*)
usage: (int return, u08[] data_in, u16 num_read) = aa_i2c_read_ext(Aardvark aardvark, u16 slave_addr, AardvarkI2cFlags flags, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_i2c_slave_disable**(*aardvark*)
usage: int return = aa_i2c_slave_disable(Aardvark aardvark)

aardvark_py.**aa_i2c_slave_enable**(*aardvark*, *addr*, *maxTxBytes*, *maxRxBytes*)
usage: int return = aa_i2c_slave_enable(Aardvark aardvark, u08 addr, u16 maxTxBytes, u16 maxRxBytes)

aardvark_py.**aa_i2c_slave_read**(*aardvark*, *data_in*)
usage: (int return, u08 addr, u08[] data_in) = aa_i2c_slave_read(Aardvark aardvark, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_i2c_slave_read_ext**(*aardvark*, *data_in*)
usage: (int return, u08 addr, u08[] data_in, u16 num_read) = aa_i2c_slave_read_ext(Aardvark aardvark, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to

the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_i2c_slave_set_response**(*aardvark*, *data_out*)
usage: int return = aa_i2c_slave_set_response(Aardvark aardvark, u08[] data_out)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

aardvark_py.**aa_i2c_slave_write_stats**(*aardvark*)
usage: int return = aa_i2c_slave_write_stats(Aardvark aardvark)

aardvark_py.**aa_i2c_slave_write_stats_ext**(*aardvark*)
usage: (int return, u16 num_written) = aa_i2c_slave_write_stats_ext(Aardvark aardvark)

aardvark_py.**aa_i2c_write**(*aardvark*, *slave_addr*, *flags*, *data_out*)
usage: int return = aa_i2c_write(Aardvark aardvark, u16 slave_addr, AardvarkI2cFlags flags, u08[] data_out)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

aardvark_py.**aa_i2c_write_ext**(*aardvark*, *slave_addr*, *flags*, *data_out*)
usage: (int return, u16 num_written) = aa_i2c_write_ext(Aardvark aardvark, u16 slave_addr, AardvarkI2cFlags flags, u08[] data_out)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

aardvark_py.**aa_i2c_write_read**(*aardvark*, *slave_addr*, *flags*, *out_data*, *in_data*)
usage: (int return, u16 num_written, u08[] in_data, u16 num_read) = aa_i2c_write_read(Aardvark aardvark, u16 slave_addr, AardvarkI2cFlags flags, u08[] out_data, u08[] in_data)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_log**(*aardvark*, *level*, *handle*)
usage: int return = aa_log(Aardvark aardvark, int level, int handle)

aardvark_py.**aa_open**(*port_number*)
usage: Aardvark return = aa_open(int port_number)

aardvark_py.**aa_open_ext**(*port_number*)
usage: (Aardvark return, AardvarkExt aa_ext) = aa_open_ext(int port_number)

aardvark_py.**aa_port**(*aardvark*)
usage: int return = aa_port(Aardvark aardvark)

aardvark_py.**aa_sleep_ms**(*milliseconds*)
usage: u32 return = aa_sleep_ms(u32 milliseconds)

aardvark_py.**aa_spi_bitrate**(*aardvark*, *bitrate_khz*)
usage: int return = aa_spi_bitrate(Aardvark aardvark, int bitrate_khz)

aardvark_py.**aa_spi_configure**(*aardvark*, *polarity*, *phase*, *bitorder*)
usage: int return = aa_spi_configure(Aardvark aardvark, AardvarkSpiPolarity polarity, AardvarkSpiPhase phase, AardvarkSpiBitorder bitorder)

aardvark_py.**aa_spi_master_ss_polarity**(*aardvark*, *polarity*)
usage: int return = aa_spi_master_ss_polarity(Aardvark aardvark, AardvarkSpiSSPolarity polarity)

aardvark_py.**aa_spi_slave_disable**(*aardvark*)
usage: int return = aa_spi_slave_disable(Aardvark aardvark)

aardvark_py.**aa_spi_slave_enable**(*aardvark*)
usage: int return = aa_spi_slave_enable(Aardvark aardvark)

aardvark_py.**aa_spi_slave_read**(*aardvark*, *data_in*)
usage: (int return, u08[] data_in) = aa_spi_slave_read(Aardvark aardvark, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_spi_slave_set_response**(*aardvark*, *data_out*)
usage: int return = aa_spi_slave_set_response(Aardvark aardvark, u08[] data_out)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

aardvark_py.**aa_spi_write**(*aardvark*, *data_out*, *data_in*)
usage: (int return, u08[] data_in) = aa_spi_write(Aardvark aardvark, u08[] data_out, u08[] data_in)

All arrays can be passed into the API as an ArrayType object or as a tuple (array, length), where array is an ArrayType object and length is an integer. The user-specified length would then serve as the length argument to the API funtion (please refer to the product datasheet). If only the array is provided, the array's intrinsic length is used as the argument to the underlying API function.

Additionally, for arrays that are filled by the API function, an integer can be passed in place of the array argument and the API will automatically create an array of that length. All output arrays, whether passed in or generated, are passed back in the returned tuple.

aardvark_py.**aa_status_string**(*status*)
usage: str return = aa_status_string(int status)

aardvark_py.**aa_target_power**(*aardvark*, *power_mask*)
usage: int return = aa_target_power(Aardvark aardvark, u08 power_mask)

aardvark_py.**aa_unique_id**(*aardvark*)
usage: u32 return = aa_unique_id(Aardvark aardvark)

aardvark_py.**aa_version**(*aardvark*)
usage: (int return, AardvarkVersion version) = aa_version(Aardvark aardvark)

aardvark_py.**array_f32**(*n*)

aardvark_py.**array_f64**(*n*)

aardvark_py.**array_s08**(*n*)

aardvark_py.**array_s16**(*n*)

aardvark_py.**array_s32**(*n*)

aardvark_py.**array_s64**(*n*)

aardvark_py.**array_u08**(*n*)

aardvark_py.**array_u16**(*n*)

aardvark_py.**array_u32**(*n*)

aardvark_py.**array_u64**(*n*)

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

aardvark_py, 17
aardvark_py.aardvark, 17

Index

A

aa_async_poll() (in module aardvark_py), 17
aa_close() (in module aardvark_py), 17
aa_configure() (in module aardvark_py), 17
aa_features() (in module aardvark_py), 17
aa_find_devices() (in module aardvark_py), 18
aa_find_devices_ext() (in module aardvark_py), 18
aa_gpio_change() (in module aardvark_py), 18
aa_gpio_direction() (in module aardvark_py), 18
aa_gpio_get() (in module aardvark_py), 18
aa_gpio_pullup() (in module aardvark_py), 18
aa_gpio_set() (in module aardvark_py), 18
aa_i2c_bitrate() (in module aardvark_py), 18
aa_i2c_bus_timeout() (in module aardvark_py), 18
aa_i2c_free_bus() (in module aardvark_py), 18
aa_i2c_monitor_disable() (in module aardvark_py), 18
aa_i2c_monitor_enable() (in module aardvark_py), 18
aa_i2c_monitor_read() (in module aardvark_py), 18
aa_i2c_pullup() (in module aardvark_py), 19
aa_i2c_read() (in module aardvark_py), 19
aa_i2c_read_ext() (in module aardvark_py), 19
aa_i2c_slave_disable() (in module aardvark_py), 19
aa_i2c_slave_enable() (in module aardvark_py), 19
aa_i2c_slave_read() (in module aardvark_py), 19
aa_i2c_slave_read_ext() (in module aardvark_py), 19
aa_i2c_slave_set_response() (in module aardvark_py), 20
aa_i2c_slave_write_stats() (in module aardvark_py), 20
aa_i2c_slave_write_stats_ext() (in module aardvark_py), 20
aa_i2c_write() (in module aardvark_py), 20
aa_i2c_write_ext() (in module aardvark_py), 20
aa_i2c_write_read() (in module aardvark_py), 20
aa_log() (in module aardvark_py), 20
aa_open() (in module aardvark_py), 20
aa_open_ext() (in module aardvark_py), 20
aa_port() (in module aardvark_py), 20
aa_sleep_ms() (in module aardvark_py), 20
aa_spi_bitrate() (in module aardvark_py), 21
aa_spi_configure() (in module aardvark_py), 21
aa_spi_master_ss_polarity() (in module aardvark_py), 21
aa_spi_slave_disable() (in module aardvark_py), 21
aa_spi_slave_enable() (in module aardvark_py), 21
aa_spi_slave_read() (in module aardvark_py), 21
aa_spi_slave_set_response() (in module aardvark_py), 21
aa_spi_write() (in module aardvark_py), 21
aa_status_string() (in module aardvark_py), 21
aa_target_power() (in module aardvark_py), 21
aa_unique_id() (in module aardvark_py), 21
aa_version() (in module aardvark_py), 21
aardvark_py (module), 17
aardvark_py.aardvark (module), 17
AardvarkExt (class in aardvark_py), 17
AardvarkVersion (class in aardvark_py), 17
array_f32() (in module aardvark_py), 21
array_f64() (in module aardvark_py), 22
array_s08() (in module aardvark_py), 22
array_s16() (in module aardvark_py), 22
array_s32() (in module aardvark_py), 22
array_s64() (in module aardvark_py), 22
array_u08() (in module aardvark_py), 22
array_u16() (in module aardvark_py), 22
array_u32() (in module aardvark_py), 22
array_u64() (in module aardvark_py), 22