aardvark_py Documentation

Release 5.30.2

Flying Camp Design

Nov 25, 2019

Contents

Official packages are available on PyPI.

https://pypi.org/project/aardvark-py/

The packages are created from the Aardvark API release package.

https://www.totalphase.com/products/aardvark-software-api

CHAPTER 1

System Requirements

- Windows 7 or 10
- Ubuntu 16.04 LTS or 18.04 LTS
- Mac OS 10.13+
- Python 2.6+ or 3.5+
- 64-bit operating system

CHAPTER 2

Installation

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

```
$ pip3 install aardvark_py
```

CHAPTER 3

Usage

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

```
$ python3
>>> 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 4

License

Please see the LICENSE.txt file in the package.

CHAPTER 5

API Documentation

5.1 aardvark_py package

5.1.1 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

5.1.2 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 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A