gpib-ctypes Documentation

Release 0.1.0dev

Tomislav Ivek

Dec 12, 2018

Contents

Contents:

CHAPTER 1

gpib-ctypes

Cross-platform Python bindings for the NI GPIB and linux-gpib C interfaces.

- Free software: GNU General Public License v2
- Documentation: https://gpib-ctypes.readthedocs.io.

1.1 Features

- cross-platform: tested on Microsoft Windows and Linux
- API-compatible with the linux-gpib Python bindings
- no Python dependencies except the standard library

1.2 Testing

Currently tested with: * NI GPIB-USB-HI, Microsoft Windows 7 32-bit, NI GPIB driver version 3.1.0.49154, * NI GPIB-USB-HI, Arch Linux 64-bit current, linux-gpib 4.1.0

More testers welcome with different hardware and OS configurations!

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

TO DO: we still do not have a stable release on pypi

To install gpib-ctypes, run this command in your terminal:

```
$ pip install gpib_ctypes
```

This is the preferred method to install gpib-ctypes, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for gpib-ctypes can be downloaded from the Github repo.

You can install directly from the repo using pip:

```
$ pip install git+https://github.com/tivek/gpib_ctypes
```

Alternatively, install from a local copy of the source. You can either clone the public repository:

```
$ git clone git://github.com/tivek/gpib_ctypes
```

Or download the tarball:

```
$ curl  -OL https://github.com/tivek/gpib_ctypes/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use `gpib-ctypes` in a project, import all submodules at once:

```
import gpib_ctypes
```

. . . or import `gpib` and `Gpib` submodules separately as below.

3.1 Handle-level GPIB API

```
# Identify instrument at board 0, primary address 23.

from gpib_ctypes import gpib

try:
    dev_handle = gpib.dev(0, 23)

    gpib.write(dev_handle, b'*IDN?')
    result = gpib.read(dev_handle, 1000)

except gpib.GpibError as err:
    # do something with err.code
    pass
```

3.2 Object-oriented GPIB API

```
# Identify instrument at board 0, primary address 23.

from gpib_ctypes import Gpib

try:
```

(continues on next page)

(continued from previous page)

```
    dev = Gpib.Gpib(0, 23)

    dev.write(b'*IDN?')
    result = dev.read(1000)

except gpib.GpibError as err:
    # do something with err.code
    pass
```

3.3 Example usage with `pyvisa` and the pure Python backend `pyvisa-py`

```
# pyvisa-py will try to load the root-level gpib module, eg. from the linux-gpib␣
↪project.
# To make pyvisa-py use gpib_ctypes.gpib instead, monkey patch it by calling gpib_
↪ctypes.make_default_gpib().

from gpib_ctypes import make_default_gpib
make_default_gpib()

import visa
rm = visa.ResourceManager('@py')

resources = rm.list_resources()
# example result: ('GPIB0::14::INSTR', 'GPIB0::23::INSTR')

dev = rm.open_resource('GPIB0::23::INSTR')
```

CHAPTER 4

gpib_ctypes

4.1 gpib_ctypes package

4.1.1 Subpackages

gpib_ctypes.gpib package

Submodules

gpib_ctypes.gpib.constants module

gpib_ctypes.gpib.gpib module

exception gpib_ctypes.gpib.gpib.**GpibError**(*funcname*)
Bases: Exception

Exception class with helpful GPIB error messages GpibError(gpib_function_name)

gpib_ctypes.gpib.gpib.**ask**(*handle*, *conf*)
Query configuration by calling ibask.

Args: handle (int): board or device handle conf (int): gpib.Iba* constant designating configuration settings

Returns: int: configuration setting value

gpib_ctypes.gpib.gpib.**clear**(*handle*)
Clear device by calling ibclr.

Args: handle (int): device handle

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**close**(*handle*)
Close board or device handle by calling ibonl.

Args: handle (int): board or device handle to close

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**command**(*handle*, *cmd*)
Write command bytes by calling ibcmd.

Args: handle (int): board handle cmd (bytes): sequence of bytes to write

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**config**(*handle*, *conf*, *value*)
Change configuration by calling ibconfig.

Args: handle (int): board or device handle conf (int): gpib.Ibc* constant designating configuration settings value (int): configuration setting value

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**dev**(*board*, *pad*, *sad=0*, *tmo=14*, *sendeoi=1*, *eos=0*)
Get a device handle by calling ibdev.

Args: board (int): board number pad (int): primary address sad (int): secondary address, default gpib.NO_SAD tmo (int): timeout constant, default gpib.T30s sendeoi (int): assert EOI on write, default 1 eos (int): end-of-string termination, default 0

Returns: int: board or device handle

gpib_ctypes.gpib.gpib.**find**(*name*)
Get a device handle based on a name from configuration file by calling ibfind.

Args: name (string)

Returns: int: board or device handle

gpib_ctypes.gpib.gpib.**ibcnt**()
Get number of transferred bytes by calling ThreadIbcntl or reading ibcnt.

Args: none

Returns: int: number of transferred bytes

gpib_ctypes.gpib.gpib.**ibloc**(*handle*)
Push device to local mode by calling ibloc.

Args: handle (int): device handle

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**ibsta**()
Get status value by calling ThreadIbsta or reading ibsta.

Args: none

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**interface_clear**(*handle*)
Clear interface by calling ibsic.

Args: handle (int): board handle

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**lines**(*board*)
Obtain the status of the control and handshaking bus lines of the bus.

Args: board (int): board handle

Returns: int: line capability and status bits

gpib_ctypes.gpib.gpib.**listener**(*board*, *pad*, *sad=0*)
Check if listener is present at address by calling ibln.

Args: board (int): board or device handle, or board number pad (int): primary address sad (int): secondary address, default gpib.NO_SAD

Returns: bool: True if listener is present, False otherwise

gpib_ctypes.gpib.gpib.**read**(*handle*, *length*)
Read a number of data bytes by calling ibread.

Args: handle (int): board or device handle length (int): number of bytes to read

Returns: bytes: sequence of bytes which was read

gpib_ctypes.gpib.gpib.**remote_enable**(*handle*, *enable*)
Set remote enable by calling ibsre.

Args: handle (int): board handle enable (int): if non-zero, set remote enable

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**serial_poll**(*handle*)
Read status byte by calling ibrsp.

Args: handle (int): device handle

Returns: int: serial poll status byte

gpib_ctypes.gpib.gpib.**spoll_bytes**(*handle*)
Get length of status byte queue by calling ibspb.

Args: handle (int): device handle

Returns: int: status byte queue length

gpib_ctypes.gpib.gpib.**timeout**(*handle*, *t*)
Set IO timeout by calling ibtmo.

Args: handle (int): board or device handle t (int): timeout, one of constants from gpib.TNONE to gpib.T100s

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**trigger**(*handle*)
Trigger device by calling ibtrg.

Args: handle (int): device handle

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**version**()
Get the GPIB library version. Not implemented on Windows.

Args: none

Returns: str: GPIB library version

gpib_ctypes.gpib.gpib.**wait**(*handle*, *eventmask*)
Wait for event by calling ibwait.

Args: handle (int): board or device handle eventmask (int): ibsta bits designating events to wait for

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**write**(*handle*, *data*)
Write data bytes by calling ibwrt.

Args: handle (int): board or device handle data (bytes): sequence of bytes to write

Returns: int: ibsta value

gpib_ctypes.gpib.gpib.**write_async**(*handle*, *data*)
Write data bytes asynchronously by calling ibwrta.

Args: handle (int): board or device handle data (bytes): sequence of bytes to write

Returns: int: ibsta value

Module contents

Python interface for the linux-gpib library or the NI GPIB C library on Windows and Linux. Adheres to the linux-gpib Python API.

All functions return the value of ibsta except where otherwise specified.

4.1.2 Submodules

4.1.3 gpib_ctypes.Gpib module

class gpib_ctypes.Gpib.**Gpib**(*name='gpib0'*, *pad=None*, *sad=0*, *timeout=13*, *send_eoi=1*, *eos_mode=0*)
Bases: object

Three ways to create a Gpib object: Gpib("name")

returns a board or device object, from a name in the config file

Gpib(board_index) returns a board object, with the given board number

Gpib(board_index, pad[, sad[, timeout[, send_eoi[, eos_mode]]]]) returns a device object, like ibdev()

ask(*option*)

clear()

close()

command(*str*)

config(*option*, *value*)

ibcnt()

ibloc()

ibsta()

interface_clear()

lines()

listener(*pad*, *sad=0*)

read(*len=512*)

remote_enable(*val*)

serial_poll()

timeout(*value*)

trigger()

wait(*mask*)

write(*str*)

write_async(*str*)

4.1.4 Module contents

Top-level package for gpib-ctypes.

gpib_ctypes.**make_default_gpib**()

Monkeypatches gpib_ctypes.gpib and gpib_ctypes.Gpib modules to be used as the only gpib and Gpib modules by the running process.

Example usage with pyvisa-py:

from gpib_ctypes import make_default_gpib make_default_gpib() # call early in __main__

import visa rm = visa.ResourceManager('@py') # rm now uses gpib_ctypes

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/tivek/gpib_ctypes/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

gpib-ctypes could always use more documentation, whether as part of the official gpib-ctypes docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/tivek/gpib_ctypes/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *gpib_ctypes* for local development.

1. Fork the *gpib_ctypes* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/gpib_ctypes.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv gpib_ctypes
$ cd gpib_ctypes/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 gpib_ctypes tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/tivek/gpib_ctypes/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_gpib_ctypes
```

CHAPTER 6

Credits

6.1 Development Lead

- Tomislav Ivek <tomislav.ivek@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.3.0 (2018-12-13)

- Provide bindings to iblines through gpib.lines and Gpib.lines

7.2 0.2.1 (2018-11-28)

- Fix gpib.ibfind string marshalling

7.3 0.2.0 (2018-11-27)

- Safe cleanup using Gpib.close()

7.4 0.1.1 (2018-11-27)

- Bugfix release

7.5 0.1.0 (2018-01-01)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index