rf-info Documentation

Release 0.8.0

Ian Perry

Jan 27, 2020

Contents:

CHAPTER 1

rf-info

Command line & Python library for obtaining details about a radio frequency

- Free software: MIT license
- Documentation: https://rf-info.readthedocs.io.
- Python 3.5, 3.6, 3.7, 3.8 & pypy3 tested. Not compatible with Python 2.x
- Linux & Windows with color, json output, and interactive terminal support

1.1 Features

Returns information about a radio frequency.

- “Radio Display” format (Dotted notaton)
- hz, khz, Mhz and Ghz representations of the frequency
- Frequency Wavelength
- ITU Band Description
- ITU Band Abbreviation
- ITU Band Number
- ISM Band Type & Description

- IEEE Band Name
- NATO Band Name
- Waveguide Band Name
- Microwave Band Name & Description
- Fixed Station & Mobile Station Designations
- IEEE Primary Band Allocations
- IEEE Secondary Band Allocations
- Detailed IEEE footnotes for each band allocation
- All active & upcomming satellite frequencys & details (406 Satellites as of 1/18/20)
- Amateur Radio Modes, License Class, Max Power (US Only)
- Broadcasting Band Number & Details (US Only)
- WIFI Frequency Details (US Only)
- Other Services CB, GMRS, Aircraft Band, Etc (US Only)

Currently supported band allocations for countries: United States (US), Canada (CA), Brazil (BR), Spain (ES), United Kingdom (GB), Russian Federation (RU), Ukraine (UA), Japan (JP), India (IN), Korea, Republic of (KR), Thailand (TH), Switzerland (CH), Chile (CL), Denmark (DK), Finland (FI), France (FR), Hungary (HU), Indonesia (ID), Iceland (IS), Italy (IT), Mexico (MX), Netherlands (NL), New Zealand (NZ), Norway (NO), Poland (PL), South Africa (ZA), Sweden (SE), Venezuela (VE), Australia (AU), Slovenia (SI), Ireland (IE), Belgium (BE), Austria (AT), Argentina (AR), Israel (IL), Romania (RO), China (CN), Uruguay (UY), Greece (GR), Panama (PA), Peru (PE)

I can easily add support for more countries upon request

Includes man pages and texinfo documentation

1.2 Install

```
$ pip install rf-info
```

1.3 Command Line Usage

```
$ rf-info <frequency> [<units>] [<country>]
```

Frequency format examples:

```
$ rf-info 89910000
$ rf-info 23,450,000
$ rf-info 12,634.534
$ rf-info 12_000_000
$ rf-info 344_500.100
```

Also supports “Radio Display” frequency representation (Dotted notation):

```
$ rf-info 124.125.000
$ rf-info 1.500.125.000
$ rf-info 000.012.500
```

Unit examples: hz, khz, Mhz, Ghz (Case Insensitive):

```
$ rf-info 123.100 mhz
$ rf-info 4.5 ghz
```

Country examples (2 digit abbriviation, 3 digit abbriviation, 3 digit number, or full name) US, USA, 040, JPN, Spain (Case Insensitive):

```
$ rf-info 144.400.000 hz US
$ rf-info 88 mhz JPN
```

1.4 Python Library Usage

```
>>> from rf_info import Frequency
>>> freq = Frequency('144.890.000')
>>> freq.details()
```

Returns a dictionary:

```
>>> {'display': '144.890.000', 'units': {'hz': 144890000, 'khz': 144890.0, 'mhz': 144.
↪89, 'ghz': 0.14489}, 'wavelength': '2m', 'itu': {'number': 8, 'band': 'Very High␣
↪Frequency', 'abbr': 'VHF'}, 'ieee': {'band': 'VHF', 'description': 'Very High␣
↪Frequency'}, 'nato': {'band': 'A'}, 'ism': {'band': None, 'description': None},
↪'waveguide': {'band': None}, 'microwave': {'band': None, 'allocation': None},
↪'country': {'name': 'United States of America', 'abbr': 'US'}, 'broadcasting': {
↪'allocated': False, 'details': ()}, 'wifi': {'allocated': False, 'details': None},
↪'amateur': {'allocated': True, 'modes': 'CW, Phone, Image, RTTY/Data', 'license':
↪'Tech, General, Extra', 'power': 'MAX'}, 'satellite': {'allocated': False, 'name':␣
↪None, 'sat-id': None, 'link': None, 'modes': None, 'callsign': None, 'status': None}
↪, 'services': None, 'station': {'fixed': False, 'mobile': False}, 'ieee_allocation
↪': {'primary': ('Amateur', 'Amateur-Satellite'), 'secondary': (), 'notes': ('[5.
↪216]: Additional allocation: in China, the band 144-146 MHz is also allocated to␣
↪the aeronautical mobile (OR) service on a secondary basis.',)}}
```

Or you can get individual items directly:

```
>>> freq.itu
>>> freq.itu['band']
>>> freq.wavelength
>>> freq.ieee_allocation['primary']
```

Also supports adding and subtracting frequencies. Either a frequency object, int, or string representation of a frequency, returns a new frequency object:

```
>>> new_freq_object = Frequency('001.123.000') + Frequency('7', 'khz')  # Adds 7 khz␣
↪to 1.123 mhz
>>> new_freq_object = Frequency('1', 'mhz') + 15000  # Adds 15 khz to 1 mhz
>>> new_freq_object = Frequency('123,000') - '000.007.000'  # Subtracts 7 khz from␣
↪123 khz
```

1.5 Output Example

```
$ rf-info 435.890.000 hz US

 Display: 435.890.000
 Hz: 435,890,000
 Khz: 435,890
 Mhz: 435.89
 Ghz: 0.43589
 Wavelength: 68cm
 ITU Band: 9 - UHF (Ultra High Frequency)
 IEEE Band: UHF (Ultra High Frequency)
 NATO Band: B
 Waveguide Band: None
 Microwave Band: None
 Country: United States of America (US)
 Broadcasting: False
 Wifi: False
 Amateur: True
   Modes: Satellite only uplink/downlink
   License: Tech, General, Extra
   Power: MAX
 Satellite: True
   Name: JAS-2 (FO-29) [24278]
   Link: Downlink
   Modes: SSB CW (DigiTalker)
   Status: Active
 Fixed Station: False
 Mobile Station: False
 Primary Allocation:
   - Radiolocation
 Secondary Allocation:
   - Amateur
   - Earth Exploration-Satellite (Active) [5.279A]
 Allocation Notes:
   - [5.279A]: The use of the frequency band 432-438 MHz by sensors in the Earth␣
↪exploration-satellite service (active) shall be in accordance with Recommendation␣
↪ITU-R RS.1260-1. Additionally, the Earth exploration-satellite service (active) in␣
↪the frequency band 432-438 MHz shall not cause harmful interference to the␣
↪aeronautical radionavigation service in China. The provisions of this footnote in␣
↪no way diminish the obligation of the Earth exploration-satellite service (active)␣
↪to operate as a secondary service in accordance with Nos. 5.29 and 5.30. (WRC-15)
```

1.6 Todo

- Add interactive terminal mode

1.7 Credits

M. Ian Perry (ianperry99@gmail.com) AD8DL

CHAPTER 2

Installation

2.1 Requirements

rf-info requires the iso3166 library, and the colorama library for the command line. These are automatically installed when rf-info is installed.

2.2 Stable Version

To install rf-info, run this command in your terminal:

```
$ pip install rf-info
```

This is the preferred method to install rf-info, 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.

2.3 Latest Version from sources

The sources for rf-info can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/cosmicc/rf_info
```

Or download the tarball:

```
$ curl -OJL https://github.com/cosmicc/rf_info/tarball/master
```

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

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

CHAPTER 3

Usage

3.1 Python

To use rf-info in a python project:

```
>>> from rf_info import Frequency
>>> freq = Frequency('112.434.000')
```

then:

```
>>> freq.details()
```

Returns a dictionary of all details:

```
>>> {'display': '144.100.000', 'hz': 144100000, 'khz': 144100.0, 'mhz': 144.1, 'ghz':␣
↪0.1441, 'wavelength': '2m', 'itu_band': 'Very High Frequency', 'itu_abbr': 'VHF',
↪'itu_num': 8, 'ieee_band': 'VHF', 'ieee_description': 'Very High Frequency', 'nato_
↪band': 'A', 'waveguide_band': None, 'country_abbr': 'US', 'country_name': 'United␣
↪States of America', 'amateur': True, 'fixed_station': False, 'mobile_station':␣
↪False, 'broadcast': False, 'primary_allocation': ['Amateur', 'Amateur-Satellite'],
↪'secondary_allocation': [], 'allocation_notes': ['[5.216]: Additional allocation:␣
↪in China, the band 144-146 MHz is also allocated to the aeronautical mobile (OR)␣
↪service on a secondary basis.']}
```

You can also get each detail individually:

```
>>> freq.itu_band
>>> freq.wavelength
>>> freq.Primary_Allocation
```

Also supports adding and subtracting frequencies. Start with a frequency object then annd or subtract another frequency object, int, or string representation of a frequency, returns a new frequency object:

```
>>> new_freq_object = Frequency('001.123.000') + Frequency('7', 'khz')  # Adds 7 khz␣
↪to 1.123 mhz
>>> new_freq_object = Frequency('1', 'mhz') + 15000  # Adds 15 khz to 1 mhz
>>> new_freq_object = Frequency('123,000') - '000.007.000'  # Subtracts 7 khz from␣
↪123 khz
```

3.2 Command Line

To use the rf-info command line tool:

```
$ rf-info <frequency> [<units>] [<country>]
```

Frequency format examples:

```
$ rf-info 89910000
$ rf-info 23,450,000
$ rf-info 12,634.534
$ rf-info 12_000_000
$ rf-info 344_500.100
```

Also supports "Radio Display" frequency representation (Dotted notation):

```
$ rf-info 124.125.000
$ rf-info 1.500.125.000
$ rf-info 000.012.500
```

Suffix examples: hz, khz, Mhz, Ghz (Case Insensitive):

```
$ rf-info 123.100 mhz
$ rf-info 4.5 ghz
```

Country examples (2 digit abbriviation, 3 digit abbriviation, 3 digit number, or full name): US, USA, 040, JPN, es, Spain (Case Insensitive):

```
$ rf-info 144.400.000 hz US
$ rf-info 88 mhz JPN
```

Example command line output:

```
$ rf-info 144.100.000 hz US
Display: 144.100.000
Hz: 144100000
Khz: 144100.0
Mhz: 144.1
Ghz: 0.1441
Wavelength: 2m
Itu_Band: Very High Frequency
Itu_Abbr: VHF
Itu_Num: 8
Ieee_Band: VHF
Ieee_Description: Very High Frequency
Nato_Band: A
Country_Abbr: US
Country_Name: United States of America
```

(continues on next page)

(continued from previous page)

```
Fixed_Station: False
Mobile_Station: False
Broadcasting: False
Amateur: True
Amateur_Details:
    General CW and weak signals
    License Class
Max Power
Primary_Allocation:
    Amateur
    Amateur-Satellite
Allocation_Notes:
    [5.216]: Additional allocation: in China, the band 144-146 MHz is also allocated␣
↪to the aeronautical mobile (OR) service on a
```

You also can print the info in raw or json formatted output:

```
$ rf-info 144.000 hz --raw
$ rf-info 144.000 hz --json
```

CHAPTER 4

rf_info

4.1 rf_info package

4.1.1 Subpackages

rf_info.data package

Submodules

rf_info.data.a_allocations module

rf_info.data.b_allocations module

rf_info.data.c_allocations module

rf_info.data.international module

rf_info.data.rangekeydict module

class rf_info.data.rangekeydict.**RangeKeyDict**(*my_dict*)
Bases: object

rf_info.data.satellites module

rf_info.data.us module

Module contents

4.1.2 Submodules

4.1.3 rf_info.cli module

rf_info.cli.**country_list**()

rf_info.cli.**country_shortlist**()

rf_info.cli.**display_json**(*frequency_obj*)

rf_info.cli.**display_raw**(*frequency_obj*)

rf_info.cli.**display_results**(*frequency*, *unit*, *country*)

rf_info.cli.**get_frequency_obj**(*frequency*, *unit*, *country*)

rf_info.cli.**main**(*argv=None*)

rf_info.cli.**verify_country**(*country*)

4.1.4 rf_info.countrymap module

4.1.5 rf_info.rf_info module

class rf_info.rf_info.**Frequency**(*freq*, *unit='hz'*, *country='us'*)

Bases: object

details()

info()

rf_info.rf_info.**parse_freq**(*freq*, *unit*)

4.1.6 Module contents

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/cosmicc/rf_info/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

rf-info could always use more documentation, whether as part of the official rf-info 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/cosmicc/rf_info/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 *rf_info* for local development.

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

```
$ git clone git@github.com:your_name_here/rf_info.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 rf_info
$ cd rf_info/
$ 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 rf_info tests
$ python setup.py test or pytest
$ 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.org/cosmicc/rf_info/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_rf_info
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Ian Perry <ianperry99@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.7.1b (2020-01-23)

- First Stable Pre-release

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index