python-xbee Documentation
Release 2.0.0

Paul Malmsten

December 29, 2010

CONTENTS

ONE

INTRODUCTION

The purpose of XBee is to allow one easy access to the advanced features of an XBee device from a Python application. It provides a semi-complete implementation of the XBee binary API protocol and allows a developer to send and receive the information they desire without dealing with the raw communication details.

Note: This library is compatible with both XBee 802.15.4 (Series 1) and XBee ZigBee (Series 2) modules, normal and PRO. The following examples are applicable to XBee 802.15.4 modules; to follow these examples with a XBee ZigBee device, change the line:

```
xbee = XBee(serial_port)
```

to:

```
xbee = ZigBee(serial_port)
```

USAGE

Note: In order to use an XBee device with this library, API mode must be enabled. For instructions about how to do this, see the documentation for your XBee device.

2.1 Synchonous Mode

The following code demonstrates a minimal use-case for the xbee package:

```python
import serial
from xbee import XBee

serial_port = serial.Serial('/dev/ttyUSB0', 9600)
xbee = XBee(serial_port)

while True:
    try:
        print xbee.wait_read_frame()
    except KeyboardInterrupt:
        break

serial_port.close()
```

This example will perpetually read from the serial port and print out any data frames which arrive from a connected XBee device. Be aware that wait_read_frame() will block until a valid frame is received from the associated XBee device.

2.2 Asynchronous Mode

The xbee package is used only slightly differently when asynchonous notification of received data is needed:

```python
import serial
import time
from xbee import XBee

serial_port = serial.Serial('/dev/ttyUSB0', 9600)

def print_data(data):
    """
    This method is called whenever data is received
    from the associated XBee device. Its first and
```

```
        only argument is the data contained within the
        frame.
        """
        print data

xbee = XBee(serial_port, callback=print_data)

while True:
    try:
        time.sleep(0.001)
    except KeyboardInterrupt:
        break

xbee.halt()
serial_port.close()
```

> **Warning:** When asychonous mode is enabled, the provided callback method is called by a background thread managed by the xbee package. Any/all thread safety considerations may apply when employing this functionality to modify external state.

Note that a background thread is automatically started to handle receiving and processing incoming data from an XBee device. This example is functionally equivalent to the non-asyncronous example above.

2.3 Additional Examples

For additional examples, look in the examples/ directory contained within the xbee package source code archive or source control.

API

3.1 Frame Data Format

Information returned from this library is a dictionary in the following format:

```
{'id':str,
 'param':binary data,
 ...}
```

The id field is always a human-readable name of the packet type received. All following fields, shown above with the key 'param', map binary data to each of the possible fields contained within the received data frame.

Note: A listing of all supported data frames and their respective fields may be found in xbee.ieee.XBee (or xbee.zigbee.ZigBee for XBee ZB devices) defined as api_responses.

3.2 Sample Data Format

Sample data is returned in the following format:

```
[ {"dio-0":True,
   "dio-1":False,
   "adc-0":100"}, ...]
```

This format is a list of dictionaries. Each dictionary represents one sample, listed in chronological order. Each sample dictionary can contain any number of digital (dio) and analog (adc) pin samples. The keys of a sample dictionary will always follow this pattern:

```
(dio|adc)-[0-9]+
```

The number of dio and adc values returned depends upon the type and configuration of the XBee device used with this library.

3.3 Sending Data to an XBee Device

In order to send data to an XBee device, use the send() method:

```
xbee.send("at", frame='A', command='MY', parameter=None)
```

This example will request the 16-bit address of the connected XBee device with an API frame marker of 'A'.

For your convenience, some optional data fields are not required. For example, the 'parameter' field may be ommitted if empty:

```
xbee.send("at", frame='A', command='MY')
```

Additionally, an alternate syntax is provided for all supported commands::

```
xbee.at(frame='A', command='MY')
```

This example is functionally equivalent to the line(s) above.

Lastly, if an API frame identifier is not needed, the command may be reduced to:

```
xbee.at(command='MY')
```

Note: A listing of all supported commands and their associated data fields may be found in xbee.ieee.XBee (xbee.zigbee.ZigBee for XBee ZB devices) defined as api_commands.

3.4 API Reference

class `xbee.base.`**`XBeeBase`**(*ser*, *shorthand=True*, *callback=None*, *escaped=False*)
    Abstract base class providing command generation and response parsing methods for XBee modules.

Constructor arguments: ser: The file-like serial port to use.

> **shorthand: boolean flag which determines whether shorthand command** calls (i.e. xbee.at(...) instead of xbee.send("at",...) are allowed.
>
> **callback: function which should be called with frame data** whenever a frame arrives from the serial port. When this is not None, a background thread to monitor the port and call the given function is automatically started.
>
> **escaped: boolean flag which determines whether the library should** operate in escaped mode. In this mode, certain data bytes in the output and input streams will be escaped and unescaped in accordance with the XBee API. This setting must match the appropriate api_mode setting of an XBee device; see your XBee device's documentation for more information.

`halt`()
    halt: None -> None

    If this instance has a separate thread running, it will be halted. This method will wait until the thread has cleaned up before returning.

`run`()
    run: None -> None

    This method overrides threading.Thread.run() and is automatically called when an instance is created with threading enabled.

`send`(*cmd*, ***kwargs*)
    send: string param=binary data ... -> None

    When send is called with the proper arguments, an API command will be written to the serial port for this XBee device containing the proper instructions and data.

    This method must be called with named arguments in accordance with the api_command specification. Arguments matching all field names other than those in reserved_names (like 'id' and 'order') should be given, unless they are of variable length (of 'None' in the specification. Those are optional).

`wait_read_frame``()`

wait_read_frame: None -> frame info dictionary

wait_read_frame calls XBee._wait_for_frame() and waits until a valid frame appears on the serial port. Once it receives a frame, wait_read_frame attempts to parse the data contained within it and returns the resulting dictionary

class `xbee.`**`XBee`**`(*args, **kwargs)`

Provides an implementation of the XBee API for IEEE 802.15.4 modules with recent firmware.

Commands may be sent to a device by instansiating this class with a serial port object (see PySerial) and then calling the send method with the proper information specified by the API. Data may be read from a device syncronously by calling wait_read_frame. For asynchronous reads, see the definition of XBeeBase.

class `xbee.zigbee.`**`ZigBee`**`(*args, **kwargs)`

Provides an implementation of the XBee API for XBee ZB (ZigBee) modules with recent firmware.

Commands may be sent to a device by instantiating this class with a serial port object (see PySerial) and then calling the send method with the proper information specified by the API. Data may be read from a device synchronously by calling wait_read_frame. For asynchronous reads, see the defintion of XBeeBase.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

X

xbee, 1

INDEX

H

halt() (xbee.base.XBeeBase method), 6

R

run() (xbee.base.XBeeBase method), 6

S

send() (xbee.base.XBeeBase method), 6

W

wait_read_frame() (xbee.base.XBeeBase method), 6

X

XBee (class in xbee), 7
xbee (module), 1
XBeeBase (class in xbee.base), 6

Z

ZigBee (class in xbee.zigbee), 7