
AlarmDecoder Documentation

Release

Nu Tech Software Solutions, Inc.

September 22, 2015

1	alarmdecoder Package	3
1.1	decoder Module	3
1.2	devices Module	6
1.3	messages Module	12
1.4	zonetracking Module	14
1.5	util Module	16
1.6	panels Module	16
2	Indices and tables	17
	Python Module Index	19

This is the API documentation for the [Alarm Decoder](#) Python library. It provides support for interacting with the [Alarm Decoder](#) (AD2) family of security alarm devices, including the AD2USB, AD2SERIAL and AD2PI.

The source code, requirements and examples for this project may be found [here](#).

Please see the [examples](#) directory for more samples, but a basic one is included below:

```
import time
from alarmdecoder import AlarmDecoder
from alarmdecoder.devices import USBDevice

def main():
    """
    Example application that prints messages from the panel to the terminal.
    """
    try:
        # Retrieve the first USB device
        device = AlarmDecoder(USBDevice.find())

        # Set up an event handler and open the device
        device.on_message += handle_message
        with device.open():
            while True:
                time.sleep(1)

    except Exception, ex:
        print 'Exception:', ex

def handle_message(sender, message):
    """
    Handles message events from the AlarmDecoder.
    """
    print sender, message.raw

if __name__ == '__main__':
    main()
```

Table of Contents:

alarmdecoder Package

1.1 decoder Module

Provides the main AlarmDecoder class.

class `alarmdecoder.decoder.AlarmDecoder` (*device*)
Bases: `object`

High-level wrapper around [AlarmDecoder](#) (AD2) devices.

on_arm

This event is called when the panel is armed.

Callback definition: *def callback(device)*

on_disarm

This event is called when the panel is disarmed.

Callback definition: *def callback(device)*

on_power_changed

This event is called when panel power switches between AC and DC.

Callback definition: *def callback(device, status)*

on_alarm

This event is called when the alarm is triggered.

Callback definition: *def callback(device, zone)*

on_alarm_restored

This event is called when the alarm stops sounding.

Callback definition: *def callback(device, zone)*

on_fire

This event is called when a fire is detected.

Callback definition: *def callback(device, status)*

on_bypass

This event is called when a zone is bypassed.

Callback definition: *def callback(device, status)*

on_boot

This event is called when the device finishes booting.

Callback definition: *def callback(device)*

on_config_received

This event is called when the device receives its configuration.

Callback definition: *def callback(device)*

on_zone_fault

This event is called when *Zonetracker* detects a zone fault.

Callback definition: *def callback(device, zone)*

on_zone_restore

This event is called when *Zonetracker* detects that a fault is restored.

Callback definition: *def callback(device, zone)*

on_low_battery

This event is called when the device detects a low battery.

Callback definition: *def callback(device, status)*

on_panic

This event is called when the device detects a panic.

Callback definition: *def callback(device, status)*

on_relay_changed

This event is called when a relay is opened or closed on an expander board.

Callback definition: *def callback(device, message)*

on_message

This event is called when standard panel *Message* is received.

Callback definition: *def callback(device, message)*

on_expander_message

This event is called when an *ExpanderMessage* is received.

Callback definition: *def callback(device, message)*

on_lrr_message

This event is called when an *LRRMessage* is received.

Callback definition: *def callback(device, message)*

on_rfx_message

This event is called when an *RFMessage* is received.

Callback definition: *def callback(device, message)*

on_sending_received

This event is called when a *!Sending.done* message is received from the AlarmDecoder.

Callback definition: *def callback(device, status, message)*

on_open

This event is called when the device has been opened.

Callback definition: *def callback(device)*

on_close

This event is called when the device has been closed.

Callback definition: *def callback(device)*

on_read

This event is called when a line has been read from the device.

Callback definition: *def callback(device, data)*

on_write

This event is called when data has been written to the device.

Callback definition: *def callback(device, data)*

KEY_F1 = u'\x01\x01\x01'

Represents panel function key #1

KEY_F2 = u'\x02\x02\x02'

Represents panel function key #2

KEY_F3 = u'\x03\x03\x03'

Represents panel function key #3

KEY_F4 = u'\x04\x04\x04'

Represents panel function key #4

KEY_PANIC = u'\x05\x05\x05'

Represents a panic keypress

BATTERY_TIMEOUT = 30

Default timeout (in seconds) before the battery status reverts.

FIRE_TIMEOUT = 30

Default tTimeout (in seconds) before the fire status reverts.

address = 18

The keypad address in use by the device.

configbits = 65280

The configuration bits set on the device.

address_mask = 4294967295

The address mask configured on the device.

emulate_zone = [False, False, False, False, False]

List containing the devices zone emulation status.

emulate_relay = [False, False, False, False]

List containing the devices relay emulation status.

emulate_lrr = False

The status of the devices LRR emulation.

deduplicate = False

The status of message deduplication as configured on the device.

mode = 0

The panel mode that the AlarmDecoder is in. Currently supports ADEMCO and DSC.

id

The ID of the [AlarmDecoder](#) device.

Returns identification string for the device

battery_timeout

Retrieves the timeout for restoring the battery status, in seconds.

Returns battery status timeout

`fire_timeout`

Retrieves the timeout for restoring the fire status, in seconds.

Returns fire status timeout

`internal_address_mask`

Retrieves the address mask used for updating internal status.

Returns address mask

`open (baudrate=None, no_reader_thread=False)`

Opens the device.

Parameters

- **baudrate** (*int*) – baudrate used for the device. Defaults to the lower-level device default.
- **no_reader_thread** (*bool*) – Specifies whether or not the automatic reader thread should be started.

`close ()`

Closes the device.

`send (data)`

Sends data to the [AlarmDecoder](#) device.

Parameters **data** (*string*) – data to send

`get_config ()`

Retrieves the configuration from the device. Called automatically by `_on_open ()`.

`save_config ()`

Sets configuration entries on the device.

`reboot ()`

Reboots the device.

`fault_zone (zone, simulate_wire_problem=False)`

Faults a zone if we are emulating a zone expander.

Parameters

- **zone** (*int*) – zone to fault
- **simulate_wire_problem** (*bool*) – Whether or not to simulate a wire fault

`clear_zone (zone)`

Clears a zone if we are emulating a zone expander.

Parameters **zone** (*int*) – zone to clear

1.2 devices Module

This module contains different types of devices belonging to the [AlarmDecoder](#) (AD2) family.

- [USBDevice](#): Interfaces with the [AD2USB](#) device.
- [SerialDevice](#): Interfaces with the [AD2USB](#), [AD2SERIAL](#) or [AD2PI](#).
- [SocketDevice](#): Interfaces with devices exposed through [ser2sock](#) or another IP to Serial solution. Also supports SSL if using [ser2sock](#).

class `alarmdecoder.devices.SSL` (*Error, WantReadError, SysCallError*)

Bases: tuple

Error

Alias for field number 0

SysCallError

Alias for field number 2

WantReadError

Alias for field number 1

class alarmdecoder.devices.**Device**

Bases: object

Base class for all [AlarmDecoder](#) (AD2) device types.

on_open

This event is called when the device has been opened.

Callback definition: *def callback(device)*

on_close

This event is called when the device has been closed.

Callback definition: *def callback(device)**

on_read

This event is called when a line has been read from the device.

Callback definition: *def callback(device, data)**

on_write

This event is called when data has been written to the device.

Callback definition: *def callback(device, data)**

id

Retrieve the device ID.

Returns identification string for the device

is_reader_alive ()

Indicates whether or not the reader thread is alive.

Returns whether or not the reader thread is alive

stop_reader ()

Stops the reader thread.

close ()

Closes the device.

class ReadThread (*device*)

Bases: `threading.Thread`

Reader thread which processes messages from the device.

READ_TIMEOUT = 10

Timeout for the reader thread.

stop ()

Stops the running thread.

run ()

The actual read process.

class `alarmdecoder.devices.USBDevice` (*interface=0, vid=None, pid=None*)
Bases: `alarmdecoder.devices.Device`

AD2USB device utilizing PyFTDI's interface.

PRODUCT_IDS = ((1027, 24577), (1027, 24597))
List of Vendor and Product IDs used to recognize **AD2USB** devices.

DEFAULT_VENDOR_ID = 1027
Default Vendor ID used to recognize **AD2USB** devices.

DEFAULT_PRODUCT_ID = 24577
Default Product ID used to recognize **AD2USB** devices.

FTDI_VENDOR_ID = 1027
DEPRECATED: Vendor ID used to recognize **AD2USB** devices.

FTDI_PRODUCT_ID = 24577
DEPRECATED: Product ID used to recognize **AD2USB** devices.

BAUDRATE = 115200
Default baudrate for **AD2USB** devices.

classmethod `find_all` (*vid=None, pid=None*)
Returns all FTDI devices matching our vendor and product IDs.

Returns list of devices

Raises `CommError`

classmethod `devices` ()
Returns a cached list of **AD2USB** devices located on the system.

Returns cached list of devices found

classmethod `find` (*device=None*)
Factory method that returns the requested `USBDevice` device, or the first device.

Parameters `device` (*tuple*) – Tuple describing the USB device to open, as returned by `find_all()`.

Returns `USBDevice` object utilizing the specified device

Raises `NoDeviceError`

classmethod `start_detection` (*on_attached=None, on_detached=None*)
Starts the device detection thread.

Parameters

- **on_attached** (*function*) – function to be called when a device is attached **Callback definition:** `def callback(thread, device)`
- **on_detached** (*function*) – function to be called when a device is detached **Callback definition:** `def callback(thread, device)`

classmethod `stop_detection` ()
Stops the device detection thread.

serial_number
Retrieves the serial number of the device.

Returns serial number of the device

description
Retrieves the description of the device.

Returns description of the device

interface

Retrieves the interface used to connect to the device.

Returns the interface used to connect to the device

open (*baudrate=115200, no_reader_thread=False*)

Opens the device.

Parameters

- **baudrate** (*int*) – baudrate to use
- **no_reader_thread** (*bool*) – whether or not to automatically start the reader thread.

Raises *NoDeviceError*

close ()

Closes the device.

fileno ()

write (*data*)

Writes data to the device.

Parameters **data** (*string*) – data to write

Raises *CommError*

read ()

Reads a single character from the device.

Returns character read from the device

Raises *CommError*

read_line (*timeout=0.0, purge_buffer=False*)

Reads a line from the device.

Parameters

- **timeout** (*float*) – read timeout
- **purge_buffer** (*bool*) – Indicates whether to purge the buffer prior to reading.

Returns line that was read

Raises *CommError, TimeoutError*

class DetectThread (*on_attached=None, on_detached=None*)

Bases: *threading.Thread*

Thread that handles detection of added/removed devices.

on_attached

This event is called when an [AD2USB](#) device has been detected.

Callback definition: def callback(thread, device*)

on_detached

This event is called when an [AD2USB](#) device has been removed.

Callback definition: def callback(thread, device*)

stop ()

Stops the thread.

run()

The actual detection process.

class `alarmdecoder.devices.SerialDevice` (*interface=None*)

Bases: `alarmdecoder.devices.Device`

AD2USB, AD2SERIAL or AD2PI device utilizing the PySerial interface.

BAUDRATE = 19200

Default baudrate for Serial devices.

static find_all (*pattern=None*)

Returns all serial ports present.

Parameters `pattern` (*string*) – pattern to search for when retrieving serial ports

Returns list of devices

Raises `CommError`

interface

Retrieves the interface used to connect to the device.

Returns interface used to connect to the device

open (*baudrate=19200, no_reader_thread=False*)

Opens the device.

Parameters

- **baudrate** (*int*) – baudrate to use with the device
- **no_reader_thread** (*bool*) – whether or not to automatically start the reader thread.

Raises `NoDeviceError`

close()

Closes the device.

fileno()

write (*data*)

Writes data to the device.

Parameters `data` (*string*) – data to write

Raises `py:class:~alarmdecoder.util.CommError`

read()

Reads a single character from the device.

Returns character read from the device

Raises `CommError`

read_line (*timeout=0.0, purge_buffer=False*)

Reads a line from the device.

Parameters

- **timeout** (*float*) – read timeout
- **purge_buffer** (*bool*) – Indicates whether to purge the buffer prior to reading.

Returns line that was read

Raises `CommError, TimeoutError`

class `alarmdecoder.devices.SocketDevice` (*interface=('localhost', 10000)*)

Bases: `alarmdecoder.devices.Device`

Device that supports communication with an `AlarmDecoder` (AD2) that is exposed via `ser2sock` or another Serial to IP interface.

interface

Retrieves the interface used to connect to the device.

Returns interface used to connect to the device

ssl

Retrieves whether or not the device is using SSL.

Returns whether or not the device is using SSL

ssl_certificate

Retrieves the SSL client certificate path used for authentication.

Returns path to the certificate path or `OpenSSL.crypto.X509`

ssl_key

Retrieves the SSL client certificate key used for authentication.

Returns jpath to the SSL key or `OpenSSL.crypto.PKey`

ssl_ca

Retrieves the SSL Certificate Authority certificate used for authentication.

Returns path to the CA certificate or `OpenSSL.crypto.X509`

open (*baudrate=None, no_reader_thread=False*)

Opens the device.

Parameters

- **baudrate** (*int*) – baudrate to use
- **no_reader_thread** (*bool*) – whether or not to automatically open the reader thread.

Raises `NoDeviceError`, `CommError`

close ()

Closes the device.

fileno ()

write (*data*)

Writes data to the device.

Parameters **data** (*string*) – data to write

Returns number of bytes sent

Raises `CommError`

read ()

Reads a single character from the device.

Returns character read from the device

Raises `CommError`

read_line (*timeout=0.0, purge_buffer=False*)

Reads a line from the device.

Parameters

- **timeout** (*float*) – read timeout
- **purge_buffer** (*bool*) – Indicates whether to purge the buffer prior to reading.

Returns line that was read

Raises *CommError, TimeoutError*

1.3 messages Module

Message representations received from the panel through the `AlarmDecoder` (AD2) devices.

- *Message*: The standard and most common message received from a panel.
- *ExpanderMessage*: Messages received from Relay or Zone expander modules.
- *RFMessage*: Message received from an RF receiver module.
- *LRRMessage*: Message received from a long-range radio module.

class `alarmdecoder.messages.BaseMessage`

Bases: `object`

Base class for messages.

raw = None

The raw message text

timestamp = None

The timestamp of the message

dict (***kwargs*)

Dictionary representation.

class `alarmdecoder.messages.Message` (*data=None*)

Bases: `alarmdecoder.messages.BaseMessage`

Represents a message from the alarm panel.

ready = False

Indicates whether or not the panel is in a ready state.

armed_away = False

Indicates whether or not the panel is armed away.

armed_home = False

Indicates whether or not the panel is armed home.

backlight_on = False

Indicates whether or not the keypad backlight is on.

programming_mode = False

Indicates whether or not we're in programming mode.

beeps = -1

Number of beeps associated with a message.

zone_bypassed = False

Indicates whether or not a zone is bypassed.

ac_power = False

Indicates whether or not the panel is on AC power.

chime_on = False

Indicates whether or not the chime is enabled.

alarm_event_occurred = False

Indicates whether or not an alarm event has occurred.

alarm_sounding = False

Indicates whether or not an alarm is sounding.

battery_low = False

Indicates whether or not there is a low battery.

entry_delay_off = False

Indicates whether or not the entry delay is enabled.

fire_alarm = False

Indicates whether or not a fire alarm is sounding.

check_zone = False

Indicates whether or not there are zones that require attention.

perimeter_only = False

Indicates whether or not the perimeter is armed.

system_fault = False

Indicates whether a system fault has occurred.

panel_type = 0

Indicates which panel type was the source of this message.

numeric_code = None

The numeric code associated with the message.

text = None

The human-readable text to be displayed on the panel LCD.

cursor_location = -1

Current cursor location on the keypad.

mask = 4294967295

Address mask this message is intended for.

bitfield = None

The bitfield associated with this message.

panel_data = None

The panel data field associated with this message.

dict (***kwargs*)

Dictionary representation.

class `alarmdecoder.messages.ExpanderMessage` (*data=None*)

Bases: `alarmdecoder.messages.BaseMessage`

Represents a message from a zone or relay expansion module.

ZONE = 0

Flag indicating that the expander message relates to a Zone Expander.

RELAY = 1

Flag indicating that the expander message relates to a Relay Expander.

type = None

Expander message type: `ExpanderMessage.ZONE` or `ExpanderMessage.RELAY`

address = -1

Address of expander

channel = -1

Channel on the expander

value = -1

Value associated with the message

dict (***kwargs*)

Dictionary representation.

class `alarmdecoder.messages.RFMessage` (*data=None*)

Bases: `alarmdecoder.messages.BaseMessage`

Represents a message from an RF receiver.

serial_number = None

Serial number of the RF device.

value = -1

Value associated with this message.

battery = False

Low battery indication

supervision = False

Supervision required indication

loop = [False, False, False, False]

Loop indicators

dict (***kwargs*)

Dictionary representation.

class `alarmdecoder.messages.LRRMessage` (*data=None*)

Bases: `alarmdecoder.messages.BaseMessage`

Represent a message from a Long Range Radio.

event_data = None

Data associated with the LRR message. Usually user ID or zone.

partition = -1

The partition that this message applies to.

event_type = None

The type of the event that occurred.

dict (***kwargs*)

Dictionary representation.

1.4 zonetracking Module

Provides zone tracking functionality for the `AlarmDecoder` (AD2) device family.

class `alarmdecoder.zonetracking.Zone` (*zone=0, name='', status=0, expander=False*)

Bases: `object`

Representation of a panel zone.

CLEAR = 0
Status indicating that the zone is cleared.

FAULT = 1
Status indicating that the zone is faulted.

CHECK = 2
Status indicating that there is a wiring issue with the zone.

STATUS = {0: 'CLEAR', 1: 'FAULT', 2: 'CHECK'}

zone = 0
Zone ID

name = ''
Zone name

status = 0
Zone status

timestamp = None
Timestamp of last update

expander = False
Does this zone exist on an expander?

class alarmdecoder.zonetracking.**Zonetracker** (*alarmdecoder_object*)

Bases: object

Handles tracking of zones and their statuses.

on_fault
This event is called when the device detects a zone fault.

Callback definition: *def callback(device, zone)*

on_restore
This event is called when the device detects that a fault is restored.

Callback definition: *def callback(device, zone)*

EXPIRE = 30
Zone expiration timeout.

zones
Returns the current list of zones being tracked.
Returns dictionary of *Zone* being tracked

faulted
Retrieves the current list of faulted zones.
Returns list of faulted zones

update (*message*)
Update zone statuses based on the current message.
Parameters *message* (*Message* or *ExpanderMessage*) – message to use to update the zone tracking

expander_to_zone (*address, channel, panel_type=0*)
Convert an address and channel into a zone number.

Parameters

- **address** (*int*) – expander address

- **channel** (*int*) – channel

Returns zone number associated with an address and channel

1.5 util Module

Provides utility classes for the [AlarmDecoder](#) (AD2) devices.

exception `alarmdecoder.util.NoDeviceError`

Bases: `exceptions.Exception`

No devices found.

exception `alarmdecoder.util.CommError`

Bases: `exceptions.Exception`

There was an error communicating with the device.

exception `alarmdecoder.util.TimeoutError`

Bases: `exceptions.Exception`

There was a timeout while trying to communicate with the device.

exception `alarmdecoder.util.InvalidMessageError`

Bases: `exceptions.Exception`

The format of the panel message was invalid.

class `alarmdecoder.util.Firmware`

Bases: `object`

Represents firmware for the [AlarmDecoder](#) devices.

STAGE_START = 0

STAGE_WAITING = 1

STAGE_BOOT = 2

STAGE_LOAD = 3

STAGE_UPLOADING = 4

STAGE_DONE = 5

static `upload` (*dev, filename, progress_callback=None*)

Uploads firmware to an [AlarmDecoder](#) device.

Parameters

- **filename** (*string*) – firmware filename
- **progress_callback** (*function*) – callback function used to report progress

Raises `NoDeviceError`, `TimeoutError`

1.6 panels Module

Representations of Panels and their templates.

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`alarmdecoder.decoder`, 3
`alarmdecoder.devices`, 6
`alarmdecoder.messages`, 12
`alarmdecoder.panels`, 16
`alarmdecoder.util`, 16
`alarmdecoder.zonetracking`, 14

A

ac_power (alarmdecoder.messages.Message attribute), 12
 address (alarmdecoder.decoder.AlarmDecoder attribute), 5
 address (alarmdecoder.messages.ExpanderMessage attribute), 13
 address_mask (alarmdecoder.decoder.AlarmDecoder attribute), 5
 alarm_event_occurred (alarmdecoder.messages.Message attribute), 13
 alarm_sounding (alarmdecoder.messages.Message attribute), 13
 AlarmDecoder (class in alarmdecoder.decoder), 3
 alarmdecoder.decoder (module), 3
 alarmdecoder.devices (module), 6
 alarmdecoder.messages (module), 12
 alarmdecoder.panels (module), 16
 alarmdecoder.util (module), 16
 alarmdecoder.zonetracking (module), 14
 armed_away (alarmdecoder.messages.Message attribute), 12
 armed_home (alarmdecoder.messages.Message attribute), 12

B

backlight_on (alarmdecoder.messages.Message attribute), 12
 BaseMessage (class in alarmdecoder.messages), 12
 battery (alarmdecoder.messages.RFMessage attribute), 14
 battery_low (alarmdecoder.messages.Message attribute), 13
 BATTERY_TIMEOUT (alarmdecoder.decoder.AlarmDecoder attribute), 5
 battery_timeout (alarmdecoder.decoder.AlarmDecoder attribute), 5
 BAUDRATE (alarmdecoder.devices.SerialDevice attribute), 10
 BAUDRATE (alarmdecoder.devices.USBDevice attribute), 8

beeps (alarmdecoder.messages.Message attribute), 12
 bitfield (alarmdecoder.messages.Message attribute), 13

C

channel (alarmdecoder.messages.ExpanderMessage attribute), 14
 CHECK (alarmdecoder.zonetracking.Zone attribute), 15
 check_zone (alarmdecoder.messages.Message attribute), 13
 chime_on (alarmdecoder.messages.Message attribute), 12
 CLEAR (alarmdecoder.zonetracking.Zone attribute), 14
 clear_zone() (alarmdecoder.decoder.AlarmDecoder method), 6
 close() (alarmdecoder.decoder.AlarmDecoder method), 6
 close() (alarmdecoder.devices.Device method), 7
 close() (alarmdecoder.devices.SerialDevice method), 10
 close() (alarmdecoder.devices.SocketDevice method), 11
 close() (alarmdecoder.devices.USBDevice method), 9
 CommError, 16
 configbits (alarmdecoder.decoder.AlarmDecoder attribute), 5
 cursor_location (alarmdecoder.messages.Message attribute), 13

D

deduplicate (alarmdecoder.decoder.AlarmDecoder attribute), 5
 DEFAULT_PRODUCT_ID (alarmdecoder.devices.USBDevice attribute), 8
 DEFAULT_VENDOR_ID (alarmdecoder.devices.USBDevice attribute), 8
 description (alarmdecoder.devices.USBDevice attribute), 8
 Device (class in alarmdecoder.devices), 7
 Device.ReadThread (class in alarmdecoder.devices), 7
 devices() (alarmdecoder.devices.USBDevice class method), 8
 dict() (alarmdecoder.messages.BaseMessage method), 12
 dict() (alarmdecoder.messages.ExpanderMessage method), 14

dict() (alarmdecoder.messages.LRRMessage method), 14
dict() (alarmdecoder.messages.Message method), 13
dict() (alarmdecoder.messages.RFMessage method), 14

E

emulate_lrr (alarmdecoder.decoder.AlarmDecoder attribute), 5
emulate_relay (alarmdecoder.decoder.AlarmDecoder attribute), 5
emulate_zone (alarmdecoder.decoder.AlarmDecoder attribute), 5
entry_delay_off (alarmdecoder.messages.Message attribute), 13
Error (alarmdecoder.devices.SSL attribute), 6
event_data (alarmdecoder.messages.LRRMessage attribute), 14
event_type (alarmdecoder.messages.LRRMessage attribute), 14
expander (alarmdecoder.zonetracking.Zone attribute), 15
expander_to_zone() (alarmdecoder.zonetracking.Zonetracker method), 15
ExpanderMessage (class in alarmdecoder.messages), 13
EXPIRE (alarmdecoder.zonetracking.Zonetracker attribute), 15

F

FAULT (alarmdecoder.zonetracking.Zone attribute), 15
fault_zone() (alarmdecoder.decoder.AlarmDecoder method), 6
faulted (alarmdecoder.zonetracking.Zonetracker attribute), 15
fileno() (alarmdecoder.devices.SerialDevice method), 10
fileno() (alarmdecoder.devices.SocketDevice method), 11
fileno() (alarmdecoder.devices.USBDevice method), 9
find() (alarmdecoder.devices.USBDevice class method), 8
find_all() (alarmdecoder.devices.SerialDevice static method), 10
find_all() (alarmdecoder.devices.USBDevice class method), 8
fire_alarm (alarmdecoder.messages.Message attribute), 13
FIRE_TIMEOUT (alarmdecoder.decoder.AlarmDecoder attribute), 5
fire_timeout (alarmdecoder.decoder.AlarmDecoder attribute), 5
Firmware (class in alarmdecoder.util), 16
FTDI_PRODUCT_ID (alarmdecoder.devices.USBDevice attribute), 8
FTDI_VENDOR_ID (alarmdecoder.devices.USBDevice attribute), 8

G

get_config() (alarmdecoder.decoder.AlarmDecoder method), 6

I

id (alarmdecoder.decoder.AlarmDecoder attribute), 5
id (alarmdecoder.devices.Device attribute), 7
interface (alarmdecoder.devices.SerialDevice attribute), 10
interface (alarmdecoder.devices.SocketDevice attribute), 11
interface (alarmdecoder.devices.USBDevice attribute), 9
internal_address_mask (alarmdecoder.decoder.AlarmDecoder attribute), 6
InvalidMessageError, 16
is_reader_alive() (alarmdecoder.devices.Device method), 7

K

KEY_F1 (alarmdecoder.decoder.AlarmDecoder attribute), 5
KEY_F2 (alarmdecoder.decoder.AlarmDecoder attribute), 5
KEY_F3 (alarmdecoder.decoder.AlarmDecoder attribute), 5
KEY_F4 (alarmdecoder.decoder.AlarmDecoder attribute), 5
KEY_PANIC (alarmdecoder.decoder.AlarmDecoder attribute), 5

L

loop (alarmdecoder.messages.RFMessage attribute), 14
LRRMessage (class in alarmdecoder.messages), 14

M

mask (alarmdecoder.messages.Message attribute), 13
Message (class in alarmdecoder.messages), 12
mode (alarmdecoder.decoder.AlarmDecoder attribute), 5

N

name (alarmdecoder.zonetracking.Zone attribute), 15
NoDeviceError, 16
numeric_code (alarmdecoder.messages.Message attribute), 13

O

on_alarm (alarmdecoder.decoder.AlarmDecoder attribute), 3
on_alarm_restored (alarmdecoder.decoder.AlarmDecoder attribute), 3
on_arm (alarmdecoder.decoder.AlarmDecoder attribute), 3

- on_attached (alarmdecoder.devices.USBDevice.DetectThread attribute), 9
- on_boot (alarmdecoder.decoder.AlarmDecoder attribute), 3
- on_bypass (alarmdecoder.decoder.AlarmDecoder attribute), 3
- on_close (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_close (alarmdecoder.devices.Device attribute), 7
- on_config_received (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_detached (alarmdecoder.devices.USBDevice.DetectThread attribute), 9
- on_disarm (alarmdecoder.decoder.AlarmDecoder attribute), 3
- on_expander_message (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_fault (alarmdecoder.zonetracking.Zonetracker attribute), 15
- on_fire (alarmdecoder.decoder.AlarmDecoder attribute), 3
- on_low_battery (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_lrr_message (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_message (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_open (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_open (alarmdecoder.devices.Device attribute), 7
- on_panic (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_power_changed (alarmdecoder.decoder.AlarmDecoder attribute), 3
- on_read (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_read (alarmdecoder.devices.Device attribute), 7
- on_relay_changed (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_restore (alarmdecoder.zonetracking.Zonetracker attribute), 15
- on_rfx_message (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_sending_received (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_write (alarmdecoder.decoder.AlarmDecoder attribute), 5
- on_write (alarmdecoder.devices.Device attribute), 7
- on_zone_fault (alarmdecoder.decoder.AlarmDecoder attribute), 4
- on_zone_restore (alarmdecoder.decoder.AlarmDecoder attribute), 4
- open() (alarmdecoder.decoder.AlarmDecoder method), 6
- open() (alarmdecoder.devices.SerialDevice method), 10
- open() (alarmdecoder.devices.SocketDevice method), 11
- open() (alarmdecoder.devices.USBDevice method), 9
- ## P
- panel_data (alarmdecoder.messages.Message attribute), 13
- panel_type (alarmdecoder.messages.Message attribute), 13
- partition (alarmdecoder.messages.LRRMessage attribute), 14
- perimeter_only (alarmdecoder.messages.Message attribute), 13
- PRODUCT_IDS (alarmdecoder.devices.USBDevice attribute), 8
- programming_mode (alarmdecoder.messages.Message attribute), 12
- ## R
- raw (alarmdecoder.messages.BaseMessage attribute), 12
- read() (alarmdecoder.devices.SerialDevice method), 10
- read() (alarmdecoder.devices.SocketDevice method), 11
- read() (alarmdecoder.devices.USBDevice method), 9
- read_line() (alarmdecoder.devices.SerialDevice method), 10
- read_line() (alarmdecoder.devices.SocketDevice method), 11
- read_line() (alarmdecoder.devices.USBDevice method), 9
- READ_TIMEOUT (alarmdecoder.devices.Device.ReadThread attribute), 7
- ready (alarmdecoder.messages.Message attribute), 12
- reboot() (alarmdecoder.decoder.AlarmDecoder method), 6
- RELAY (alarmdecoder.messages.ExpanderMessage attribute), 13
- RFMessage (class in alarmdecoder.messages), 14
- run() (alarmdecoder.devices.Device.ReadThread method), 7
- run() (alarmdecoder.devices.USBDevice.DetectThread method), 9
- ## S
- save_config() (alarmdecoder.decoder.AlarmDecoder method), 6
- send() (alarmdecoder.decoder.AlarmDecoder method), 6
- serial_number (alarmdecoder.devices.USBDevice attribute), 8
- serial_number (alarmdecoder.messages.RFMessage attribute), 14

SerialDevice (class in alarmdecoder.devices), 10
SocketDevice (class in alarmdecoder.devices), 10
ssl (alarmdecoder.devices.SocketDevice attribute), 11
SSL (class in alarmdecoder.devices), 6
ssl_ca (alarmdecoder.devices.SocketDevice attribute), 11
ssl_certificate (alarmdecoder.devices.SocketDevice attribute), 11
ssl_key (alarmdecoder.devices.SocketDevice attribute), 11
STAGE_BOOT (alarmdecoder.util.Firmware attribute), 16
STAGE_DONE (alarmdecoder.util.Firmware attribute), 16
STAGE_LOAD (alarmdecoder.util.Firmware attribute), 16
STAGE_START (alarmdecoder.util.Firmware attribute), 16
STAGE_UPLOADING (alarmdecoder.util.Firmware attribute), 16
STAGE_WAITING (alarmdecoder.util.Firmware attribute), 16
start_detection() (alarmdecoder.devices.USBDevice class method), 8
STATUS (alarmdecoder.zonetracking.Zone attribute), 15
status (alarmdecoder.zonetracking.Zone attribute), 15
stop() (alarmdecoder.devices.Device.ReadThread method), 7
stop() (alarmdecoder.devices.USBDevice.DetectThread method), 9
stop_detection() (alarmdecoder.devices.USBDevice class method), 8
stop_reader() (alarmdecoder.devices.Device method), 7
supervision (alarmdecoder.messages.RFMessage attribute), 14
SysCallError (alarmdecoder.devices.SSL attribute), 7
system_fault (alarmdecoder.messages.Message attribute), 13

T

text (alarmdecoder.messages.Message attribute), 13
TimeoutError, 16
timestamp (alarmdecoder.messages.BaseMessage attribute), 12
timestamp (alarmdecoder.zonetracking.Zone attribute), 15
type (alarmdecoder.messages.ExpanderMessage attribute), 13

U

update() (alarmdecoder.zonetracking.Zonetracker method), 15
upload() (alarmdecoder.util.Firmware static method), 16
USBDevice (class in alarmdecoder.devices), 7

USBDevice.DetectThread (class in alarmdecoder.devices), 9

V

value (alarmdecoder.messages.ExpanderMessage attribute), 14
value (alarmdecoder.messages.RFMessage attribute), 14

W

WantReadError (alarmdecoder.devices.SSL attribute), 7
write() (alarmdecoder.devices.SerialDevice method), 10
write() (alarmdecoder.devices.SocketDevice method), 11
write() (alarmdecoder.devices.USBDevice method), 9

Z

ZONE (alarmdecoder.messages.ExpanderMessage attribute), 13
zone (alarmdecoder.zonetracking.Zone attribute), 15
Zone (class in alarmdecoder.zonetracking), 14
zone_bypassed (alarmdecoder.messages.Message attribute), 12
zones (alarmdecoder.zonetracking.Zonetracker attribute), 15
Zonetracker (class in alarmdecoder.zonetracking), 15