
PyAbacus Documentation

Release 1.0.6

Tausand Electronica

Jan 03, 2019

CONTENTS

1	Contents	3
1.1	pyAbacus.core	3
1.2	pyAbacus.exceptions	5
1.3	pyAbacus.constants	5
2	Indices and tables	7
	Python Module Index	9
	Index	11



pyAbacus was build to simplify the usage of [Tausands](#) tools.

CONTENTS

1.1 pyAbacus.core

class pyAbacus.core.**AbacusSerial** (*port, bounce_timeout=40*)
Builds a serial port from pyserial.

findIdn ()

flush ()

getIdn ()

getNChannels ()

readSerial ()

testDevice ()

writeSerial (*command, address, data_16o32*)

class pyAbacus.core.**CountersValues** (*n_channels*)

getCountersID ()

getNumericAddresses ()

getTimeLeft ()

getValue (*channel*)

getValues (*channels*)

getValuesFormatted (*channels*)

setCountersID (*id*)

setTimeLeft (*time*)

setValueFromArray (*address, value*)

time_left = None
in ms

class pyAbacus.core.**Settings2Ch**

getAddressAndValue (*timer*)

getSetting (*timer*)

getSettingStr (*timer*)

setSetting (*setting, value*)

class `pyAbacus.core.Settings48Ch`

4 and 8 channel devices use as time base a second. Nevertheless 2 channel uses ns for all timers with the exception of the sampling time (ms).

exponentRepresentationToValue (*c, e*)

exponentsToBits (*c, e*)

fromBitsToValue (*bits*)

getAddressAndValue (*timer*)

getChannels ()

getSetting (*timer*)

For all timers: returns nanoseconds, for sampling returns ms.

getSettingStr (*timer*)

initAddresses ()

setSetting (*setting, value*)

For all timers: value is in nanoseconds, for sampling in ms.

valueToExponentRepresentation (*number*)

class `pyAbacus.core.Settings4Ch`

4 and 8 channel devices use as time base a second. Nevertheless 2 channel uses ns for all timers with the exception of the sampling time (ms).

class `pyAbacus.core.Settings8Ch`

4 and 8 channel devices use as time base a second. Nevertheless 2 channel uses ns for all timers with the exception of the sampling time (ms).

class `pyAbacus.core.Stream` (*abacus_port, counters, output_function=<built-in function print>*)

setCounters (*counters*)

start ()

stop ()

`pyAbacus.core.close` (*abacus_port*)

`pyAbacus.core.dataArraysToCounters` (*abacus_port, addresses, data*)

`pyAbacus.core.dataArraysToSettings` (*abacus_port, addresses, data*)

`pyAbacus.core.dataStreamToDataArrays` (*input_string*)

`pyAbacus.core.findDevices` (*print_on=True*)

`pyAbacus.core.getAllCounters` (*abacus_port*)

`pyAbacus.core.getAllSettings` (*abacus_port*)

`pyAbacus.core.getChannelsFromName` (*name*)

`pyAbacus.core.getCountersID` (*abacus_port*)

`pyAbacus.core.getFollowingCounters` (*abacus_port, counters*)

`pyAbacus.core.getIdn` (*abacus_port*)

`pyAbacus.core.getSetting` (*abacus_port, setting*)


```

pyAbacus.core.getTimeLeft (abacus_port)
pyAbacus.core.open (abacus_port)
pyAbacus.core.readSerial (abacus_port)
pyAbacus.core.renameDuplicates (old)
pyAbacus.core.setAllSettings (abacus_port, new_settings)
pyAbacus.core.setSetting (abacus_port, setting, value)
pyAbacus.core.writeSerial (abacus_port, command, address, data_16o32)

```

1.2 pyAbacus.exceptions

```

exception pyAbacus.exceptions.AbacusError (message=")
    An unexpected error occurred.

exception pyAbacus.exceptions.BaseError (message)

exception pyAbacus.exceptions.CheckSumError
    An error occurred while doing check sum.

exception pyAbacus.exceptions.InvalidValueError (message=")
    The selected value is not valid

exception pyAbacus.exceptions.TimeoutError (message=")
    A time out error occurred

```

1.3 pyAbacus.constants

```

pyAbacus.constants.ADDRESS_DIRECTORY_2CH = {'coincidence_window_ms': 22, 'coincidence_windo
    Memory addresses

pyAbacus.constants.BAUDRATE = 115200
    Default baudrate for the serial port communication

pyAbacus.constants.BOUNCE_TIMEOUT = 40
    Number of times a specific transmittion is tried

pyAbacus.constants.COINCIDENCE_WINDOW_DEFAULT_VALUE = 5
    Default coincidence window time value (ns).

pyAbacus.constants.COINCIDENCE_WINDOW_MAXIMUM_VALUE = 50000
    Maximum coincidence window time value (ns).

pyAbacus.constants.COINCIDENCE_WINDOW_MINIMUM_VALUE = 5
    Minimum coincidence window time value (ns).

pyAbacus.constants.COINCIDENCE_WINDOW_STEP_VALUE = 5
    Increase ratio on the coincidence window time value (ns).

pyAbacus.constants.COUNTERS_VALUES = {}
    Global counters values variable

pyAbacus.constants.CURRENT_OS = 'linux'
    Current operative system

```

`pyAbacus.constants.DELAY_DEFAULT_VALUE = 100`
Default delay time value (ns).

`pyAbacus.constants.DELAY_MAXIMUM_VALUE = 100`
Maximum delay time value (ns).

`pyAbacus.constants.DELAY_MINIMUM_VALUE = 0`
Minimum delay time value (ns).

`pyAbacus.constants.DELAY_STEP_VALUE = 5`
Increase ratio on the delay time value (ns).

`pyAbacus.constants.END_COMMUNICATION = 4`
End of message

`pyAbacus.constants.MAXIMUM_WRITING_TRIES = 20`
Number of tries done to write a value

`pyAbacus.constants.READ_VALUE = 14`
Reading operation signal

`pyAbacus.constants.SAMPLING_DEFAULT_VALUE = 100`
Default sampling time value (ms)

`pyAbacus.constants.SAMPLING_VALUES = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000]`
From (1, 2, 5) ms to 1000 s

`pyAbacus.constants.SETTINGS = {}`
Global settings variable

`pyAbacus.constants.SLEEP_DEFAULT_VALUE = 25`
Default sleep time value (ns).

`pyAbacus.constants.SLEEP_MAXIMUM_VALUE = 100`
Maximum sleep time value (ns).

`pyAbacus.constants.SLEEP_MINIMUM_VALUE = 0`
Minimum sleep time value (ns).

`pyAbacus.constants.SLEEP_STEP_VALUE = 5`
Increase ratio on the sleep time value (ns).

`pyAbacus.constants.START_COMMUNICATION = 2`
Begin message signal

`pyAbacus.constants.TIMEOUT = 0.5`
Maximum time without answer from the serial port

`pyAbacus.constants.WRITE_VALUE = 15`
Writing operation signal

INDICES AND TABLES

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

PYTHON MODULE INDEX

p

`pyAbacus.constants`, [5](#)
`pyAbacus.core`, [3](#)
`pyAbacus.exceptions`, [5](#)

A

AbacusError, 5
 AbacusSerial (class in *pyAbacus.core*), 3
 ADDRESS_DIRECTORY_2CH (in module *pyAbacus.constants*), 5

B

BaseError, 5
 BAUDRATE (in module *pyAbacus.constants*), 5
 BOUNCE_TIMEOUT (in module *pyAbacus.constants*), 5

C

ChecksumError, 5
 close() (in module *pyAbacus.core*), 4
 COINCIDENCE_WINDOW_DEFAULT_VALUE (in module *pyAbacus.constants*), 5
 COINCIDENCE_WINDOW_MAXIMUM_VALUE (in module *pyAbacus.constants*), 5
 COINCIDENCE_WINDOW_MINIMUM_VALUE (in module *pyAbacus.constants*), 5
 COINCIDENCE_WINDOW_STEP_VALUE (in module *pyAbacus.constants*), 5
 COUNTERS_VALUES (in module *pyAbacus.constants*), 5
 CountersValues (class in *pyAbacus.core*), 3
 CURRENT_OS (in module *pyAbacus.constants*), 5

D

dataArraysToCounters() (in module *pyAbacus.core*), 4
 dataArraysToSettings() (in module *pyAbacus.core*), 4
 dataStreamToDataArrays() (in module *pyAbacus.core*), 4
 DELAY_DEFAULT_VALUE (in module *pyAbacus.constants*), 5
 DELAY_MAXIMUM_VALUE (in module *pyAbacus.constants*), 6
 DELAY_MINIMUM_VALUE (in module *pyAbacus.constants*), 6
 DELAY_STEP_VALUE (in module *pyAbacus.constants*), 6

E

END_COMMUNICATION (in module *pyAbacus.constants*), 6
 exponentRepresentationToValue() (*pyAbacus.core.Settings48Ch* method), 4
 exponentsToBits() (*pyAbacus.core.Settings48Ch* method), 4

F

findDevices() (in module *pyAbacus.core*), 4
 findIdn() (*pyAbacus.core.AbacusSerial* method), 3
 flush() (*pyAbacus.core.AbacusSerial* method), 3
 fromBitsToValue() (*pyAbacus.core.Settings48Ch* method), 4

G

getAddressAndValue() (*pyAbacus.core.Settings2Ch* method), 3
 getAddressAndValue() (*pyAbacus.core.Settings48Ch* method), 4
 getAllCounters() (in module *pyAbacus.core*), 4
 getAllSettings() (in module *pyAbacus.core*), 4
 getChannels() (*pyAbacus.core.Settings48Ch* method), 4
 getChannelsFromName() (in module *pyAbacus.core*), 4
 getCountersID() (in module *pyAbacus.core*), 4
 getCountersID() (*pyAbacus.core.CountersValues* method), 3
 getFollowingCounters() (in module *pyAbacus.core*), 4
 getIdn() (in module *pyAbacus.core*), 4
 getIdn() (*pyAbacus.core.AbacusSerial* method), 3
 getNChannels() (*pyAbacus.core.AbacusSerial* method), 3
 getNumericAddresses() (*pyAbacus.core.CountersValues* method), 3
 getSetting() (in module *pyAbacus.core*), 4
 getSetting() (*pyAbacus.core.Settings2Ch* method), 3
 getSetting() (*pyAbacus.core.Settings48Ch* method), 4

`getSettingStr()` (*pyAbacus.core.Settings2Ch method*), 3
`getSettingStr()` (*pyAbacus.core.Settings48Ch method*), 4
`getTimeLeft()` (*in module pyAbacus.core*), 4
`getTimeLeft()` (*pyAbacus.core.CountersValues method*), 3
`getValue()` (*pyAbacus.core.CountersValues method*), 3
`getValues()` (*pyAbacus.core.CountersValues method*), 3
`getValuesFormatted()` (*pyAbacus.core.CountersValues method*), 3

I

`initAddreses()` (*pyAbacus.core.Settings48Ch method*), 4
`InvalidValueError`, 5

M

`MAXIMUM_WRITING_TRIES` (*in module pyAbacus.constants*), 6

O

`open()` (*in module pyAbacus.core*), 5

P

`pyAbacus.constants` (*module*), 5
`pyAbacus.core` (*module*), 3
`pyAbacus.exceptions` (*module*), 5

R

`READ_VALUE` (*in module pyAbacus.constants*), 6
`readSerial()` (*in module pyAbacus.core*), 5
`readSerial()` (*pyAbacus.core.AbacusSerial method*), 3
`renameDuplicates()` (*in module pyAbacus.core*), 5

S

`SAMPLING_DEFAULT_VALUE` (*in module pyAbacus.constants*), 6
`SAMPLING_VALUES` (*in module pyAbacus.constants*), 6
`setAllSettings()` (*in module pyAbacus.core*), 5
`setCounters()` (*pyAbacus.core.Stream method*), 4
`setCountersID()` (*pyAbacus.core.CountersValues method*), 3
`setSetting()` (*in module pyAbacus.core*), 5
`setSetting()` (*pyAbacus.core.Settings2Ch method*), 3
`setSetting()` (*pyAbacus.core.Settings48Ch method*), 4
`setTimeLeft()` (*pyAbacus.core.CountersValues method*), 3

`SETTINGS` (*in module pyAbacus.constants*), 6
`Settings2Ch` (*class in pyAbacus.core*), 3
`Settings48Ch` (*class in pyAbacus.core*), 4
`Settings4Ch` (*class in pyAbacus.core*), 4
`Settings8Ch` (*class in pyAbacus.core*), 4
`setValueFromArray()` (*pyAbacus.core.CountersValues method*), 3
`SLEEP_DEFAULT_VALUE` (*in module pyAbacus.constants*), 6
`SLEEP_MAXIMUM_VALUE` (*in module pyAbacus.constants*), 6
`SLEEP_MINIMUM_VALUE` (*in module pyAbacus.constants*), 6
`SLEEP_STEP_VALUE` (*in module pyAbacus.constants*), 6
`start()` (*pyAbacus.core.Stream method*), 4
`START_COMMUNICATION` (*in module pyAbacus.constants*), 6
`stop()` (*pyAbacus.core.Stream method*), 4
`Stream` (*class in pyAbacus.core*), 4

T

`testDevice()` (*pyAbacus.core.AbacusSerial method*), 3
`time_left` (*pyAbacus.core.CountersValues attribute*), 3
`TIMEOUT` (*in module pyAbacus.constants*), 6
`TimeOutError`, 5

V

`valueToExponentRepresentation()` (*pyAbacus.core.Settings48Ch method*), 4

W

`WRITE_VALUE` (*in module pyAbacus.constants*), 6
`writeSerial()` (*in module pyAbacus.core*), 5
`writeSerial()` (*pyAbacus.core.AbacusSerial method*), 3