

---

# **trio-serial**

***Release 0.4.0***

**Jörn Heissler**

**Sep 30, 2023**



**CONTENTS:**

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Example</b>	<b>5</b>
<b>3</b>	<b>API reference</b>	<b>7</b>
<b>4</b>	<b>Contributing and Support</b>	<b>11</b>
<b>5</b>	<b>Development and Testing</b>	<b>13</b>
<b>6</b>	<b>Changelog</b>	<b>15</b>
<b>7</b>	<b>License</b>	<b>17</b>
<b>8</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



This project is an adaption of the [pyserial](#) project for [trio](#).



## INSTALLATION

```
$ pip install trio-serial
```





**EXAMPLE**

```
from trio import run
from trio_serial import SerialStream

async def main():
    async with SerialStream("/dev/ttyUSB0") as ser:
        for i in range(10):
            buf = await ser.receive_some()
            await ser.send_all(buf)
            await ser.send_all(buf)

run(main)
```



## API REFERENCE

```
class trio_serial.abstract.AbstractSerialStream(port, *, exclusive=False, baudrate=115200,  
                                              bytesize=8, parity=Parity.NONE,  
                                              stopbits=StopBits.ONE, xonxoff=False, rtscts=False)
```

Operating system independant public interface of `SerialStream`.

Create new `SerialStream` object.

### Parameters

- **port** (*str*) – Name of port. Format depends on implementation. This could be “/dev/ttyUSB0” on Linux or “COM7” on Windows.
- **exclusive** (*bool*) – Lock port for exclusive use
- **baudrate** (*int*) – Initial Port speed
- **bytesize** (*int*) – Number of bits per byte
- **parity** (*Parity*) – Parity
- **stopbits** (*StopBits*) – Number of stop bits
- **xonxoff** (*bool*) – Software Flow Control with XON/XOFF
- **rtscts** (*bool*) – Hardware Flow Control with RTS/CTS

**abstract async aclose()**

Cleanly close the port.

Do nothing if already closed.

### Return type

None

**abstract async aopen()**

Open the port and configure it with the initial state from `__init__()`.

### Return type

None

**abstract async discard\_input()**

Discard any unread input.

### Return type

None

**abstract async discard\_output()**

Discard any unwritten output.

**Return type**

None

**abstract async get\_cts()**

Retrieve current *Clear To Send* state.

**Returns**

Current CTS state

**Return type**

bool

**abstract async get\_hangup()**

Retrieve current *Hangup on Close* state.

**Returns**

Current *Hangup on Close* state

**Return type**

bool

**abstract async get\_rts()**

Retrieve current *Ready To Send* state.

**Returns**

Current RTS state

**Return type**

bool

**property port: str**

Get the port name.

**Returns**

port name or device

**async receive\_some(max\_bytes=None)**

Receive some bytes from the serial port.

**Parameters**

**max\_bytes** (*int* / *None*) – Maximum number of bytes to receive.

**Returns**

On success, between 1 and **max\_bytes** bytes. On End-of-file (e.g. serial port is gone) an empty bytes object is returned.

**Return type**

bytes

**async send\_all(data)**

Send data to the serial port. :param data: Data to send

**Parameters**

**data** (*ByteString*) –

**Return type**

None

**abstract async send\_break(duration=0.25)**

Transmit a continuous stream of zero-valued bits for a specific duration.

**Params:**

duration: Number of seconds

**Parameters**

**duration** (*float*) –

**Return type**

None

**abstract async set\_hangup**(*value*)

*Set Hangup on Close state.*

**Parameters**

**value** (*bool*) – New Hangup on Close state

**Return type**

None

**abstract async set\_rts**(*value*)

*Set Ready To Send state.*

**Parameters**

**value** (*bool*) – New Ready To Send state

**Return type**

None

**async wait\_send\_all\_might\_not\_block**()

*Wait until sending might not block (it still might block).*

**Return type**

None

**class trio\_serial.abstract.Parity**(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Enumeration of parity types.

**EVEN** = 2

Even parity

**MARK** = 4

Parity bit always 1

**NONE** = 1

No parity

**ODD** = 3

Odd parity

**SPACE** = 5

Parity bit always 0

**class trio\_serial.abstract.StopBits**(*value, names=None, \*, module=None, qualname=None, type=None, start=1, boundary=None*)

Enumeration of stop bit lengths.

**ONE** = 1

One bit

**ONE\_POINT\_FIVE = 2**

One and a half bits

**TWO = 3**

Two bits

## CONTRIBUTING AND SUPPORT

Please create a new [Issue](#) or [Pull request](#) on GitHub to contribute changes or ask questions.





## **DEVELOPMENT AND TESTING**

To test the package locally, run: `.. code-block:: console`

```
$ /path/to/pip install -U .
```

where pip could be in a virtual environment.



## CHANGELOG

### 6.1 0.4.0 - 2023-09-30

- Remove loop on `receive_some` method. (#9)
- Update dependencies.

### 6.2 0.3.0 - 2021-05-15

- Fix cleanup issue when running in sync context. Remove `close` method. (#4)

### 6.3 0.2.1 - 2021-04-09

- Do not handle modem bits (e.g. rts) in constructor / `aopen`. Use `set_rts()/get_rts()` instead. (#2)
- Add new methods to control “hangup on close”:
  - `get_hangup()`
  - `set_hangup()`

### 6.4 0.1.2 - 2021-02-07

- More relaxed dependencies

### 6.5 0.1.1 - 2021-01-31

- Add new methods:
  - `port()`
  - `discard_input()`
  - `discard_output()`
  - `send_break()`

## **6.6 0.1.0 - 2020-12-21**

- Initial release.

## LICENSE

This documentation is covered under the [CC BY-SA 4.0 license](#).



## INDICES AND TABLES

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





## A

`AbstractSerialStream` (class in `trio_serial.abstract`), 7  
`aclose()` (`trio_serial.abstract.AbstractSerialStream` method), 7  
`aopen()` (`trio_serial.abstract.AbstractSerialStream` method), 7

## D

`discard_input()` (`trio_serial.abstract.AbstractSerialStream` method), 7  
`discard_output()` (`trio_serial.abstract.AbstractSerialStream` method), 7

## E

`EVEN` (`trio_serial.abstract.Parity` attribute), 9

## G

`get_cts()` (`trio_serial.abstract.AbstractSerialStream` method), 8  
`get_hangup()` (`trio_serial.abstract.AbstractSerialStream` method), 8  
`get_rts()` (`trio_serial.abstract.AbstractSerialStream` method), 8

## M

`MARK` (`trio_serial.abstract.Parity` attribute), 9

## N

`NONE` (`trio_serial.abstract.Parity` attribute), 9

## O

`ODD` (`trio_serial.abstract.Parity` attribute), 9  
`ONE` (`trio_serial.abstract.StopBits` attribute), 9  
`ONE_POINT_FIVE` (`trio_serial.abstract.StopBits` attribute), 9

## P

`Parity` (class in `trio_serial.abstract`), 9  
`port` (`trio_serial.abstract.AbstractSerialStream` property), 8

## R

`receive_some()` (`trio_serial.abstract.AbstractSerialStream` method), 8

## S

`send_all()` (`trio_serial.abstract.AbstractSerialStream` method), 8  
`send_break()` (`trio_serial.abstract.AbstractSerialStream` method), 8  
`set_hangup()` (`trio_serial.abstract.AbstractSerialStream` method), 9  
`set_rts()` (`trio_serial.abstract.AbstractSerialStream` method), 9

`SPACE` (`trio_serial.abstract.Parity` attribute), 9

`StopBits` (class in `trio_serial.abstract`), 9

## T

`TWO` (`trio_serial.abstract.StopBits` attribute), 10

## W

`wait_send_all_might_not_block()` (`trio_serial.abstract.AbstractSerialStream` method), 9