

---

# **smbus2 Documentation**

***Release 0.4.1***

**Karl-Petter Lindegaard**

**Apr 26, 2021**



---

## Contents

---

<b>Python Module Index</b>	<b>5</b>
<b>Index</b>	<b>7</b>



smbus2 - A drop-in replacement for smbus-cffi/smbus-python

**class** `smbus2.SMBus` (*bus=None, force=False*)

**block\_process\_call** (*i2c\_addr, register, data, force=None*)

Executes a SMBus Block Process Call, sending a variable-size data block and receiving another variable-size response

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Register to read/write to
- **data** (*list*) – List of bytes
- **force** (*Boolean*) –

**Returns** List of bytes

**Return type** list

**close** ()

Close the i2c connection.

**enable\_pec** (*enable=True*)

Enable/Disable PEC (Packet Error Checking) - SMBus 1.1 and later

**Parameters** **enable** (*Boolean*) –

**i2c\_rdwr** (*\*i2c\_msgs*)

Combine a series of i2c read and write operations in a single transaction (with repeated start bits but no stop bits in between).

This method takes i2c\_msg instances as input, which must be created first with `i2c_msg.read()` or `i2c_msg.write()`.

**Parameters** **i2c\_msgs** (*i2c\_msg*) – One or more i2c\_msg class instances.

**Return type** `None`

**open** (*bus*)

Open a given i2c bus.

**Parameters** **bus** (*int or str*) – i2c bus number (e.g. 0 or 1) or an absolute file path (e.g. '/dev/i2c-42').

**Raises** **TypeError** – if type(bus) is not in (int, str)

**pec**

Get and set SMBus PEC. 0 = disabled (default), 1 = enabled.

**process\_call** (*i2c\_addr, register, value, force=None*)

Executes a SMBus Process Call, sending a 16-bit value and receiving a 16-bit response

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Register to read/write to
- **value** (*int*) – Word value to transmit

- **force** (*Boolean*) –

**Return type** *int*

**read\_block\_data** (*i2c\_addr, register, force=None*)

Read a block of up to 32-bytes from a given register.

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Start register
- **force** (*Boolean*) –

**Returns** List of bytes

**Return type** *list*

**read\_byte** (*i2c\_addr, force=None*)

Read a single byte from a device.

**Return type** *int*

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **force** (*Boolean*) –

**Returns** Read byte value

**read\_byte\_data** (*i2c\_addr, register, force=None*)

Read a single byte from a designated register.

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Register to read
- **force** (*Boolean*) –

**Returns** Read byte value

**Return type** *int*

**read\_i2c\_block\_data** (*i2c\_addr, register, length, force=None*)

Read a block of byte data from a given register.

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Start register
- **length** (*int*) – Desired block length
- **force** (*Boolean*) –

**Returns** List of bytes

**Return type** *list*

**read\_word\_data** (*i2c\_addr, register, force=None*)

Read a single word (2 bytes) from a given register.

**Parameters**

- **i2c\_addr** (*int*) – i2c address

- **register** (*int*) – Register to read
- **force** (*Boolean*) –

Returns 2-byte word

Return type *int*

**write\_block\_data** (*i2c\_addr*, *register*, *data*, *force=None*)

Write a block of byte data to a given register.

Parameters

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Start register
- **data** (*list*) – List of bytes
- **force** (*Boolean*) –

Return type *None*

**write\_byte** (*i2c\_addr*, *value*, *force=None*)

Write a single byte to a device.

Parameters

- **i2c\_addr** (*int*) – i2c address
- **value** (*int*) – value to write
- **force** (*Boolean*) –

**write\_byte\_data** (*i2c\_addr*, *register*, *value*, *force=None*)

Write a byte to a given register.

Parameters

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Register to write to
- **value** (*int*) – Byte value to transmit
- **force** (*Boolean*) –

Return type *None*

**write\_i2c\_block\_data** (*i2c\_addr*, *register*, *data*, *force=None*)

Write a block of byte data to a given register.

Parameters

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Start register
- **data** (*list*) – List of bytes
- **force** (*Boolean*) –

Return type *None*

**write\_quick** (*i2c\_addr*, *force=None*)

Perform quick transaction. Throws IOError if unsuccessful. :param i2c\_addr: i2c address :type i2c\_addr: int :param force: :type force: Boolean

**write\_word\_data** (*i2c\_addr*, *register*, *value*, *force=None*)

Write a single word (2 bytes) to a given register.

**Parameters**

- **i2c\_addr** (*int*) – i2c address
- **register** (*int*) – Register to write to
- **value** (*int*) – Word value to transmit
- **force** (*Boolean*) –

**Return type** *None*

**class** `smbus2.i2c_msg`

As defined in `i2c.h`.

**addr**

Structure/Union member

**buf**

Structure/Union member

**flags**

Structure/Union member

**len**

Structure/Union member

**static read** (*address*, *length*)

Prepares an i2c read transaction.

**Parameters**

- **address** – Slave address.
- **length** – Number of bytes to read.

**Type** *address*: int

**Type** *length*: int

**Returns** New *i2c\_msg* instance for read operation.

**Return type** *i2c\_msg*

**static write** (*address*, *buf*)

Prepares an i2c write transaction.

**Parameters**

- **address** (*int*) – Slave address.
- **buf** (*list*) – Bytes to write. Either list of values or str.

**Returns** New *i2c\_msg* instance for write operation.

**Return type** *i2c\_msg*



### S

`smbus2`, [1](#)



## A

`addr` (*smbus2.i2c\_msg* attribute), 4

## B

`block_process_call()` (*smbus2.SMBus* method), 1

`buf` (*smbus2.i2c\_msg* attribute), 4

## C

`close()` (*smbus2.SMBus* method), 1

## E

`enable_pec()` (*smbus2.SMBus* method), 1

## F

`flags` (*smbus2.i2c\_msg* attribute), 4

## I

`i2c_msg` (class in *smbus2*), 4

`i2c_rdwr()` (*smbus2.SMBus* method), 1

## L

`len` (*smbus2.i2c\_msg* attribute), 4

## O

`open()` (*smbus2.SMBus* method), 1

## P

`pec` (*smbus2.SMBus* attribute), 1

`process_call()` (*smbus2.SMBus* method), 1

## R

`read()` (*smbus2.i2c\_msg* static method), 4

`read_block_data()` (*smbus2.SMBus* method), 2

`read_byte()` (*smbus2.SMBus* method), 2

`read_byte_data()` (*smbus2.SMBus* method), 2

`read_i2c_block_data()` (*smbus2.SMBus* method), 2

`read_word_data()` (*smbus2.SMBus* method), 2

## S

`SMBus` (class in *smbus2*), 1

`smbus2` (module), 1

## W

`write()` (*smbus2.i2c\_msg* static method), 4

`write_block_data()` (*smbus2.SMBus* method), 3

`write_byte()` (*smbus2.SMBus* method), 3

`write_byte_data()` (*smbus2.SMBus* method), 3

`write_i2c_block_data()` (*smbus2.SMBus* method), 3

`write_quick()` (*smbus2.SMBus* method), 3

`write_word_data()` (*smbus2.SMBus* method), 3