smbus2

build passing | python 2.7 | 3.4 | 3.5 | 3.6 | 3.7 | 3.8 | 3.9 | pypi v0.4.1

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

v: latest

`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 byte 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

v: latest

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`**