SMBus / I2C Functions

SMBus (System Management Bus) is a subset from the I2C protocol. When writing a driver for an I2C device try to use the SMBus commands if possible (if the device uses only that subset of the I2C protocol) as it makes it possible to use the device driver on both SMBus adapters and I2C adapters.

Note: The address parameter is the 7 bit version of the address (excluding the read / write bit). The address will be shifted left 1 bit when added to the read/write bit.

```
long write_quick(int addr)
```
Send only the read / write bit

```
long read_byte(int addr)
```
Read a single byte from a device, without specifying a device register.

```
long write_byte(int addr, char val)
```
Send a single byte to a device

```
long read_byte_data(int addr, char cmd)
```
Read Byte Data transaction.

```
long write_byte_data(int addr,char cmd, char val)
```
Write Byte Data transaction.

```
long read_word_data(int addr, char cmd)
```
Read Word Data transaction.

```
long write_word_data(int addr, char cmd, int val)
```
Write Word Data transaction.

```
long process_call(int addr, char cmd, int val)
```
Process Call transaction.

```
long[] read_block_data(int addr, char cmd)
```
Read Block Data transaction.

```
write_block_data(int addr, char cmd, long vals[])
```
Write up to 32 bytes to a device. This function adds an initial byte indicating the length of the vals array before the vals array. Use write_i2c_block_data instead!

```
long[] block_process_call(int addr, char cmd, long vals[])
```
Block Process Call transaction.

I2C Access Functions

```
long[] read_i2c_block_data(int addr, char cmd)
```

Block Read transaction.

```
write_i2c_block_data(int addr, char cmd, long vals[])
```

Block Write transaction.

Code Example

```
#!/usr/bin/python

import smbus
# 0 = /dev/i2c-0 (port I2C0), 1 = /dev/i2c-1 (port I2C1)
bus = smbus.SMBus(1)
#7 bit address (will be left shifted to add the read write bit)
DEVICE_ADDRESS = 0x15
DEVICE_REG_MODE1 = 0x00
DEVICE_REG_LEDOUT0 = 0x1d

#Write a single register
bus.write_byte_data(DEVICE_ADDRESS, DEVICE_REG_MODE1, 0x80)

#Write an array of registers
ledout_values = [0xff, 0xff, 0xff, 0xff, 0xff, 0xff]
bus.write_i2c_block_data(DEVICE_ADDRESS, DEVICE_REG_LEDOUT0, ledout_values)
```